

USER GUIDE

Essential Studio for EJ2 React

Version - v25.1.35 | Release Date - March 15, 2024

Kanban	26
Getting Started.....	26
Prerequisites	26
Dependencies.....	26
Create the React application.....	26
Import the Syncfusion CSS styles	27
Add Kanban component to the application.....	28
Run the application	30
Populating cards.....	30
Enable swimlane	32
Creating a Next.js Application Using Syncfusion React Components	34
What is Next.js?	34
Prerequisites	34
Create a Next.js application	34
Install Syncfusion React packages.....	35
Import Syncfusion CSS styles	35
Add Syncfusion React component	36
Run the application	36
Columns in React Kanban component.....	37
Single-key mapping	37
Multi-key mapping	41
Header text	45
Header template	45
Toggle columns	50
Stacked headers	59
Cards in React Kanban component.....	63
Drag-and-drop.....	64
Header.....	64
Content	68
Template	68
Selection.....	74
Data binding in React Kanban component	78
Local data	78
Remote data.....	82
Loading data via ajax.....	113

Swimlane in React Kanban component	117
Render swimlane row	117
Custom row text.....	121
Template	126
Sorting.....	131
Drag-and-drop.....	135
Create empty row	139
Calculate cards count.....	143
Enable frozen rows	147
Drag and drop in React Kanban component.....	151
Internal drag and drop.....	152
External drag and drop	160
Sort in React Kanban component	189
Index.....	189
DataSource Order	198
Custom	202
Change the direction.....	207
Dialog in React Kanban component.....	211
Default Dialog.....	211
Custom Fields.....	215
Dialog Template	229
Prevent Dialog.....	239
Persisting data in server.....	243
Tooltip in React Kanban component.....	251
Tooltip template	251
Validation in React Kanban component	257
Minimum card limit.....	257
Maximum card limit.....	257
Virtualization in React Kanban component	262
Virtual scrolling	262
Limitations for virtual scrolling	274
Localization in React Kanban component.....	274
Loading translations.....	274
Right to left (RTL)	280
Dimensions in React Kanban component	286

Auto height and width	286
Height and width in pixel	290
Height and width in percentage	294
Persistence in React Kanban component	298
Responsive mode in React Kanban component	303
Layouts	303
Scrolling	304
Selection	305
Style in React Kanban component	308
To set fixed position to the Kanban header	311
Accessibility in React Kanban component	311
WAI-ARIA attributes	312
Keyboard interaction	313
Ensuring accessibility	313
See also	313
How To	313
Header double click in React Kanban component	313
Dynamically change columns in React Kanban component	319
Filter cards in React Kanban component	324
Search cards in React Kanban component	330
Ej1 api migration in React Kanban component	337
Columns	337
Cards	339
DataSource	341
Common	342
Swimlane	344
Stacked Headers	345
WIP Validation	345
Keyboard	346
Toggle Columns	346
Dialog Editing	347
Dialog Editing Fields	349
Tooltip	350
Context Menu	350
WorkFlows	351

Filtering	351
Searching.....	352
External Drag And Drop	352
Scrolling.....	353
Card Selection and Hover.....	353
Toolbar	354
Responsive	354
State Persistence.....	354
Right to Left - RTL.....	354
Linear Gauge	355
Getting Started with React Linear Gauge	355
Dependencies.....	355
Installation and configuration.....	355
Module Injection.....	356
Adding the Linear Gauge Title.....	357
Axis Range	358
Setting the value of the pointer	360
Dimensions in React Linear gauge component.....	361
Size for Linear Gauge	361
Axis in React Linear gauge component.....	363
Setting the start value and end value of the axis.....	363
Line Customization.....	364
Ticks Customization	365
Labels Customization	367
Orientation.....	371
Inverted Axis	372
Opposed Axis.....	373
Multiple Axes	374
Ranges in React Linear gauge component.....	374
Customizing the range	375
Setting the range color for the labels	377
Multiple ranges	378
Gradient Color.....	379
Pointers in React Linear Gauge Component.....	382
Types of pointer	383

Multiple pointers	394
Pointer animation	395
Gradient Color.....	396
Annotations in React Linear gauge component.....	400
Adding annotation	400
Customization	401
Multiple annotations	404
Animation in React Linear Gauge component.....	405
User interaction in React Linear gauge component	408
Tooltip	408
Pointer Drag	413
Print and export in React Linear gauge component	414
Print.....	414
Export.....	415
Appearance in React Linear gauge component.....	418
Customizing the Linear Gauge area	418
Setting up the Linear Gauge title	419
Customizing the Linear Gauge container.....	420
Fitting the Linear Gauge to the control.....	423
Accessibility in React Linear Gauge component	424
WAI-ARIA attributes.....	425
Screen reading in Linear Gauge	425
Ensuring accessibility	425
See also	425
Internationalization in React Linear gauge component.....	426
Numeric Format	426
Events in React Linear gauge component.....	427
animationComplete	427
annotationRender	428
axisLabelRender	429
beforePrint	430
dragEnd	431
dragMove	432
dragStart	433
gaugeMouseDown	434

gaugeMouseLeave	435
gaugeMouseMove	435
gaugeMouseUp	436
load	437
loaded	438
resized	438
tooltipRender	439
valueChange	440
Methods in React Linear gauge component	441
setPointerValue	441
setAnnotationValue	443
refresh	444
Ej1 api migration in React Linear gauge component	446
Linear gauge dimensions	446
Line customization	446
Ticks	447
Labels	449
Axis	450
Ranges	451
Bar Pointer	452
Marker Pointer	454
Annotation	456
Tooltip	458
Appearance of Linear Gauge	459
Gauge Container type	459
Events	460
ListBox	462
Getting Started	462
Dependencies	462
Installation and configuration	463
Adding syncfusion packages	463
Adding CSS reference	463
Adding ListBox component	464
Binding data source	464
Run the application	465

Accessibility in React ListBox component	466
WAI-ARIA attributes	467
Keyboard interaction	467
Ensuring accessibility	468
See also	468
Data binding in React List box component	468
Local Data	469
Remote Data	471
Drag and drop in React List box component	472
Single listbox	472
Multiple listbox	473
Dual list box in React List box component	475
Icons and templates in React List box component	477
Icons	477
Templates	478
Selection in React List box component	479
Single selection	480
Multiple selection	481
Sorting and grouping in React List box component	482
Sorting	482
Grouping	483
Style and appearance in React List box component	484
Horizontal ListBox	484
How To	486
Add items in React List box component	486
Enable or disable items in React List box component	488
Enable scroller in React List box component	489
Form submit in React List box component	490
Select items in React List box component	491
ListView	492
Getting Started	492
Dependencies	492
Installation and Configuration	493
Adding Syncfusion packages	493
Adding ListView component	493

Bind dataSource	495
Running the application	496
See Also	497
Data binding in React Listview component	497
Bind to local data	498
Bind to remote data	500
Grouping in React Listview component	501
Customization	503
Check list in React Listview component.....	503
Checkbox Position	504
Nested list in React Listview component.....	505
Customizing templates in React Listview component	515
Header template	515
Template	517
Group template.....	522
Virtualization in React Listview component	526
Module injection	526
Getting started	526
Conditional rendering	531
Scrolling in React Listview component	532
Style in React Listview component	537
Customizing ListView	537
Customizing the list items	537
Customizing ListView's header.....	538
Customizing group header of ListView	538
Customizing the hover state of ListView control	538
Customizing selected item of ListView control.....	539
Accessibility in React ListView component.....	539
WAI-ARIA attributes.....	540
Keyboard interaction	540
Ensuring accessibility	541
See also	541
Ej1 api migration in React Listview component.....	541
How To	543
Get selected items from listview in React Listview component	543

Use dynamic templates in listview based on device in React Listview component	545
Create dual list from listview in React Listview component	550
Load list items in child list dynamically in React Listview component.....	563
Customize listview with dynamic tags in React Listview component.....	567
Show items count in group header in React Listview component	577
Filter list items in the listview in React Listview component.....	579
Add and remove list items from listview in React Listview component	581
Trace all events in listview in React Listview component.....	584
Create mobile contact layout from listview in React Listview component	588
Display spinner until list items are loaded in React Listview component	591
Hide checkbox in listview in React Listview component	593
Manipulate listview as grid layout in React Listview component.....	605
default-list .e-list-item {	605
Drag and drop list items in React Listview component	615
Load html content via ajax in React Listview component	618
Render listview with hyper link navigation in React Listview component	619
Customize listview as chat window in React Listview component	620
Integrate paging with listview in React Listview component	626
Maps	627
Getting Started.....	627
Dependencies.....	627
Installation and Configuration	627
Module Injection.....	629
Render shapes from GeoJSON data	630
Bind data source to map	631
Apply Color Mapping	632
Add Title for Maps.....	634
Enable Legend.....	635
Add Data Label	636
Enable Tooltip	638
Creating a Next.js Application Using Syncfusion React Components	639
What is Next.js?	639
Prerequisites	639
Create a Next.js application	639
Install Syncfusion React packages.....	640

Add Syncfusion React component	640
Run the application	640
Populate data in React Maps component	641
Geometry types.....	641
Shape data	641
Data source	641
Data binding.....	643
Binding complex data source	647
Layers in React Maps component.....	650
Multilayer	650
Sublayer	650
Displaying different layer in the view	652
Rendering custom shapes	653
Providers	653
Openstreetmap in React Maps component.....	653
Bing maps in React Maps component	661
Azure maps in React Maps component	668
Other maps in React Maps component	674
Customization in React Maps component.....	684
Setting the size for Maps	684
Maps title	685
Setting theme.....	687
Customizing Maps container	688
Customizing Maps area.....	689
Customizing the shapes	690
Setting color to the shapes from the data source	692
Applying border to individual shapes	693
Projection type.....	695
Color mapping in React Maps component	696
Types of color mapping.....	696
Multiple colors for a single shape	703
Color for items excluded from color mapping.....	704
Color mapping for bubbles	706
Data label in React Maps component.....	708
Adding data labels.....	708

Customization	710
Label animation.....	712
Smart labels.....	714
Intersect action	715
Adding data label as a template	716
Polygon shape in React Maps component.....	718
Adding polygon shape.....	718
Tooltip	723
Markers in React Maps component.....	733
Adding marker.....	733
Adding marker template	734
Customization	736
Marker shapes	738
Multiple marker groups	740
Customize marker shapes from data source	742
Repositioning the marker using drag and drop	745
Marker zooming	752
Marker clustering.....	753
Customization of marker cluster.....	755
Expanding the marker cluster	757
Tooltip for marker	759
Bubble in React Maps component.....	760
Bubble shapes	762
Customization	763
Setting colors to the bubbles from the data source	765
Setting the range of the bubble size	767
Multiple bubble groups.....	768
Enable tooltip for bubble	772
Annotations in React Maps component	774
Annotation	774
Annotation customization	775
Multiple Annotation.....	778
Legend in React Maps component	779
Modes of legend	780
Positioning of the legend	781

Legend for shapes	783
Enable Legend for bubbles.....	796
Enable legend for markers	798
Navigation line in React Maps component.....	801
Customization	801
Enabling the arrows	803
User interactions in React Maps component	804
Zooming	804
Selection.....	818
Highlight	832
Tooltip	843
Print in React Maps component	847
Print.....	847
Export.....	848
State persistence in React Maps component	854
State Persistence.....	854
Accessibility in React Maps component	855
WAI-ARIA attributes.....	856
Screen reading in Maps.....	856
Keyboard Navigation.....	856
Ensuring accessibility	857
See also	857
Internationalization in React Maps component	857
Globalization	857
Numeric Format	857
Localization in React Maps component	860
Methods in React Maps component.....	862
Methods	862
getMinMaxLatitudeLongitude	862
Ej1 api migration in React Maps component.....	866
Size Customization	866
Title and Subtitle Customization	866
Zooming Customization	866
Layer Customization.....	868
Shape Customization	870

Marker Customization	871
Bubble Customization	874
DataLabel Customization	876
Legend Customization.....	877
Highlight And Selection Customization.....	881
Navigation Line Customization	882
Tooltip Customization	884
Annotation Cutomization.....	885
Maps Other Properties Customization	886
Events.....	887
How To	890
Annotation in React Maps component.....	890
Custom path in React Maps component	893
Drilldown in React Maps component	894
Marker type in React Maps component	897
Multiple layer in React Maps component.....	901
Zooming in React Maps component	903
MaskedTextBox.....	904
Getting Started.....	904
Dependencies.....	904
Installation and configuration.....	904
Adding Syncfusion packages	905
Adding MaskedTextBox to the application	905
Set the mask.....	906
Adding CSS reference.....	906
Run the application	907
See Also	908
Mask configuration in React Maskedtextbox component.....	908
Standard mask elements.....	908
Custom mask elements.....	913
Prompt character	915
Accessibility in React Maskedtextbox component	916
WAI-ARIA attributes.....	917
Ensuring accessibility	918
See also	918

React functional component in React Maskedtextbox component	918
Style appearance in React Maskedtextbox component	921
Customizing the appearance of MaskedTextBox wrapper element.....	921
Customizing the MaskedTextBox element on hovering	921
How To	922
Perform custom validation using form validator in React Maskedtextbox component.....	922
Set cursor position while focus on the input textbox in React Maskedtextbox component	924
Display numeric keypad when focus on mobile devices in React Maskedtextbox component	926
Customize the ui appearance of the control in React Maskedtextbox component	927
Ej1 api migration in React Maskedtextbox component.....	928
Common.....	928
Mask Configuration.....	931
Validation	933
Mention.....	934
Getting Started.....	934
Dependencies.....	934
Setup your development environment	934
Adding syncfusion packages	935
Adding Style sheet to the Application.....	935
Adding Mention component.....	936
Binding data source	938
Run the application	940
Display Mention character.....	943
Working with data in React Mention component	945
Binding local data.....	946
Binding remote data	953
See Also	958
Filtering data in React Mention component.....	958
Limit the minimum filter character.....	958
Change the filter type	961
Allow spacing between search.....	964
Customize the suggestion item count	967
See Also	970
Sorting in React Mention component.....	970
Template in React Mention component.....	973

Item template	973
Display template	976
No records template	979
Spinner template	982
Localization in React Mention component	985
Loading translations	985
See Also	988
Customization in React Mention component	988
Show or hide mention character	988
Adding the suffix character after selection	991
Configure the popup list	994
Trigger character	996
Accessibility in React Mention component	997
WAI-ARIA attributes	998
Keyboard interaction	998
Ensuring accessibility	1001
See also	1001
Menu	1001
Getting Started	1001
Dependencies	1001
Setup your development environment	1001
Adding Syncfusion packages	1002
Adding Style sheet to the Application	1002
Add Menu to the project	1002
Run the application	1005
See Also	1007
Icons and sub menu items in React Menu component	1007
Icons	1007
Navigation	1009
Multilevel nesting	1012
See Also	1015
Data source binding and custom menu items in React Menu component	1015
Data binding	1015
Custom menu items	1023
See Also	1027

Use case scenarios in React Menu component	1027
Scrollable menu	1027
Menu in toolbar	1033
Hamburger menu.....	1036
Mobile view.....	1044
Accessibility in React Menu component.....	1049
WAI-ARIA attributes.....	1050
Keyboard interaction	1050
Ensuring accessibility	1050
See also	1051
Style and appearance in React Menu component.....	1051
How To	1051
Render with separator in React Menu component	1051
Change orientation in React Menu component	1053
Customize menu using events in React Menu component	1055
Customize menu items in React Menu component.....	1058
Create mnemonic ui in menuitem in React Menu component	1067
Change sub menu position in React Menu component.....	1069
Rounded corner in React Menu component	1071
Change animation settings in React Menu component.....	1073
Right to left in React Menu component	1075
Set title in React Menu component	1077
Menu item click in React Menu component.....	1079
Ej1 api migration in React Menu component	1082
Properties.....	1082
Methods.....	1084
Events.....	1085
Message	1087
Getting Started.....	1087
Prerequisites	1087
Dependencies.....	1087
Create the React application.....	1087
Add Syncfusion React packages	1088
Import the Syncfusion CSS styles	1088
Add Message component to the application.....	1089

Run the application	1089
Severities in React Message component	1091
Variants in React Message component	1093
Icons in React Message component	1096
No Icon	1096
Custom Icon	1098
Close Icon	1101
Customization in React Message component.....	1106
Content Alignment	1106
Rounded and Square.....	1107
CSS Message.....	1109
Template in React Message component	1113
Accessibility in React Message component	1116
WAI-ARIA attributes.....	1117
Keyboard interaction	1117
Ensuring accessibility	1117
See also	1117
MultiSelect	1117
Getting Started.....	1117
Dependencies.....	1117
Installation and configuration.....	1118
Adding syncfusion packages	1118
Adding MultiSelect component	1118
Adding CSS reference.....	1119
Binding data source	1120
Run the application	1121
Configure the Popup List.....	1122
See Also	1124
Data binding in React Multi select component	1124
Binding local data.....	1124
Binding remote data	1130
Value binding in ##Platform_Name## Multi select control	1132
Primitive Data Types	1132
Object Data Types	1134
Templates in React Multi select component	1135

Item template	1135
Value template.....	1138
Group template.....	1141
Header template	1144
Footer template	1148
No records template	1150
Action failure template	1152
See Also	1154
Grouping in React Multi select component.....	1154
Customization	1157
Grouping with CheckBox.....	1157
See Also	1160
Filtering in React Multi select component.....	1160
Limit the minimum filter character.....	1163
Change the filter type	1167
Case sensitive filtering	1170
Diacritics Filtering.....	1174
See Also	1176
Custom value in React Multi select component	1176
Virtualization in MultiSelect Dropdown	1178
Binding local data.....	1178
Binding remote data	1179
Customizing items count in virtualization.....	1181
Grouping with virtualization	1182
Filtering with virtualization	1184
Checkbox with virtualization.....	1186
Custom value with virtualization	1187
Preselect values with virtualization	1188
Checkbox in React Multi select component	1189
Select All.....	1192
Selection Limit.....	1195
Selection Reordering.....	1197
See Also	1199
Chip customization in React Multi select component	1200
Localization in React Multi select component.....	1202

Loading translations.....	1203
See Also	1206
Style in React Multi select component	1206
Customizing the background color of wrapper element	1206
Customizing the appearance of the delimiter wrapper element	1206
Customizing the appearance of chips	1207
Customizing the dropdown icon's color	1207
Customizing the focus color	1207
Customizing the disabled component's text color	1207
Customizing the color of the placeholder text	1208
Customizing the placeholder to add mandatory indicator(*).....	1208
Customizing the float label element's focusing color	1208
Customizing the outline theme's focus color	1208
Customizing the background color of focus, hover, and active item's	1209
Customizing the appearance of pop-up element	1209
Customizing the color of the checkbox.....	1209
Accessibility in React Multi select component	1210
WAI-ARIA attributes.....	1211
Keyboard interaction	1211
Ensuring accessibility	1215
See also	1215
How To	1215
Icons support in React Multi select component	1215
Cascading in React Multi select component	1216
NumericTextBox.....	1221
Getting Started.....	1221
Dependencies.....	1221
Installation and configuration.....	1221
Adding Syncfusion packages	1222
Adding NumericTextBox to the application	1222
Adding CSS reference.....	1223
Run the application	1223
Range validation.....	1224
Formatting the value.....	1225
Precision of numbers	1226

See Also	1228
Formats in React Numerictextbox component.....	1228
Standard formats	1229
Custom formats	1230
Globalization in React Numerictextbox component.....	1232
Localization	1232
Internationalization.....	1233
Right to Left(RTL)	1530
Accessibility in React Numerictextbox component	1532
WAI-ARIA attributes.....	1533
Keyboard interaction	1533
Ensuring accessibility	1534
See also	1535
React functional component in React Numerictextbox component	1535
Style appearance in React Numerictextbox component	1538
Customizing the appearance of NumericTextBox wrapper element.....	1538
Customizing the NumericTextBox icons	1539
How To	1539
Customize the ui appearance of the control in React Numerictextbox component	1539
Customize the spin buttons up and down arrow in React Numerictextbox component	1540
Customize the step value and hide spin buttons in React Numerictextbox component	1540
Maintain trailing zeros in numerictextbox in React Numerictextbox component	1541
Perform custom validation using form validator in React Numerictextbox component	1542
Prevent nullable input in numerictextbox in React Numerictextbox component.....	1545
Ej1 api migration in React Numerictextbox component.....	1546
Common.....	1546
Globalization	1548
Group	1548
Numeric configuration	1549
Number Formats	1550
Validation	1550
Pager	1551
Getting started.....	1551
Dependencies.....	1551
Setup for Local Development.....	1551

Adding Syncfusion packages	1552
Adding CSS reference.....	1552
Adding Pager component	1552
Page Size	1553
Page Count	1553
Run the application	1554
Style and appearance in React Pager component	1555
Customizing the Pager	1555
Accessibility in React Pager component	1556
WAI-ARIA.....	1556
Keyboard navigation	1556
PDF Viewer.....	1557
Getting Started.....	1557
Getting Started with Standalone PDF Viewer component	1557
Getting Started.....	1562
Getting Started with the React PDF Viewer Component in the Preact Framework.....	1568
Feature module in React Pdfviewer component	1572
See also	1573
Open PDF files.....	1573
Opening a PDF from URL.....	1573
Opening a PDF from base64 data	1576
Opening a PDF from database	1576
Saving PDF file.....	1579
Save PDF file to Server	1579
Download PDF file as a copy	1581
Save PDF file to Database	1582
Toolbar in React Pdfviewer component	1585
Show/Hide the default toolbar	1585
Show/Hide the default toolbaritem.....	1588
See also	1591
Navigation in React Pdfviewer component	1591
Toolbar page navigation option.....	1591
Bookmark navigation	1593
Thumbnail navigation	1595
Hyperlink navigation	1596

Table of content navigation	1597
See also	1599
Magnification in React Pdfviewer component	1600
See also	1602
Accessibility in Syncfusion React PDF Viewer components	1602
WAI-ARIA attributes	1603
Keyboard interaction	1603
Ensuring accessibility	1607
See also	1608
Text search in React Pdfviewer component	1608
See also	1609
Annotation	1610
Text markup annotation in React Pdfviewer component	1610
Shape annotation in React Pdfviewer component	1627
Stamp annotation in React Pdfviewer component	1634
Sticky notes annotation in React Pdfviewer component	1639
Measurement annotation in React Pdfviewer component	1644
Free text annotation in React Pdfviewer component	1652
Ink annotation in React Pdfviewer component	1660
Comments in React Pdfviewer component	1664
Handwritten signature in React PDF Viewer component	1668
Interaction mode in React Pdfviewer component	1674
Selection mode	1674
Panning Mode	1675
See also	1677
Form Designer	1677
Create programmatically in React Pdfviewer component	1677
Create with user interface interaction in React Pdfviewer component	1724
Print in React Pdfviewer component	1730
See also	1733
Download in React Pdfviewer component	1733
See also	1735
Globalization in React Pdfviewer component	1735
Accessibility in Syncfusion React PDF Viewer components	1752
WAI-ARIA attributes	1753

Keyboard interaction	1753
Ensuring accessibility	1758
See also	1758
How To	1758
Toolbar customization in React Pdfviewer component.....	1758
magnificationToolBarItems {.....	1762
magnificationToolBar {.....	1762
magnificationToolBar.e-toolbar .e-tbar-btn {	1762
topToolBar {	1763
topToolBar .e-pv-download-document-icon::before {	1763
topToolBar .e-pv-print-document-icon::before {.....	1764
Load document in React Pdfviewer component.....	1769
Unload document in React Pdfviewer component.....	1771
Lock annotation in React Pdfviewer component.....	1772
Add signature in React Pdfviewer component	1772
Extract text in React Pdfviewer component	1774
Import export annotation object in React Pdfviewer component	1775
Authorization token in React Pdfviewer component	1782
Delete annotation in React Pdfviewer component	1783
Open thumbnail in React Pdfviewer component	1783
Convert pixel to point in server side in React Pdfviewer component	1784
Retrieve id of the document in React Pdfviewer component	1784
Perform form field double click event in React Pdfviewer component.....	1784
Clear annotation in React Pdfviewer component.....	1785
Resolve "Unable to find an entry point named FPDFText_GetCharAngle" error	1786
How to clear the "Web-service is not listening" to error	1787
Retry Timeout	1789
Load N number of pages on initial loading	1790
Customize toolbar in PDF Viewer component.....	1791
Know the supported conformance PDF documents in React PDF Viewer component.....	1795
Organize Pages Feature in React Pdfviewer component.....	1796
Customize context menu in React Pdfviewer component.....	1797
Open and Close Bookmark pane programmatically	1807
Troubleshooting.....	1808
Why Do I Have to Manually Copy Files from node_modules into My App?.....	1808

Troubleshoot error 'cp' is not recognized as a command1808

Document Loading Issues in Version 23.1 or Newer1809

Kanban

Getting Started

This article provides a step-by-step introduction to get started with the Syncfusion React Kanban component.

Prerequisites

[System requirements for Syncfusion React UI components](#)

Dependencies

The following list of dependencies are required to use the **Kanban** component in the application.

```
`ts
|-- @syncfusion/ej2-react-kanban
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-kanban
|-- @syncfusion/ej2-layouts
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
`,`
```

Create the React application

To set-up, a React application, choose any of the following ways. The best and easiest way is to use the [Create-react-app](#). It sets up the development environment in JavaScript and improvises the application for production. Refer to the [installation instructions](#) of the **Create-react-app**

```
`bash
npx create-react-app my-app
cd my-app
npm start
`,`
```

or

```
`bash
yarn create react-app my-app
cd my-app
```

```
yarn start
```

```
,
```

To set-up a React application in the **TypeScript** environment, run the following command.

```
`bash
```

```
npx create-react-app my-app --template typescript
```

```
cd my-app
```

```
npm start
```

```
,
```

Besides using the [npx](#) runner tool, also create an application from the **npm init**. To begin with **npm init**, upgrade the **npm** version to **npm 6+**.

```
`bash
```

```
npm init react-app my-app
```

```
cd my-app
```

```
npm start
```

```
,
```

Adding Syncfusion packages

Once you have created the React application, install the required Syncfusion React component package in the application. All Syncfusion React (Essential JS 2) packages are published on the [npmjs](#) public registry.

To install the Kanban component package, use the following command.

```
`bash
```

```
npm install @syncfusion/ej2-react-kanban --save
```

```
,
```

or

```
`bash
```

```
yarn add @syncfusion/ej2-react-kanban
```

```
,
```

Import the Syncfusion CSS styles

After installing the Syncfusion component packages in the application, import the required themes based on the components used.

The Syncfusion React component comes with built-in [themes](#), which are available in installed packages. It is quite simple to adapt the Syncfusion React components based on the application style by referring to any of the built-in themes. Import the **Material** theme for the Kanban component.

Import the CSS styles for the Kanban component and its dependencies in the **src/App.css** file.

```
`css
```

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-layouts/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-kanban/styles/material.css";
`
```

Check out the [Themes topic](#) to know more about built-in themes and different ways to refer to themes in React applications.

Add Kanban component to the application

Start adding the required components to the application. Add the Kanban component in the `src/App.js` or `src/App.tsx` file using the following code.

- Before adding the Kanban component to the markup, import the Kanban component in the `src/App.js` or `src/App.tsx` file.

```
`bash
```

```
import { KanbanComponent } from '@syncfusion/ej2-react-kanban';
```

- Then, add the Kanban component in the application using the following code sample.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
function App() {
    return (<KanbanComponent id="kanban" keyField="Status">
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('kanban'));
```

INDEX.TSX


```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
function App() {
  return (
    <KanbanComponent id="kanban" keyField="Status">
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('kanban'));
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Columns Header</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Columns Header" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
```

```

        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Run the application

All are set. Now, run the application using the following command.

```
`bash
```

```
npm start
```

```
,
```

or

```
`bash
```

```
yarn start
```

```
,
```

The output will display the kanban header.

Populating cards

To populate the empty Kanban with cards, define the local JSON data or remote data using the `dataSource` property. To define `dataSource`, the mandatory fields in JSON object should be relevant to `keyField`. In the following example, you can see the cards defined with default fields such as ID, Summary, and Status.

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    const data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>

```

```

    </KanbanComponent>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    const data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Cards</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Cards" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Enable swimlane

Swimlane can be enabled by mapping the fields `swimlaneSettings.keyField` to appropriate column name in `dataSource`. This enables the grouping of the cards based on the mapped column values.

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    const data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
export default App;

```

```
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
  const data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban with Swimlane Rows</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban swimlane rows" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
```

```
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></script>
<script src="systemjs.config.js"></script>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>
```

Creating a Next.js Application Using Syncfusion React Components

This section provides a step-by-step guide for setting up a Next.js application and integrating the Syncfusion React Kanban component.

What is Next.js?

[Next.js](#) is a React framework that makes it easy to build fast, SEO-friendly, and user-friendly web applications. It provides features such as server-side rendering, automatic code splitting, routing, and API routes, making it an excellent choice for building modern web applications.

Prerequisites

Before getting started with the Next.js application, ensure the following prerequisites are met:

- [Node.js 18.17](#) or later.
- The application is compatible with macOS, Windows, and Linux operating systems.

Create a Next.js application

To create a new **Next.js** application, use one of the commands that are specific to either NPM or Yarn.

NPM

```
npx create-next-app@latest
```

YARN

```
yarn create next-app
```

Using one of the above commands will lead you to set up additional configurations for the project as below:

1. Define the project name: Users can specify the name of the project directly. Let's specify the name of the project as `ej2-nextjs-kanban`.

CMD

```
✓ What is your project named? » ej2-nextjs-kanban
```

2. Select the required packages.

CMD

```
✓ What is your project named? ... ej2-nextjs-kanban
✓ Would you like to use TypeScript? ... No / `Yes`
✓ Would you like to use ESLint? ... No / `Yes`
✓ Would you like to use Tailwind CSS? ... `No` / Yes
✓ Would you like to use `src/` directory? ... No / `Yes`
✓ Would you like to use App Router? (recommended) ... No / `Yes`
✓ Would you like to customize the default import alias? ... `No` / Yes
Creating a new Next.js app in D:\ej2-nextjs-kanban.
```

3. Once complete the above mentioned steps to create `ej2-nextjs-kanban`, navigate to the directory using the below command:

CMD

```
cd ej2-nextjs-kanban
```

The application is ready to run with default settings. Now, let's add Syncfusion components to the project.

Install Syncfusion React packages

Syncfusion React component packages are available at [npmjs.com](https://www.npmjs.com). To use Syncfusion React components in the project, install the corresponding npm package.

Here, the [React Kanban component](#) is used as an example. To install the React kanban component in the project, use the following command:

NPM

```
npm install @syncfusion/ej2-react-kanban --save
```

YARN

```
yarn add @syncfusion/ej2-react-kanban
```

Import Syncfusion CSS styles

Syncfusion React components come with [built-in themes](#), which are available in the installed packages. It's easy to adapt the Syncfusion React components to match the style of your application by referring to one of the built-in themes.

Import the **Material** theme into the **src/app/globals.css** file and removed the existing styles in that file, as shown below:

GLOBALS.CSS

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-layouts/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-kanban/styles/material.css";
```

To know more about built-in themes and CSS reference for individual components, refer to the [themes](#) section.

Add Syncfusion React component

Follow the below steps to add the React Kanban component to the Next.js project:

1. Define the kanban component in the **src/app/page.tsx** file, as shown below:

PAGE.TSX

```
'use client'
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from '@syncfusion/ej2-react-kanban';
function App(this: any) {
  let data = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-services/api/Kanban',
    adaptor: new ODataAdaptor
  });
  function DialogOpen(args: { cancel: boolean; }) {
    args.cancel = true;
  }
  return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ contentField: "Summary", headerField: "Id" }}
    allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
    <ColumnsDirective>
    <ColumnDirective headerText="To Do" keyField="Open"/>
    <ColumnDirective headerText="In Progress" keyField="InProgress"/>
    <ColumnDirective headerText="Testing" keyField="Testing"/>
    <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
  </KanbanComponent>);
}
export default App;
```

Run the application

To run the application, use the following command:

NPM

```
npm run dev
```


YARN

```
yarn run dev
```

To learn more about the functionality of the Kanban component, refer to the [documentation](#).

[View the NEXT.js Kanban sample in the GitHub repository.](#)

Columns in React Kanban component

The **Kanban** columns represent the each stage of the process. The column definitions are used as the **dataSource** schema in the Kanban. The Kanban operations such as drag-and-drop, swimlane, and toggle columns are performed based on column definitions.

Single-key mapping

Kanban columns are categorized by mapping the **key** from the datasource using the **keyField** property. The corresponding **value** in the datasource is mapped inside the columns **keyField**. Based on this categorization, Kanban columns are split on this board.

The **keyField** property is mandatory to render the columns in the Kanban board.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Single Key Columns</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban single key columns" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}

```

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Single Key Columns</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban single key columns" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>

```

```

<style>
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
</style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Multi-key mapping

Kanban board allows to render a single column by mapping multiple keys using `keyField` property. In below sample, specified the multiple keys(Open, Validate) to a single column.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open,
Validate"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open,
Validate" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Multiple Key Columns</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban columns with multiple keys" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open, Validate"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App(){
  let data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ contentField: "Summary", headerField: "Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open, Validate" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Multiple Key Columns</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban columns with multiple keys" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
</style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Header text

You can provide the column header text of Kanban columns using the `headerText` property. If you have not specified any header text, it will render the header without any text.

Header template

You can customize the column header with any HTML or CSS element using the `template` property as shown in the following code.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  columnTemplate(props) {
    return (<div className="header-template-wrap">
      <div className={"header-icon e-icons " +
props.keyField}></div>
      <div className="header-text">{props.headerText}</div>
    </div>);
  }
  render() {

```

```

        return <KanbanComponent id="kanban" keyField="Status"
cssClass="kanban-header" dataSource={this.data} cardSettings={{
contentField: "Summary", headerField: "Id" }}>
    <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"
template={this.columnTemplate.bind(this)} />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" template={this.columnTemplate.bind(this)} />
        <ColumnDirective headerText="Review" keyField="Review"
template={this.columnTemplate.bind(this)} />
        <ColumnDirective headerText="Done" keyField="Close"
template={this.columnTemplate.bind(this)} />
    </ColumnsDirective>
</KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    private columnTemplate(props: { [key: string]: string }): JSX.Element {
        return (
            <div className="header-template-wrap">
                <div className={"header-icon e-icons " +
props.keyField}></div>
                <div className="header-text">{props.headerText}</div>
            </div>
        );
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status" cssClass="kanban-
header" dataSource={this.data} cardSettings={{ contentField: "Summary",
headerField: "Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"
template={this.columnTemplate.bind(this)} />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" template={this.columnTemplate.bind(this)} />
                <ColumnDirective headerText="Review" keyField="Review"
template={this.columnTemplate.bind(this)} />
                <ColumnDirective headerText="Done" keyField="Close"
template={this.columnTemplate.bind(this)} />
            </ColumnsDirective>
        </KanbanComponent>
    }
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

```

        </ColumnsDirective>
      </KanbanComponent>
    }
  };
  ReactDOM.render(<App />, document.getElementById('kanban'));
  {% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Column Header Template</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban column header template" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link href="index.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>

```

```
</body>
</html>
```

[Functional-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function columnTemplate(props) {
        return (<div className="header-template-wrap">
            <div className={"header-icon e-icons " +
props.keyField}></div>
            <div className="header-text">{props.headerText}</div>
        </div>);
    }
    return (<KanbanComponent id="kanban" keyField="Status" cssClass="kanban-
header" dataSource={data} cardSettings={{ contentField: "Summary",
headerField: "Id" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"
template={columnTemplate.bind(this)}>
            <ColumnDirective headerText="In Progress" keyField="InProgress"
template={columnTemplate.bind(this)}>
            <ColumnDirective headerText="Review" keyField="Review"
template={columnTemplate.bind(this)}>
            <ColumnDirective headerText="Done" keyField="Close"
template={columnTemplate.bind(this)}>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function columnTemplate(props: { [key: string]: string }) {
        return (
            <div className="header-template-wrap">
```

```

        <div className={"header-icon e-icons " +
props.keyField}></div>
        <div className="header-text">{props.headerText}</div>
    </div>
    );
}
return (
    <KanbanComponent id="kanban" keyField="Status" cssClass="kanban-
header" dataSource={data} cardSettings={{ contentField: "Summary",
headerField: "Id" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"
template={columnTemplate.bind(this)} />
            <ColumnDirective headerText="In Progress" keyField="InProgress"
template={columnTemplate.bind(this)} />
            <ColumnDirective headerText="Review" keyField="Review"
template={columnTemplate.bind(this)} />
            <ColumnDirective headerText="Done" keyField="Close"
template={columnTemplate.bind(this)} />
        </ColumnsDirective>
    </KanbanComponent>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Column Header Template</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban column header template" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link href="index.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Toggle columns

Kanban allows to expand or collapse its columns using `allowToggle` inside the `columns` property. When enable the property, it will render the expand or collapse icon to the column header.

By default, collapsed column width is set to `50px`.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true}/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress" allowToggle={true}/>
        <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true}/>

```

```

        <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true}/>
        </ColumnsDirective>
    </KanbanComponent>;
}
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true} />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" allowToggle={true} />
                <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true} />
                <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true} />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Toggle Column</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Toggle Column" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
        <ColumnsDirective>

```



```

        <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true}/>
        <ColumnDirective headerText="In Progress" keyField="InProgress"
allowToggle={true}/>
        <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true}/>
        <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true}/>
    </ColumnsDirective>
</KanbanComponent>;
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true} />
                <ColumnDirective headerText="In Progress" keyField="InProgress"
allowToggle={true} />
                <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true} />
                <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true} />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Toggle Column</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Toggle Column" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Initially collapsed column

By default, all columns are on expanded state when loading the Kanban board initially. But, you can render the columns with collapsed state using the `isExpanded` property.

The `isExpanded` property only works when enabling the `allowToggle` property on particular column.

In the following example, the backlog column is collapsed on initialization of Kanban board.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';

```

```

import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true} isExpanded={false}/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress" allowToggle={true}/>
        <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true} isExpanded={false}/>
        <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true}/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true} isExpanded={false} />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" allowToggle={true} />
        <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true} isExpanded={false} />
        <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true} />

```

```

        </ColumnsDirective>
      </KanbanComponent>
    }
  };
ReactDOM.render(<App />, document.getElementById('kanban'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Rendering Column with Collapsed State</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban rendering column with collapsed
state" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>

```

```
</body>
</html>
```

[Functional-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true} isExpanded={false}/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"
allowToggle={true}/>
            <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true} isExpanded={false}/>
            <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true}/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true} isExpanded={false} />
                <ColumnDirective headerText="In Progress" keyField="InProgress"
allowToggle={true} />
                <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true} isExpanded={false} />
                <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true} />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
```

```

    </ColumnsDirective>
  </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Rendering Column with Collapsed State</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban rendering column with collapsed
state" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>

```

```
</body>
</html>
```

Stacked headers

Stacked headers are the additional headers to column header that will group the similar columns.

Define the grouping of columns **key** value to the **keyFields** property and provide the custom header text name to grouped columns using the **text** property, which is placed inside the **stackedHeaders** property.

In the following code, the kanban columns 'InProgress, Review' are grouped under 'Development Phase' category.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
StackedHeadersDirective, StackedHeaderDirective } from "@syncfusion/ej2-
react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="Open" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="In Review"
keyField="Review"/>
        <ColumnDirective headerText="Completed"
keyField="Close"/>
      </ColumnsDirective>
      <StackedHeadersDirective>
        <StackedHeaderDirective text='To Do '
keyFields='Open'></StackedHeaderDirective>
        <StackedHeaderDirective text='Development Phase '
keyFields='InProgress, Review'></StackedHeaderDirective>
        <StackedHeaderDirective text='Done '
keyFields='Close'></StackedHeaderDirective>
      </StackedHeadersDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
```

```
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
StackedHeadersDirective, StackedHeaderDirective } from "@syncfusion/ej2-
react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="Open" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="In Review"
keyField="Review" />
        <ColumnDirective headerText="Completed" keyField="Close"
/>
      </ColumnsDirective>
      <StackedHeadersDirective>
        <StackedHeaderDirective text='To Do'
keyFields='Open'></StackedHeaderDirective>
        <StackedHeaderDirective text='Development Phase'
keyFields='InProgress, Review'></StackedHeaderDirective>
        <StackedHeaderDirective text='Done'
keyFields='Close'></StackedHeaderDirective>
      </StackedHeadersDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Stacked Headers</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban stacked headers" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
```



```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
StackedHeadersDirective, StackedHeaderDirective } from "@syncfusion/ej2-
react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
        <ColumnsDirective>

```

```

        <ColumnDirective headerText="Open" keyField="Open"/>
        <ColumnDirective headerText="In Progress" keyField="InProgress"/>
        <ColumnDirective headerText="In Review" keyField="Review"/>
        <ColumnDirective headerText="Completed" keyField="Close"/>
    </ColumnsDirective>
    <StackedHeadersDirective>
        <StackedHeaderDirective text='To Do '
keyFields='Open'></StackedHeaderDirective>
        <StackedHeaderDirective text='Development Phase'
keyFields='InProgress, Review'></StackedHeaderDirective>
        <StackedHeaderDirective text='Done '
keyFields='Close'></StackedHeaderDirective>
    </StackedHeadersDirective>
</KanbanComponent>;
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
StackedHeadersDirective, StackedHeaderDirective } from "@syncfusion/ej2-
react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="Open" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="In Review" keyField="Review" />
                <ColumnDirective headerText="Completed" keyField="Close" />
            </ColumnsDirective>
            <StackedHeadersDirective>
                <StackedHeaderDirective text='To Do '
keyFields='Open'></StackedHeaderDirective>
                <StackedHeaderDirective text='Development Phase'
keyFields='InProgress, Review'></StackedHeaderDirective>
                <StackedHeaderDirective text='Done '
keyFields='Close'></StackedHeaderDirective>
            </StackedHeadersDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```
<!DOCTYPE html>
```

```

<html lang="en">
<head>
  <title>Kanban Stacked Headers</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban stacked headers" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Cards in React Kanban component

The cards are main elements in Kanban board, which represent the task information with header and content. The header and content of a card is fetched from the corresponding mapping fields. The card layout can be customized with template also.

Drag-and-drop

Transit or change the card position using the drag-and-drop functionality. By default, the `allowDragAndDrop` property is enabled on the Kanban board, which is used to change the card position by column-to-column or within the column.

Added dotted border on Kanban cells except the dragged clone cells when dragging, which indicates the possible ways for dropping the cards into the cells.

Header

The card header is achieved by mapping the `headerField` property, which is placed inside the `cardSettings` property. By default, the `showHeader` property enabled by Kanban board that is used to show the header at the top of the card.

The `headerField` property of `cardSettings` is mandatory to render the cards in the Kanban board. It acts as a unique field that is used to avoid the duplication of card data. You can not change the `headerField` of mapped data value using the `updateCard` public method or server-side update of data.

In the following demo, the `showHeader` property is disabled on Kanban board.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id", showHeader: false }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}>, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id", showHeader: false }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Card without Header</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban card without header" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
```

```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id", showHeader:
false }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';

```

```
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id", showHeader:
false }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Card without Header</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban card without header" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
```

```

<style>
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
</style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Content

The card's content is fetched from data source using the `contentField` property, which is placed inside the `cardSettings` property. If the `contentField` property is not used, card is rendered with empty content.

Template

You can customize the default card layout using template as per your application needs. This can be achieved by template of the `cardSettings` property.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  cardTemplate(props) {
    return (<div className="card-template">
      <div className='e-card-content'>
        <table className="card-template-wrap">
          <tbody>
            <tr>
              <td className="CardHeader">Id:</td>
              <td>{props.Id}</td>
            </tr>
            <tr>
              <td className="CardHeader">Type:</td>
              <td>{props.Type}</td>
            </tr>
          </tbody>
        </table>
      </div>
    </div>);
  }
}
ReactDOM.render(<App data={kanbanData}/>, document.getElementById('kanban'));

```



```

        </tr>
        <tr>
            <td
className="CardHeader">Priority:</td>
            <td>{props.Priority}</td>
        </tr>
        <tr>
            <td className="CardHeader">Summary:</td>
            <td>{props.Summary}</td>
        </tr>
    </tbody>
</table>
</div>
</div>);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ headerField: "Id", template:
this.cardTemplate.bind(this) }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    private cardTemplate(props): JSX.Element {
        return (<div className="card-template">
            <div className='e-card-content'>
                <table className="card-template-wrap">
                    <tbody>
                        <tr>
                            <td className="CardHeader">Id:</td>
                            <td>{props.Id}</td>
                        </tr>
                        <tr>

```

```

                <td className="CardHeader">Type:</td>
                <td>{props.Type}</td>
            </tr>
            <tr>
                <td
className="CardHeader">Priority:</td>
                <td>{props.Priority}</td>
            </tr>
            <tr>
                <td className="CardHeader">Summary:</td>
                <td>{props.Summary}</td>
            </tr>
        </tbody>
    </table>
</div>
</div>
    );
}
render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ headerField: "Id", template:
this.cardTemplate.bind(this) }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" />
            <ColumnDirective headerText="In Progress"
keyField="InProgress" />
            <ColumnDirective headerText="Done" keyField="Close" />
        </ColumnsDirective>
    </KanbanComponent>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Card Template</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban card template" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link href="index.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function cardTemplate(props) {
        return (<div className="card-template">
            <div className='e-card-content'>
                <table className="card-template-wrap">
                    <tbody>
                        <tr>
                            <td className="CardHeader">Id:</td>
                            <td>{props.Id}</td>
                        </tr>
                        <tr>
                            <td className="CardHeader">Type:</td>
                            <td>{props.Type}</td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>);
    }
}

```

```

        <tr>
            <td className="CardHeader">Priority:</td>
            <td>{props.Priority}</td>
        </tr>
        <tr>
            <td className="CardHeader">Summary:</td>
            <td>{props.Summary}</td>
        </tr>
    </tbody>
</table>
</div>
</div>);
}
return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ headerField: "Id", template: cardTemplate.bind(this) }}>
    <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress" keyField="InProgress"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function cardTemplate(props) {
        return (
            <div className="card-template">
                <div className="e-card-content">
                    <table className="card-template-wrap">
                        <tbody>
                            <tr>
                                <td className="CardHeader">Id:</td>
                                <td>{props.Id}</td>
                            </tr>
                            <tr>
                                <td className="CardHeader">Type:</td>
                                <td>{props.Type}</td>
                            </tr>
                            <tr>
                                <td className="CardHeader">Priority:</td>
                                <td>{props.Priority}</td>
                            </tr>
                            <tr>
                                <td className="CardHeader">Summary:</td>

```

```

                <td>{props.Summary}</td>
            </tr>
        </tbody>
    </table>
</div>
</div>
);
}
return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ headerField: "Id", template: cardTemplate.bind(this) }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" />
            <ColumnDirective headerText="In Progress" keyField="InProgress" />
            <ColumnDirective headerText="Done" keyField="Close" />
        </ColumnsDirective>
    </KanbanComponent>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Card Template</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban card template" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link href="index.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>

```

```

<style>
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
</style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Selection

Kanban board allows to select single and multiple selection of cards when mouse or keyboard interactions using `selectionType` property. The property contains following types.

- **None:** No cards are allowed to select from Kanban board.
- **Single:** Only one card allowed to select at a time in the Kanban board.
- **Multiple:** Multiple cards are allowed to select in a board.

Multiple Selection

Select the multiple cards randomly using Ctrl + mouse click and select the multiple cards continuously using Shift + mouse click action on Kanban board. Set `Multiple` in `selectionType` to enable the multiple selection in a board.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
    dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
    "Id", selectionType: 'Multiple' }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>

```

```

        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id", selectionType: 'Multiple' }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Multiple Cards Selection</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban multiple cards selection" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id", selectionType:
'Multiple' }}>
        <ColumnsDirective>

```



```

        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress" keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id", selectionType:
'Multiple' }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Multiple Cards Selection</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban multiple cards selection" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008c8f;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Data binding in React Kanban component

The Kanban uses **DataManager**, which supports both RESTful data service binding and JavaScript object array binding. The **dataSource** property of Kanban can be assigned either with the instance of **DataManager** or JavaScript object array collection, as it supports the following two data binding methods:

- Local data
- Remote data

Local data

To bind local JSON data to the Kanban, you can simply assign a JavaScript object array to the **dataSource** property. The JSON object data can also be provided as an instance of **DataManager** and assigned to the Kanban **dataSource** property.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';

```

```

import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
}

```

```
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Local Data</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Local Data" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

[Functional-component]**INDEX.JSX**

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

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

```

<head>
  <title>Kanban Local Data</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Local Data" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

By default, **DataManager** uses **JsonAdaptor** for binding local data.

Remote data

To bind remote data to kanban component, assign service data as an instance of **DataManager** to the **dataSource** property. To interact with remote data source, provide the endpoint **url**.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component {
  data = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
    adaptor: new ODataAdaptor
  });
  DialogOpen(args) {
    args.cancel = true;
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} allowDragAndDrop={false} dialogOpen={this.DialogOpen.bind(this)}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs
} from "@syncfusion/ej2-react-kanban";
class App extends React.Component<{}, {}>{
  public data = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
    adaptor: new ODataAdaptor
  });
  private DialogOpen(args: DialogEventArgs): void {
    args.cancel = true;
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} allowDragAndDrop={false} dialogOpen={this.DialogOpen.bind(this)}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />

```

```

        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>
</KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Remote Data</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Remote Data" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>

```



```
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

[Functional-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
function App() {
  let data = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
    adaptor: new ODataAdaptor
  });
  function DialogOpen(args) {
    args.cancel = true;
  }
  return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
    <ColumnsDirective>
      <ColumnDirective headerText="To Do" keyField="Open"/>
      <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
      <ColumnDirective headerText="Testing" keyField="Testing"/>
      <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
  </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs
} from "@syncfusion/ej2-react-kanban";
function App() {
  let data = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
    adaptor: new ODataAdaptor
  });
  function DialogOpen(args: DialogEventArgs): void {

```

```

        args.cancel = true;
    }
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
        cardSettings={{ contentField: "Summary", headerField: "Id" }}
        allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
                keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Remote Data</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Remote Data" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008c8f;
            height: 40px;

```

```

        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

By default, [DataManager](#) uses **ODataAdaptor** for remote data-binding.

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.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component {
    data = new DataManager({
        url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
        adaptor: new ODataAdaptor,
        crossDomain: true
    });
    DialogOpen(args) {
        args.cancel = true;
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} allowDragAndDrop={false} dialogOpen={this.DialogOpen.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;

```

```
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs } from "@syncfusion/ej2-react-kanban";
class App extends React.Component<{}>, {}>{
    public data = new DataManager({
        url: 'https://ej2services.syncfusion.com/production/web-services/api/Kanban',
        adaptor: new ODataAdaptor,
        crossDomain: true
    });
    private DialogOpen(args: DialogEventArgs): void {
        args.cancel = true;
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} cardSettings={{ contentField: "Summary", headerField: "Id" }} allowDragAndDrop={false} dialogOpen={this.DialogOpen.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
                keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
                />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban OData</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban OData" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-navigations/styles/material.css" rel="stylesheet" />
```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
function App() {
    let data = new DataManager({
        url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
        adaptor: new ODataAdaptor,
        crossDomain: true
    });
    function DialogOpen(args) {
        args.cancel = true;
    }
}

```

```

    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
    <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs } from '@syncfusion/ej2-react-kanban';
function App() {
    let data = new DataManager({
        url: 'https://ej2services.syncfusion.com/production/web-services/api/Kanban',
        adaptor: new ODataAdaptor,
        crossDomain: true
    });
    function DialogOpen(args: DialogEventArgs): void {
        args.cancel = true;
    }
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
            </ColumnsDirective>
            <ColumnDirective headerText="Testing" keyField="Testing" />
            <ColumnDirective headerText="Done" keyField="Close" />
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban OData</title>

```

```

<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Kanban OData" />
<meta name="author" content="Syncfusion" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

OData v4 services

The ODataV4 is an improved version of OData protocols, and the [DataManager](#) can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [odata documentation](#). To bind OData v4 service, use the **ODataV4Adaptor**.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';

```

```

import * as ReactDOM from 'react-dom';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component {
    data = new DataManager({
        url:
'https://services.odata.org/v4/northwind/northwind.svc/Suppliers',
        adaptor: new ODataV4Adaptor
    });
    DialogOpen(args) {
        args.cancel = true;
    }
    render() {
        return <KanbanComponent id="kanban" keyField="ContactTitle"
dataSource={this.data} cardSettings={{ contentField: "ContactName",
headerField: "SupplierID" }} allowDragAndDrop={false}
dialogOpen={this.DialogOpen.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="Order Administrator"
keyField="Order Administrator"/>
                <ColumnDirective headerText="Sales Representative"
keyField="Sales Representative"/>
                <ColumnDirective headerText="Export Administrator"
keyField="Export Administrator"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs
} from "@syncfusion/ej2-react-kanban";
class App extends React.Component<{}, {}>{
    public data = new DataManager({
        url:
'https://services.odata.org/v4/northwind/northwind.svc/Suppliers',
        adaptor: new ODataV4Adaptor
    });
    private DialogOpen(args: DialogEventArgs): void {
        args.cancel = true;
    }
    render() {
        return <KanbanComponent id="kanban" keyField="ContactTitle"
dataSource={this.data} cardSettings={{ contentField: "ContactName",
headerField: "SupplierID" }} allowDragAndDrop={false}
dialogOpen={this.DialogOpen.bind(this)}>
            <ColumnsDirective>

```



```

        <ColumnDirective headerText="Order Administrator"
keyField="Order Administrator" />
        <ColumnDirective headerText="Sales Representative"
keyField="Sales Representative" />
        <ColumnDirective headerText="Export Administrator"
keyField="Export Administrator" />
    </ColumnsDirective>
</KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban ODataV4</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban ODataV4" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>

```

```

</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from '@syncfusion/ej2-react-kanban';
function App() {
  let data = new DataManager({
    url:
'https://services.odata.org/v4/northwind/northwind.svc/Suppliers',
    adaptor: new ODataV4Adaptor
  });
  function DialogOpen(args) {
    args.cancel = true;
  }
  return (<KanbanComponent id="kanban" keyField="ContactTitle"
dataSource={data} cardSettings={{ contentField: "ContactName", headerField:
"SupplierID" }} allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
    <ColumnsDirective>
      <ColumnDirective headerText="Order Administrator"
keyField="Order Administrator"/>
      <ColumnDirective headerText="Sales Representative"
keyField="Sales Representative"/>
      <ColumnDirective headerText="Export Administrator"
keyField="Export Administrator"/>
    </ColumnsDirective>
  </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs } from '@syncfusion/ej2-react-kanban';
function App() {
  let data = new DataManager({
    url:
'https://services.odata.org/v4/northwind/northwind.svc/Suppliers',
    adaptor: new ODataV4Adaptor

```

```

});
function DialogOpen(args: DialogEventArgs): void {
    args.cancel = true;
}
return (
    <KanbanComponent id="kanban" keyField="ContactTitle" dataSource={data}
cardSettings={{ contentField: "ContactName", headerField: "SupplierID" }}
allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
        <ColumnsDirective>
            <ColumnDirective headerText="Order Administrator"
keyField="Order Administrator" />
            <ColumnDirective headerText="Sales Representative"
keyField="Sales Representative" />
            <ColumnDirective headerText="Export Administrator"
keyField="Export Administrator" />
        </ColumnsDirective>
    </KanbanComponent>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban ODataV4</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban ODataV4" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>

```

```

        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Web API

You can use **WebApiAdaptor** to bind kanban with Web API created using OData endpoint.

[Class-component]

```

{% raw %}
`ts
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component<{}, {}>{
    public data = new DataManager({
        url: '/api/Tasks',
        adaptor: new WebApiAdaptor
    });
    private DialogOpen(args: DialogEventArgs): void {
        args.cancel = true;
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status" dataSource={this.data} cardSettings={{
            contentField: "Summary", headerField: "Id" }} allowDragAndDrop={false}
            dialogOpen={this.DialogOpen.bind(this)}>
            <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" />

```

```

<ColumnDirective headerText="In Progress" keyField="InProgress" />
<ColumnDirective headerText="Testing" keyField="Testing" />
<ColumnDirective headerText="Done" keyField="Close" />
</ColumnsDirective>
</KanbanComponent>
}
};

```

```
ReactDOM.render(<App />, document.getElementById('kanban'));
```

```
,
```

```
{% endraw %}
```

```
[Functional-component]
```

```
{% raw %}
```

```
`ts
```

```
import * as React from 'react';
```

```
import * as ReactDOM from 'react-dom';
```

```
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
```

```
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs } from
"@syncfusion/ej2-react-kanban";
```

```
function App(){
```

```
let data = new DataManager({
```

```
url: '/api/Tasks',
```

```
adaptor: new WebApiAdaptor
```

```
});
```

```
function DialogOpen(args: DialogEventArgs): void {
```

```
args.cancel = true;
```

```
}
```

```
return(
```

```
<KanbanComponent id="kanban" keyField="Status" dataSource={data} cardSettings={{ contentField:
"Summary", headerField: "Id" }} allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
```

```
<ColumnsDirective>
```

```
<ColumnDirective headerText="To Do" keyField="Open" />
```

```
<ColumnDirective headerText="In Progress" keyField="InProgress" />
```

```
<ColumnDirective headerText="Testing" keyField="Testing" />
```

```
<ColumnDirective headerText="Done" keyField="Close" />
```

```

</ColumnsDirective>
</KanbanComponent>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));

```

```
{% endraw %}
```

Below server-side controller code to get the Kanban data.

```

`ts
[HttpGet]
public object Get()
{
var data = OrdersDetails.GetAllRecords().ToList();
return data;
}

```

URL adaptor

The CRUD (Create, Read, Update and Delete) actions can be performed easily on Kanban cards using the various adaptors available within the **DataManager**. Most preferably, we will be using **UrlAdaptor** for performing CRUD actions on Kanban.

The CRUD operation in Kanban can be mapped to server-side controller actions using the properties **insertUrl**, **removeUrl**, **updateUrl**, and **crudUrl**.

- **insertUrl** – You can perform a single insertion operation on the server-side.
- **updateUrl** – You can update single data on the server-side.
- **removeUrl** – You can remove single data on the server-side.
- **crudUrl** – You can perform bulk data operation on the server-side.

[Class-component]

```

{% raw %}
`ts
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from "@syncfusion/ej2-react-kanban";

```

```

class App extends React.Component<{}, {}>{
  public data = new DataManager({
    url: 'Home/DataSource',
    updateUrl: 'Home/Update',
    insertUrl: 'Home/Insert',
    removeUrl: 'Home/Delete',
    adaptor: new UrlAdaptor(),
    crossDomain: true
  });
  render() {
    return <KanbanComponent id="kanban" keyField="Status" dataSource={this.data} cardSettings={{
      contentField: "Summary", headerField: "Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));

```

```
{% endraw %}
```

[Functional-component]

```
{% raw %}
```

```
`ts
```

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from "@syncfusion/ej2-react-kanban";

function App(){
  let data = new DataManager({

```

```

url: 'Home/DataSource',
updateUrl: 'Home/Update',
insertUrl: 'Home/Insert',
removeUrl: 'Home/Delete',
adaptor: new UrlAdaptor(),
crossDomain: true
});
return(
<KanbanComponent id="kanban" keyField="Status" dataSource={data} cardSettings={{ contentField:
"Summary", headerField: "Id" }}>
<ColumnsDirective>
<ColumnDirective headerText="To Do" keyField="Open" />
<ColumnDirective headerText="In Progress" keyField="InProgress" />
<ColumnDirective headerText="Testing" keyField="Testing" />
<ColumnDirective headerText="Done" keyField="Close" />
</ColumnsDirective>
</KanbanComponent>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
`
{% endraw %}

```

The server-side controller code to handle the CRUD operations are as follows.

```

`ts
private NORTHWNDEntities db = new NORTHWNDEntities();
public ActionResult DataSource() {
var DataSource = db.Tasks.ToList();
return Json(DataSource, JsonRequestBehavior.AllowGet);
}
public ActionResult Insert(Params value) {
//Insert card data into the database
return Json(value, JsonRequestBehavior.AllowGet);
}

```



```
public ActionResult Update(Params value) {
//Update card data into the database
return Json(value, JsonRequestBehavior.AllowGet);
}

public void Delete(Params value) {
//Delete card data from the database
}

public class Params {
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Assignee { get; set; }
}
`ts

private NORTHWNDEntities db = new NORTHWNDEntities();
public ActionResult DataSource() {
var DataSource = db.Tasks.ToList();
return Json(DataSource, JsonRequestBehavior.AllowGet);
}

public ActionResult Insert(Params value) {
//Insert card data into the database
return Json(value, JsonRequestBehavior.AllowGet);
}

public ActionResult Update(Params value) {
//Update card data into the database
return Json(value, JsonRequestBehavior.AllowGet);
}

public void Delete(Params value) {
//Delete card data from the database
}

public class Params {
public int Id { get; set; }
```

```

public string Status { get; set; }
public string Summary { get; set; }
public string Assignee { get; set; }
}

```

The `crudUrl` is used to update the bulk data sent to the server-side. Multiple selections and `sortBy` as `Index` properties are used for `crudUrl` properties to update the modified bulk data to the server-side.

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 `TaskId` for the cards by overriding the built-in response processing using the `processResponse` method of the `ODataAdaptor`.

[Class-component]

TASKIDADAPTOR.JSX

```

import { setValue } from '@syncfusion/ej2-base';
import { ODataAdaptor } from '@syncfusion/ej2-data';
export class TaskIdAdaptor extends ODataAdaptor {
    processResponse() {
        let i = 0;
        /* calling base class processResponse function */
        const original = super.processResponse.apply(this, arguments);
        /* adding Task Id */
        original.forEach((item) => setValue('Id', 'Task - ' + ++i, item));
        return original;
    }
}

```

TASKIDADAPTOR.TSX

```

import { setValue } from '@syncfusion/ej2-base';
import { ODataAdaptor } from '@syncfusion/ej2-data';
export class TaskIdAdaptor extends ODataAdaptor {
    public processResponse() {
        let i = 0;
        /* calling base class processResponse function */
        const original = super.processResponse.apply(this, arguments);
        /* adding Task Id */
        original.forEach((item: any) => setValue('Id', 'Task - ' + ++i,
item));
        return original;
    }
}

```

[Functional-component]

TASKIDADAPTOR.JSX

```

import { setValue } from '@syncfusion/ej2-base';

```

```
import { ODataAdaptor } from '@syncfusion/ej2-data';
export class TaskIdAdaptor extends ODataAdaptor {
    processResponse() {
        let i = 0;
        /* calling base class processResponse function */
        const original = super.processResponse.apply(this, arguments);
        /* adding Task Id */
        original.forEach((item) => setValue('Id', 'Task - ' + ++i, item));
        return original;
    }
}
```

TASKIDADAPTOR.TSX

```
import { setValue } from '@syncfusion/ej2-base';
import { ODataAdaptor } from '@syncfusion/ej2-data';
export class TaskIdAdaptor extends ODataAdaptor {
    public processResponse() {
        let i = 0;
        /* calling base class processResponse function */
        const original = super.processResponse.apply(this, arguments);
        /* adding Task Id */
        original.forEach((item: any) => setValue('Id', 'Task - ' + ++i,
item));
        return original;
    }
}
```

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 kanban [query](#) property.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor, Query } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component {
    data = new DataManager({
        url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
        adaptor: new ODataAdaptor
    });
    DialogOpen(args) {
        args.cancel = true;
    }
    query = new Query().addParams('ej2kanban', 'true');
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
```

```

    "Id" }} query={this.query} allowDragAndDrop={false}
    dialogOpen={this.DialogOpen.bind(this)}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor, Query } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs }
from "@syncfusion/ej2-react-kanban";
class App extends React.Component<{}>, {}>{
    public data = new DataManager({
        url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
        adaptor: new ODataAdaptor
    });
    private DialogOpen(args: DialogEventArgs): void {
        args.cancel = true;
    }
    private query: Query = new Query().addParams('ej2kanban', 'true');
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} query={this.query} allowDragAndDrop={false}
dialogOpen={this.DialogOpen.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```
<!DOCTYPE html>
```

```

<html lang="en">
<head>
  <title>Kanban Additional Parameter</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Additional Parameter" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor, Query } from '@syncfusion/ej2-data';

```

```

import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
function App() {
  let data = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
    adaptor: new ODataAdaptor
  });
  function DialogOpen(args) {
    args.cancel = true;
  }
  let query = new Query().addParams('ej2kanban', 'true');
  return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} query={query}
allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
    <ColumnsDirective>
      <ColumnDirective headerText="To Do" keyField="Open"/>
      <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
      <ColumnDirective headerText="Testing" keyField="Testing"/>
      <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
  </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataAdaptor, Query } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs }
from "@syncfusion/ej2-react-kanban";
function App() {
  let data = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
    adaptor: new ODataAdaptor
  });
  function DialogOpen(args: DialogEventArgs): void {
    args.cancel = true;
  }
  let query: Query = new Query().addParams('ej2kanban', 'true');
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} query={query}
allowDragAndDrop={false} dialogOpen={DialogOpen.bind(this)}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress"
/>
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

```

    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Additional Parameter</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Additional Parameter" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008c8f;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

The parameters added using the [query](#) property will be sent along with the data request for every kanban action.

Handling HTTP error

During server interaction from the kanban, 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.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component {
  kanban;
  data = new DataManager({
    url: 'http://some.com/invalidUrl'
  });
  render() {
    this.onActionFailure = this.onActionFailure.bind(this);
    return <KanbanComponent ref={kanban => this.kanban = kanban}
id="kanban" keyField="Status" dataSource={this.data} cardSettings={{
contentField: "Summary", headerField: "Id" }}
actionFailure={this.onActionFailure}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
  onActionFailure = (e) => {
    const span = document.createElement('span');
    if (this.kanban) {
      this.kanban.element.parentNode.insertBefore(span,
this.kanban.element);
      span.style.color = "#FF0000";
      span.innerHTML = "Server exception: 404 Not found";
    }
  };
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```


INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogEventArgs, ActionEventArgs } from "@syncfusion/ej2-react-kanban";
class App extends React.Component<{}>, {}>{
    private kanban: KanbanComponent;
    public data = new DataManager({
        url: 'http://some.com/invalidUrl'
    });
    render() {
        this.onActionFailure = this.onActionFailure.bind(this);
        return <KanbanComponent ref={kanban => this.kanban = kanban}
id="kanban" keyField="Status" dataSource={this.data} cardSettings={{
contentField: "Summary", headerField: "Id" }}
actionFailure={this.onActionFailure}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
    public onActionFailure = (e: ActionEventArgs) => {
        const span: HTMLElement = document.createElement('span');
        if (this.kanban) {
            (this.kanban.element.parentNode as
HTMLElement).insertBefore(span, this.kanban.element);
            span.style.color = "#FF0000";
            span.innerHTML = "Server exception: 404 Not found";
        }
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Action Failure</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Action Failure" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
function App() {
    let data = new DataManager({
        url: 'http://some.com/invalidUrl'
    });
    let kanban;
    return (<KanbanComponent ref={kanban => kanban = kanban} id="kanban"
keyField="Status" dataSource={data} cardSettings={{ contentField: "Summary",
headerField: "Id" }} actionFailure={onActionFailure.bind(this)}>
        <ColumnsDirective>

```

```

        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress" keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>);
function onActionFailure(e) {
    const span = document.createElement('span');
    if (kanban) {
        kanban.element.parentNode.insertBefore(span, kanban.element);
        span.style.color = "#FF0000";
        span.innerHTML = "Server exception: 404 Not found";
    }
}
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogEventArgs, ActionEventArgs } from "@syncfusion/ej2-react-kanban";
function App() {
    let data = new DataManager({
        url: 'http://some.com/invalidUrl'
    });
    let kanbanObj: KanbanComponent | null;
    return (
        <KanbanComponent ref={(kanban) => { kanbanObj = kanban }} id="kanban"
        keyField="Status" dataSource={data} cardSettings={{ contentField: "Summary",
        headerField: "Id" }} actionFailure={onActionFailure}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
    function onActionFailure (e: ActionEventArgs) {
        const span: HTMLElement = document.createElement('span');
        if (kanbanObj) {
            (kanbanObj.element.parentNode as HTMLElement).insertBefore(span,
            kanbanObj.element);
            span.style.color = "#FF0000";
            span.innerHTML = "Server exception: 404 Not found";
        }
    }
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Action Failure</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Action Failure" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

The [actionFailure](#) event will be triggered not only for the server errors, but also when there is an exception while processing the kanban actions.

Loading data via ajax

You can use Kanban [dataSource](#) property to bind the datasource to Kanban from external ajax request. In the following code, we have fetched the datasource from the server using ajax request and provided that to the [dataSource](#) property by using the **onSuccess** event of ajax.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { Ajax } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component {
  kanban;
  componentDidMount() {
    const ajax = new
    Ajax("https://ej2services.syncfusion.com/production/web-
    services/api/Orders", "GET");
    ajax.send();
    ajax.onSuccess = (data) => {
      if (this.kanban) {
        this.kanban.dataSource = JSON.parse(data);
      }
    };
  }
  render() {
    return <KanbanComponent ref={kanban => this.kanban = kanban}
    id="kanban" keyField="ShipCountry" dataSource={this.data} cardSettings={{
    contentField: "ShippedDate", headerField: "OrderID" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="Denmark" keyField="Denmark"/>
        <ColumnDirective headerText="Brazil" keyField="Brazil"/>
        <ColumnDirective headerText="Switzerland"
keyField="Switzerland"/>
        <ColumnDirective headerText="Germany" keyField="Germany"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { Ajax } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective, Kanban } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component<{}, {}>{
  public kanban: Kanban | null;
```

```

public componentDidMount() {
    const ajax = new
    Ajax("https://ej2services.syncfusion.com/production/web-
    services/api/Orders", "GET");
    ajax.send();
    ajax.onSuccess = (data: any) => {
        if (this.kanban) {
            this.kanban.dataSource = JSON.parse(data);
        }
    }
    render() {
        return <KanbanComponent ref={kanban => this.kanban = kanban}
        id="kanban" keyField="ShipCountry" dataSource={this.data} cardSettings={{
        contentField: "ShippedDate", headerField: "OrderID" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="Denmark" keyField="Denmark"
            />
                <ColumnDirective headerText="Brazil" keyField="Brazil" />
                <ColumnDirective headerText="Switzerland"
            keyField="Switzerland" />
                <ColumnDirective headerText="Germany" keyField="Germany"
            />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban AJAX Binding</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban AJAX Binding" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    kanban/styles/material.css" rel="stylesheet" />

```

```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { Ajax } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
function App() {
    let kanban;
    function componentDidMount() {
        const ajax = new
Ajax("https://ej2services.syncfusion.com/production/web-
services/api/Orders", "GET");
        ajax.send();
        ajax.onSuccess = (data) => {
            if (kanban) {
                kanban.dataSource = JSON.parse(data);
            }
        };
    }
    return (<KanbanComponent ref={kanban => kanban = kanban} id="kanban"
keyField="ShipCountry" dataSource={data} cardSettings={{ contentField:
"ShippedDate", headerField: "OrderID" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="Denmark" keyField="Denmark"/>
            <ColumnDirective headerText="Brazil" keyField="Brazil"/>
            <ColumnDirective headerText="Switzerland" keyField="Switzerland"/>
            <ColumnDirective headerText="Germany" keyField="Germany"/>

```

```

        </ColumnsDirective>
      </KanbanComponent>);
    }
    ReactDOM.render(<App />, document.getElementById('kanban'));
    {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { Ajax } from '@syncfusion/ej2-base';
import { useEffect } from 'react';
import { KanbanComponent, ColumnsDirective, ColumnDirective, Kanban } from
"@syncfusion/ej2-react-kanban";
function App() {
  let kanbanObj: KanbanComponent | null;
  React.useEffect(() => {
    const ajax = new Ajax("https://ej2services.syncfusion.com/production/web-
services/api/Orders", "GET");
    ajax.send();
    ajax.onSuccess = (data: any) => {
      if (kanbanObj) {
        kanbanObj.dataSource = JSON.parse(data);
      }
    };
  });
  return (
    <KanbanComponent ref={(kanban) => { kanbanObj = kanban }} id="kanban"
    keyField="ShipCountry" cardSettings={{ contentField: "ShippedDate",
    headerField: "OrderID" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="Denmark" keyField="Denmark" />
        <ColumnDirective headerText="Brazil" keyField="Brazil" />
        <ColumnDirective headerText="Switzerland" keyField="Switzerland" />
        <ColumnDirective headerText="Germany" keyField="Germany" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban AJAX Binding</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban AJAX Binding" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />

```



```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</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.

Swimlane in React Kanban component

Swimlanes are horizontal categorizations of cards on the Kanban board. It is used for grouping of cards, which brings transparency to the workflow process.

Render swimlane row

Cards can be grouped based on **keyField** and displayed in rows, which are separated by columns. It is mandatory to define the **keyField** that is mapped from the datasource for rendering swimlane rows in the Kanban board.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

```

        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Swimlane</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban board with swimlane" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]**INDEX.JSX**

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
```

```
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Swimlane</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban board with swimlane" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

Custom row text

Customize the swimlane row header text by using the `textField` property mapped from datasource.

It is not mandatory to define the `textField` to `swimlaneSettings`. It will automatically consider the `keyField` to swimlane row header text.

If the mapping `textField` key is not present in the datasource, it will consider the swimlane `keyField` as swimlane row header text.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", textField: 'AssigneeName '
}}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
```

```

    return <KanbanComponent id="kanban" keyField="Status"
    dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
    "Id" }} swimlaneSettings={{ keyField: "Assignee", textField: 'AssigneeName'
    }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" />
            <ColumnDirective headerText="In Progress"
    keyField="InProgress" />
            <ColumnDirective headerText="Testing" keyField="Testing"
    />
            <ColumnDirective headerText="Done" keyField="Close" />
        </ColumnsDirective>
    </KanbanComponent>
    }
    };
    ReactDOM.render(<App />, document.getElementById('kanban'));
    {% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Swimlane Custom Header Row Text</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Swimlane custom row text" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    kanban/styles/material.css" rel="stylesheet" />
    <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
    rel="stylesheet" />
    <script
    src="https://cdn.jsdelivr.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
    ript>
    <script src="systemjs.config.js"></script>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;

```

```

        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", textField: 'AssigneeName' }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", textField: 'AssigneeName' }}>

```



```

        <ColumnsDirective>
          <ColumnDirective headerText="To Do" keyField="Open" />
          <ColumnDirective headerText="In Progress"
keyField="InProgress" />
          <ColumnDirective headerText="Testing" keyField="Testing" />
          <ColumnDirective headerText="Done" keyField="Close" />
        </ColumnsDirective>
      </KanbanComponent>
    );
  }
  ReactDOM.render(<App />, document.getElementById('kanban'));
  {% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Swimlane Custom Header Row Text</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Swimlane custom row text" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>

```

```

</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Template

You can customize the Kanban swimlane row by using the `template` property, which is specified within the `swimlaneSettings` property. In this demo, the swimlane header is customized with HTML element.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  rowTemplate(props) {
    var src = props.keyField + '.png';
    return (<div className='swimlane-template e-swimlane-template-
table'>
      <div className="e-swimlane-row-text"><img src={src}
alt={props.keyField}/>
      <span>{props.textField}</span></div>
    </div>);
  }
  template = this.rowTemplate;
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", textField: 'AssigneeName',
template: this.template.bind(this) }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}>, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  public rowTemplate(props): any {
    var src = props.keyField + '.png';
    return (
      <div className='swimlane-template e-swimlane-template-table'>
        <div className="e-swimlane-row-text"><img src={src}
alt={props.keyField} />
          <span>{props.textField}</span></div>
        </div>
      </div>
    );
  }
  public template: any = this.rowTemplate;
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", textField: 'AssigneeName',
template: this.template.bind(this) }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Swimlane Template</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Swimlane template" />
  <meta name="author" content="Syncfusion" />
</head>
```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link href="index.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008c8f;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function rowTemplate(props) {

```

```

        var src = props.keyField + '.png';
        return (<div className='swimlane-template e-swimlane-template-
table'>
            <div className="e-swimlane-row-text"><img src={src}
alt={props.keyField}/>
                <span>{props.textField}</span></div>
            </div>);
    }
    let template = rowTemplate;
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", textField: 'AssigneeName',
template: template.bind(this) }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function rowTemplate(props): any {
        var src = props.keyField + '.png';
        return (
            <div className='swimlane-template e-swimlane-template-table'>
                <div className="e-swimlane-row-text"><img src={src}
alt={props.keyField} />
                    <span>{props.textField}</span></div>
                </div>
            );
    }
    let template: any = rowTemplate;
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", textField: 'AssigneeName',
template: template.bind(this) }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress"
/>
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

```

        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>
</KanbanComponent>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Swimlane Template</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Swimlane template" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link href="index.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>

```

```

        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Sorting

Swimlane rows are rendered on descending order when using the `sortBy` property set to `Descending` order. By default, swimlane rows are rendered by **Ascending** order.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", sortBy: 'Descending' }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);

```

```

        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
        "Id" }} swimlaneSettings={{ keyField: "Assignee", sortBy: 'Descending' }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
                keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
                />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Swimlane Sorting</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Swimlane sorting" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;

```



```

        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", sortBy: 'Descending' }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (

```

```

    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ contentField: "Summary", headerField: "Id" }}
    swimlaneSettings={{ keyField: "Assignee", sortBy: 'Descending' }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Swimlane Sorting</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Swimlane sorting" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>

```

```

    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Drag-and-drop

By default, The Kanban does not allow dragging the cards across the swimlane rows. Enabling the `dragAndDrop` property allows you to drag the cards across the swimlane rows, which is specified inside `swimlaneSettings` property.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", allowDragAndDrop: true }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';

```

```

import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}>, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee",allowDragAndDrop: true }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Across Swimlane Drag and Drop</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Swimlane drag and drop" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
  let data = extend([], kanbanData, null, true);
  return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", allowDragAndDrop: true }}>
    <ColumnsDirective>
      <ColumnDirective headerText="To Do" keyField="Open"/>
      <ColumnDirective headerText="In Progress" keyField="InProgress"/>
      <ColumnDirective headerText="Testing" keyField="Testing"/>
      <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
  </KanbanComponent>);
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';

```

```
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
  let data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", allowDragAndDrop: true }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  )
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Across Swimlane Drag and Drop</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Swimlane drag and drop" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
```

```

        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Create empty row

You can render the empty swimlane row by enabling the `showEmptyRow` property. If mapping `keyField` does not have cards, empty swimlane row will be rendered.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", showEmptyRow: true }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}>, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", showEmptyRow: true }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Swimlane Row with Empty Rows</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Empty swimlane row" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
```



```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", showEmptyRow: true }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';

```

```

import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", showEmptyRow: true }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Swimlane Row with Empty Rows</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Empty swimlane row" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>

```

```

<style>
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
</style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Calculate cards count

Users can show or hide the cards count by swimlane row in header when enabling the `showItemCount` property, which is enabled by default on the Kanban board.

Provided localization support for **items** text.

In below demo, disabled on `showItemCount` property on rendering swimlane row without total count.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
    dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
    "Id" }} swimlaneSettings={{ keyField: "Assignee", showItemCount: false }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
        keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}

```

```
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", showItemCount: false }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Swimlane Cards Count</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Swimlane row cards count" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", showItemCount: false }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));

```

```
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
  let data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", showItemCount: false }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Swimlane Cards Count</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Swimlane row cards count" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
```

```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Enable frozen rows

Frozen rows provide an option to make the current swimlane row header text always visible on top of the content while scrolling the Kanban content. The swimlane header text will be changed dynamically, when you scroll to another swimlane row.

By default, the `enableFrozenRows` property is set as `false`. If you wish to show the swimlane frozen rows, you can enable the `enableFrozenRows` property.

This feature support only when using Kanban content scrolling.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", enableFrozenRows: true }}
height="500px">

```

```

        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>;
}
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", enableFrozenRows: true }}
height="500px">
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Swimlane Cards Count</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Swimlane row cards count" />

```



```

    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);

```

```

    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", enableFrozenRows: true }}
height="500px">
    <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", enableFrozenRows: true }}
height="500px">
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Swimlane Cards Count</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Swimlane row cards count" />
    <meta name="author" content="Syncfusion" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Drag and drop in React Kanban component

All cards can be dragged and dropped across the columns or within the columns or swimlane row or kanban to an external source and vice versa.

The following drag and drop types are available in the Kanban board.

- Internal drag and drop
- Column drag and drop
- Swimlane drag and drop
- External drag and drop
- Kanban to Kanban
- Kanban to External source and vice versa.

Dropped card position varies based on the `sortSettings` property.

Internal drag and drop

Allows the user to drag and drop the cards within the kanban board. Based on this, we can categorize into two ways.

- Column drag and drop
- Swimlane drag and drop

Column drag and drop

By default, all cards can be dragged and dropped across the columns and within the columns. You cannot drag and drop the cards when disabling the `allowDragAndDrop` property.

You can prevent the drag or drop behavior of the particular column by disabling the `allowDrag` or `allowDrop` property.

You can also control the flow of transition cards between the columns by using the `transitionColumns` property.

In the following example, disable the drag and drop behavior on the Kanban board.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} allowDragAndDrop={false}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} allowDragAndDrop={false}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Cards</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Cards" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
```

```

    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
allowDragAndDrop={false}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
  let data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ contentField: "Summary", headerField: "Id" }}
    allowDragAndDrop={false}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Cards</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Cards" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
</style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Swimlane drag and drop

By default, Swimlane allows drag and drop across the columns within the swimlane row. Kanban does not allow dragging the cards across the swimlane rows.

Enabling the `dragAndDrop` property allows you to drag the cards across the swimlane rows, which is specified inside the `swimlaneSettings` property.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", allowDragAndDrop: true }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>

```



```

        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}>, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} swimlaneSettings={{ keyField: "Assignee", allowDragAndDrop: true }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Across Swimlane Drag and Drop</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Swimlane drag and drop" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee", allowDragAndDrop: true }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>

```

```

        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
        cardSettings={{ contentField: "Summary", headerField: "Id" }}
        swimlaneSettings={{ keyField: "Assignee", allowDragAndDrop: true }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Across Swimlane Drag and Drop</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Swimlane drag and drop" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
</style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

External drag and drop

Allows the user to drag and drop the cards from one kanban to another kanban or Kanban to an external source and vice versa.

Kanban to kanban

Drag and drop the card from one kanban to another kanban and vice versa. This can be achieved by specifying the `externalDropId` property which is used to specify the id of the dropped kanban element and the `dragStop` event which is used to delete the card on dragged Kanban and add the card on dropped Kanban using the `deleteCard` and `addCard` public methods.

Before adding a card to dropped kanban, you can manually change the card data `headerField` when the same card data `headerField` is dropped to another Kanban.

In the following example, Drag the card from one Kanban and drop it into another kanban using the `dragStop` event. In this event, remove the card from the dragged Kanban by using the `deleteCard` public method and add the card to the dropped Kanban by using the `addCard` public method.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest } from '@syncfusion/ej2-base';

```

```

import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanAData, kanbanBData } from './datasource';
class App extends React.Component {
  kanbanA;
  kanbanB;
  constructor() {
    super(...arguments);
    this.dataA = extend([], kanbanAData, null, true);
    this.dataB = extend([], kanbanBData, null, true);
    this.externalKanbanADropId = ["#KanbanB"];
    this.externalKanbanBDropId = ["#KanbanA"];
  }
  onKanbanADragStop(args) {
    let kanbanBElement = closest(args.event.target, '#KanbanB');
    if (kanbanBElement) {
      this.kanbanA.deleteCard(args.data);
      args.data.forEach((card, i) => {
        const index = this.kanbanB.kanbanData.findIndex((colData) =>
colData[this.kanbanB.cardSettings.headerField] ===
card[this.kanbanB.cardSettings.headerField]);
        if (index !== -1) {
          card[this.kanbanB.cardSettings.headerField] =
Math.max(...this.kanbanB.kanbanData.map((obj) =>
parseInt(obj[this.kanbanB.cardSettings.headerField], 10))) + ++i;
        }
      });
      this.kanbanB.addCard(args.data, args.dropIndex);
      args.cancel = true;
    }
  }
  onKanbanBDragStop(args) {
    let kanbanAElement = closest(args.event.target, '#KanbanA');
    if (kanbanAElement) {
      this.kanbanB.deleteCard(args.data);
      args.data.forEach((card, i) => {
        const index = this.kanbanA.kanbanData.findIndex((colData) =>
colData[this.kanbanA.cardSettings.headerField] ===
card[this.kanbanA.cardSettings.headerField]);
        if (index !== -1) {
          card[this.kanbanA.cardSettings.headerField] =
Math.max(...this.kanbanA.kanbanData.map((obj) =>
parseInt(obj[this.kanbanA.cardSettings.headerField], 10))) + ++i;
        }
      });
      this.kanbanA.addCard(args.data, args.dropIndex);
      args.cancel = true;
    }
  }
  render() {
    return <div>
      <div className="container-fluid">
        <div className="row">
          <div className="col-sm-6">
            <h4>Kanban A</h4>
            <KanbanComponent id="KanbanA" ref={kanbanA =>
this.kanbanA = kanbanA} keyField="Status" dataSource={this.dataA}

```

```

cardSettings={{ contentField: "Summary", headerField: "Id" }}
dragStop={this.onKanbanADragStop.bind(this)}
externalDropId={this.externalKanbanADropId}>
    <ColumnsDirective>
        <ColumnDirective headerText="To Do"
keyField="Open"/>
        <ColumnDirective headerText="Done"
keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>
</div>
<div className="col-sm-6">
    <h4>Kanban B</h4>
    <KanbanComponent id="KanbanB" ref={kanbanB =>
this.kanbanB = kanbanB} keyField="Status" dataSource={this.dataB}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dragStop={this.onKanbanBDragStop.bind(this)}
externalDropId={this.externalKanbanBDropId}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do"
keyField="Open"/>
            <ColumnDirective headerText="Done"
keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>
</div>
</div>
</div>
</div>
</div>
}
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DragEventArgs,
Kanban } from "@syncfusion/ej2-react-kanban";
import { kanbanAData, kanbanBData } from './datasource';
class App extends React.Component<{}, {}>{
    public kanbanA: Kanban | null;
    public kanbanB: Kanban | null;
    constructor() {
        super(...arguments);
        this.dataA = extend([], kanbanAData, null, true);
        this.dataB = extend([], kanbanBData, null, true);
        this.externalKanbanADropId = ["#KanbanB"];
        this.externalKanbanBDropId = ["#KanbanA"];
    }
    private onKanbanADragStop(args: DragEventArgs): void {

```

```

    let kanbanBElement: Element = closest(args.event.target as Element,
    '#KanbanB');
    if (kanbanBElement) {
        this.kanbanA.deleteCard(args.data);
        args.data.forEach((card: Record<string, any>, i: number) => {
            const index: number =
this.kanbanB.kanbanData.findIndex((colData: Record<string, any>) =>
            colData[this.kanbanB.cardSettings.headerField] ===
card[this.kanbanB.cardSettings.headerField]);
            if (index !== -1) {
                card[this.kanbanB.cardSettings.headerField] =
Math.max(...this.kanbanB.kanbanData.map(
                    (obj: Record<string, string>) =>
parseInt(obj[this.kanbanB.cardSettings.headerField], 10))) + ++i;
            }
        });
        this.kanbanB.addCard(args.data, args.dropIndex);
        args.cancel = true;
    }
}

private onKanbanBDragStop(args: DragEventArgs): void {
    let kanbanAElement: Element = closest(args.event.target as Element,
    '#KanbanA');
    if (kanbanAElement) {
        this.kanbanB.deleteCard(args.data);
        args.data.forEach((card: Record<string, any>, i: number) => {
            const index: number =
this.kanbanA.kanbanData.findIndex((colData: Record<string, any>) =>
            colData[this.kanbanA.cardSettings.headerField] ===
card[this.kanbanA.cardSettings.headerField]);
            if (index !== -1) {
                card[this.kanbanA.cardSettings.headerField] =
Math.max(...this.kanbanA.kanbanData.map(
                    (obj: Record<string, string>) =>
parseInt(obj[this.kanbanA.cardSettings.headerField], 10))) + ++i;
            }
        });
        this.kanbanA.addCard(args.data, args.dropIndex);
        args.cancel = true;
    }
}

render() {
    return <div>
        <div className="container-fluid">
            <div className="row">
                <div className="col-sm-6">
                    <h4>Kanban A</h4>
                    <KanbanComponent id="KanbanA" ref={kanbanA =>
this.kanbanA = kanbanA} keyField="Status" dataSource={this.dataA}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dragStop={this.onKanbanADragStop.bind(this)}
externalDropId={this.externalKanbanADropId}>
                        <ColumnsDirective>
                            <ColumnDirective headerText="To Do"
keyField="Open" />
                            <ColumnDirective headerText="Done"
keyField="Close" />
                    </KanbanComponent>
                </div>
            </div>
        </div>
    </div>

```

```

        </ColumnsDirective>
      </KanbanComponent>
    </div>
    <div className="col-sm-6">
      <h4>Kanban B</h4>
      <KanbanComponent id="KanbanB" ref={kanbanB =>
        this.kanbanB = kanbanB} keyField="Status" dataSource={this.dataB}
        cardSettings={{ contentField: "Summary", headerField: "Id" }}
        dragStop={this.onKanbanBDragStop.bind(this)}
        externalDropId={this.externalKanbanBDropId}>
        <ColumnsDirective>
          <ColumnDirective headerText="To Do"
            keyField="Open" />
          <ColumnDirective headerText="Done"
            keyField="Close" />
        </ColumnsDirective>
      </KanbanComponent>
    </div>
  </div>
</div>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban to Kanban drag and drop</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban to Kanban drag and drop" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-kanban/styles/material.css" rel="stylesheet" />
  <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
    .row {
      display: flex;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanAData, kanbanBData } from './datasource';
function App() {
  let kanbanA;
  let kanbanB;
  let dataA = extend([], kanbanAData, null, true);
  let dataB = extend([], kanbanBData, null, true);
  let externalKanbanADropId = ["#KanbanB"];
  let externalKanbanBDropId = ["#KanbanA"];
  function onKanbanADragStop(args) {
    let kanbanBElement = closest(args.event.target, '#KanbanB');
    if (kanbanBElement) {
      kanbanA.deleteCard(args.data);
      args.data.forEach((card, i) => {
        const index = kanbanB.kanbanData.findIndex((colData) =>
colData[kanbanB.cardSettings.headerField] ===
card[kanbanB.cardSettings.headerField]);
        if (index !== -1) {
          card[kanbanB.cardSettings.headerField] =
Math.max(...kanbanB.kanbanData.map((obj) =>
parseInt(obj[kanbanB.cardSettings.headerField], 10))) + ++i;
        }
      });
    }
  }
}

```

```

    });
    kanbanB.addCard(args.data, args.dropIndex);
    args.cancel = true;
  }
}
function onKanbanBDragStop(args) {
  let kanbanAElement = closest(args.event.target, '#KanbanA');
  if (kanbanAElement) {
    kanbanB.deleteCard(args.data);
    args.data.forEach((card, i) => {
      const index = kanbanA.kanbanData.findIndex((colData) =>
colData[kanbanA.cardSettings.headerField] ===
card[kanbanA.cardSettings.headerField]);
      if (index !== -1) {
        card[kanbanA.cardSettings.headerField] =
Math.max(...kanbanA.kanbanData.map((obj) =>
parseInt(obj[kanbanA.cardSettings.headerField], 10))) + ++i;
      }
    });
    kanbanA.addCard(args.data, args.dropIndex);
    args.cancel = true;
  }
}
return (<div>
  <div className="container-fluid">
    <div className="row">
      <div className="col-sm-6">
        <h4>Kanban A</h4>
        <KanbanComponent id="KanbanA" ref={kanbanA => kanbanA =
kanbanA} keyField="Status" dataSource={dataA} cardSettings={{ contentField:
"Summary", headerField: "Id" }} dragStop={onKanbanADragStop.bind(this)}
externalDropId={externalKanbanADropId}>
          <ColumnsDirective>
            <ColumnDirective headerText="To Do"
keyField="Open"/>
            <ColumnDirective headerText="Done"
keyField="Close"/>
          </ColumnsDirective>
        </KanbanComponent>
      </div>
      <div className="col-sm-6">
        <h4>Kanban B</h4>
        <KanbanComponent id="KanbanB" ref={kanbanB => kanbanB =
kanbanB} keyField="Status" dataSource={dataB} cardSettings={{ contentField:
"Summary", headerField: "Id" }} dragStop={onKanbanBDragStop.bind(this)}
externalDropId={externalKanbanBDropId}>
          <ColumnsDirective>
            <ColumnDirective headerText="To Do"
keyField="Open"/>
            <ColumnDirective headerText="Done"
keyField="Close"/>
          </ColumnsDirective>
        </KanbanComponent>
      </div>
    </div>
  </div>
</div>);

```

```

}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DragEventArgs,
Kanban } from "@syncfusion/ej2-react-kanban";
import { kanbanAData, kanbanBData } from './datasource';
function App() {
    let kanbanA: Kanban | null;
    let kanbanB: Kanban | null;
    let dataA = extend([], kanbanAData, null, true);
    let dataB = extend([], kanbanBData, null, true);
    let externalKanbanADropId = ["#KanbanB"];
    let externalKanbanBDropId = ["#KanbanA"];
    function onKanbanADragStop(args: DragEventArgs): void {
        let kanbanBElement: Element = closest(args.event.target as Element,
'#KanbanB');
        if (kanbanBElement) {
            kanbanA.deleteCard(args.data);
            args.data.forEach((card: Record<string, any>, i: number) => {
                const index: number = kanbanB.kanbanData.findIndex((colData:
Record<string, any>) =>
                    colData[kanbanB.cardSettings.headerField] ===
card[kanbanB.cardSettings.headerField]);
                if (index !== -1) {
                    card[kanbanB.cardSettings.headerField] =
Math.max(...kanbanB.kanbanData.map(
                        (obj: Record<string, string>) =>
parseInt(obj[kanbanB.cardSettings.headerField], 10))) + ++i;
                }
            });
            kanbanB.addCard(args.data, args.dropIndex);
            args.cancel = true;
        }
    }
    function onKanbanBDragStop(args: DragEventArgs): void {
        let kanbanAElement: Element = closest(args.event.target as Element,
'#KanbanA');
        if (kanbanAElement) {
            kanbanB.deleteCard(args.data);
            args.data.forEach((card: Record<string, any>, i: number) => {
                const index: number = kanbanA.kanbanData.findIndex((colData:
Record<string, any>) =>
                    colData[kanbanA.cardSettings.headerField] ===
card[kanbanA.cardSettings.headerField]);
                if (index !== -1) {
                    card[kanbanA.cardSettings.headerField] =
Math.max(...kanbanA.kanbanData.map(
                        (obj: Record<string, string>) =>
parseInt(obj[kanbanA.cardSettings.headerField], 10))) + ++i;

```

```

    });
    kanbanA.addCard(args.data, args.dropIndex);
    args.cancel = true;
  }
}
return (
  <div>
    <div className="container-fluid">
      <div className="row">
        <div className="col-sm-6">
          <h4>Kanban A</h4>
          <KanbanComponent id="KanbanA" ref={kanbanA => kanbanA =
kanbanA} keyField="Status" dataSource={dataA} cardSettings={{ contentField:
"Summary", headerField: "Id" }} dragStop={onKanbanADragStop.bind(this)}
externalDropId={externalKanbanADropId}>
            <ColumnsDirective>
              <ColumnDirective headerText="To Do"
keyField="Open" />
              <ColumnDirective headerText="Done"
keyField="Close" />
            </ColumnsDirective>
          </KanbanComponent>
        </div>
        <div className="col-sm-6">
          <h4>Kanban B</h4>
          <KanbanComponent id="KanbanB" ref={kanbanB => kanbanB =
kanbanB} keyField="Status" dataSource={dataB} cardSettings={{ contentField:
"Summary", headerField: "Id" }} dragStop={onKanbanBDragStop.bind(this)}
externalDropId={externalKanbanBDropId}>
            <ColumnsDirective>
              <ColumnDirective headerText="To Do"
keyField="Open" />
              <ColumnDirective headerText="Done"
keyField="Close" />
            </ColumnsDirective>
          </KanbanComponent>
        </div>
      </div>
    </div>
  </div>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban to Kanban drag and drop</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban to Kanban drag and drop" />
  <meta name="author" content="Syncfusion" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .row {
        display: flex;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Treeview to Kanban

Drag the card from the Kanban board and drop it to the Treeview component and vice versa.

In the following sample, remove the data from the Kanban board using the `deleteCard` public method and add to the Treeview component using the `addNodes` public method at Kanban `dragStop` event when dragging the card and dropping it to the Treeview component. Remove the data from Treeview using the `removeNodes` public method and add to Kanban board using the `openDialog` public method when dragging the list from the Treeview component and dropping it to the kanban board.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { TreeViewComponent } from '@syncfusion/ej2-react-navigations';
import { kanbanData, treeViewData } from '../datasource';
class App extends React.Component {
  kanbanObj;
  treeViewObj;
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
    this.treeData = extend([], treeViewData, null, true);
    this.fields = { dataSource: this.treeData, id: 'Id', text: 'Status'
};

    this.allowDragAndDrops = true;
    this.externalKanbanDropId = ["#treeView"];
  }
  treeTemplate(props) {
    return (<div><div id="treelist"><div id="treeviewlist">{props.Id} -
{props.Status}</div>
      </div></div>);
  }
  onKanbanADragStop(args) {
    let treeViewElement = closest(args.event.target, '#treeView');
    if (treeViewElement) {
      this.kanbanObj.deleteCard(args.data);
      this.treeViewObj.addNodes(args.data);
      args.cancel = true;
    }
  }
  onTreeDragStop(args) {
    let kanbanElement = closest(args.event.target, '#Kanban');
    if (kanbanElement) {
      let treeData = this.treeViewObj.fields.dataSource;
      const filteredData = treeData.filter((item) => item.Id ===
parseInt(args.draggedNodeData.id, 10));
      this.treeViewObj.removeNodes([args.draggedNodeData.id]);
      this.kanbanObj.openDialog('Add', filteredData[0]);
      args.cancel = true;
    }
  }
  render() {
    return <div>
      <div className="container-fluid">
        <div className="row">
          <div className="col-sm-6">
            <h4>Kanban</h4>
            <KanbanComponent id="Kanban" ref={kanban =>
this.kanbanObj = kanban} keyField="Status" dataSource={this.data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dragStop={this.onKanbanADragStop.bind(this)}
externalDropId={this.externalKanbanDropId}>

```

```

                                <ColumnsDirective>
                                    <ColumnDirective headerText="To Do"
keyField="Open"/>
                                    <ColumnDirective headerText="Done"
keyField="Close"/>
                                </ColumnsDirective>
                            </KanbanComponent>
                        </div>
                    <div className="col-sm-6">
                        <h4>TreeView</h4>
                        <TreeViewComponent id="treeView" ref={treeView
=> this.treeViewObj = treeView} nodeTemplate={this.treeTemplate.bind(this)}
fields={this.fields} nodeDragStop={this.onTreeDragStop.bind(this)}
allowDragAndDrop={this.allowDragAndDrops}/>
                    </div>
                </div>
            </div>
        </div>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DragEventArgs,
Kanban } from "@syncfusion/ej2-react-kanban";
import { DragAndDropEventArgs } from '@syncfusion/ej2-navigations';
import { TreeViewComponent } from '@syncfusion/ej2-react-navigations';
import { kanbanData, treeViewData } from './datasource';
class App extends React.Component<{}, {}>{
    public kanbanObj: Kanban | null;
    public treeViewObj: TreeViewComponent | null;
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
        this.treeData = extend([], treeViewData, null, true);
        this.fields = { dataSource: this.treeData, id: 'Id', text: 'Status'
};

        this.allowDragAndDrops = true;
        this.externalKanbanDropId = ["#treeView"];
    }
    private treeTemplate(props: KanbanDataModel): JSX.Element {
        return (<div><div id="treelist"><div id="treeviewlist">{props.Id} -
{props.Status}</div>
                </div></div>);
    }
    private onKanbanADragStop(args: DragEventArgs): void {
        let treeViewElement: Element = closest(args.event.target as Element,
'#treeView');
        if (treeViewElement) {

```

```

        this.kanbanObj.deleteCard(args.data);
        this.treeViewObj.addNodes(args.data);
        args.cancel = true;
    }
}

private onTreeDragStop(args: DragAndDropEventArgs): void {
    let kanbanElement: Element = closest(args.event.target as Element,
    '#Kanban');
    if (kanbanElement) {
        let treeData = this.treeViewObj.fields.dataSource as { [key:
string]: Object }[];
        const filteredData =
            treeData.filter((item: any) => item.Id ===
parseInt(args.draggedNodeData.id as string, 10));
        this.treeViewObj.removeNodes([args.draggedNodeData.id] as
string[]);
        this.kanbanObj.openDialog('Add', filteredData[0]);
        args.cancel = true;
    }
}

render() {
    return <div>
        <div className="container-fluid">
            <div className="row">
                <div className="col-sm-6">
                    <h4>Kanban</h4>
                    <KanbanComponent id="Kanban" ref={kanban =>
this.kanbanObj = kanban} keyField="Status" dataSource={this.data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dragStop={this.onKanbanADragStop.bind(this)}
externalDropId={this.externalKanbanDropId} >
                        <ColumnsDirective>
                            <ColumnDirective headerText="To Do"
keyField="Open" />
                            <ColumnDirective headerText="Done"
keyField="Close" />
                        </ColumnsDirective>
                    </KanbanComponent>
                </div>
                <div className="col-sm-6">
                    <h4>TreeView</h4>
                    <TreeViewComponent id="treeView" ref={treeView
=> this.treeViewObj = treeView} nodeTemplate={this.treeTemplate.bind(this)}
fields={this.fields} nodeDragStop={this.onTreeDragStop.bind(this)}
allowDragAndDrop={this.allowDragAndDrops} />
                </div>
            </div>
        </div>
    </div>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML


```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban to treeview drag and drop</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban to treeview drag and drop" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
    .row {
      display: flex;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { TreeViewComponent } from '@syncfusion/ej2-react-navigations';
import { kanbanData, treeViewData } from './datasource';
function App() {
    let kanbanObj;
    let treeViewObj;
    let data = extend([], kanbanData, null, true);
    let treeData = extend([], treeViewData, null, true);
    let fields = { dataSource: treeData, id: 'Id', text: 'Status' };
    let allowDragAndDrops = true;
    let externalKanbanDropId = ["#treeView"];
    function treeTemplate(props) {
        return (<div><div id="treelist"><div id="treeviewlist">{props.Id} -
{props.Status}</div></div></div>);
    }
    function onKanbanADragStop(args) {
        let treeViewElement = closest(args.event.target, '#treeView');
        if (treeViewElement) {
            kanbanObj.deleteCard(args.data);
            treeViewObj.addNodes(args.data);
            args.cancel = true;
        }
    }
    function onTreeDragStop(args) {
        let kanbanElement = closest(args.event.target, '#Kanban');
        if (kanbanElement) {
            let treeData = treeViewObj.fields.dataSource;
            const filteredData = treeData.filter((item) => item.Id ===
parseInt(args.draggedNodeData.id, 10));
            treeViewObj.removeNodes([args.draggedNodeData.id]);
            kanbanObj.openDialog('Add', filteredData[0]);
            args.cancel = true;
        }
    }
    return (<div>
        <div className="container-fluid">
            <div className="row">
                <div className="col-sm-6">
                    <h4>Kanban</h4>
                    <KanbanComponent id="Kanban" ref={kanban => kanbanObj =
kanban} keyField="Status" dataSource={data} cardSettings={{ contentField:
"Summary", headerField: "Id" }} dragStop={onKanbanADragStop.bind(this)}
externalDropId={externalKanbanDropId}>
                        <ColumnsDirective>
                            <ColumnDirective headerText="To Do"
keyField="Open"/>
                            <ColumnDirective headerText="Done"
keyField="Close"/>
                        </ColumnsDirective>
                    </KanbanComponent>
                </div>
                <div className="col-sm-6">

```

```

        <h4>TreeView</h4>
        <TreeViewComponent id="treeView" ref={treeView =>
treeViewObj = treeView} nodeTemplate={treeTemplate.bind(this)}
fields={fields} nodeDragStop={onTreeDragStop.bind(this)}
allowDragAndDrop={allowDragAndDrops}/>
        </div>
    </div>
</div>
</div>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DragEventArgs,
Kanban } from '@syncfusion/ej2-react-kanban';
import { DragAndDropEventArgs } from '@syncfusion/ej2-navigations';
import { TreeViewComponent } from '@syncfusion/ej2-react-navigations';
import { kanbanData, treeViewData } from './datasource';
function App() {
    let kanbanObj: Kanban | null;
    let treeViewObj: TreeViewComponent | null;
    let data = extend([], kanbanData, null, true);
    let treeData = extend([], treeViewData, null, true);
    let fields = { dataSource: treeData, id: 'Id', text: 'Status' };
    let allowDragAndDrops = true;
    let externalKanbanDropId = ["#treeView"];
    function treeTemplate(props: KanbanDataModel): JSX.Element {
        return (<div><div id="treelist"><div id="treeviewlist">{props.Id} -
{props.Status}</div></div></div>);
    }
    function onKanbanADragStop(args: DragEventArgs): void {
        let treeViewElement: Element = closest(args.event.target as Element,
'#treeView');
        if (treeViewElement) {
            kanbanObj.deleteCard(args.data);
            treeViewObj.addNodes(args.data);
            args.cancel = true;
        }
    }
    function onTreeDragStop(args: DragAndDropEventArgs): void {
        let kanbanElement: Element = closest(args.event.target as Element,
'#Kanban');
        if (kanbanElement) {
            let treeData = treeViewObj.fields.dataSource as { [key: string]:
Object }[];
            const filteredData =
                treeData.filter((item: any) => item.Id ===
parseInt(args.draggedNodeData.id as string, 10));
            treeViewObj.removeNodes([args.draggedNodeData.id as string[]]);
            kanbanObj.openDialog('Add', filteredData[0]);

```

```

        args.cancel = true;
    }
}
return(
    <div>
        <div className="container-fluid">
            <div className="row">
                <div className="col-sm-6">
                    <h4>Kanban</h4>
                    <KanbanComponent id="Kanban" ref={kanban => kanbanObj =
kanban} keyField="Status" dataSource={data} cardSettings={{ contentField:
"Summary", headerField: "Id" }} dragStop={onKanbanADragStop.bind(this)}
externalDropId={externalKanbanDropId} >
                        <ColumnsDirective>
                            <ColumnDirective headerText="To Do"
keyField="Open" />
                            <ColumnDirective headerText="Done"
keyField="Close" />
                        </ColumnsDirective>
                    </KanbanComponent>
                </div>
                <div className="col-sm-6">
                    <h4>TreeView</h4>
                    <TreeViewComponent id="treeView" ref={treeView =>
treeViewObj = treeView} nodeTemplate={treeTemplate.bind(this)}
fields={fields} nodeDragStop={onTreeDragStop.bind(this)}
allowDragAndDrop={allowDragAndDrops} />
                </div>
            </div>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban to treeview drag and drop</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban to treeview drag and drop" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .row {
        display: flex;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Schedule to Kanban

Drag the card from the Kanban board and drop it to the Schedule component and vice versa.

In the following sample, remove the data from the Kanban board using the `deleteCard` public method and add to the schedule component using the `addNodes` public method at Kanban `dragStop` event when dragging the card and dropping it to the Treeview component. Remove the data from Treeview using the `removeNodes` public method and add to Kanban board using the `addCard` public method when dragging the list from the Treeview component and dropping it to the kanban board.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest, removeClass } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { ScheduleComponent, ResourcesDirective, ResourceDirective,
ViewsDirective, ViewDirective, Inject, TimelineViews, Resize, DragAndDrop,
TimelineMonth } from '@syncfusion/ej2-react-schedule';

```

```

import { kanbanData, scheduleData } from './datasource';
class App extends React.Component {
  kanbanObj;
  scheduleObj;
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
    this.scheduleDataSource = extend([], scheduleData, null, true);
    this.externalKanbanDropId = ["#Schedule"];
  }
  departmentData = [
    { Text: 'GENERAL', Id: 1, Color: '#bbdc00' },
    { Text: 'DENTAL', Id: 2, Color: '#9e5fff' }
  ];
  consultantData = [
    { Text: 'Alice', Id: 1, GroupId: 1, Color: '#bbdc00', Designation: 'Cardiologist' },
    { Text: 'Nancy', Id: 2, GroupId: 2, Color: '#9e5fff', Designation: 'Orthodontist' },
    { Text: 'Robert', Id: 3, GroupId: 1, Color: '#bbdc00', Designation: 'Optometrist' },
    { Text: 'Robson', Id: 4, GroupId: 2, Color: '#9e5fff', Designation: 'Periodontist' },
    { Text: 'Laura', Id: 5, GroupId: 1, Color: '#bbdc00', Designation: 'Orthopedic' },
    { Text: 'Margaret', Id: 6, GroupId: 2, Color: '#9e5fff', Designation: 'Endodontist' }
  ];
  getConsultantName(value) {
    return value.resourceData[value.resource.textField];
  }
  getConsultantDesignation(value) {
    return value.resourceData.Designation;
  }
  resourceHeaderTemplate(props) {
    return (<div className="template-wrap"><div className="specialist-category"><div className="specialist-name">
      {this.getConsultantName(props)}</div><div className="specialist-designation">{this.getConsultantDesignation(props)}</div></div></div>);
  }
  onKanbanDragStop(args) {
    let scheduleElement = closest(args.event.target, '#Schedule');
    if (scheduleElement) {
      this.kanbanObj.deleteCard(args.data);
      this.scheduleObj.openEditor(args.data[0], 'Add', true);
      args.cancel = true;
    }
  }
  scheduleDragStop(args) {
    let kanbanElement = closest(args.event.target, '#Kanban');
    if (kanbanElement) {
      this.scheduleObj.deleteEvent(args.data.Id);
      removeClass([this.scheduleObj.element.querySelector('.e-selected-cell')], 'e-selected-cell');
      this.kanbanObj.openDialog('Add', args.data);
      args.cancel = true;
    }
  }
}

```

```

    }
    render() {
        return <div>
            <div className="container-fluid">
                <div className="row">
                    <div className="col-sm-6">
                        <h4>Kanban</h4>
                        <KanbanComponent id="Kanban" ref={kanban =>
this.kanbanObj = kanban} keyField="DepartmentName" dataSource={this.data}
cardSettings={{ contentField: "Name", headerField: "Id" }}
dragStop={this.onKanbanDragStop.bind(this)}
externalDropId={this.externalKanbanDropId}>
                            <ColumnsDirective>
                                <ColumnDirective headerText="GENERAL"
keyField="GENERAL"/>
                            </ColumnsDirective>
                        </KanbanComponent>
                    </div>
                    <div className="col-sm-6">
                        <h4>Schedule</h4>
                        <ScheduleComponent id="Schedule" ref={schedule =>
this.scheduleObj = schedule} cssClass='schedule-drag-drop' width='100%'
height='650px' selectedDate={new Date(2018, 7, 1)} currentView='TimelineDay'
resourceHeaderTemplate={this.resourceHeaderTemplate.bind(this)}
eventSettings={{
                            dataSource: this.scheduleDataSource,
                            fields: {
                                subject: { title: 'Patient Name', name: 'Name' },
                                startTime: { title: "From", name: "StartTime" },
                                endTime: { title: "To", name: "EndTime" },
                                description: { title: 'Reason', name: 'Description' }
                            }
                        }} group={{ enableCompactView: false, resources: ['Departments',
'Departments', 'Consultants'] }} dragStop={this.scheduleDragStop.bind(this)}>
                            <ResourcesDirective>
                                <ResourceDirective field='DepartmentID'
title='Department' name='Departments' allowMultiple={false}
dataSource={this.departmentData} textField='Text' idField='Id'
colorField='Color'>
                                    </ResourceDirective>
                                <ResourceDirective field='ConsultantID'
title='Consultant' name='Consultants' allowMultiple={false}
dataSource={this.consultantData} textField='Text' idField='Id'
groupIDField="GroupId" colorField='Color'>
                                    </ResourceDirective>
                                </ResourcesDirective>
                            <ViewsDirective>
                                <ViewDirective option='TimelineDay' />
                                <ViewDirective option='TimelineMonth' />
                            </ViewsDirective>
                            <Inject services={[TimelineViews,
TimelineMonth, Resize, DragAndDrop]} />
                        </ScheduleComponent>
                    </div>
                </div>
            </div>
        </div>;
    }
}

```

```

    }
  }
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest, removeClass } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DragEventArgs,
Kanban } from "@syncfusion/ej2-react-kanban";
import {
  ScheduleComponent, ResourcesDirective, ResourceDirective,
  ViewsDirective, ViewDirective, Inject, TimelineViews,
  Resize, DragAndDrop, TimelineMonth
} from '@syncfusion/ej2-react-schedule';
import { kanbanData, scheduleData } from './datasource';
class App extends React.Component<{}, {}>{
  public kanbanObj: Kanban | null;
  public scheduleObj: ScheduleComponent | null;
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
    this.scheduleDataSource = extend([], scheduleData, null, true);
    this.externalKanbanDropId = ["#Schedule"];
  }
  private departmentData: Object[] = [
    { Text: 'GENERAL', Id: 1, Color: '#bbdc00' },
    { Text: 'DENTAL', Id: 2, Color: '#9e5fff' }
  ];
  private consultantData: Object[] = [
    { Text: 'Alice', Id: 1, GroupId: 1, Color: '#bbdc00', Designation:
'Cardiologist' },
    { Text: 'Nancy', Id: 2, GroupId: 2, Color: '#9e5fff', Designation:
'Orthodontist' },
    { Text: 'Robert', Id: 3, GroupId: 1, Color: '#bbdc00', Designation:
'Optometrist' },
    { Text: 'Robson', Id: 4, GroupId: 2, Color: '#9e5fff', Designation:
'Periodontist' },
    { Text: 'Laura', Id: 5, GroupId: 1, Color: '#bbdc00', Designation:
'Orthopedic' },
    { Text: 'Margaret', Id: 6, GroupId: 2, Color: '#9e5fff', Designation:
'Endodontist' }
  ];
  private getConsultantName(value: ResourceDetails | TreeViewArgs): string
{
    return (value as ResourceDetails).resourceData[(value as
ResourceDetails).resource.textField] as string;
  }
  private getConsultantDesignation(value: ResourceDetails): string {
    return (value as ResourceDetails).resourceData.Designation as string;
  }
  private resourceHeaderTemplate(props: any): JSX.Element {

```



```

    return (<div className="template-wrap"><div className="specialist-
category"><div className="specialist-name">
    {this.getConsultantName(props)}</div><div className="specialist-
designation">{this.getConsultantDesignation(props)}</div></div></div>);
    }
    private onKanbanDragStop(args: DragEventArgs): void {
        let scheduleElement: Element = closest(args.event.target as Element,
'#Schedule');
        if (scheduleElement) {
            this.kanbanObj.deleteCard(args.data);
            this.scheduleObj.openEditor(args.data[0], 'Add', true);
            args.cancel = true;
        }
    }
    private scheduleDragStop(args: ScheduleDragEventArgs): void {
        let kanbanElement: Element = closest(args.event.target as Element,
'#Kanban');
        if (kanbanElement) {
            this.scheduleObj.deleteEvent(args.data.Id);
            removeClass([this.scheduleObj.element.querySelector('.e-
selected-cell')], 'e-selected-cell');
            this.kanbanObj.openDialog('Add', args.data);
            args.cancel = true;
        }
    }
    render() {
        return <div>
            <div className="container-fluid">
                <div className="row">
                    <div className="col-sm-6">
                        <h4>Kanban</h4>
                        <KanbanComponent id="Kanban" ref={kanban =>
this.kanbanObj = kanban} keyField="DepartmentName" dataSource={this.data}
cardSettings={{ contentField: "Name", headerField: "Id" }}
dragStop={this.onKanbanDragStop.bind(this)}
externalDropId={this.externalKanbanDropId}>
                            <ColumnsDirective>
                                <ColumnDirective headerText="GENERAL"
keyField="GENERAL" />
                            </ColumnsDirective>
                        </KanbanComponent>
                    </div>
                    <div className="col-sm-6">
                        <h4>Schedule</h4>
                        <ScheduleComponent id="Schedule" ref={schedule =>
this.scheduleObj = schedule} cssClass='schedule-drag-drop' width='100%'
height='650px' selectedDate={new Date(2018, 7, 1)}
currentView='TimelineDay'
resourceHeaderTemplate={this.resourceHeaderTemplate.bind(this)}
eventSettings={{
                            dataSource: this.scheduleDataSource,
                            fields: {
                                subject: { title: 'Patient Name',
name: 'Name' },
                                startTime: { title: "From", name:
"StartTime" },

```

```

                                endTime: { title: "To", name:
"EndTime" },
                                description: { title: 'Reason', name:
'Description' }
                                }
                                }}
                                group={{ enableCompactView: false,
resources: ['Departments', 'Consultants'] }}
dragStop={this.scheduleDragStop.bind(this)} >
                                <ResourcesDirective>
                                <ResourceDirective field='DepartmentID'
title='Department' name='Departments' allowMultiple={false}
                                dataSource={this.departmentData}
textField='Text' idField='Id' colorField='Color'>
                                </ResourceDirective>
                                <ResourceDirective field='ConsultantID'
title='Consultant' name='Consultants' allowMultiple={false}
                                dataSource={this.consultantData}
textField='Text' idField='Id' groupIDField="GroupId" colorField='Color'>
                                </ResourceDirective>
                                </ResourcesDirective>
                                <ViewsDirective>
                                <ViewDirective option='TimelineDay' />
                                <ViewDirective option='TimelineMonth' />
                                </ViewsDirective>
                                <Inject services={[TimelineViews,
TimelineMonth, Resize, DragAndDrop]} />
                                </ScheduleComponent>
                                </div>
                                </div>
                                </div>
                                </div>
                                }
                                };
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban to schedule drag and drop</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban to schedule drag and drop" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
calendars/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-excel-
export/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-file-
utils/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
schedule/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
compression/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .row {
        display: flex;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest, removeClass } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";

```

```

import { ScheduleComponent, ResourcesDirective, ResourceDirective,
ViewsDirective, ViewDirective, Inject, TimelineViews, Resize, DragAndDrop,
TimelineMonth } from '@syncfusion/ej2-react-schedule';
import { kanbanData, scheduleData } from './datasource';
function App() {
    let kanbanObj;
    let scheduleObj;
    let data = extend([], kanbanData, null, true);
    let scheduleDataSource = extend([], scheduleData, null, true);
    let externalKanbanDropId = ["#Schedule"];
    let departmentData = [
        { Text: 'GENERAL', Id: 1, Color: '#bbdc00' },
        { Text: 'DENTAL', Id: 2, Color: '#9e5fff' }
    ];
    let consultantData = [
        { Text: 'Alice', Id: 1, GroupId: 1, Color: '#bbdc00', Designation:
'Cardiologist' },
        { Text: 'Nancy', Id: 2, GroupId: 2, Color: '#9e5fff', Designation:
'Orthodontist' },
        { Text: 'Robert', Id: 3, GroupId: 1, Color: '#bbdc00', Designation:
'Optometrist' },
        { Text: 'Robson', Id: 4, GroupId: 2, Color: '#9e5fff', Designation:
'Periodontist' },
        { Text: 'Laura', Id: 5, GroupId: 1, Color: '#bbdc00', Designation:
'Orthopedic' },
        { Text: 'Margaret', Id: 6, GroupId: 2, Color: '#9e5fff',
Designation: 'Endodontist' }
    ];
    function getConsultantName(value) {
        return value.resourceData[value.resource.textField];
    }
    function getConsultantDesignation(value) {
        return value.resourceData.Designation;
    }
    function resourceHeaderTemplate(props) {
        return (<div className="template-wrap"><div className="specialist-
category"><div className="specialist-name">
{getConsultantName(props)}</div><div className="specialist-
designation">{getConsultantDesignation(props)}</div></div></div>);
    }
    function onKanbanDragStop(args) {
        let scheduleElement = closest(args.event.target, '#Schedule');
        if (scheduleElement) {
            kanbanObj.deleteCard(args.data);
            scheduleObj.openEditor(args.data[0], 'Add', true);
            args.cancel = true;
        }
    }
    function scheduleDragStop(args) {
        let kanbanElement = closest(args.event.target, '#Kanban');
        if (kanbanElement) {
            scheduleObj.deleteEvent(args.data.Id);
            removeClass([scheduleObj.element.querySelector('.e-selected-
cell')], 'e-selected-cell');
            kanbanObj.openDialog('Add', args.data);
            args.cancel = true;
        }
    }
}

```

```

    }
    return (<div>
      <div className="container-fluid">
        <div className="row">
          <div className="col-sm-6">
            <h4>Kanban</h4>
            <KanbanComponent id="Kanban" ref={kanban => kanbanObj =
kanban} keyField="DepartmentName" dataSource={data} cardSettings={{
contentField: "Name", headerField: "Id" }}
dragStop={onKanbanDragStop.bind(this)}
externalDropId={externalKanbanDropId}>
              <ColumnsDirective>
                <ColumnDirective headerText="GENERAL"
keyField="GENERAL"/>
              </ColumnsDirective>
            </KanbanComponent>
          </div>
          <div className="col-sm-6">
            <h4>Schedule</h4>
            <ScheduleComponent id="Schedule" ref={schedule => scheduleObj
= schedule} cssClass='schedule-drag-drop' width='100%' height='650px'
selectedDate={new Date(2018, 7, 1)} currentView='TimelineDay'
resourceHeaderTemplate={resourceHeaderTemplate.bind(this)} eventSettings={{
  dataSource: scheduleDataSource,
  fields: {
    subject: { title: 'Patient Name', name: 'Name' },
    startTime: { title: "From", name: "StartTime" },
    endTime: { title: "To", name: "EndTime" },
    description: { title: 'Reason', name: 'Description' }
  }
}} group={{ enableCompactView: false, resources: ['Departments',
'Departments', 'Consultants'] }} dragStop={scheduleDragStop.bind(this)}>
              <ResourcesDirective>
                <ResourceDirective field='DepartmentID'
title='Department' name='Departments' allowMultiple={false}
dataSource={departmentData} textField='Text' idField='Id'
colorField='Color'>
                </ResourceDirective>
                <ResourceDirective field='ConsultantID'
title='Consultant' name='Consultants' allowMultiple={false}
dataSource={consultantData} textField='Text' idField='Id'
groupIDField="GroupId" colorField='Color'>
                </ResourceDirective>
              </ResourcesDirective>
              <ViewsDirective>
                <ViewDirective option='TimelineDay'/>
                <ViewDirective option='TimelineMonth'/>
              </ViewsDirective>
              <Inject services={[TimelineViews, TimelineMonth,
Resize, DragAndDrop]}/>
            </ScheduleComponent>
          </div>
        </div>
      </div>
    </div>);
  }
  ReactDOM.render(<App />, document.getElementById('kanban'));

```

```
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, closest, removeClass } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DragEventArgs,
Kanban } from "@syncfusion/ej2-react-kanban";
import {
    ScheduleComponent, ResourcesDirective, ResourceDirective,
    ViewsDirective, ViewDirective, Inject, TimelineViews,
    Resize, DragAndDrop, TimelineMonth
} from '@syncfusion/ej2-react-schedule';
import { kanbanData, scheduleData } from '../datasource';
function App() {
    let kanbanObj: Kanban | null;
    let scheduleObj: ScheduleComponent | null;
    let data = extend([], kanbanData, null, true);
    let scheduleDataSource = extend([], scheduleData, null, true);
    let externalKanbanDropId = ["#Schedule"];
    let departmentData: Object[] = [
        { Text: 'GENERAL', Id: 1, Color: '#bbdc00' },
        { Text: 'DENTAL', Id: 2, Color: '#9e5fff' }
    ];
    let consultantData: Object[] = [
        { Text: 'Alice', Id: 1, GroupId: 1, Color: '#bbdc00', Designation:
'Cardiologist' },
        { Text: 'Nancy', Id: 2, GroupId: 2, Color: '#9e5fff', Designation:
'Orthodontist' },
        { Text: 'Robert', Id: 3, GroupId: 1, Color: '#bbdc00', Designation:
'Optometrist' },
        { Text: 'Robson', Id: 4, GroupId: 2, Color: '#9e5fff', Designation:
'Periodontist' },
        { Text: 'Laura', Id: 5, GroupId: 1, Color: '#bbdc00', Designation:
'Orthopedic' },
        { Text: 'Margaret', Id: 6, GroupId: 2, Color: '#9e5fff',
Designation: 'Endodontist' }
    ];
    function getConsultantName(value: ResourceDetails | TreeViewArgs):
string {
        return (value as ResourceDetails).resourceData[(value as
ResourceDetails).resource.textField] as string;
    }
    function getConsultantDesignation(value: ResourceDetails): string {
        return (value as ResourceDetails).resourceData.Designation as string;
    }
    function resourceHeaderTemplate(props: any): JSX.Element {
        return (<div className="template-wrap"><div className="specialist-
category"><div className="specialist-name">
{getConsultantName(props)}</div><div className="specialist-
designation">{getConsultantDesignation(props)}</div></div></div>);
    }
    function onKanbanDragStop(args: DragEventArgs): void {
```

```

    let scheduleElement: Element = closest(args.event.target as Element,
    '#Schedule');
    if (scheduleElement) {
        kanbanObj.deleteCard(args.data);
        scheduleObj.openEditor(args.data[0], 'Add', true);
        args.cancel = true;
    }
}
function scheduleDragStop(args: ScheduleDragEventArgs): void {
    let kanbanElement: Element = closest(args.event.target as Element,
    '#Kanban');
    if (kanbanElement) {
        scheduleObj.deleteEvent(args.data.Id);
        removeClass([scheduleObj.element.querySelector('.e-selected-
cell')], 'e-selected-cell');
        kanbanObj.openDialog('Add', args.data);
        args.cancel = true;
    }
}
return (
    <div>
        <div className="container-fluid">
            <div className="row">
                <div className="col-sm-6">
                    <h4>Kanban</h4>
                    <KanbanComponent id="Kanban" ref={kanban => kanbanObj =
kanban} keyField="DepartmentName" dataSource={data} cardSettings={{
contentField: "Name", headerField: "Id" }}
dragStop={onKanbanDragStop.bind(this)}
externalDropId={externalKanbanDropId}>
                        <ColumnsDirective>
                            <ColumnDirective headerText="GENERAL"
keyField="GENERAL" />
                        </ColumnsDirective>
                    </KanbanComponent>
                </div>
                <div className="col-sm-6">
                    <h4>Schedule</h4>
                    <ScheduleComponent id="Schedule" ref={schedule => scheduleObj
= schedule} cssClass='schedule-drag-drop' width='100%' height='650px'
selectedDate={new Date(2018, 7, 1)}
currentView='TimelineDay'
resourceHeaderTemplate={resourceHeaderTemplate.bind(this)}
eventSettings={{
                    dataSource: scheduleDataSource,
                    fields: {
                        subject: { title: 'Patient Name', name: 'Name' },
                        startTime: { title: "From", name: "StartTime" },
                        endTime: { title: "To", name: "EndTime" },
                        description: { title: 'Reason', name:
'Description' }
                    }
                }}
                    group={{ enableCompactView: false, resources:
['Departments', 'Consultants'] }}
                    dragStop={scheduleDragStop.bind(this)} >
                        <ResourcesDirective>

```

```

        <ResourceDirective field='DepartmentID'
title='Department' name='Departments' allowMultiple={false}
        dataSource={departmentData} textField='Text'
idField='Id' colorField='Color'>
        </ResourceDirective>
        <ResourceDirective field='ConsultantID'
title='Consultant' name='Consultants' allowMultiple={false}
        dataSource={consultantData} textField='Text'
idField='Id' groupIDField="GroupId" colorField='Color'>
        </ResourceDirective>
    </ResourcesDirective>
    <ViewsDirective>
        <ViewDirective option='TimelineDay' />
        <ViewDirective option='TimelineMonth' />
    </ViewsDirective>
    <Inject services={[TimelineViews, TimelineMonth,
Resize, DragAndDrop]} />
    </ScheduleComponent>
</div>
</div>
</div>
</div>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban to schedule drag and drop</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban to schedule drag and drop" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
calendars/styles/material.css" rel="stylesheet" />

```



```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-excel-
export/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-file-
utils/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
schedule/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
compression/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .row {
        display: flex;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Sort in React Kanban component

The Kanban provides built-in support to arrange the cards in their columns based on the JSON data order and drop the cards in the columns based on the dropped clone. Initially, users can change the arrangement of cards in the columns and position of the dropped card by using the [sortBy](#) property. The [sortBy](#) property contains three enumeration values as follows.

- Index
- DataSourceOrder
- Custom

Index

SortBy **Index** property can be used with or without [field](#) mapping.

Index without field mapping

By default, SortBy **Index** property support without any [field](#) mapping. In this behavior, cards are loaded based on the JSON data order and cards are dropped based on the dropped clone.

[Class-component]**INDEX.JSX**

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
    </ColumnsDirective>
  </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

```

        <ColumnDirective headerText="Testing" keyField="Testing"
    />
        <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>
</KanbanComponent>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Single Key Columns</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban single key columns" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>

```

```

        <div id='loader'>Loading....</div>
      </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
  let data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ contentField: "Summary", headerField: "Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress" keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
  let data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ contentField: "Summary", headerField: "Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Single Key Columns</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban single key columns" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Index with field mapping

SortBy Index property also supports with [field](#) mapping. In this behavior, cards are loaded based on mapping field values, and cards are dropped based on the dropped clone.

Cards are placed in a particular position in the columns where you can drop the cards by specifying the [field](#) property, which is mapped from the data source. This property allows the users to drop the cards in the Kanban board where the dropped clone is created exactly. It is also helpful to render the cards based on the [field](#) property value.

The [field](#) property mapping key value must be in **number** format.

The following cases will dynamically change their [field](#) value when dropping the cards.

- If the cell has no cards, the dropped card [field](#) value does not change.
- If the cell has one card and dropped a card to the last position or previous/next cards that do not have continuous order, then the dropped card [field](#) value will be changed based on their previous card value.
- If the cell has one card and dropped a card on the previous position, then it will compare both the values, and the dropped card [field](#) value will be changed if the cards have continuous order otherwise values will not be changed.
- When the previous and next cards do not have continuous order, the dropped card [field](#) value will be changed based on the previous card value.
- When the previous and next cards have continuous order or odd/even value, then the [field](#) value of the dropped card and the cards followed by the dropped card will be changed based on the **previous** card value with continuous order.

For Example,

Continuous Order -

Consider, Column A has Card A with priority value **1**, Card B with priority value **2**, and Card C with priority value **3**.

and Column B has Card D with priority value **5**, then the dropped Card D will be placed between Card A and Card B. Now, the Cards D, B, and C will be dynamically changed to the priority values as **2, 3, and 4** respectively.

Odd/Even order -

Consider, Column A has Card A with priority value **1**, Card B with priority value **3**, and Card C with priority value **5**.

and Column B has Card D with priority value **5**, then the Dropped Card D will be placed between Card A and Card B. Now, the Cards D, B, and C will be dynamically changed to the priority values as **2, 3, and 5** respectively.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
```

```

    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} sortSettings={{ sortBy: "Index", field: "RankId" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}>, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} sortSettings={{ sortBy: "Index", field: "RankId" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Single Key Columns</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban single key columns" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';

```



```

import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
sortSettings={{ sortBy: "Index", field: "RankId" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
sortSettings={{ sortBy: "Index", field: "RankId" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress"
/>
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Single Key Columns</title>
    <meta charset="utf-8" />

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Kanban single key columns" />
<meta name="author" content="Syncfusion" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

DataSource Order

The SortBy `DataSourceOrder` property does not require any [field](#) mapping. In this behavior, cards are loaded based on the JSON data order, and also cards are dropped based on the JSON data order.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';

```

```

import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} sortSettings={{ sortBy: "DataSourceOrder" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} sortSettings={{ sortBy: "DataSourceOrder" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));

```

```
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Single Key Columns</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban single key columns" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

[Functional-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
sortSettings={{ sortBy: "DataSourceOrder" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
sortSettings={{ sortBy: "DataSourceOrder" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress"
/>
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Single Key Columns</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban single key columns" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Custom

Custom with field mapping

The SortBy **Custom** property must require datasource [field](#) mapping. In this behavior, cards are loaded based on the [field](#) mapping value and also cards are dropped based on the [field](#) mapping value.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} sortSettings={{ sortBy: "Custom", field: "Summary" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} sortSettings={{ sortBy: "Custom", field: "Summary" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

```

        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Single Key Columns</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban single key columns" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></script>
  <script src="systemjs.config.js"></script>
  <style>
    #loader {
      color: #008c8f;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>

```



```
</body>
</html>
```

[Functional-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
sortSettings={{ sortBy: "Custom", field: "Summary" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
sortSettings={{ sortBy: "Custom", field: "Summary" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Single Key Columns</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban single key columns" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

Change the direction

Kanban board also provides support for aligning the cards in the columns using the [direction](#) property inside the [sortSettings](#) property. Based on this, cards can be aligned in the columns either in **Ascending** or **Descending** order. Sorting direction will be performed based on [sortBy](#) property.

By default, cards are aligned in the columns based on **Ascending** order.

In the following sample, cards are aligned in **Descending** order.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} sortSettings={{ sortBy: "Custom", field: "Summary", direction:
"Descending" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
  }
}
```

```

        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
        "Id" }} sortSettings={{ sortBy: "Custom", field: "Summary", direction:
        "Descending" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
                keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
                />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Single Key Columns</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban single key columns" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    kanban/styles/material.css" rel="stylesheet" />
    <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {

```

```

        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
        cardSettings={{ contentField: "Summary", headerField: "Id" }}
        sortSettings={{ sortBy: "Custom", field: "Summary", direction: "Descending"
        }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress" keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (

```

```

    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ contentField: "Summary", headerField: "Id" }}
    sortSettings={{ sortBy: "Custom", field: "Summary", direction: "Descending"
    }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" />
            <ColumnDirective headerText="In Progress" keyField="InProgress" />
            <ColumnDirective headerText="Testing" keyField="Testing" />
            <ColumnDirective headerText="Done" keyField="Close" />
        </ColumnsDirective>
    </KanbanComponent>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Single Key Columns</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban single key columns" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdn.jsdelivr.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
    <script src="systemjs.config.js"></script>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;

```

```

        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Dialog in React Kanban component

The Kanban provides built-in support to add, edit and delete a card using dialog module. User can edit a card using the following ways.

- Built-in dialog module
- Custom Fields
- Dialog template

Default Dialog

When double-click on the cards, the dialog is opened with below fields to edit a card. This dialog contains **Delete**, **Save** and **Cancel** buttons.

- To edit a card, modify the card details and click the **Save** button.
- To delete a card, click **Delete** button.
- Click on the **Cancel** button to cancel the editing action.

The dialog displays with the following fields which mapped to dialog fields by default.

Key | Type | Text

cardSettings.headerField | Input | ID

keyField | DropDown | -

cardSettings.contentField | TextArea | -

cardSettings.priority(If applicable) | Numeric | -

swimlaneSettings.keyField(If applicable) | DropDown | -

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {

```

```

        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogFieldsModel } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```
<!DOCTYPE html>
```



```

<html lang="en">
<head>
  <title>Kanban Cards</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Cards" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';

```

```
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogFieldsModel } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

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

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Custom Fields

You can change the default fields of dialog using `fields` property inside the `dialogSettings` property. The `key` property used to map the `dataSource` value and rendered the corresponding component based on specified `type` property.

The following types are available in dialog fields.

- String
- Numeric
- TextArea
- DropDown
- TextBox
- Input

If `type` is not defined in the fields, then it renders as the HTML input element in dialog.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  fields = [
    { key: 'Id', type: 'Input' },
    { key: 'Status', type: 'DropDown' },
    { key: 'Estimate', type: 'Numeric' },
    { key: 'Summary', type: 'TextArea' }
  ];
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} dialogSettings={{ fields: this.fields }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogFieldsModel } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  private fields: DialogFieldsModel[] = [
```

```

        { key: 'Id', type: 'Input' },
        { key: 'Status', type: 'DropDown' },
        { key: 'Estimate', type: 'Numeric' },
        { key: 'Summary', type: 'TextArea' }
    ];
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
        "Id" }} dialogSettings={{ fields: this.fields }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
        keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
        />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Custom Label Dialog</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Custom Label Dialog" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>

```

```

<style>
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
</style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
  let data = extend([], kanbanData, null, true);
  let fields = [
    { key: 'Id', type: 'Input' },
    { key: 'Status', type: 'DropDown' },
    { key: 'Estimate', type: 'Numeric' },
    { key: 'Summary', type: 'TextArea' }
  ];
  return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dialogSettings={{ fields: fields }}>
    <ColumnsDirective>
      <ColumnDirective headerText="To Do" keyField="Open"/>
      <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
      <ColumnDirective headerText="Testing" keyField="Testing"/>
      <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
  </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';

```

```

import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogFieldsModel } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    let fields: DialogFieldsModel[] = [
        { key: 'Id', type: 'Input' },
        { key: 'Status', type: 'DropDown' },
        { key: 'Estimate', type: 'Numeric' },
        { key: 'Summary', type: 'TextArea' }
    ];
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dialogSettings={{ fields: fields }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Custom Label Dialog</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Custom Label Dialog" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />

```

```

    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Custom Fields label

By default, the fields **key** mapping value is considered as a **label** and you can change this label by using **text** property.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    fields = [
        { text: 'ID', key: 'Id', type: 'TextBox' },
        { key: 'Status', type: 'DropDown' },
        { key: 'Estimate', type: 'Numeric' },
        { key: 'Summary', type: 'TextArea' }
    ];
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} dialogSettings={{ fields: this.fields }}>
            <ColumnsDirective>

```



```

        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogFieldsModel } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    private fields: DialogFieldsModel[] = [
        { text: 'ID', key: 'Id', type: 'TextBox' },
        { key: 'Status', type: 'DropDown' },
        { key: 'Estimate', type: 'Numeric' },
        { key: 'Summary', type: 'TextArea' }
    ];
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} dialogSettings={{ fields: this.fields }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

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

```

```

<title>Kanban Label Dialog</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Kanban Label Dialog" />
<meta name="author" content="Syncfusion" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";

```

```

import { kanbanData } from './datasource';
function App() {
  let data = extend([], kanbanData, null, true);
  let fields = [
    { text: 'ID', key: 'Id', type: 'TextBox' },
    { key: 'Status', type: 'DropDown' },
    { key: 'Estimate', type: 'Numeric' },
    { key: 'Summary', type: 'TextArea' }
  ];
  return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dialogSettings={{ fields: fields }}>
    <ColumnsDirective>
      <ColumnDirective headerText="To Do" keyField="Open"/>
      <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
      <ColumnDirective headerText="Testing" keyField="Testing"/>
      <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
  </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogFieldsModel } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
  let data = extend([], kanbanData, null, true);
  let fields: DialogFieldsModel[] = [
    { text: 'ID', key: 'Id', type: 'TextBox' },
    { key: 'Status', type: 'DropDown' },
    { key: 'Estimate', type: 'Numeric' },
    { key: 'Summary', type: 'TextArea' }
  ];
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dialogSettings={{ fields: fields }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));

```

```
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Label Dialog</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Label Dialog" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

Fields Validation

The dialog fields can be validated while click on the **Save** button. This can be achieved by using **validationRules** property.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  fields = [
    { text: 'ID', key: 'Id', type: 'Input' },
    { key: 'Status', type: 'DropDown' },
    { key: 'Estimate', type: 'Numeric', validationRules: { range: [0,
1000] } },
    { key: 'Summary', type: 'TextArea', validationRules: { required:
true } }
  ];
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} dialogSettings={{ fields: this.fields }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogFieldsModel } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
```

```

constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
}

private fields: DialogFieldsModel[] = [
    { text: 'ID', key: 'Id', type: 'Input' },
    { key: 'Status', type: 'DropDown' },
    { key: 'Estimate', type: 'Numeric', validationRules: { range: [0,
1000] } },
    { key: 'Summary', type: 'TextArea', validationRules: { required:
true } }
];

render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} dialogSettings={{ fields: this.fields }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" />
            <ColumnDirective headerText="In Progress"
keyField="InProgress" />
            <ColumnDirective headerText="Testing" keyField="Testing"
/>
            <ColumnDirective headerText="Done" keyField="Close" />
        </ColumnsDirective>
    </KanbanComponent>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Dialog Fields Validation</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Dialog Fields Validation" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />

```

```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    let fields = [
        { text: 'ID', key: 'Id', type: 'Input' },
        { key: 'Status', type: 'DropDown' },
        { key: 'Estimate', type: 'Numeric', validationRules: { range: [0,
1000] } },
        { key: 'Summary', type: 'TextArea', validationRules: { required:
true } }
    ];
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dialogSettings={{ fields: fields }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </div>

```

```

        </KanbanComponent>);
    }
    ReactDOM.render(<App />, document.getElementById('kanban'));
    {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective,
DialogFieldsModel } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    let fields: DialogFieldsModel[] = [
        { text: 'ID', key: 'Id', type: 'Input' },
        { key: 'Status', type: 'DropDown' },
        { key: 'Estimate', type: 'Numeric', validationRules: { range: [0,
1000] } },
        { key: 'Summary', type: 'TextArea', validationRules: { required:
true } }
    ];
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dialogSettings={{ fields: fields }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Dialog Fields Validation</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Dialog Fields Validation" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />

```



```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Dialog Template

Using the dialog template, you can render your own dialog by defining the `template` property. Initialize the template as SCRIPT element Id or HTML string which holds the template and map it to the template property.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {
        super(...arguments);
    }

```

```

        this.data = extend([], kanbanData, null, true);
    }
    dialogTemplate(props) {
        return (<KanbanDialogFormTemplate {...props}/>);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
        "Id" }} dialogSettings={{ template: this.dialogTemplate }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
export class KanbanDialogFormTemplate extends React.Component {
    assigneeData = [
        'Nancy Davloio', 'Andrew Fuller', 'Janet Leverling',
        'Steven walker', 'Robert King', 'Margaret hamilt', 'Michael Suyama'
    ];
    statusData = ['Open', 'InProgress', 'Testing', 'Close'];
    priorityData = ['Low', 'Normal', 'Critical', 'Release Breaker', 'High'];
    tagsHtmlAttributes = { name: "Tags" };
    constructor(props) {
        super(props);
        this.state = extend({}, {}, props, true);
    }
    onChange(args) {
        let key = args.target.name;
        let value = args.target.value;
        this.setState({ [key]: value });
    }
    render() {
        let data = this.state;
        return (<div>
            <table>
                <tbody>
                    <tr>
                        <td className="e-label">ID</td>
                        <td>
                            <div className="e-float-input e-control-
wrapper">
                                <input id="Id" name="Id" type="text"
className="e-field" value={data.Id} disabled/>
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td className="e-label">Status</td>
                        <td>

```

```

                                <DropDownListComponent id='Status' name="Status"
dataSource={this.statusData} className="e-field" placeholder='Status'
value={data.Status}></DropDownListComponent>
                                </td>
                                </tr>
                                <tr>
                                <td className="e-label">Assignee</td>
                                <td>
                                <DropDownListComponent id='Assignee'
name="Assignee" className="e-field" dataSource={this.assigneeData}
placeholder='Assignee' value={data.Assignee}></DropDownListComponent>
                                </td>
                                </tr>
                                <tr>
                                <td className="e-label">Priority</td>
                                <td>
                                <DropDownListComponent type="text"
name="Priority" id="Priority" popupHeight='300px' className="e-field"
value={data.Priority} dataSource={this.priorityData}
placeholder='Priority'></DropDownListComponent>
                                </td>
                                </tr>
                                <tr>
                                <td className="e-label">Summary</td>
                                <td>
                                <div className="e-float-input e-control-
wrapper">
                                <textarea name="Summary" className="e-field"
value={data.Summary} onChange={this.onChange.bind(this)}></textarea>
                                </div>
                                </td>
                                </tr>
                                </tbody>
                                </table>
                                </div>);
        }
    }
    {% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    private dialogTemplate(props: KanbanDataModel): JSX.Element {
        return (<KanbanDialogFormTemplate {...props} />);
    }
}

```

```

render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} dialogSettings={{ template: this.dialogTemplate }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" />
            <ColumnDirective headerText="In Progress"
keyField="InProgress" />
            <ColumnDirective headerText="Testing" keyField="Testing"
/>
            <ColumnDirective headerText="Done" keyField="Close" />
        </ColumnsDirective>
    </KanbanComponent>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
export class KanbanDialogFormTemplate extends React.Component<{}, {}> {
    public assigneeData: string[] = [
        'Nancy Davloio', 'Andrew Fuller', 'Janet Leverling',
        'Steven walker', 'Robert King', 'Margaret hamilt', 'Michael Suyama'
    ];
    public statusData: string[] = ['Open', 'InProgress', 'Testing',
'Close'];
    public priorityData: string[] = ['Low', 'Normal', 'Critical', 'Release
Breaker', 'High'];
    public tagsHtmlAttributes = { name: "Tags" };
    constructor(props) {
        super(props);
        this.state = extend({}, {}, props, true);
    }
    onChange(args: any): void {
        let key: string = args.target.name;
        let value: string = args.target.value;
        this.setState({ [key]: value });
    }
    render(): any {
        let data: KanbanDataModel = this.state;
        return (<div>
            <table>
                <tbody>
                    <tr>
                        <td className="e-label">ID</td>
                        <td>
                            <div className="e-float-input e-control-
wrapper">
                                <input id="Id" name="Id" type="text"
className="e-field" value={data.Id} disabled />
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td className="e-label">Status</td>
                        <td>
                            <DropDownListComponent id='Status' name="Status"
dataSource={this.statusData} className="e-field" placeholder='Status'
value={data.Status}></DropDownListComponent>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
    );
}

```

```

        </tr>
        <tr>
            <td className="e-label">Assignee</td>
            <td>
                <DropDownListComponent id='Assignee'
name="Assignee" className="e-field" dataSource={this.assigneeData}
placeholder='Assignee' value={data.Assignee}></DropDownListComponent>
            </td>
        </tr>
        <tr>
            <td className="e-label">Priority</td>
            <td>
                <DropDownListComponent type="text"
name="Priority" id="Priority" popupHeight='300px' className="e-field"
value={data.Priority} dataSource={this.priorityData}
placeholder='Priority'></DropDownListComponent>
            </td>
        </tr>
        <tr>
            <td className="e-label">Summary</td>
            <td>
                <div className="e-float-input e-control-
wrapper">
                    <textarea name="Summary" className="e-field"
value={data.Summary} onChange={this.onChange.bind(this)}></textarea>
                </div>
            </td>
        </tr>
    </tbody>
</table>
</div>);
    }
}
export interface KanbanDataModel {
    Id?: string;
    Title?: string;
    Status?: string;
    Summary?: string;
    Type?: string;
    Priority?: string;
    Tags?: string;
    Estimate?: number;
    Assignee?: string;
    RankId?: number;
    Color?: string;
}
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Dialog Template</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

```

```

<meta name="description" content="Kanban Dialog Template" />
<meta name="author" content="Syncfusion" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
import { DropDownListComponent } from "@syncfusion/ej2-react-dropdowns";
function App() {

```

```

let data = extend([], kanbanData, null, true);
let kanbanObj;
function dialogTemplate(props) {
  return (<KanbanDialogFormTemplate {...props}/>);
}
return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dialogSettings={{ template: dialogTemplate.bind(this) }}>
  <ColumnsDirective>
    <ColumnDirective headerText="To Do" keyField="Open"/>
    <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
    <ColumnDirective headerText="Testing" keyField="Testing"/>
    <ColumnDirective headerText="Done" keyField="Close"/>
  </ColumnsDirective>
</KanbanComponent>);
}
function KanbanDialogFormTemplate(props) {
  let assigneeData = [
    'Nancy Davloio', 'Andrew Fuller', 'Janet Leverling',
    'Steven walker', 'Robert King', 'Margaret hamilt', 'Michael Suyama'
  ];
  let statusData = ['Open', 'InProgress', 'Testing', 'Close'];
  let priorityData = ['Low', 'Normal', 'Critical', 'Release Breaker',
'High'];
  let tagsHtmlAttributes = { name: "Tags" };
  const [state, setState] = React.useState(extend({}, {}, props, true));
  function onChange(args) {
    let key = args.target.name;
    let value = args.target.value;
    setState({ [key]: value });
  }
  let data = state;
  return (<div>
    <table>
      <tbody>
        <tr>
          <td className="e-label">ID</td>
          <td>
            <div className="e-float-input e-control-wrapper">
              <input id="Id" name="Id" type="text"
className="e-field" value={data.Id} disabled/>
            </div>
          </td>
        </tr>
        <tr>
          <td className="e-label">Status</td>
          <td>
            <DropDownListComponent id='Status' name="Status"
dataSource={statusData} className="e-field" placeholder='Status'
value={data.Status}></DropDownListComponent>
          </td>
        </tr>
        <tr>
          <td className="e-label">Assignee</td>
          <td>

```

```

        <DropDownListComponent id='Assignee'
name="Assignee" className="e-field" dataSource={assigneeData}
placeholder='Assignee' value={data.Assignee}></DropDownListComponent>
    </td>
</tr>
<tr>
    <td className="e-label">Priority</td>
    <td>
        <DropDownListComponent type="text" name="Priority"
id="Priority" popupHeight='300px' className="e-field" value={data.Priority}
dataSource={priorityData} placeholder='Priority'></DropDownListComponent>
    </td>
</tr>
<tr>
    <td className="e-label">Summary</td>
    <td>
        <div className="e-float-input e-control-wrapper">
            <textarea name="Summary" className="e-field"
value={data.Summary} onChange={onChange.bind(this)}></textarea>
        </div>
    </td>
</tr>
</tbody>
</table>
</div>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let data = extend([], kanbanData, null, true);
    let kanbanObj: KanbanComponent;
    function dialogTemplate(props: KanbanDataModel) {
        return (<KanbanDialogFormTemplate {...props} />);
    }
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
        cardSettings={{ contentField: "Summary", headerField: "Id" }}
        dialogSettings={{ template: dialogTemplate.bind(this) }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" />
            <ColumnDirective headerText="In Progress" keyField="InProgress"
/>
            <ColumnDirective headerText="Testing" keyField="Testing" />
            <ColumnDirective headerText="Done" keyField="Close" />
        </ColumnsDirective>
    </KanbanComponent>

```



```

    );
  }
function KanbanDialogFormTemplate (props) {
  let assigneeData: string[] = [
    'Nancy Davloio', 'Andrew Fuller', 'Janet Leverling',
    'Steven walker', 'Robert King', 'Margaret hamilt', 'Michael Suyama'
  ];
  let statusData: string[] = ['Open', 'InProgress', 'Testing', 'Close'];
  let priorityData: string[] = ['Low', 'Normal', 'Critical', 'Release
Breaker', 'High'];
  let tagsHtmlAttributes = { name: "Tags" };
  const [state, setState] = React.useState(extend({}, {}, props, true));

  function onChange(args: any): void {
    let key: string = args.target.name;
    let value: string = args.target.value;
    setState({ [key]: value });
  }
  let data: KanbanDataModel = state;
  return (
    <div>
      <table>
        <tbody>
          <tr>
            <td className="e-label">ID</td>
            <td>
              <div className="e-float-input e-control-wrapper">
                <input id="Id" name="Id" type="text"
className="e-field" value={data.Id} disabled />
              </div>
            </td>
          </tr>
          <tr>
            <td className="e-label">Status</td>
            <td>
              <DropDownListComponent id='Status' name="Status"
dataSource={statusData} className="e-field" placeholder='Status'
value={data.Status}></DropDownListComponent>
            </td>
          </tr>
          <tr>
            <td className="e-label">Assignee</td>
            <td>
              <DropDownListComponent id='Assignee'
name="Assignee" className="e-field" dataSource={assigneeData}
placeholder='Assignee' value={data.Assignee}></DropDownListComponent>
            </td>
          </tr>
          <tr>
            <td className="e-label">Priority</td>
            <td>
              <DropDownListComponent type="text" name="Priority"
id="Priority" popupHeight='300px' className="e-field" value={data.Priority}
dataSource={priorityData} placeholder='Priority'></DropDownListComponent>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  );
}

```

```

        <td className="e-label">Summary</td>
        <td>
            <div className="e-float-input e-control-wrapper">
                <textarea name="Summary" className="e-field"
value={data.Summary} onChange={onChange.bind(this)}></textarea>
            </div>
        </td>
    </tr>
</tbody>
</table>
</div>
);
}
export interface KanbanDataModel {
    Id?: string;
    Title?: string;
    Status?: string;
    Summary?: string;
    Type?: string;
    Priority?: string;
    Tags?: string;
    Estimate?: number;
    Assignee?: string;
    RankId?: number;
    Color?: string;
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Dialog Template</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Dialog Template" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />

```

```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Prevent Dialog

The Kanban allows to prevent to open a dialog on card double-click by enabling `args.cancel` in `dialogOpen` event.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    DialogOpen(args) {
        args.cancel = true;
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} dialogOpen={this.DialogOpen.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>

```

```

        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    private DialogOpen(args: DialogEventArgs): void {
        args.cancel = true;
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
        "Id" }} dialogOpen={this.DialogOpen.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Prevent Dialog Open</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Prevent Dialog Open" />
    <meta name="author" content="Syncfusion" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function DialogOpen(args) {
        args.cancel = true;
    }

```

```

    }
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dialogOpen={DialogOpen.bind(this)}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective, DialogEventArgs } from "@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function DialogOpen(args: DialogEventArgs): void {
        args.cancel = true;
    }
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dialogOpen={DialogOpen.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Prevent Dialog Open</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Prevent Dialog Open" />

```

```

    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008c00;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Persisting data in server

The modified card data can be persisted in the database using the RESTful web services. All the CRUD operations in the Kanban 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 below section covers how to get the edited data details on the server-side using the [UrlAdaptor](#).

URL adaptor

You can use the [UrlAdaptor](#) of [DataManager](#) when binding data source for remote data. In the initial load of Kanban, data are fetched from remote data and bound to the Kanban using `url` property of [DataManager](#).

You can map the CRUD operation in Kanban can be mapped to server-side controller actions using the properties `insertUrl`, `removeUrl`, `updateUrl`, and `crudUrl`.

- `insertUrl` – You can perform single insertion operation on server-side.
- `updateUrl` – You can update single data on server-side.
- `removeUrl` – You can remove single data on server-side.
- `crudUrl` – You can perform bulk data operation on server-side.

The following code example describes the above behavior.

[Class-component]

```
{% raw %}  
`ts  
import * as React from 'react';  
import * as ReactDOM from 'react-dom';  
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';  
import { KanbanComponent, ColumnsDirective, ColumnDirective } from "@syncfusion/ej2-react-kanban";  
  
class App extends React.Component<{}, {}>{  
  public data = new DataManager({  
    url: 'Home/DataSource',  
    updateUrl: 'Home/Update',  
    insertUrl: 'Home/Insert',  
    removeUrl: 'Home/Delete',  
    adaptor: new UrlAdaptor(),  
    crossDomain: true  
  });  
  
  render() {  
    return <KanbanComponent id="kanban" keyField="Status" dataSource={this.data} cardSettings={{  
      contentField: "Summary", headerField: "Id" }}>  
      <ColumnsDirective>  
        <ColumnDirective headerText="To Do" keyField="Open" />  
        <ColumnDirective headerText="In Progress" keyField="InProgress" />  
        <ColumnDirective headerText="Testing" keyField="Testing" />  
        <ColumnDirective headerText="Done" keyField="Close" />  
      </ColumnsDirective>  
    </KanbanComponent>  
  }  
}
```



```

}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
`
{% endraw %}
[Functional-component]
{% raw %}
`ts
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from "@syncfusion/ej2-react-kanban";
function App(){
let data = new DataManager({
url: 'Home/DataSource',
updateUrl: 'Home/Update',
insertUrl: 'Home/Insert',
removeUrl: 'Home/Delete',
adaptor: new UrlAdaptor(),
crossDomain: true
});
return (
<KanbanComponent id="kanban" keyField="Status" dataSource={data} cardSettings={{ contentField:
"Summary", headerField: "Id" }}>
<ColumnsDirective>
<ColumnDirective headerText="To Do" keyField="Open" />
<ColumnDirective headerText="In Progress" keyField="InProgress" />
<ColumnDirective headerText="Testing" keyField="Testing" />
<ColumnDirective headerText="Done" keyField="Close" />
</ColumnsDirective>
</KanbanComponent>
);
}

```

```
ReactDOM.render(<App />, document.getElementById('kanban'));
```

```
,
```

```
{% endraw %}
```

The server-side controller code to handle the CRUD operations are as follows.

```
`ts
```

```
private NORTHWNDEntities db = new NORTHWNDEntities();
```

```
public ActionResult DataSource() {
```

```
var DataSource = db.Tasks.ToList();
```

```
return Json(DataSource, JsonRequestBehavior.AllowGet);
```

```
}
```

```
public ActionResult Insert(Params value) {
```

```
//Insert card data into the database
```

```
return Json(value, JsonRequestBehavior.AllowGet);
```

```
}
```

```
public ActionResult Update(Params value) {
```

```
//Update card data into the database
```

```
return Json(value, JsonRequestBehavior.AllowGet);
```

```
}
```

```
public void Delete(Params value) {
```

```
//Delete card data from the database
```

```
}
```

```
public class Params {
```

```
public int Id { get; set; }
```

```
public string Status { get; set; }
```

```
public string Summary { get; set; }
```

```
public string Assignee { get; set; }
```

```
}
```

```
,
```

Insert card

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.

```
`ts
```

```
public ActionResult Insert(Params value)
```

```
{
//Insert card in the database
}
```

The newly added record details are bound to the `value` parameter.

Update card

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 above behavior.

```
`ts
public ActionResult Update(Params value)
{
//Update card data in the database
}
```

The updated record details are bound to the `value` parameter.

Delete card

Using the `removeUrl` property, the controller action mapping URL can be specified to perform delete operation on the server-side.

The following code example describes the above behavior.

```
`ts
public void Delete(int key)
{
//Delete card in the database
}
```

The deleted card primary key value is bound to the `key` parameter.

CRUD URL

Using the `crudUrl` property, the controller action mapping URL can be specified to perform all the CRUD operations at the server-side using a single method instead of specifying a separate controller action method for CRUD (insert, update and delete) operations.

The action parameter of `crudUrl` is used to get the corresponding CRUD action.

The following code example describes the above behavior.

```
[Class-component]
```

```
{% raw %}
```

```

`ts
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from "@syncfusion/ej2-react-kanban";

class App extends React.Component<{}, {}>{
  public data = new DataManager({
    url: 'Home/DataSource',
    updateUrl: 'Home/UpdateData',
    insertUrl: 'Home/UpdateData',
    removeUrl: 'Home/UpdateData',
    crudUrl: 'Home/UpdateData',
    adaptor: new UrlAdaptor(),
    crossDomain: true
  });
  render() {
    return <KanbanComponent id="kanban" keyField="Status" dataSource={this.data} cardSettings={{
      contentField: "Summary", headerField: "Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
`
`

{% endraw %}
[Functional-component]
{% raw %}
`ts

```

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from "@syncfusion/ej2-react-kanban";

function App(){
  let data = new DataManager({
    url: 'Home/DataSource',
    updateUrl: 'Home/UpdateData',
    insertUrl: 'Home/UpdateData',
    removeUrl: 'Home/UpdateData',
    crudUrl: 'Home/UpdateData',
    adaptor: new UrlAdaptor(),
    crossDomain: true
  });
  return(
    <KanbanComponent id="kanban" keyField="Status" dataSource={data} cardSettings={{ contentField:
    "Summary", headerField: "Id" }}>
    <ColumnsDirective>
    <ColumnDirective headerText="To Do" keyField="Open" />
    <ColumnDirective headerText="In Progress" keyField="InProgress" />
    <ColumnDirective headerText="Testing" keyField="Testing" />
    <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>
    </KanbanComponent>
  );
}

ReactDOM.render(<App />, document.getElementById('kanban'));
`ts

private NORTHWNDEntities db = new NORTHWNDEntities();
public ActionResult DataSource() {
  var DataSource = db.Tasks.ToList();

```

```
return Json(DataSource, JsonRequestBehavior.AllowGet);
}

public ActionResult UpdateData(EditParams param) {
    if (param.action == "insert" || (param.action == "batch" && param.added != null)) {
        if (param.action == "insert") {
            db.Tasks.Add(param.value);
        } else {
            foreach (var temp in param.added) {
                db.Tasks.Add(temp);
            }
        }
    }

    if (param.action == "update" || (param.action == "batch" && param.changed != null)) {
        if (param.action == "update") {
            Task old = db.Tasks.Where(o => o.Id == param.value.Id).SingleOrDefault();
            if (old != null) {
                db.Entry(old).CurrentValues.SetValues(param.value);
            }
        } else {
            foreach (var temp in param.changed) {
                Task old = db.Tasks.Where(o => o.Id == temp.Id).SingleOrDefault();
                if (old != null) {
                    db.Entry(old).CurrentValues.SetValues(temp);
                }
            }
        }
    }

    if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) {
        if (param.action == "remove") {
            int key = Convert.ToInt32(param.key);
            db.Tasks.Remove(db.Tasks.Where(o => o.Id == key).SingleOrDefault());
        } else {
            foreach (var temp in param.deleted) {
```

```

db.Tasks.Remove(db.Tasks.Where(o => o.Id == temp.Id).SingleOrDefault());
}
}
}
db.SaveChanges();
return Json(param, JsonRequestBehavior.AllowGet);
}
public class EditParams {
public string key { get; set; }
public string action { get; set; }
public List<Tasks> added { get; set; }
public List<Tasks> changed { get; set; }
public List<Tasks> deleted { get; set; }
public Tasks value { get; set; }
}
,

```

The `crudUrl` is used to update the bulk data sent to the server-side. Multiple selections and `sortBy` as `Index` properties are used for `crudUrl` properties to update the modified bulk data to the server-side.

Tooltip in React Kanban component

The tooltip is used to show the card information when the cursor hover over the card elements using the `enableTooltip` property. Tooltip content is dynamically set based on hovering over the card elements.

If you wish to show tooltip on Kanban board custom elements, you need to add `e-tooltip-text` class name of a particular element.

Tooltip template

You can customize the tooltip content with any HTML or CSS element and styling using the `tooltipTemplate` property. In the following demo, the tooltip is customized with HTML elements.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {

```

```

        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    template(props) {
        return (<div className="e-kanbantooltiptemp">
            <table>
                <tr>
                    <td className="e-kanban-card-details">
                        <table>
                            <colgroup>
                                <col style={{ width: "30%" }}/>
                                <col style={{ width: "70%" }}/>
                            </colgroup>
                            <tbody><tr><td
                                className="CardHeader">Assignee:</td><td>{props.Assignee}</td></tr>
                                <tr><td
                                className="CardHeader">Type:</td><td>{props.Type}</td></tr><tr>
                                <td
                                className="CardHeader">Estimate:</td><td>{props.Estimate}</td></tr>
                                <tr><td
                                className="CardHeader">Summary:</td><td>{props.Summary}</td></tr></tbody>
                            </table>
                        </td></tr>
                    </table>
                </div>);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
            dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
            "Id" }} enableTooltip={true} tooltipTemplate={this.template.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
                keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
    }
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```



```

        this.data = extend([], kanbanData, null, true);
    }
    private template(props): JSX.Element {
        return (<div className="e-kanbantooltiptemp">
            <table>
                <tr>
                    <td className="e-kanban-card-details">
                        <table>
                            <colgroup>
                                <col style={{ width: "30%" }} />
                                <col style={{ width: "70%" }} />
                            </colgroup>
                            <tbody><tr><td
className="CardHeader">Assignee:</td><td>{props.Assignee}</td></tr>
                                <tr><td
className="CardHeader">Type:</td><td>{props.Type}</td></tr><tr>
                                    <td
className="CardHeader">Estimate:</td><td>{props.Estimate}</td></tr>
                                    <tr><td
className="CardHeader">Summary:</td><td>{props.Summary}</td></tr></tbody>
                                </table>
                            </td></tr>
                        </table>
                    </div>);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} enableTooltip={true} tooltipTemplate={this.template.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Tooltip Template</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Tooltip template" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link href="index.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function template(props) {
        return <div className="e-kanbantooltip" >
            <table>

```

```

        <tr>
            <td className="e-kanban-card-details">
                <table>
                    <colgroup>
                        <col style={{ width: "30%" }}/>
                        <col style={{ width: "70%" }}/>
                    </colgroup>
                    <tbody><tr><td
className="CardHeader">Assignee:</td><td>{props.Assignee}</td></tr>
                    <tr><td
className="CardHeader">Type:</td><td>{props.Type}</td></tr><tr>
                        <td
className="CardHeader">Estimate:</td><td>{props.Estimate}</td></tr>
                    <tr><td
className="CardHeader">Summary:</td><td>{props.Summary}</td></tr></tbody>
                </table>
            </td></tr>
        </table>
    </div>);
}
return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
enableTooltip={true} tooltipTemplate={template.bind(this)}>
    <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function template(props): JSX.Element {
        return (<div className="e-kanbantooltiptemp">
            <table>
                <tr>
                    <td className="e-kanban-card-details">
                        <table>
                            <colgroup>
                                <col style={{ width: "30%" }} />
                                <col style={{ width: "70%" }} />
                            </colgroup>

```

```

        <tbody><tr><td
className="CardHeader">Assignee:</td><td>{props.Assignee}</td></tr>
        <tr><td
className="CardHeader">Type:</td><td>{props.Type}</td></tr><tr>
        <td
className="CardHeader">Estimate:</td><td>{props.Estimate}</td></tr>
        <tr><td
className="CardHeader">Summary:</td><td>{props.Summary}</td></tr></tbody>
        </table>
      </td></tr>
    </table>
  </div>
  );
}
return (
  <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
enableTooltip={true} tooltipTemplate={template.bind(this)}>
    <ColumnsDirective>
      <ColumnDirective headerText="To Do" keyField="Open" />
      <ColumnDirective headerText="In Progress"
keyField="InProgress" />
      <ColumnDirective headerText="Testing" keyField="Testing" />
      <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>
  </KanbanComponent>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Tooltip Template</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Tooltip template" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link href="index.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Validation in React Kanban component

Validate particular column using the `minCount` or `maxCount` properties. The corresponding columns gets different appearance when validation fails. In default layout, `constraintType` property accept only `Column` type. In swimlane layout, accept both `Column` and `Swimlane` constraint type.

There are two types of constraints:

1. Column
2. Swimlane

By default, the column count validation is performed based on Kanban **columns**.

Minimum card limit

The `minCount` property is used to specify the minimum cards hold on particular column or swimlane cell. If the column or swimlane total card count falls short of the minimum count value, it shows the column or cell background colour with validation fails.

Maximum card limit

The `maxCount` property is used to specify the maximum cards hold on particular column or swimlane cell. If the column or swimlane cell total card count exceeds the maximum count value, it shows the column or cell background colour with validation fails.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"
minCount={6}/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress" maxCount={5}/>
        <ColumnDirective headerText="Testing" keyField="Testing"
maxCount={4} minCount={3}/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"
minCount={6} />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" maxCount={5} />
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
```

```

        <ColumnDirective headerText="Testing" keyField="Testing"
maxCount={4} minCount={3} />
        <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>
</KanbanComponent>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Column Validation</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban column validation" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>

```

```

        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"
minCount={6}/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"
maxCount={5}/>
            <ColumnDirective headerText="Testing" keyField="Testing"
maxCount={4} minCount={3}/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" minCount={6}
/>
                <ColumnDirective headerText="In Progress" keyField="InProgress"
maxCount={5} />
                <ColumnDirective headerText="Testing" keyField="Testing"
maxCount={4} minCount={3} />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```



```

        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Column Validation</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban column validation" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></script>
  <script src="systemjs.config.js"></script>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>

```

```
</body>
</html>
```

Virtualization in React Kanban component

Kanban allows you to load a large amount of data without any performance degradation. This feature can be enabled by setting the [enableVirtualization](#) property in the Kanban to `true`.

Virtual scrolling

Virtual scrolling optimizes data rendering within each column when using large datasets. Only a subset of cards that are visible and about to be loaded on the screen are rendered. The number of records displayed in the Kanban is determined implicitly by the height of the Kanban area and the card height. The [cardHeight](#) property of Kanban can be used to set the cards' height in pixel value. By default, the card height will be `auto`.

When the Kanban column is scrolled, the virtual scrolling feature dynamically loads additional data on demand into view and unloads the data that is no longer visible.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component {
  constructor() {
    super(...arguments);
  }
  generateKanbanDataVirtualScrollData() {
    var kanbanData = [];
    var BUG_TASKS = [
      'UI component not displaying images in IE browser',
      'Button not responding on hover action',
      'Text overlapping in mobile view',
      'Dropdown menu not functioning properly',
      'Form validation error',
      'Alignment issue in tables',
      'Column not loading completely',
      'Broken UI Designs',
      'Font size inconsistency',
      'UI element misaligned on scroll'
    ];
    var FEATURE_TASKS = [
      'Implement new user registration flow',
      'Add pagination to search results',
      'Improve accessibility for visually impaired users',
      'Create custom dashboard for users',
      'Develop user profile editing functionality',
      'Integrate with third-party API for weather data',
      'Implement social media sharing for articles',
      'Add support for multiple languages',
      'Create onboarding tutorial for new users',
      'Implement push notifications for mobile app'
    ];
  }
}
```

```

    ];
    var EPIC_TASKS = [
        'Revamp UI design for entire application',
        'Develop mobile application for iOS and Android',
        'Create API for integration with external systems',
        'Implement machine learning algorithms for personalized
recommendations',
        'Upgrade database infrastructure for scalability',
        'Integrate with payment gateway for subscription model',
        'Develop chatbot for customer support',
        'Implement real-time collaboration features for team projects',
        'Create analytics dashboard for administrators',
        'Introduce gamification elements to increase user engagement',
    ];
    var assignee = ['Andrew Fuller', 'Janet Leverling', 'Steven walker',
'Robert King', 'Margaret hamilt', 'Nancy Davloio', 'Margaret Buchanan',
'Laura Bergs', 'Anton Fleet', 'Jack Kathryn', 'Martin Davolio', 'Fleet
Jack'];
    var status = ['Open', 'InProgress', 'Review', 'Testing', 'Close'];
    var priority = ['Ultra-Critical', 'Critical', 'High', 'Normal',
'Low'];
    var types = ['Epic', 'Bug', 'Story'];
    var tagsField = ['Feature', 'Bug', 'Enhancement', 'Documentation',
'Automation', 'Mobile', 'Web', 'iOS', 'Safari', 'Chrome', 'Firefox', 'Manual
Testing'];
    var storyPoints = ['1', '2', '3', '3.5', '4', '4.5', '5', '6',
'7.5', '8'];
    var count = 600000;
    for (let a = 500000, id = 500000; a < count; a++) {
        var typeValue = types[Math.floor(Math.random() * types.length)];
        var summary = typeValue === 'Bug' ?
BUG_TASKS[Math.floor(Math.random() * BUG_TASKS.length)] :
        typeValue === 'Story' ?
FEATURE_TASKS[Math.floor(Math.random() * FEATURE_TASKS.length)] :
        EPIC_TASKS[Math.floor(Math.random() *
EPIC_TASKS.length)];
        kanbanData.push({
            Id: id,
            Type: typeValue,
            Priority: priority[Math.floor(Math.random() *
priority.length)],
            Status: status[Math.floor(Math.random() * status.length)],
            Assignee: assignee[Math.floor(Math.random() *
assignee.length)],
            StoryPoints: storyPoints[Math.floor(Math.random() *
storyPoints.length)],
            Tags: [tagsField[Math.floor(Math.random() *
tagsField.length)], tagsField[Math.floor(Math.random() *
tagsField.length)]],
            Title: 'Task ' + id,
            Summary: summary,
        });
        id++;
    }
    return kanbanData;
}
render() {

```

```

    return <KanbanComponent id="KanbanVirtualScrolling"
enableVirtualization={true} keyField="Status"
dataSource={this.generateKanbanDataVirtualScrollData()} enableTooltip={true}
    cardSettings={{ headerField: "Id", contentField: "Summary",
selectionType: 'Multiple' }}
    dialogSettings={{
        fields: [
            { key: 'Id', text: 'ID', type: 'TextBox' },
            { key: 'Status', text: 'Status', type: 'DropDown' },
            { key: 'StoryPoints', text: 'Story Points', type:
'Numeric' },
            { key: 'Summary', text: 'Summary', type: 'TextArea' }
        ]
    }} >
    <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Code Review" keyField="Review"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>
    </KanbanComponent>;
}
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
    }
    private generateKanbanDataVirtualScrollData(): Record<string, any>[] {
        const kanbanData: Record<string, any>[] = [];
        const BUG_TASKS: string[] = [
            'UI component not displaying images in IE browser',
            'Button not responding on hover action',
            'Text overlapping in mobile view',
            'Dropdown menu not functioning properly',
            'Form validation error',
            'Alignment issue in tables',
            'Column not loading completely',
            'Broken UI Designs',
            'Font size inconsistency',
            'UI element misaligned on scroll'
        ];
        const FEATURE_TASKS: string[] = [
            'Implement new user registration flow',

```

```

    'Add pagination to search results',
    'Improve accessibility for visually impaired users',
    'Create custom dashboard for users',
    'Develop user profile editing functionality',
    'Integrate with third-party API for weather data',
    'Implement social media sharing for articles',
    'Add support for multiple languages',
    'Create onboarding tutorial for new users',
    'Implement push notifications for mobile app'
  ];
  const EPIC_TASKS: string[] = [
    'Revamp UI design for entire application',
    'Develop mobile application for iOS and Android',
    'Create API for integration with external systems',
    'Implement machine learning algorithms for personalized
recommendations',
    'Upgrade database infrastructure for scalability',
    'Integrate with payment gateway for subscription model',
    'Develop chatbot for customer support',
    'Implement real-time collaboration features for team projects',
    'Create analytics dashboard for administrators',
    'Introduce gamification elements to increase user engagement',
  ];
  const assignee: string[] = ['Andrew Fuller', 'Janet Leverling', 'Steven
walker', 'Robert King', 'Margaret hamilt', 'Nancy Davloio', 'Margaret
Buchanan', 'Laura Bergs', 'Anton Fleet', 'Jack Kathryn', 'Martin Davolio',
'Fleet Jack'];
  const status: string[] = ['Open', 'InProgress', 'Review', 'Testing',
'Close'];
  const priority: string[] = ['Ultra-Critical', 'Critical', 'High',
'Normal', 'Low'];
  const types: string[] = ['Epic', 'Bug', 'Story'];
  const tagsField: string[] = ['Feature', 'Bug', 'Enhancement',
'Documentation', 'Automation', 'Mobile', 'Web', 'iOS', 'Safari', 'Chrome',
'Firefox', 'Manual Testing'];
  const storyPoints: string[] = ['1', '2', '3', '3.5', '4', '4.5', '5',
'6', '7.5', '8'];
  const count: number = 60000;
  for (let a: number = 50000, id: number = 50000; a < count; a++) {
    const typeValue: string = types[Math.floor(Math.random() *
types.length)];
    const summary: string = typeValue === 'Bug' ?
BUG_TASKS[Math.floor(Math.random() * BUG_TASKS.length)] :
typeValue === 'Story' ? FEATURE_TASKS[Math.floor(Math.random() *
FEATURE_TASKS.length)] :
EPIC_TASKS[Math.floor(Math.random() * EPIC_TASKS.length)];
    kanbanData.push({
      Id: id,
      Type: typeValue,
      Priority: priority[Math.floor(Math.random() * priority.length)],
      Status: status[Math.floor(Math.random() * status.length)],
      Assignee: assignee[Math.floor(Math.random() * assignee.length)],
      StoryPoints: storyPoints[Math.floor(Math.random() *
storyPoints.length)],
      Tags: [tagsField[Math.floor(Math.random() * tagsField.length)],
tagsField[Math.floor(Math.random() * tagsField.length)]],
      Title: 'Task ' + id,

```

```

        Summary: summary,
    });
    id++;
}
return kanbanData;
}
render() {
    return <KanbanComponent id="KanbanVirtualScrolling"
enableVirtualization={true} keyField="Status"
dataSource={this.generateKanbanDataVirtualScrollData()} enableTooltip={true}
cardSettings={{ headerField: "Id", contentField: "Summary",
selectionType: 'Multiple' }}
dialogSettings={{
fields: [
{ key: 'Id', text: 'ID', type: 'TextBox' },
{ key: 'Status', text: 'Status', type: 'DropDown' },
{ key: 'StoryPoints', text: 'Story Points', type: 'Numeric' },
{ key: 'Summary', text: 'Summary', type: 'TextArea' }
]
}} >
<ColumnsDirective>
    <ColumnDirective headerText="To Do" keyField="Open" />
    <ColumnDirective headerText="In Progress" keyField="InProgress" />
    <ColumnDirective headerText="Code Review" keyField="Review" />
    <ColumnDirective headerText="Done" keyField="Close" />
</ColumnsDirective>
</KanbanComponent>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Local Data</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Local Data" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/ej2-base/styles/material.css"
rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/ej2-react-
popups/styles/material.css" rel="stylesheet" />

```

```

    <link href="https://cdn.syncfusion.com/ej2/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
function App() {
    return (<KanbanComponent id="KanbanVirtualScrolling"
enableVirtualization={true} keyField="Status"
    dataSource={generateKanbanDataVirtualScrollData()} enableTooltip={true}
    cardSettings={{ headerField: "Id", contentField: "Summary",
selectionType: 'Multiple' }}
    dialogSettings={{
        fields: [
            { key: 'Id', text: 'ID', type: 'TextBox' },
            { key: 'Status', text: 'Status', type: 'DropDown' },
            { key: 'StoryPoints', text: 'Story Points', type: 'Numeric' },
            { key: 'Summary', text: 'Summary', type: 'TextArea' }
        ]
    }}>
    <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress" keyField="InProgress" />
        <ColumnDirective headerText="Code Review" keyField="Review" />
        <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>

```

```

</KanbanComponent>);
function generateKanbanDataVirtualScrollData() {
  var kanbanData = [];
  var BUG_TASKS = [
    'UI component not displaying images in IE browser',
    'Button not responding on hover action',
    'Text overlapping in mobile view',
    'Dropdown menu not functioning properly',
    'Form validation error',
    'Alignment issue in tables',
    'Column not loading completely',
    'Broken UI Designs',
    'Font size inconsistency',
    'UI element misaligned on scroll'
  ];
  var FEATURE_TASKS = [
    'Implement new user registration flow',
    'Add pagination to search results',
    'Improve accessibility for visually impaired users',
    'Create custom dashboard for users',
    'Develop user profile editing functionality',
    'Integrate with third-party API for weather data',
    'Implement social media sharing for articles',
    'Add support for multiple languages',
    'Create onboarding tutorial for new users',
    'Implement push notifications for mobile app'
  ];
  var EPIC_TASKS = [
    'Revamp UI design for entire application',
    'Develop mobile application for iOS and Android',
    'Create API for integration with external systems',
    'Implement machine learning algorithms for personalized
recommendations',
    'Upgrade database infrastructure for scalability',
    'Integrate with payment gateway for subscription model',
    'Develop chatbot for customer support',
    'Implement real-time collaboration features for team projects',
    'Create analytics dashboard for administrators',
    'Introduce gamification elements to increase user engagement'
  ];
  var assignee = ['Andrew Fuller', 'Janet Leverling', 'Steven walker',
'Robert King', 'Margaret hamilt', 'Nancy Davloio', 'Margaret Buchanan',
'Laura Bergs', 'Anton Fleet', 'Jack Kathryn', 'Martin Davolio', 'Fleet
Jack'];
  var status = ['Open', 'InProgress', 'Review', 'Testing', 'Close'];
  var priority = ['Ultra-Critical', 'Critical', 'High', 'Normal', 'Low'];
  var types = ['Epic', 'Bug', 'Story'];
  var tagsField = ['Feature', 'Bug', 'Enhancement', 'Documentation',
'Automation', 'Mobile', 'Web', 'iOS', 'Safari', 'Chrome', 'Firefox', 'Manual
Testing'];
  var storyPoints = ['1', '2', '3', '3.5', '4', '4.5', '5', '6', '7.5',
'8'];
  var count = 600000;
  for (let a = 500000, id = 500000; a < count; a++) {
    var typeValue = types[Math.floor(Math.random() * types.length)];
    var summary = typeValue === 'Bug' ? BUG_TASKS[Math.floor(Math.random()
* BUG_TASKS.length)] :

```



```

        typeValue === 'Story' ? FEATURE_TASKS[Math.floor(Math.random() *
FEATURE_TASKS.length)] :
        EPIC_TASKS[Math.floor(Math.random() * EPIC_TASKS.length)];
        kanbanData.push({
            Id: id,
            Type: typeValue,
            Priority: priority[Math.floor(Math.random() * priority.length)],
            Status: status[Math.floor(Math.random() * status.length)],
            Assignee: assignee[Math.floor(Math.random() * assignee.length)],
            StoryPoints: storyPoints[Math.floor(Math.random() *
storyPoints.length)],
            Tags: [tagsField[Math.floor(Math.random() * tagsField.length)],
tagsField[Math.floor(Math.random() * tagsField.length)]],
            Title: 'Task ' + id,
            Summary: summary,
        });
        id++;
    }
    return kanbanData;
}
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
function App() {
    return (
        <KanbanComponent id="KanbanVirtualScrolling" enableVirtualization={true}
keyField="Status"
            dataSource={generateKanbanDataVirtualScrollData()}
enableTooltip={true}
            cardSettings={{ headerField: "Id", contentField: "Summary",
selectionType: 'Multiple' }}
            dialogSettings={{
                fields: [
                    { key: 'Id', text: 'ID', type: 'TextBox' },
                    { key: 'Status', text: 'Status', type: 'DropDown' },
                    { key: 'StoryPoints', text: 'Story Points', type: 'Numeric' },
                    { key: 'Summary', text: 'Summary', type: 'TextArea' }
                ]
            }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Code Review" keyField="Review" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
    function generateKanbanDataVirtualScrollData(): Record<string, any>[] {

```

```

const kanbanData: Record<string, any>[] = [];
const BUG_TASKS: string[] = [
  'UI component not displaying images in IE browser',
  'Button not responding on hover action',
  'Text overlapping in mobile view',
  'Dropdown menu not functioning properly',
  'Form validation error',
  'Alignment issue in tables',
  'Column not loading completely',
  'Broken UI Designs',
  'Font size inconsistency',
  'UI element misaligned on scroll'
];
const FEATURE_TASKS: string[] = [
  'Implement new user registration flow',
  'Add pagination to search results',
  'Improve accessibility for visually impaired users',
  'Create custom dashboard for users',
  'Develop user profile editing functionality',
  'Integrate with third-party API for weather data',
  'Implement social media sharing for articles',
  'Add support for multiple languages',
  'Create onboarding tutorial for new users',
  'Implement push notifications for mobile app'
];
const EPIC_TASKS: string[] = [
  'Revamp UI design for entire application',
  'Develop mobile application for iOS and Android',
  'Create API for integration with external systems',
  'Implement machine learning algorithms for personalized recommendations',
  'Upgrade database infrastructure for scalability',
  'Integrate with payment gateway for subscription model',
  'Develop chatbot for customer support',
  'Implement real-time collaboration features for team projects',
  'Create analytics dashboard for administrators',
  'Introduce gamification elements to increase user engagement'
];
const assignee: string[] = ['Andrew Fuller', 'Janet Leverling', 'Steven walker', 'Robert King', 'Margaret hamilt', 'Nancy Davloio', 'Margaret Buchanan', 'Laura Bergs', 'Anton Fleet', 'Jack Kathryn', 'Martin Davolio', 'Fleet Jack'];
const status: string[] = ['Open', 'InProgress', 'Review', 'Testing', 'Close'];
const priority: string[] = ['Ultra-Critical', 'Critical', 'High', 'Normal', 'Low'];
const types: string[] = ['Epic', 'Bug', 'Story'];
const tagsField: string[] = ['Feature', 'Bug', 'Enhancement', 'Documentation', 'Automation', 'Mobile', 'Web', 'iOS', 'Safari', 'Chrome', 'Firefox', 'Manual Testing'];
const storyPoints: string[] = ['1', '2', '3', '3.5', '4', '4.5', '5', '6', '7.5', '8'];
const count: number = 60000;
for (let a: number = 50000, id: number = 50000; a < count; a++) {
  const typeValue: string = types[Math.floor(Math.random() * types.length)];

```

```

    const summary: string = typeValue === 'Bug' ?
BUG_TASKS[Math.floor(Math.random() * BUG_TASKS.length)] :
    typeValue === 'Story' ? FEATURE_TASKS[Math.floor(Math.random() *
FEATURE_TASKS.length)] :
    EPIC_TASKS[Math.floor(Math.random() * EPIC_TASKS.length)];
    kanbanData.push({
      Id: id,
      Type: typeValue,
      Priority: priority[Math.floor(Math.random() * priority.length)],
      Status: status[Math.floor(Math.random() * status.length)],
      Assignee: assignee[Math.floor(Math.random() * assignee.length)],
      StoryPoints: storyPoints[Math.floor(Math.random() *
storyPoints.length)],
      Tags: [tagsField[Math.floor(Math.random() * tagsField.length)],
tagsField[Math.floor(Math.random() * tagsField.length)]],
      Title: 'Task ' + id,
      Summary: summary,
    });
    id++;
  }
  return kanbanData;
}
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Local Data</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban Local Data" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/ej2-base/styles/material.css"
rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Configure the remote data service

When the remote data is configured for the [dataSource](#), the service method will receive an additional `KanbanVirtualization` parameter to handle the initial data load for Kanban Virtualization.

To handle Kanban virtual scrolling, the server-side code needs to handle the `Where` and `Take` queries differently using the `KanbanVirtualization` parameter. The following is the example code for handling Kanban virtualization's initial data load using the `KanbanVirtualization` parameter.

```

`ts
public IActionResult LoadCard([FromBody] ExtendedDataManagerRequest dm)
{
    kanbanData = _context.KanbanCards.ToList();
    IEnumerable<KanbanCard> DataSource = kanbanData.AsEnumerable();
    DataOperations operation = new DataOperations();
    // For normal kanban data load Where query handling.
    if (dm.Where != null && dm.Where.Count > 0 && dm.KanbanVirtualization != "KanbanVirtualization")
    {
        dm.Where[0].value = dm.Where[0].value.ToString();
        DataSource = operation.PerformFiltering(DataSource, dm.Where, dm.Where[0].Operator);
    }
    if (dm.Skip != 0)
    {
        DataSource = operation.PerformSkip(DataSource, dm.Skip);
    }
}

```

```
}  
  
// For normal Kanban data load Take query handling.  
if (dm.Take != 0 && dm.KanbanVirtualization != "KanbanVirtualization")  
{  
    DataSource = operation.PerformTake(DataSource, dm.Take);  
}  
  
// For Kanban virtual scrolling data load Where and Take query handling.  
var columnCount = new List<KeyValuePair<string, int>>();  
if (dm.KanbanVirtualization == "KanbanVirtualization" && dm.Where != null && dm.Where.Count > 0  
&& dm.Take != 0)  
{  
    IEnumerable<KanbanCard> currentData = new List<KanbanCard>();  
    List<WhereFilter> currentFilter = new List<WhereFilter>();  
    for (int i = 0; i < dm.Where.Count; i++)  
    {  
        dm.Where[i].value = dm.Where[i].value.ToString();  
        currentFilter.Add(dm.Where[i]);  
        var filterData = operation.PerformFiltering(DataSource, currentFilter, dm.Where[i].Operator);  
        columnCount.Add(new KeyValuePair<string, int>(dm.Where[i].value.ToString(), filterData.Count()));  
        filterData = operation.PerformTake(filterData, dm.Take);  
        currentData = currentData.Concat(filterData);  
        currentFilter.Clear();  
    }  
    DataSource = currentData;  
}  
  
// To return the data for Kanban virtual scrolling.  
if (dm.KanbanVirtualization == "KanbanVirtualization") {  
    return Json(new { result = DataSource, count = columnCount });  
}  
  
// To return the data for Kanban virtual scrolling.  
else  
{  
    return Json(DataSource);  
}
```

```

}
}
,

```

Limitations for virtual scrolling

- When virtualization is enabled in a Kanban board and the card height is not explicitly set, it will not default to `auto` height. Instead, a fixed height of `100px` will be applied to the cards. It's important to note that the card height should be specified in pixel values, as percentage values are not accepted.
- When a card is dragged and dropped, the index position of the card will not be preserved when scrolling through the column.
- Virtualization is not supported for swimlanes in the Kanban board.

Localization in React Kanban component

The localization library allows you to localize the default text content of the Kanban to different cultures using the `locale` property.

In Kanban, total count and min or max count text alone will be localized based on culture.

```

| Locale key | en-US (default) |
|-----|-----|
| items | items |
| min | Min |
| max | Max |
| cardsSelected | Cards Selected |
| addTitle | Add New Card |
| editTitle | Edit Card Details |
| deleteTitle | Delete Card |
| deleteContent | Are you sure you want to delete this card? |
| save | Save |
| delete | Delete |
| cancel | Cancel |
| yes | Yes |
| no | No |
| close | Close |
| noCard | No cards to display |
| unassigned | Unassigned |

```

Loading translations

To load translation object in an application, use `load` function of `L10n` class.

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

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, L10n } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
L10n.load({
    'de': {
        'kanban': {
            'items': 'Artikel',
            'min': 'Min',
            'max': 'Max',
            'cardsSelected': 'Karten ausgewählt',
            'addTitle': 'Neue Karte hinzufügen',
            'editTitle': 'Kartendetails bearbeiten',
            'deleteTitle': 'Karte löschen',
            'deleteContent': 'Möchten Sie diese Karte wirklich löschen?',
            'save': 'speichern',
            'delete': 'Löschen',
            'cancel': 'Stornieren',
            'yes': 'Ja',
            'no': 'Nein',
            'close': 'Schließen',
            'noCard': 'Keine Karten zum Anzeigen',
            'unassigned': 'nicht zugewiesen'
        }
    }
});
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} locale='de' cardSettings={{ contentField: "Summary",
headerField: "Id" }} swimlaneSettings={{ keyField: "Assignee" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"
minCount={6}/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress" maxCount={3}/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, L10n } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
L10n.load({
    'de': {
        'kanban': {
            'items': 'Artikel',
            'min': 'Min',
            'max': 'Max',
            'cardsSelected': 'Karten ausgewählt',
            'addTitle': 'Neue Karte hinzufügen',
            'editTitle': 'Kartendetails bearbeiten',
            'deleteTitle': 'Karte löschen',
            'deleteContent': 'Möchten Sie diese Karte wirklich löschen?',
            'save': 'speichern',
            'delete': 'Löschen',
            'cancel': 'Stornieren',
            'yes': 'Ja',
            'no': 'Nein',
            'close': 'Schließen',
            'noCard': 'Keine Karten zum Anzeigen',
            'unassigned': 'nicht zugewiesen'
        }
    }
});
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} locale='de' cardSettings={{ contentField: "Summary",
headerField: "Id" }} swimlaneSettings={{ keyField: "Assignee" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"
minCount={6} />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" maxCount={3} />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML


```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban Localization</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban localization" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';

```

```

import { extend, L10n } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
L10n.load({
    'de': {
        'kanban': {
            'items': 'Artikel',
            'min': 'Min',
            'max': 'Max',
            'cardsSelected': 'Karten ausgewählt',
            'addTitle': 'Neue Karte hinzufügen',
            'editTitle': 'Kartendetails bearbeiten',
            'deleteTitle': 'Karte löschen',
            'deleteContent': 'Möchten Sie diese Karte wirklich löschen?',
            'save': 'speichern',
            'delete': 'Löschen',
            'cancel': 'Stornieren',
            'yes': 'Ja',
            'no': 'Nein',
            'close': 'Schließen',
            'noCard': 'Keine Karten zum Anzeigen',
            'unassigned': 'nicht zugewiesen'
        }
    }
});
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
locale='de' cardSettings={{ contentField: "Summary", headerField: "Id" }}
swimlaneSettings={{ keyField: "Assignee" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"
minCount={6}/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"
maxCount={3}/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, L10n } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
L10n.load({
    'de': {
        'kanban': {
            'items': 'Artikel',

```

```

        'min': 'Min',
        'max': 'Max',
        'cardsSelected': 'Karten ausgewählt',
        'addTitle': 'Neue Karte hinzufügen',
        'editTitle': 'Kartendetails bearbeiten',
        'deleteTitle': 'Karte löschen',
        'deleteContent': 'Möchten Sie diese Karte wirklich löschen?',
        'save': 'speichern',
        'delete': 'Löschen',
        'cancel': 'Stornieren',
        'yes': 'Ja',
        'no': 'Nein',
        'close': 'Schließen',
        'noCard': 'Keine Karten zum Anzeigen',
        'unassigned': 'nicht zugewiesen'
    }
}
});
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
        locale="de" cardSettings={{ contentField: "Summary", headerField: "Id" }}
        swimlaneSettings={{ keyField: "Assignee" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" minCount={6}
            />
                <ColumnDirective headerText="In Progress" keyField="InProgress"
            maxCount={3} />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Localization</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban localization" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Right to left (RTL)

The Kanban provides an option to switch its text direction and layout 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 right-to-left mode in Kanban, set the `enableRtl` to true.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, L10n } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
L10n.load({
    'ar': {
        'kanban': {
            'items': 'العناصر',
            'min': 'أنا',
            'max': 'ماكس',
            'cardsSelected': 'تم تحديد البطاقات',

```

```

        'addTitle': 'إضافة بطاقة جديدة',
        'editTitle': 'تحرير تفاصيل البطاقة',
        'deleteTitle': 'حذف البطاقة',
        'deleteContent': 'هل أنت متأكد أنك تريد حذف هذه البطاقة?',
        'save': 'حفظ',
        'delete': 'حذف',
        'cancel': 'إلغاء',
        'yes': 'نعم',
        'no': 'لا',
        'close': 'قريب',
        'noCard': 'لا توجد بطاقات لعرضها',
        'unassigned': 'غير معين'
    }
}
});
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} locale='ar' enableRtl={true} cardSettings={{
        contentField: "Summary", headerField: "Id" }} swimlaneSettings={{ keyField:
        "Assignee" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"
minCount={2}/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress" maxCount={3}/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, L10n } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
L10n.load({
    'ar': {
        'kanban': {
            'items': 'العناصر',
            'min': 'أنا',
            'max': 'ماكس',
            'cardsSelected': 'تم تحديد البطاقات',
            'addTitle': 'إضافة بطاقة جديدة',

```

```

        'editTitle': 'تحرير تفاصيل البطاقة',
        'deleteTitle': 'حذف البطاقة',
        'deleteContent': 'هل أنت متأكد أنك تريد حذف هذه البطاقة?',
        'save': 'حفظ',
        'delete': 'حذف',
        'cancel': 'إلغاء',
        'yes': 'نعم',
        'no': 'لا',
        'close': 'قريب',
        'noCard': 'لا توجد بطاقات لعرضها',
        'unassigned': 'غير معين'
    }
}
});
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} locale='ar' enableRtl={true} cardSettings={{
        contentField: "Summary", headerField: "Id" }} swimlaneSettings={{ keyField:
        "Assignee" }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"
minCount={2} />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" maxCount={3} />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban RTL</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban RTL mode" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, L10n } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
L10n.load({
    'ar': {
        'kanban': {
            'items': 'العناصر',
            'min': 'أنا',
            'max': 'ماكس',
            'cardsSelected': 'تم تحديد البطاقات',
            'addTitle': 'إضافة بطاقة جديدة',
            'editTitle': 'تحرير تفاصيل البطاقة',
            'deleteTitle': 'حذف البطاقة',
            'deleteContent': 'هل أنت متأكد أنك تريد حذف هذه البطاقة؟',
            'save': 'حفظ',

```

```

        'delete': 'حذف',
        'cancel': 'إلغاء',
        'yes': 'نعم',
        'no': 'لا',
        'close': 'قريب',
        'noCard': 'لا توجد بطاقات لعرضها',
        'unassigned': 'غير معين'
    }
}
});
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
    locale='ar' enableRtl={true} cardSettings={{ contentField: "Summary",
    headerField: "Id" }} swimlaneSettings={{ keyField: "Assignee" }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" minCount={2}/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"
    maxCount={3}/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend, L10n } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
L10n.load({
    'ar': {
        'kanban': {
            'items': 'العناصر',
            'min': 'أنا',
            'max': 'ماكس',
            'cardsSelected': 'تم تحديد البطاقات',
            'addTitle': 'إضافة بطاقة جديدة',
            'editTitle': 'تحرير تفاصيل البطاقة',
            'deleteTitle': 'حذف البطاقة',
            'deleteContent': 'هل أنت متأكد أنك تريد حذف هذه البطاقة?',
            'save': 'حفظ',
            'delete': 'حذف',
            'cancel': 'إلغاء',
            'yes': 'نعم',
            'no': 'لا',
            'close': 'قريب',
            'noCard': 'لا توجد بطاقات لعرضها',
            'unassigned': 'غير معين'
        }
    }
});

```



```

});
function App() {
  let data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    locale='ar' enableRtl={true} cardSettings={{ contentField: "Summary",
    headerField: "Id" }} swimlaneSettings={{ keyField: "Assignee" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" minCount={2}
      />
        <ColumnDirective headerText="In Progress" keyField="InProgress"
    maxCount={3} />
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Kanban RTL</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban RTL mode" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008c9f;

```

```

        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Dimensions in React Kanban component

The Kanban dimensions refers to both height and width of the entire layout and it accepts three types of values.

- Auto
- Pixel
- Percentage

Auto height and width

When height and width of the Kanban 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 Kanban, its width or height will be the sum of its children. By default, Kanban is assigned with **auto** values for both the height and width properties.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} width="auto" height="auto">
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
            </ColumnsDirective>
        </KanbanComponent>
    }
}
ReactDOM.render(<App />, document.getElementById('root'));

```

```

        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} width="auto" height="auto">
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Auto Height and Width</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban auto height and width" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008c9f;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} width="auto"
height="auto">
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress" keyField="InProgress"/>

```

```

        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} width="auto"
height="auto">
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Auto Height and Width</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban auto height and width" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Height and width in pixel

The Kanban height and width will be rendered exactly as per the given pixel values. It accepts both string and number values.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} width={650} height="550px">
            <ColumnsDirective>

```

```

        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} width={650} height="550px">
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Set Kanban Height and Width on Pixel</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban height and width to pixel" />
    <meta name="author" content="Syncfusion" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
    let data = extend([], kanbanData, null, true);

```



```

    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} width={650}
height="550px">
    <ColumnsDirective>
    <ColumnDirective headerText="To Do" keyField="Open"/>
    <ColumnDirective headerText="In Progress" keyField="InProgress"/>
    <ColumnDirective headerText="Testing" keyField="Testing"/>
    <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} width={650}
height="550px">
    <ColumnsDirective>
    <ColumnDirective headerText="To Do" keyField="Open" />
    <ColumnDirective headerText="In Progress" keyField="InProgress" />
    <ColumnDirective headerText="Testing" keyField="Testing" />
    <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>
    </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Set Kanban Height and Width on Pixel</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban height and width to pixel" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Height and width in percentage

When height and width of the Kanban are given in percentage, it will make the Kanban as wide as the parent container.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component {
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }

```

```

    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
        "Id" }} width="100%" height="100%">
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
        keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    render() {
        return <KanbanComponent id="kanban" keyField="Status"
        dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
        "Id" }} width="100%" height="100%">
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
        keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
        />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

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

```

```

<title>Set Kanban Height and Width on Percentage</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Kanban height and width to percentage"
/>
<meta name="author" content="Syncfusion" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';

```

```
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} width="100%"
height="100%">
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App(){
    let data = extend([], kanbanData, null, true);
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} width="100%"
height="100%">
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress" keyField="InProgress"
/>
                <ColumnDirective headerText="Testing" keyField="Testing" />
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Set Kanban Height and Width on Percentage</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```

<meta name="description" content="Kanban height and width to percentage"
/>
<meta name="author" content="Syncfusion" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Persistence in React Kanban component

State persistence refers to the Kanban 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 Kanban datasource, column and swimlane expand/collapse state in the local storage when the [enablePersistence](#) is defined as true.

[Class-component]

INDEX.JSX

```
{% raw %}
```

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} enablePersistence={true} swimlaneSettings={{ keyField: "Assignee"
}}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true}/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress" allowToggle={true}/>
        <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true}/>
        <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true}/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component<{}> {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} enablePersistence={true} swimlaneSettings={{ keyField: "Assignee"
}}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true} />

```

```

        <ColumnDirective headerText="In Progress"
keyField="InProgress" allowToggle={true} />
        <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true} />
        <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true} />
    </ColumnsDirective>
</KanbanComponent>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Persistence</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Persistence" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
poppers/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>

```



```

</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App() {
  let data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ contentField: "Summary", headerField: "Id" }}
    enablePersistence={true} swimlaneSettings={{ keyField: "Assignee" }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true}/>
        <ColumnDirective headerText="In Progress" keyField="InProgress"
allowToggle={true}/>
        <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true}/>
        <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true}/>
      </ColumnsDirective>
    </KanbanComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
function App(){
  let data = extend([], kanbanData, null, true);
  return (
    <KanbanComponent id="kanban" keyField="Status" dataSource={data}
    cardSettings={{ contentField: "Summary", headerField: "Id" }}
    enablePersistence={true} swimlaneSettings={{ keyField: "Assignee" }}>
      <ColumnsDirective>

```

```

        <ColumnDirective headerText="To Do" keyField="Open"
allowToggle={true} />
        <ColumnDirective headerText="In Progress" keyField="InProgress"
allowToggle={true} />
        <ColumnDirective headerText="Testing" keyField="Testing"
allowToggle={true} />
        <ColumnDirective headerText="Done" keyField="Close"
allowToggle={true} />
    </ColumnsDirective>
</KanbanComponent>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kanban Persistence</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban Persistence" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;

```

```
    }  
  </style>  
</head>  
<body>  
  <div id='kanban'>  
    <div id='loader'>Loading....</div>  
  </div>  
</body>  
</html>
```

Responsive mode in React Kanban component

The Kanban component has support for responsive behavior based on the client browser's width and height.

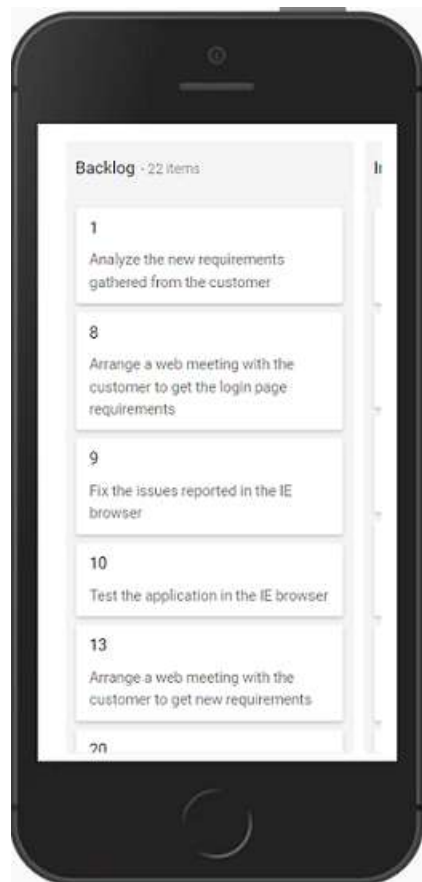
Layouts

Possible layouts are:

- Default Layout
- Swimlane Layout

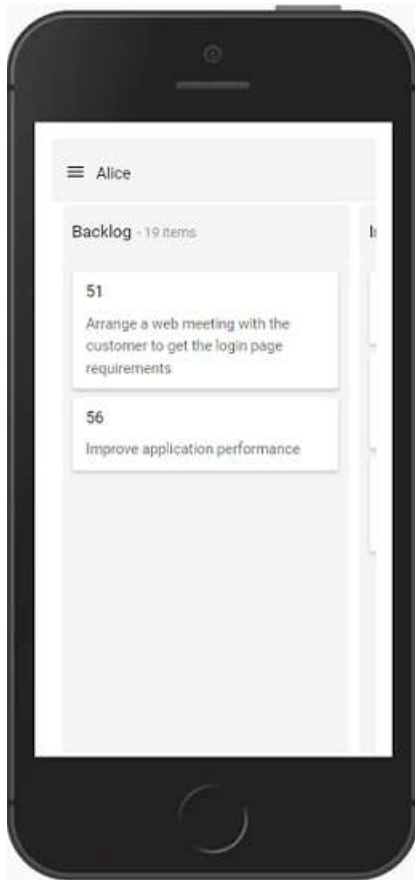
Default Layout

Kanban user interface is customized and redesigned for the best view on small screens. In responsive mode, the first column occupies 80% and the second column occupies 20% of the screen layout. Tap and hold the Kanban card to drag and drop it. Swipe left or right to view the columns.



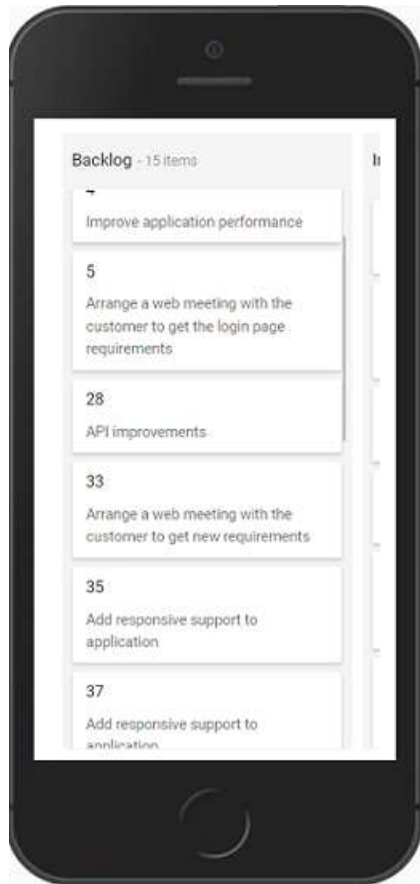
Swimlane Layout

Kanban swimlane header is rendered with menu icon on top of the kanban board. It will show all the available swimlane groups of the header text with a popup when clicking the menu icon. Swimlane selected grouped header text resultant data is rendered on the Kanban board. By default, the first swimlane grouped header text is selected and the resultant data is shown on the Kanban board. The Kanban board data will be changed when changing the swimlane group header text.



Scrolling

Column scrolling will be shown when exceeding the screen size in the columns.

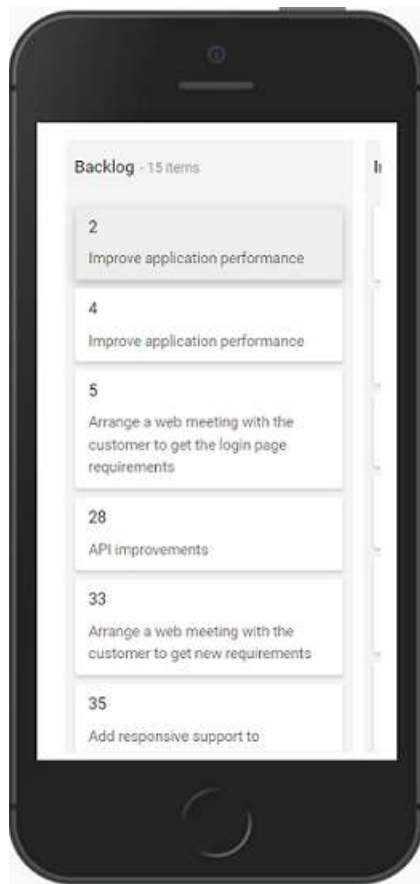


Selection

Select particular cards in the Kanban board by tapping the card.

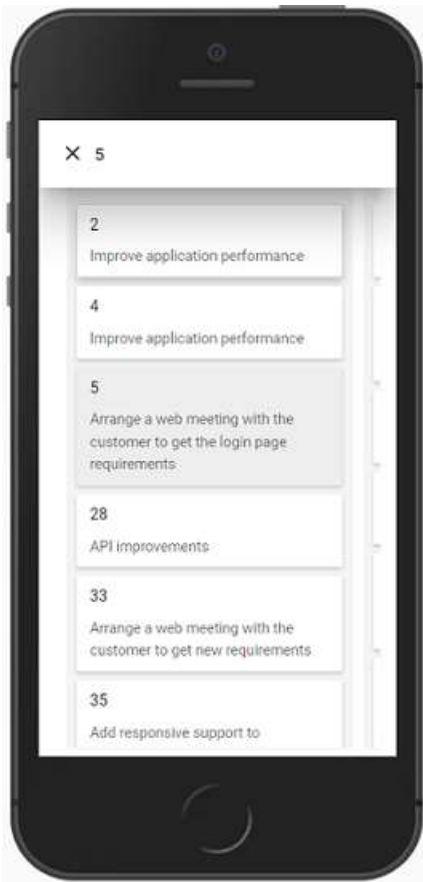
Single Selection

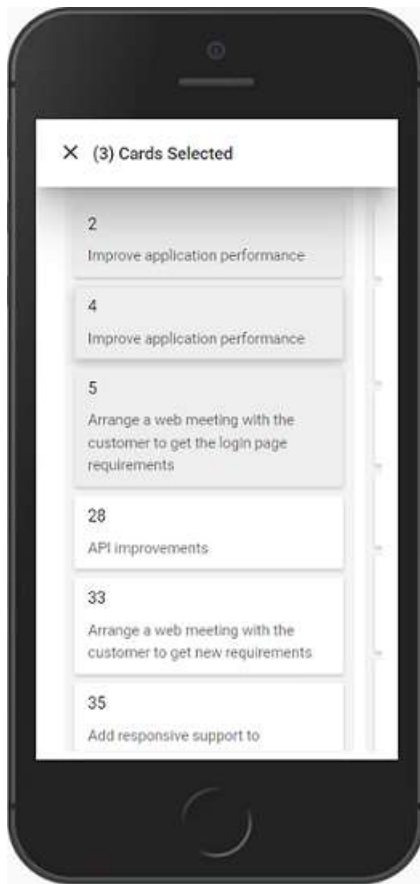
Single card will be selected when you tap the card once and selection will be removed when you select another card.



Multiple Selection

Enable `selectionType` as `Multiple` to select multiple cards. It will open the popup on the screen top. Selected card header text will be shown when selecting single card with a tap and hold action. If single card is selected, only tap action is required to select multiple cards. Multiple Selected card count will be shown on the popup when selecting multiple cards.





Style in React Kanban component

To modify the Kanban appearance, you need to override the default CSS of Kanban. Also, you have an option to create your own custom theme using our [Theme Studio](#). Please find the list of CSS classes in Kanban.

Css class	Purpose
-----	-----
<code>.e-kanban .e-kanban-table</code>	customize the kanban.
<code>.e-kanban .e-kanban-header .e-header-cells</code>	Header cells of kanban.
<code>.e-kanban .e-kanban-header .e-header-cells .e-header-wrap .e-header-title</code>	Header title of kanban.
<code>.e-kanban .e-kanban-header .e-header-cells.e-min-color</code>	Header cells minimum color of kanban.
<code>.e-kanban .e-kanban-header .e-header-cells.e-max-color</code>	Header cells maximum color of kanban.
<code>.e-kanban .e-kanban-header .e-header-cells.e-collapsed.e-min-color</code>	Header cells of collapsed column minimum color in column constraint type of kanban.
<code>.e-kanban .e-kanban-header .e-header-cells.e-collapsed.e-max-color</code>	Header cells of collapsed column maximum color in column constraint type of kanban.
<code>.e-kanban .e-kanban-header .e-header-cells .e-header-text</code>	Header text of Kanban.
<code>.e-kanban .e-kanban-header .e-header-cells .e-item-count</code>	Header cells Item count of Kanban.

| .e-kanban .e-kanban-header .e-header-cells .e-limits | Header cells limits in column constraint type of kanban. |

| .e-kanban .e-kanban-header .e-header-cells .e-limits .e-min-count | Header cells minimum count of kanban. |

| .e-kanban .e-kanban-header .e-header-cells .e-limits .e-max-count | Header cells maximum count of kanban. |

| .e-kanban .e-kanban-content | Customize kanban Content. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-limits | Content cells limits in swimlane constraint type of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-limits .e-min-count | Content cells minimum count of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-limits .e-max-count | Content cells maximum count of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-min-color | Content cells minimum color of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-max-color | Content cells maximum color of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-collapsed .e-collapse-header-text | Content cells of collapsed header text. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-collapsed .e-collapse-header-text .e-item-count | Content cells of collapsed header text Item count. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-show-add-button | Add button in content cells of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-show-add-button .e-show-add-icon | Customize content cells add icon of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-empty-card | Empty content cells of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card | Customize cards in kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-header .e-card-header-title | Cards header title of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-footer | Cards footer of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-content | Cards content of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card.e-card-color | Cards color of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-tags | Customize Card tags of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-tag |
Card tag of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-swimlane-row-expand | Content cells of swimlane row expand of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-swimlane-row-collapse | Content cells of swimlane row collapse of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-swimlane-text | Content cells of swimlane header text of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-item-count | Content cells of swimlane items count of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells | swimlane content cells of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells.e-dropping |
Customize swimlane content cells card dropping of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells .e-card-wrapper |
Swimlane content cells of card wrapper. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells.e-min-color |
Swimlane content cells of minimum color of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells.e-max-color |
Swimlane content cells of maximum color of kanban. |Customize the kanban CSS theme. Please find the list of CSS classes in Kanban. | .e-kanban .e-kanban-table .e-header-cells | Header cells of kanban. |

| .e-kanban .e-kanban-table .e-header-cells .e-header-text | Header text of Kanban. |

| .e-kanban .e-kanban-table .e-header-cells .e-item-count | Header cells Item count of Kanban. |

| .e-kanban .e-kanban-table .e-header-cells .e-column-expand | Header cells of toggle icon in column expand. |

| .e-kanban .e-kanban-table .e-header-cells .e-column-collapse | Header cells of toggle icon in column collapse. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row:not(.e-swimlane-row) .e-content-cells |
swimlane content cells of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row.e-swimlane-row .e-swimlane-text | Content cells of swimlane header text of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row.e-swimlane-row .e-item-count | Content cells of swimlane items count of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row .e-show-add-button .e-show-add-icon | Add icon in content cells of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card.e-selection | Selected card of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card .e-card-header | Cards header in kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card .e-card-content | Cards content in kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card .e-card-tag.e-card-label | Cards label in kanban. |

To set fixed position to the Kanban header

The Fixed header in Kanban control can be customized in following ways,

By setting a fixed height to the Kanban content,

```
`css
.e-kanban .e-kanban-content {
height: 500px;
}
`
```

By customizing the CSS for the Kanban header.

```
`css
.e-kanban-header {
position: -webkit-sticky;
position: sticky;
z-index: 100;
top: 0;
}
`
```

Note: It will not affect the Kanban content's height.

Accessibility in React Kanban component

The Kanban 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 Kanban component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](https://cdn.syncfusion.com/content/images/documentation/full.png) | ![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) |

| [Section 508 Support](https://cdn.syncfusion.com/content/images/documentation/full.png) | ![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) |

| [Screen Reader Support](https://cdn.syncfusion.com/content/images/documentation/full.png) | ![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) |

|

| [Right-To-Left Support](https://cdn.syncfusion.com/content/images/documentation/full.png) | ![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) |

| [Color Contrast](https://cdn.syncfusion.com/content/images/documentation/full.png) |

![Intermediate](https://cdn.syncfusion.com/content/images/documentation/partial.png) |

| [Mobile Device Support](#) | ![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png)
|

| [Keyboard Navigation Support](#) |
![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) |

| [Accessibility Checker Validation](#) |
![Intermediate](https://cdn.syncfusion.com/content/images/documentation/partial.png) |

| [Axe-core Accessibility Validation](#) |
![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) - All features of the component meet the requirement.

![Intermediate](https://cdn.syncfusion.com/content/images/documentation/partial.png) - Some features of the component do not meet the requirement.

![No](https://cdn.syncfusion.com/content/images/documentation/not-supported.png) - The component does not meet the requirement.

WAI-ARIA attributes

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

| Attributes | Purpose |

| --- | --- |

| **aria-label** | It helps to provides information about elements in a kanban component for assistive technology. |

| **aria-expanded** | Attributes indicate the state of a collapsible element. |

| **aria-selected** | This attribute is assigned to the Kanban component for the selection of elements, and its default value is **false**. The value changes to true when the user selects a Kanban card. |

| **aria-grabbed** | Indicates whether the attribute is set to true. It has been selected for dragging. If this attribute is set to false, the element can be grabbed for a drag-and-drop operation but will not be currently grabbed. |

| **aria-describedby** | This attribute contains the ID of the Kanban header column to indicate that the attribute establishes an association between the Kanban header column and the Kanban column body. |

| **aria-roledescription** | This attribute is assigned to the Kanban component and is used to provide alternative descriptions for card elements. |

Keyboard interaction

The Kanban 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 Kanban component.

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

| --- | --- |

| **Home** | To select the first card in the kanban |

| **End** | To select the last card in the kanban |

| **Arrow Up** | Select the card through the up arrow |

| **Arrow Down** | Select the card through the down arrow |

| **Arrow Right** | Move the column selection to the right |

| **Arrow Left** | Move the column selection to the left |

| **Ctrl + Enter** | Used to select the multi cards |

| **Ctrl + Space** | Used to select the multi cards |

| **Shift + Arrow Up** | Used to select the multiple cards towards up |

| **Shift + Arrow Down** | Used to select the multiple cards towards down |

| **Shift + Tab** | Reverse order of the tab action |

| **Enter** | Open the selected cards |

| **Tab** | To navigate the Kanban column |

| **Delete** | To delete the selected cards |

| **ESC** | Escape from the modified details |

| **Space** | Used to open the card edit dialog based on the column selection |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

How To

Header double click in React Kanban component

You can bind the header double click event by using the [dataBound](#) event at the initial rendering. You can get the column header text when you double click on the headers.

[Class-component]

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { DialogUtility } from '@syncfusion/ej2-popups';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from '../datasource';
class App extends React.Component {
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  OnDataBound() {
    let headerEle = document.querySelector('.e-header-row');
    headerEle.addEventListener("dblclick", function (e) {
      let target = closest(e.target, '.e-header-cells');
      DialogUtility.alert({
        title: 'Header',
        content: "Double clicked on " + target.querySelector('.e-
header-text').innerText + " header",
        showCloseIcon: true,
        closeOnEscape: true,
        animationSettings: { effect: 'Zoom' }
      });
    });
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} dataBound={this.OnDataBound.bind(this)}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
      </ColumnsDirective>
    </KanbanComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { DialogUtility } from '@syncfusion/ej2-popups';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
```

```

import { kanbanData } from './datasource';
class App extends React.Component<{}>, {}>{
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  private OnDataBound(): void {
    let headerEle: HTMLElement = document.querySelector('.e-header-
row');
    headerEle.addEventListener("dblclick", function (e: Event) {
      let target = closest(e.target, '.e-header-cells');
      DialogUtility.alert({
        title: 'Header',
        content: "Double clicked on " + target.querySelector('.e-
header-text').innerText + " header",
        showCloseIcon: true,
        closeOnEscape: true,
        animationSettings: { effect: 'Zoom' }
      });
    });
  }
  render() {
    return <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} dataBound={this.OnDataBound.bind(this)}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Auto Height and Width</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban auto height and width" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { DialogUtility } from '@syncfusion/ej2-popups';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function OnDataBound() {
        let headerEle = document.querySelector('.e-header-row');
        headerEle.addEventListener("dblclick", function (e) {
            let target = closest(e.target, '.e-header-cells');
            DialogUtility.alert({
                title: 'Header',

```



```

        content: "Double clicked on " + target.querySelector('.e-
header-text').innerText + " header",
        showCloseIcon: true,
        closeOnEscape: true,
        animationSettings: { effect: 'Zoom' }
    });
});
}
return (<KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dataBound={OnDataBound.bind(this)}>
    <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open"/>
        <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
        <ColumnDirective headerText="Testing" keyField="Testing"/>
        <ColumnDirective headerText="Done" keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { DialogUtility } from '@syncfusion/ej2-popups';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    function OnDataBound(): void {
        let headerEle: HTMLElement = document.querySelector('.e-header-
row');
        headerEle.addEventListener("dblclick", function (e: Event) {
            let target = closest(e.target, '.e-header-cells');
            DialogUtility.alert({
                title: 'Header',
                content: "Double clicked on " + target.querySelector('.e-
header-text').innerText + " header",
                showCloseIcon: true,
                closeOnEscape: true,
                animationSettings: { effect: 'Zoom' }
            });
        });
    }
    return (
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }}
dataBound={OnDataBound.bind(this)}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />

```

```

        <ColumnDirective headerText="In Progress" keyField="InProgress"
    />
        <ColumnDirective headerText="Testing" keyField="Testing" />
        <ColumnDirective headerText="Done" keyField="Close" />
    </ColumnsDirective>
</KanbanComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Auto Height and Width</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban auto height and width" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>

```

```

        <div id='kanban'>
            <div id='loader'>Loading....</div>
        </div>
    </body>
</html>

```

Dynamically change columns in React Kanban component

You can dynamically change the Kanban columns by using the [columns](#) property.

In the below sample, you can dynamically change the [allowToggle](#) property at the particular column when you click on the button. You can also change the initially created columns to the new Kanban columns by using the [columns](#) property when you click on the button.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { kanbanData } from './datasource';
class App extends React.Component {
    kanbanObj;
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    particularColumn() {
        this.kanbanObj.columns[1].allowToggle = true;
    }
    column() {
        this.kanbanObj.columns = [
            { headerText: 'To Do', keyField: 'Open' },
            { headerText: 'Done', keyField: 'Close' }
        ];
    }
    render() {
        return <div className='control-wrapper'>
            <ButtonComponent id='particularColumn' className="e-btn"
onClick={this.particularColumn.bind(this)}>Enable Allow
Toggle</ButtonComponent>
            <ButtonComponent id='column' className="e-btn"
onClick={this.column.bind(this)}>Change Columns</ButtonComponent>
            <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} ref={(kanban) => { this.kanbanObj = kanban; }}>
                <ColumnsDirective>
                    <ColumnDirective headerText="To Do"
keyField="Open"/>
                    <ColumnDirective headerText="In Progress"
keyField="InProgress"/>

```

```

                                <ColumnDirective headerText="Testing"
keyField="Testing"/>
                                <ColumnDirective headerText="Done"
keyField="Close"/>
                                </ColumnsDirective>
                                </KanbanComponent>
                                </div>;
        }
    }
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
    private kanbanObj: KanbanComponent;
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    particularColumn() {
        this.kanbanObj.columns[1].allowToggle= true;
    }
    column() {
        this.kanbanObj.columns = [
            { headerText: 'To Do', keyField: 'Open' },
            { headerText: 'Done', keyField: 'Close' }
        ]
    }
    render() {
        return <div className='control-wrapper'>
            <ButtonComponent id='particularColumn' className="e-btn"
onClick={this.particularColumn.bind(this)} >Enable Allow
Toggle</ButtonComponent>
            <ButtonComponent id='column' className="e-btn"
onClick={this.column.bind(this)}>Change Columns</ButtonComponent>
            <KanbanComponent id="kanban" keyField="Status"
dataSource={this.data} cardSettings={{ contentField: "Summary", headerField:
"Id" }} ref={(kanban) => { this.kanbanObj = kanban }}>
                <ColumnsDirective>
                    <ColumnDirective headerText="To Do"
keyField="Open"/>
                    <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                    <ColumnDirective headerText="Testing"
keyField="Testing"/>

```

```

        <ColumnDirective headerText="Done"
keyField="Close"/>
    </ColumnsDirective>
</KanbanComponent>
</div>
}
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Auto Height and Width</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban auto height and width" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>

```

```

        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    let kanbanObj;
    function particularColumn() {
        kanbanObj.columns[1].allowToggle = true;
    }
    function column() {
        kanbanObj.columns = [
            { headerText: 'To Do', keyField: 'Open' },
            { headerText: 'Done', keyField: 'Close' }
        ];
    }
    return (<div className='control-wrapper'>
        <ButtonComponent id='particularColumn' className="e-btn"
onClick={particularColumn.bind(this)}>Enable Allow Toggle</ButtonComponent>
        <ButtonComponent id='column' className="e-btn"
onClick={column.bind(this)}>Change Columns</ButtonComponent>
        <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} ref={(kanban)
=> { kanbanObj = kanban; }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';

```

```

import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { kanbanData } from './datasource';
function App() {
    let data = extend([], kanbanData, null, true);
    let kanbanObj: KanbanComponent;
    function particularColumn() {
        kanbanObj.columns[1].allowToggle= true;
    }
    function column() {
        kanbanObj.columns = [
            { headerText: 'To Do', keyField: 'Open' },
            { headerText: 'Done', keyField: 'Close' }
        ]
    }
    return (
        <div className='control-wrapper'>
            <ButtonComponent id='particularColumn' className="e-btn"
onClick={particularColumn.bind(this)} >Enable Allow Toggle</ButtonComponent>
            <ButtonComponent id='column' className="e-btn"
onClick={column.bind(this)}>Change Columns</ButtonComponent>
            <KanbanComponent id="kanban" keyField="Status" dataSource={data}
cardSettings={{ contentField: "Summary", headerField: "Id" }} ref={{(kanban)
=> { kanbanObj = kanban }}}>
                <ColumnsDirective>
                    <ColumnDirective headerText="To Do" keyField="Open"/>
                    <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                    <ColumnDirective headerText="Testing" keyField="Testing"/>
                    <ColumnDirective headerText="Done" keyField="Close"/>
                </ColumnsDirective>
            </KanbanComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Auto Height and Width</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban auto height and width" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Filter cards in React Kanban component

You can filter the collection of cards from the dataSource and display it on the Kanban board by using the [query](#) property.

In the below sample, you can filter the cards based on the 'where' query and display the filtered data to the Kanban board.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query } from '@syncfusion/ej2-data';
import { kanbanData } from '../datasource';
class App extends React.Component {

```



```

kanbanObj;
priorityObj;
constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
    this.priorityData = ['None', 'High', 'Normal', 'Low'];
    this.value = 'None';
}
change(args) {
    let filterQuery = new Query();
    if (args.value !== 'None') {
        if (args.element.id === 'priority') {
            filterQuery = new Query().where('Priority', 'equal',
args.value);
        }
        this.kanbanObj.query = filterQuery;
    }
}
;
render() {
    return <div className='control-wrapper'>
        <DropDownListComponent id='priority' ref={(kanban) => {
this.priorityObj = kanban; }} dataSource={this.priorityData}
change={this.change.bind(this)} value={this.value} placeholder='Select a
priority'></DropDownListComponent>
        <KanbanComponent ref={(kanban) => { this.kanbanObj = kanban; }}
id="kanban" keyField="Status" dataSource={this.data} cardSettings={{
contentField: "Summary", headerField: "Id", showHeader: false }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open"/>
                <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                <ColumnDirective headerText="Testing" keyField="Testing"/>
                <ColumnDirective headerText="Done" keyField="Close"/>
            </ColumnsDirective>
        </KanbanComponent>
    </div>;
}
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { DropDownListComponent, ChangeEventArgs } from '@syncfusion/ej2-
react-dropdowns';
import { Query } from '@syncfusion/ej2-data';
import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{

```

```

private kanbanObj: KanbanComponent;
private priorityObj: DropDownListComponent;
constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
    this.priorityData = ['None', 'High', 'Normal', 'Low'];
    this.value = 'None';
}
change(args: ChangeEventArgs): void {
    let filterQuery: Query = new Query();
    if (args.value !== 'None') {
        if (args.element.id === 'priority') {
            filterQuery = new Query().where('Priority', 'equal',
args.value);
        }
        this.kanbanObj.query = filterQuery;
    }
};
render() {
    return <div className='control-wrapper'>
        <DropDownListComponent id='priority' ref={(kanban) => {
this.priorityObj = kanban; }} dataSource={this.priorityData}
change={this.change.bind(this)} value={this.value} placeholder='Select a
priority'></DropDownListComponent>
        <KanbanComponent ref={(kanban) => { this.kanbanObj = kanban }}
id="kanban" keyField="Status" dataSource={this.data} cardSettings={{
contentField: "Summary", headerField: "Id", showHeader: false }}>
            <ColumnsDirective>
                <ColumnDirective headerText="To Do" keyField="Open" />
                <ColumnDirective headerText="In Progress"
keyField="InProgress" />
                <ColumnDirective headerText="Testing" keyField="Testing"
/>
                <ColumnDirective headerText="Done" keyField="Close" />
            </ColumnsDirective>
        </KanbanComponent>
    </div>
};
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Auto Height and Width</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Kanban auto height and width" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query } from '@syncfusion/ej2-data';
import { kanbanData } from './datasource';
function App() {
    let kanbanObj;
    let priorityObj;
    let data = extend([], kanbanData, null, true);
    let priorityData = ['None', 'High', 'Normal', 'Low'];
    let value = 'None';

```

```

function change(args) {
    let filterQuery = new Query();
    if (args.value !== 'None') {
        if (args.element.id === 'priority') {
            filterQuery = new Query().where('Priority', 'equal',
args.value);
        }
    }
    kanbanObj.query = filterQuery;
}
;
return (<div className='control-wrapper'>
    <DropDownListComponent id='priority' ref={(kanban) => {
priorityObj = kanban; }} dataSource={priorityData}
change={change.bind(this)} value={value} placeholder='Select a
priority'></DropDownListComponent>
    <KanbanComponent ref={(kanban) => { kanbanObj = kanban; }}
id="kanban" keyField="Status" dataSource={data} cardSettings={{
contentField: "Summary", headerField: "Id", showHeader: false }}>
        <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open"/>
            <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
            <ColumnDirective headerText="Testing" keyField="Testing"/>
            <ColumnDirective headerText="Done" keyField="Close"/>
        </ColumnsDirective>
    </KanbanComponent>
</div>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { DropDownListComponent, ChangeEventArgs } from '@syncfusion/ej2-
react-dropdowns';
import { Query } from '@syncfusion/ej2-data';
import { kanbanData } from './datasource';
function App() {
    let kanbanObj: KanbanComponent;
    let priorityObj: DropDownListComponent;
    let data = extend([], kanbanData, null, true);
    let priorityData = ['None', 'High', 'Normal', 'Low'];
    let value = 'None';
    function change(args: ChangeEventArgs): void {
        let filterQuery: Query = new Query();
        if (args.value !== 'None') {
            if (args.element.id === 'priority') {
                filterQuery = new Query().where('Priority', 'equal',
args.value);
            }
        }
    }
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

```

    }
  }
  kanbanObj.query = filterQuery;
};
return (
  <div className='control-wrapper'>
    <DropDownListComponent id='priority' ref={(kanban) => {
priorityObj = kanban; }} dataSource={priorityData}
change={change.bind(this)} value={value} placeholder='Select a
priority'></DropDownListComponent>
    <KanbanComponent ref={(kanban) => { kanbanObj = kanban }}
id="kanban" keyField="Status" dataSource={data} cardSettings={{
contentField: "Summary", headerField: "Id", showHeader: false }}>
      <ColumnsDirective>
        <ColumnDirective headerText="To Do" keyField="Open" />
        <ColumnDirective headerText="In Progress"
keyField="InProgress" />
        <ColumnDirective headerText="Testing" keyField="Testing"
/>
        <ColumnDirective headerText="Done" keyField="Close" />
      </ColumnsDirective>
    </KanbanComponent>
  </div>
);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Auto Height and Width</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban auto height and width" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />

```

```

    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
    <style>
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
    </style>
</head>
<body>
    <div id='kanban'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Search cards in React Kanban component

You can search the cards in Kanban by using the `query` property.

In the following sample, the searching operation starts as soon as you start typing characters in the external text box. It will search the cards based on the `Id` and `Summary` using the `search` query with `contains` operator.

[Class-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { TextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { Query } from '@syncfusion/ej2-data';
import { kanbanData } from '../datasource';
class App extends React.Component {
    textBoxObj;
    kanbanObj;
    constructor() {
        super(...arguments);
        this.data = extend([], kanbanData, null, true);
    }
    searchClick(e) {
        let searchValue = e.value;
        let searchQuery = new Query();

```

```

        if (searchValue !== '') {
            searchQuery = new Query().search(searchValue, ['Id', 'Summary'],
'contains', true);
        }
        this.kanbanObj.query = searchQuery;
    }
;
reset() {
    this.textBoxObj.value = '';
    this.kanbanObj.query = new Query();
}
;
render() {
    return <div className='control-wrapper'>
        <table>
            <tbody>
                <td style={{ width: '200px' }}>
                    <TextBoxComponent id="search" ref={(kanban) => {
this.textBoxObj = kanban; }} showClearButton={true} placeholder="Enter
search text" input={this.searchClick.bind(this)} />
                </td>
                <td>
                    <ButtonComponent id='reset' className="e-btn"
onClick={this.reset.bind(this)}>Reset</ButtonComponent></td>
                </tbody>
            </table>
            <KanbanComponent ref={(kanban) => { this.kanbanObj = kanban; }}
id="kanban" keyField="Status" dataSource={this.data} cardSettings={{
contentField: "Summary", headerField: "Id", showHeader: false }}>
                <ColumnsDirective>
                    <ColumnDirective headerText="To Do" keyField="Open"/>
                    <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                    <ColumnDirective headerText="Testing" keyField="Testing"/>
                    <ColumnDirective headerText="Done" keyField="Close"/>
                </ColumnsDirective>
            </KanbanComponent>
        </div>;
    }
}
;
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { TextBoxComponent, InputEventArgs } from '@syncfusion/ej2-react-
inputs';
import { Query } from '@syncfusion/ej2-data';

```

```

import { kanbanData } from './datasource';
class App extends React.Component<{}, {}>{
  private textBoxObj: TextBoxComponent;
  private kanbanObj: KanbanComponent;
  constructor() {
    super(...arguments);
    this.data = extend([], kanbanData, null, true);
  }
  searchClick(e: InputEventArgs): void {
    let searchValue: string = e.value;
    let searchQuery: Query = new Query();
    if (searchValue !== '') {
      searchQuery = new Query().search(searchValue, ['Id', 'Summary'],
'contains', true);
    }
    this.kanbanObj.query = searchQuery;
  };
  reset(): void {
    this.textBoxObj.value = '';
    this.kanbanObj.query = new Query();
  };
  render() {
    return <div className='control-wrapper'>
      <table>
        <tbody>
          <td style={{ width: '200px' }}>
            <TextBoxComponent id="search" ref={(kanban) => {
this.textBoxObj = kanban; }} showClearButton={true} placeholder="Enter
search text" input={this.searchClick.bind(this)} />
          </td>
          <td>
            <ButtonComponent id='reset' className="e-btn"
onClick={this.reset.bind(this)}>Reset</ButtonComponent></td>
          </tbody>
        </table>
        <KanbanComponent ref={(kanban) => { this.kanbanObj = kanban }}
id="kanban" keyField="Status" dataSource={this.data} cardSettings={{
contentField: "Summary", headerField: "Id", showHeader: false }}>
          <ColumnsDirective>
            <ColumnDirective headerText="To Do" keyField="Open" />
            <ColumnDirective headerText="In Progress"
keyField="InProgress" />
            <ColumnDirective headerText="Testing" keyField="Testing"
/>
            <ColumnDirective headerText="Done" keyField="Close" />
          </ColumnsDirective>
        </KanbanComponent>
      </div>
    }
  };
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.HTML

```
<!DOCTYPE html>
```



```

<html lang="en">
<head>
  <title>Auto Height and Width</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban auto height and width" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';

```

```

import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { TextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { Query } from '@syncfusion/ej2-data';
import { kanbanData } from './datasource';
function App() {
    let textBoxObj;
    let kanbanObj;
    let data = extend([], kanbanData, null, true);
    function searchClick(e) {
        let searchValue = e.value;
        let searchQuery = new Query();
        if (searchValue !== '') {
            searchQuery = new Query().search(searchValue, ['Id', 'Summary'],
'contains', true);
        }
        kanbanObj.query = searchQuery;
    }
    ;
    function reset() {
        textBoxObj.value = '';
        kanbanObj.query = new Query();
    }
    ;
    return (<div className='control-wrapper'>
        <table>
            <tbody>
                <td>
                    <TextBoxComponent id="search" ref={(kanban) => {
textBoxObj = kanban; }} showClearButton={true} placeholder="Enter search
text" input={searchClick.bind(this)} />
                </td>
                <td>
                    <ButtonComponent id='reset' className="e-btn"
onClick={reset.bind(this)}>Reset</ButtonComponent></td>
                </tbody>
            </table>
            <KanbanComponent ref={(kanban) => { kanbanObj = kanban; }}
id="kanban" keyField="Status" dataSource={data} cardSettings={{
contentField: "Summary", headerField: "Id", showHeader: false }}>
                <ColumnsDirective>
                    <ColumnDirective headerText="To Do" keyField="Open"/>
                    <ColumnDirective headerText="In Progress"
keyField="InProgress"/>
                    <ColumnDirective headerText="Testing" keyField="Testing"/>
                    <ColumnDirective headerText="Done" keyField="Close"/>
                </ColumnsDirective>
            </KanbanComponent>
        </div>);
}
ReactDOM.render(<App />, document.getElementById('kanban'));
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { extend } from '@syncfusion/ej2-base';
import { KanbanComponent, ColumnsDirective, ColumnDirective } from
"@syncfusion/ej2-react-kanban";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { TextBoxComponent, InputEventArgs } from '@syncfusion/ej2-react-
inputs';
import { Query } from '@syncfusion/ej2-data';
import { kanbanData } from '../datasource';

function App() {
    let textBoxObj: TextBoxComponent;
    let kanbanObj: KanbanComponent;
    let data = extend([], kanbanData, null, true);
    function searchClick(e: InputEventArgs): void {
        let searchValue: string = e.value;
        let searchQuery: Query = new Query();
        if (searchValue !== '') {
            searchQuery = new Query().search(searchValue, ['Id', 'Summary'],
'contains', true);
        }
        kanbanObj.query = searchQuery;
    };
    function reset(): void {
        textBoxObj.value = '';
        kanbanObj.query = new Query();
    };
    return (
        <div className='control-wrapper'>
            <table>
                <tbody>
                    <td>
                        <TextBoxComponent id="search" ref={(kanban) => {
textBoxObj = kanban; }} showClearButton={true} placeholder="Enter search
text" input={searchClick.bind(this)} />
                    </td>
                    <td>
                        <ButtonComponent id='reset' className="e-btn"
onClick={reset.bind(this)}>Reset</ButtonComponent></td>
                </tbody>
            </table>
            <KanbanComponent ref={(kanban) => { kanbanObj = kanban }}
id="kanban" keyField="Status" dataSource={data} cardSettings={{
contentField: "Summary", headerField: "Id", showHeader: false }}>
                <ColumnsDirective>
                    <ColumnDirective headerText="To Do" keyField="Open" />
                    <ColumnDirective headerText="In Progress" keyField="InProgress"
/>
                    <ColumnDirective headerText="Testing" keyField="Testing" />
                    <ColumnDirective headerText="Done" keyField="Close" />
                </ColumnsDirective>
            </KanbanComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('kanban'));
```

```
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Auto Height and Width</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Kanban auto height and width" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
layouts/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
kanban/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.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>
  <style>
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
  </style>
</head>
<body>
  <div id='kanban'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

Ej1 api migration in React Kanban component

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

Columns

{% raw %}

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

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

| Default | **Property** : *columns* </br></br> <EJ.Kanban><columns></br></columns></EJ.Kanban> |

Property :

columns</br></br><KanbanComponent><ColumnsDirective></br></ColumnsDirective></KanbanComponent></br> |

| Header Text | **Property** : *headerText* </br> </br> <EJ.Kanban><columns> </br><column headerText="Backlog"></br></column></br></columns></EJ.Kanban> | **Property** : *headerText* </br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>headerText="Backlog"/></br></ColumnsDirective></br></KanbanComponent></br> |

| Key Field | **Property** : *key* </br></br> <EJ.Kanban><columns></br><column key="Open"></br></column></br></columns></EJ.Kanban> | **Property**: *keyField* </br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>keyField='Open'/></br></ColumnsDirective></br></KanbanComponent></br> |

| Initial Collapsed</br>Columns | **Property**: *isCollapsed*</br></br><EJ.Kanban><columns></br><column key="Open"></br>headerText="Backlog" </br>[isCollapsed]="true"></br></column></br></columns></EJ.Kanban> | **Property**: *isExpanded* </br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>keyField='Open'/></br>allowToggle='true'</br>[isExpanded]='true'/></br></ColumnsDirective></br></KanbanComponent></br> |

| Cell Add card button | **Property**: *showAddButton* </br></br><EJ.Kanban><columns></br><column key="Open"></br>headerText="Backlog" </br>[showAddButton]="true"></br></column></br></columns></EJ.Kanban> | **Property**: *showAddButton* </br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>headerText='To do'</br>keyField='Open'</br>[showAddButton]='true'/></br></ColumnsDirective></br></KanbanComponent></br> |

| Column card count | **Property**: *enableTotalCount* </br></br><EJ.Kanban [enableTotalCount]="true"></br></br></EJ.Kanban> | **Property**: *showItemCount* </br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>headerText='To do'</br>keyField='Open'</br>[showItemCount]='true'/></br></ColumnsDirective></br></KanbanComponent></br> |

| Template | **Property**: *headerTemplate* </br></br><EJ.Kanban><columns></br><column key="Open"></br>headerText="Backlog" </br>headerTemplate="#template"></br></column></br>

```

r></columns></EJ.Kanban> | Property:
template</br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>headerText='To
do'</br>keyField='Open'</br>template='#headerTemplate'/></br></ColumnsDirective></br></K
anbanComponent></br> |

```

```

| Allow Drop | Property: allowDrop </br> </br><EJ.Kanban><columns></br><column
key="Open"</br>headerText="Backlog"</br>[allowDrop]="false"></br></column></br></column
ns></EJ.Kanban> | Property: allowDrop
</br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>headerTex
t='To
do'</br>keyField='Open'</br>[allowDrop]="false"/></br></ColumnsDirective></br></KanbanCo
mponent></br> |

```

```

| Allow Drag | Property: allowDrag </br> </br><EJ.Kanban><columns></br><column
key="Open"</br>headerText="Backlog"</br>[allowDrag]="false"></br></column></br></column
ns></EJ.Kanban> | Property: allowDrag
</br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>headerTex
t='To
do'</br>keyField='Open'</br>[allowDrag]="false"/></br></ColumnsDirective></br></KanbanCo
mponent></br> |

```

```

| Total Count text | Property: totalCount </br> </br><EJ.Kanban><columns></br><column
key="Open"</br>headerText="Backlog"</br>totalCount-text="Backlog
Count"></br></column></br></columns></EJ.Kanban> | Not Available |

```

```

| Width | Property: width</br></br><EJ.Kanban><columns></br><column
key="Open"</br>headerText="Backlog"</br>width="200"></br></column></br></columns></EJ
.Kanban> | Not Available |

```

```

| Visible | Property: visible</br></br><EJ.Kanban><columns></br><column
key="Open"</br>headerText="Backlog"</br>visible={false}></br></column></br></columns></E
J.Kanban> | Not Available |

```

```

| Add/Delete Columns | Method: columns(column, key,</br> [action])</br></br>Add
:</br><EJ.Kanban></br></EJ.Kanban></br>var kanbanObj =</br> $$(".e-
kanban").</br>ejKanban("instance");</br>kanbanObj.columns</br>("Review", "Review",
"add");</br></br>Delete: </br>kanbanObj.columns</br>("Review", "Review", "remove");</br> |
Method: addColumn(columnOptions,</br> index)</br></br><KanbanComponent</br>ref={ kanban
=> this.</br>kanbanInstance =
kanban}></br></KanbanComponent></br>this.kanbanInstance</br>.addColumn({ </br>headerText:
"Review",</br> keyField: "Review"</br>}, 2);</br></br>Method: deleteColumn(index)
</br></br>this.kanbanInstance.deleteColumn(2); |

```

```

| Show Columns | Method: showColumns(headerText)
</br></br><EJ.Kanban></br></EJ.Kanban></br>var kanbanObj =</br> $$(".e-
kanban").</br>ejKanban("instance");</br>kanbanObj.</br>showColumns("Testing");</br></br> |
Method: showColumn(key) </br></br><KanbanComponent</br>ref={ kanban

```

```

=></br>this.kanbanInstance</br>=
kanban}></br></KanbanComponent></br>this.kanbanInstance</br>.showColumn</br>("Testing");</br>
|
| Hide Columns | Method: hideColumns(headerText)
</br></br><EJ.Kanban></br></EJ.Kanban></br>var kanbanObj =</br> $("e-
kanban").</br>ejKanban("instance");</br>kanbanObj.</br>hideColumns("Testing");</br></br> |
Method: hideColumns(key) </br></br><KanbanComponent</br>ref={ kanban
=></br>this.kanbanInstance =</br>
kanban}></br></KanbanComponent></br>this.kanbanInstance</br>.hideColumn</br>("Testing");</br>
|
| Get Visible</br>Column Names | Method:
getVisibleColumnNames()</br></br><EJ.Kanban></br></EJ.Kanban></br>var kanbanObj =</br> $("e-
kanban").</br>ejKanban("instance");</br>kanbanObj.</br>getVisibleColumnNames();</br></br> | Not
Applicable |
| Get Column</br>By Header Text | Method:
getColumnByHeaderText</br>(headerText)</br></br><EJ.Kanban></br></EJ.Kanban></br>var
kanbanObj =</br> $("e-
kanban").</br>ejKanban("instance");</br>kanbanObj.</br>getColumnByHeaderText</br>("Testing");</br>
| Not Applicable |
| Get Column Data | Not Applicable | Method:
getColumnData()</br></br><KanbanComponent</br>ref={ kanban
=></br>this.kanbanInstance</br>=
kanban}></br></KanbanComponent></br>this.kanbanInstance</br>.getColumnData();</br>
| Triggers after</br>cell is click | Event:
cellClick</br></br><EJ.Kanban</br>cellClick={cellClick}></br></EJ.Kanban></br>function
cellClick(args){}</br> | Not Applicable |

```

Cards

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

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

```

| Card unique field | Property : </br>fields.primaryKey</br></br><EJ.Kanban</br>fields-
primaryKey="Id"></br></EJ.Kanban> | Property :
</br>cardSettings.headerField</br><KanbanComponent</br>cardSettings={{</br>headerField:
"Id"}}></br></KanbanComponent></br> |
| Content | Property: </br>fields.content </br></br><EJ.Kanban</br>fields-
content</br>="Summary"></br></EJ.Kanban> | Property :
</br>cardSettings.contentField</br><KanbanComponent</br>cardSettings={{</br>contentField:
'Summary'></br></KanbanComponent></br> |
| Tag | Property: </br>fields.tag </br></br><EJ.Kanban</br>fields-tag="Tags"></br></EJ.Kanban> |
Property : </br>cardSettings.tagsField</br><KanbanComponent</br>cardSettings={{</br>tagsField:
'Tags'></br></KanbanComponent></br> |

```

| Left border color | **Property:** `</br>fields.color</br></br><EJ.Kanban fields-color="Type" cardSettings-colorMapping={colormap}></br></EJ.Kanban></br>var colormap = {</br>"cb2027": "Bug,Story",</br>"#67ab47": "Improvement",</br>"#fbae19": "Epic",</br>"#6a5da8": "Others"</br>};</br>` | **Property:**

`</br>cardSettings.grabberField</br><KanbanComponent</br>cardSettings={{</br>grabberField: "color"></br></KanbanComponent></br>` |

| Header | **Property:** `</br>fields.title</br></br><EJ.Kanban</br> fields-title="Assignee"</br></EJ.Kanban>` | Card Unique mapping</br> field is displayed </br>on card header. |

| Image | **Property:** `</br>fields.imageUrl</br></br><EJ.Kanban</br> fields-imageUrl="ImgUrl"</br></EJ.Kanban></br>` | Not Applicable |

| CSS class | **Not Applicable** | **Property :** `</br>cardSettings.</br>footerCssField</br></br><KanbanComponent</br>cardSettings={{</br>footerCssField: "classNames"></br></KanbanComponent></br>` |

| Template | **Property:** `</br>cardSettings.template</br></br><EJ.Kanban</br>cardSettings-template</br>="#template"</br></EJ.Kanban>` | **Property:** `</br>cardSettings.template</br></br><KanbanComponent</br>cardSettings={{</br>template:</br>>this.cardTemplate.bind(this)></br></KanbanComponent></br>` |

| Toggle Card | **Method:** `toggleCard</br>($div or id)</br></br><EJ.Kanban></br></EJ.Kanban></br>var kanbanObj =</br>$(".e-kanban").</br>ejKanban("instance");</br>kanbanObj.</br>toggleCard("2");</br></br>` | **Not Applicable** |

| Get Card Details | **Not Applicable** | **Method:** `</br>getCardDetails(target)</br></br><KanbanComponent</br>ref={ kanban =></br>this.kanbanInstance =</br>kanban}></br></KanbanComponent></br>this.kanbanInstance</br>.getCardDetails(</br>obj.element</br>.querySelector(".e-card"));` |

| Get Selected Cards | **Not Applicable** | **Method:** `</br>getSelectedCards()</br></br><KanbanComponent</br>ref={ kanban =></br>this.kanbanInstance =</br>kanban}></br></KanbanComponent></br>this.kanbanInstance</br>.getSelectedCards();` |

| Card Click | **Event:** `cardClick</br></br><EJ.Kanban</br>cardClick={cardClick}></br></EJ.Kanban></br>function cardClick(args){</br>` | **Event:** `cardClick` `</br></br><KanbanComponent</br>cardClick={this.cardClick.bind(this)}</br></KanbanComponent></br>private cardClick() {}</br>` |

| Card Double Click | **Event:** `cardDoubleClick</br></br><EJ.Kanban</br>cardDoubleClick={cardDoubleClick}></br></EJ.Kanban></br>function cardDoubleClick(args){</br>` | **Event:** `cardDoubleClick` `</br></br><KanbanComponent</br>cardDoubleClick={this.cardDoubleClick.bind(this)}</br></KanbanComponent></br>private cardDoubleClick() {}</br>` |

| Triggers when start the drag | **Event:**
`cardDragStart`
 <EJ.Kanban>
 cardDragStart={cardDragStart}</EJ.Kanban>
 function cardDragStart(args){ } | **Event:**
`dragStart`
 <KanbanComponent>
 dragStart={this.dragStart.bind(this)}</KanbanComponent>
 private dragStart() { } |

| Triggers when card is dragged | **Event:**
`cardDrag`
 <EJ.Kanban>
 cardDrag={cardDrag}</EJ.Kanban>
 function cardDrag(args){ } | **Event:** `drag`
 <KanbanComponent>
 drag={this.drag.bind(this)}</KanbanComponent>
 private drag() { } |

| Triggers when card dragging stops | **Event:**
`cardDragStop`
 <EJ.Kanban>
 cardDragStop={cardDragStop}</EJ.Kanban>
 function cardDragStop(args){ } | **Event:**
`dragStop`
 <KanbanComponent>
 dragStop={this.dragStop.bind(this)}</KanbanComponent>
 private dragStop() { } |

| Triggers after save the data when dropped | **Event:**
`cardDrop`
 <EJ.Kanban>
 cardDrop={cardDrop}</EJ.Kanban>
 function cardDrop(args){ } | **Not Applicable** |

| Triggers after cell is click | **Event:**
`cellClick`
 <EJ.Kanban>
 cellClick={cellClick}</EJ.Kanban>
 function cellClick(args){ } | **Not Applicable** |

| Triggers each card rendered | **Event:**
`queryCellInfo`
 <EJ.Kanban>
 queryCellInfo={queryCellInfo}</EJ.Kanban>
 function queryCellInfo(args){ } | **Event:**
`cardRendered`
 <KanbanComponent>
 cardRendered={this.cardRendered.bind(this)}</KanbanComponent>
 private cardRendered() { } |

DataSource

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

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

| Card unique field | **Property** : `fields.primaryKey`
 <EJ.Kanban>
 fields-
 primaryKey="Id"</EJ.Kanban> | **Property** :
 <KanbanComponent>
 cardSettings={{headerField:
 "Id"}}</KanbanComponent> |

| **DataSource** | **Property:** `dataSource`
 <EJ.Kanban>
 dataSource={window.kanbanData}</EJ.Kanban> | **Method:**
 <EJ.Kanban>
 <EJ.Kanban>
 var kanbanObj = \$(".e-kanban").ejKanban("instance");
 kanbanObj.dataSource(newDataSource);</br>
 > | **Property:** `dataSource`
 <KanbanComponent>
 dataSource={this.data}</KanbanComponent> | **Method:**
 <KanbanComponent>
 ref={ kanban
 =>this.kanbanInstance } |

```
kanban}></br></KanbanComponent></br>this.kanbanInstance</br>.dataSource(newDataSource);</br>|
```

| Triggers before</br>data load | **Event:**

```
load</br></br><EJ.Kanban</br>load={load}></br></EJ.Kanban></br>function load(args){</br>|
```

Event:

```
dataBinding</br></br><KanbanComponent</br>dataBinding={this.dataBinding.bind(this)}</br></KanbanComponent></br>private dataBinding() {}</br>|
```

| Triggers after</br>data bounded | **Event:**

```
dataBound</br></br><EJ.Kanban</br>dataBound={dataBound}></br></EJ.Kanban></br>function dataBound(args){</br>| Event:
```

```
dataBound</br></br><KanbanComponent</br>dataBound={this.dataBound.bind(this)}</br></KanbanComponent></br>private dataBound() {}</br>|
```

Common

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

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

| Drag And Drop | **Property:**

```
allowDragAndDrop</br></br><EJ.Kanban</br>allowDragAndDrop={true}></br></EJ.Kanban>|
```

Property:

```
allowDragAndDrop</br></br><KanbanComponent</br>allowDragAndDrop={true}></br></KanbanComponent></br>|
```

| Key Field | **Property:** *keyField*</br></br><EJ.Kanban</br>keyField="Status"></br></EJ.Kanban> | **Property :**

```
keyField</br></br><KanbanComponent</br>keyField="Status"></br></KanbanComponent></br>|
```

| Title | **Property:** *allowTitle*</br></br><EJ.Kanban</br>allowTitle={true}></br></EJ.Kanban> | **Not Applicable** |

| CssClass | **Property:** *cssClass*</br></br><EJ.Kanban</br>cssClass="gradient-green"></br></EJ.Kanban> | **Property:** *cssClass*</br></br><KanbanComponent</br>cssClass="custom-class"></br></KanbanComponent></br>|

| Print | **Property:**

```
allowPrinting</br></br><EJ.Kanban</br>allowPrinting={true}></br></EJ.Kanban></br></br>Method: print()</br></br><EJ.Kanban></br></EJ.Kanban></br>var kanbanObj =</br> $(".e-kanban").</br>ejKanban("instance");</br>kanbanObj.</br>print(); | Not Applicable |
```

| Touch | **Property:**

```
enableTouch</br></br><EJ.Kanban</br>enableTouch={true}></br></EJ.Kanban></br></br>| Not Applicable |
```

| Locale | **Property:** *locale*</br></br><EJ.Kanban</br>locale="de-DE"></br></EJ.Kanban> |

Property: *locale*</br></br><KanbanComponent</br>locale="de-DE"></br></KanbanComponent></br>|

| Query | **Property:** `query`

```
<EJ.Kanban query={query}></EJ.Kanban>
var query = ej.Query().select("");
| Property:
query</>
<KanbanComponent query={query}></KanbanComponent>
var query = new Query().select("");</> |
```

| Refresh | **Method:**

```
</>refresh([templateRefresh])</>
<EJ.Kanban></EJ.Kanban>
var kanbanObj = $(".e-kanban").ejKanban("instance");
kanbanObj.refresh();
| Method:
refresh()</>
<KanbanComponent ref={ kanban =>this.kanbanInstance}>=
kanban}></KanbanComponent>
this.kanbanInstance.refresh();</> |
```

| Refresh Template | **Method:**

```
</>refreshTemplate()</>
<EJ.Kanban></EJ.Kanban>
var kanbanObj = $(".e-kanban").ejKanban("instance");
kanbanObj.refreshTemplate();
| Not Applicable |
```

| Destroy | **Method:** `destroy()`

```
<EJ.Kanban></EJ.Kanban>
var kanbanObj = $(".e-kanban").ejKanban("instance");
kanbanObj.destroy();
| Method:
destroy()</>
<KanbanComponent ref={ kanban =>this.kanbanInstance}>=
kanban}></KanbanComponent>
this.kanbanInstance.destroy();</> |
```

| Get Header Table | **Method:** `getHeaderTable()`

```
<EJ.Kanban></EJ.Kanban>
var kanbanObj = $(".e-kanban").ejKanban("instance");
kanbanObj.getHeaderTable();
| Not Applicable |
```

| Show Spinner | **Not Applicable** | **Method:** `showSpinner()`

```
<KanbanComponent ref={
kanban =>this.kanbanInstance}>=
kanban}></KanbanComponent>
this.kanbanInstance.showSpinner();</> |
```

| Hide Spinner | **Not Applicable** | **Method:** `hideSpinner()`

```
<KanbanComponent ref={
kanban =>this.kanbanInstance}>=
kanban}></KanbanComponent>
this.kanbanInstance.hideSpinner();</> |
```

| Triggers before every action | **Event:**

```
actionBegin</>
<EJ.Kanban>actionBegin={actionBegin}</EJ.Kanban>
function actionBegin(args){</>
| Event:
actionBegin</>
<KanbanComponent>actionBegin={this.actionBegin.bind(this)}</>
</KanbanComponent>
private actionBegin(event) {}</> |
```

| Triggers on successfully completion of actions | **Event:**

```
actionComplete</>
<EJ.Kanban>actionComplete={actionComplete}</EJ.Kanban>
function actionComplete(args){</>
| Event:
actionComplete</>
<KanbanComponent>actionComplete={this.actionComplete.bind(t
his)}</>
</KanbanComponent>
private actionComplete(event) {}</> |
```

| Triggers on action failure | **Event:**

```
actionFailure</>
<EJ.Kanban>actionFailure={actionFailure}</EJ.Kanban>
function actionFailure(args){</>
| Event:
actionFailure</>
<KanbanComponent>actionFailure={this.actionFailure.bind(this)}</>
</KanbanComponent>
private actionFailure(event) {}</> |
```

| Triggers after Kanban rendered | **Event:**

```
create</br></br><EJ.Kanban</br>create={create}></br></EJ.Kanban></br>function
```

```
create(args){</br> | Event:
```

```
created</br></br><KanbanComponent</br>created={this.created.bind(this)}></br></KanbanComp  
onent></br>private created(event) {}</br> |
```

| Triggers when header click | **Event:**

```
headerClick</br></br><EJ.Kanban</br>headerClick={headerClick}></br></EJ.Kanban></br>function
```

```
headerClick(args){</br> | Not Applicable |
```

| Triggers when destroy | **Event:**

```
destroy</br></br><EJ.Kanban</br>destroy={destroy}></br></EJ.Kanban></br>function
```

```
destroy(args){</br> | Event:
```

```
destroy</br></br><KanbanComponent</br>destroy={this.destroy.bind(this)}></br></KanbanComp  
onent></br>private destroy(event) {}</br> |
```

Swimlane

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

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

| Default | **Property:** *swimlaneKey*</br></br><EJ.Kanban</br>fields-

```
swimlaneKey="Assignee"></br></EJ.Kanban> | Property:
```

```
keyField</br></br><KanbanComponent</br>swimlaneSettings={{</br>keyField: "Assignee" }}  
></br></KanbanComponent></br> |
```

| Header | **Property:**

```
headers</br></br><EJ.Kanban</br>headers={headers}></br></EJ.Kanban></br>var headers =
```

```
</br>[ { </br>text: "Andrew", </br>key: "Andrew Fuller" } ]</br> } </br> | Property:
```

```
textField</br></br><KanbanComponent</br>swimlaneSettings={{</br>textField: "AssigneeName"  
}} ></br></KanbanComponent></br> |
```

| Drag And Drop | **Property:**

```
allowDragAndDrop</br></br><EJ.Kanban</br>swimlaneSettings={{allowDragAndDrop:
```

```
true}}></br></EJ.Kanban> | Property:
```

```
allowDragAndDrop</br></br><KanbanComponent</br>swimlaneSettings={{</br>allowDragAndDr  
op: true}} ></br></KanbanComponent></br> |
```

| Card Count | **Property:** *showCount*</br></br><EJ.Kanban</br>swimlaneSettings={{showCount:

```
true}}></br></EJ.Kanban> | Property:
```

```
showItemCount</br></br><KanbanComponent</br>swimlaneSettings={{</br>showItemCount:  
true}} ></br></KanbanComponent></br> |
```

| Empty Row | **Property:**

```
</br>showEmptySwimlane</br></br><EJ.Kanban</br>swimlaneSettings={{showEmptySwimlane:
```

```
true}}></br></EJ.Kanban> | Property:
```

```
showEmptyRow</br></br><KanbanComponent</br>swimlaneSettings={{</br>showEmptyRow:  
true}} ></br></KanbanComponent></br> |
```

| Sorting | **Not Available** | **Property:**

```
</br>sortDirection</br></br><KanbanComponent</br>swimlaneSettings={{</br>sortDirection:</br>
>"Descending"}} ></br></KanbanComponent></br> |
```

| Expand All | **Method:** *expandAll()*</br></br><EJ.Kanban></br></EJ.Kanban></br>var kanbanObj
=</br> \$\$(".e-kanban").</br>ejKanban("instance");</br>kanbanObj.KanbanSwimlane</br>.expandAll();
| **Not Applicable** |

| Collapse All | **Method:** *collapseAll()*</br></br><EJ.Kanban></br></EJ.Kanban></br>var kanbanObj
=</br> \$\$(".e-kanban").</br>ejKanban("instance");</br>kanbanObj.KanbanSwimlane</br>.collapseAll();
| **Not Applicable** |

| Toggle | **Method:** *toggle(\$div or key)*</br></br><EJ.Kanban></br></EJ.Kanban></br>var kanbanObj
=</br> \$\$(".e-
kanban").</br>ejKanban("instance");</br>kanbanObj.</br>KanbanSwimlane</br>.toggle(\$(".e-
slexpandcollapse")</br>.eq(1)); | **Not Applicable** |

| Get Swimlane Data | **Not Applicable** | **Method:**
</br>getSwimlaneData(keyField)</br></br><KanbanComponent</br>ref={ kanban
=></br>this.kanbanInstance</br>=
kanban}></br></KanbanComponent></br>this.kanbanInstance</br>.getSwimlaneData("Janet");</br>
> |

| Triggers before swimlane</br>icon click event | **Event:**
swimlaneClick</br></br><EJ.Kanban</br>swimlaneClick={swimlaneClick}></br></EJ.Kanban></br>
function swimlaneClick(args){}</br> | **Not Applicable** |

Stacked Headers

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

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

| Multiple stacked headers | **Property:**
stackedHeaderColumns</br></br><EJ.Kanban</br>stackedHeaderRows={stackedHeaderRow}></br>
></EJ.Kanban></br>var stackedHeaderRow = </br>[{ stackedHeaderColumns: [{</br> headerText:
"Status",</br>column: "Backlog,</br>In Progress, Testing, </br>Done"}] },</br> {
stackedHeaderColumns: [{</br> headerText: "Unresolved",</br>column: "Backlog,</br>In
Progress"}] }]; | **Not Applicable** |

| Single Stacked Header | **Property:**
stackedHeaderColumns</br></br><EJ.Kanban</br>stackedHeaderRows={stackedHeaderRow}></br>
></EJ.Kanban></br>var stackedHeaderRow = </br>[{ stackedHeaderColumns: [{</br> headerText:
"Status",</br>column: "Backlog,</br>In Progress, Testing, </br>Done"}] },</br> {
stackedHeaderColumns: [{</br>headerText: "Unresolved",</br>column: "Backlog,</br>In
Progress"}] }]; | **Property:**
</br>stackedHeaders</br><KanbanComponent></br><StackedHeadersDirective></br><StackedHe
aderDirective</br>text='To
Do'</br>keyFields='Open,</br>InProgress'></br></StackedHeaderDirective></br></KanbanCom
ponent></br> |

WIP Validation

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

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

| Constraints Type | **Property:**

```
</br>constraints.type</br></br><EJ.Kanban></br><columns></br><column</br>headerText="Backlog"</br>key="Open"</br>constraints-type</br>="swimlane" /></br></EJ.Kanban> | Property:
</br>constraintType</br></br><KanbanComponent</br>constraintType="swimlane"></br></KanbanComponent></br> |
```

| Maximum card Count</br>at particular</br>column/swimlane | **Property:**

```
</br>constraints.max</br></br><EJ.Kanban></br><columns></br><column</br>headerText="Backlog"</br>key="Open"</br>constraints-type</br>="swimlane" /> constraints-max</br>="5" /></br></columns></EJ.Kanban> | Property:
</br>maxCount</br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>headerText='Backlog'</br>keyField='Open'</br>maxCount='5'></br></ColumnsDirective></br></KanbanComponent></br> |
```

| Minimum card Count</br>at particular column |

```
Property:</br>constraints.min</br></br><EJ.Kanban></br><columns></br><column</br>headerText="Backlog"</br>key="Open"</br>constraints-type</br>="swimlane" /> constraints-min</br>="5" /></br></columns></EJ.Kanban> | Property:
minCount</br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>headerText='Backlog'</br>keyField='Open'</br>minCount='5'></br></ColumnsDirective></br></KanbanComponent></br> |
```

Keyboard

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

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

| KeyBoard | **Property:**

```
</br>allowKeyboardNavigation</br></br><EJ.Kanban</br>allowKeyboardNavigation={true}></br></EJ.Kanban> | Property:
allowDragAndDrop</br></br><KanbanComponent</br>allowKeyboard={true}></br></KanbanComponent></br> |
```

| Settings | **Property:**

```
</br>keySettings</br></br><EJ.Kanban</br>keySettings={keySettings}></br></EJ.Kanban></br>var keySettings = {</br> focus: "e",</br> insertCard: "45"</br>} | Not Applicable |
```

Toggle Columns

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

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

| Default | **Property:**

```
</br>allowToggleColumn</br></br><EJ.Kanban</br>allowToggleColumn={true}></br></EJ.Kanban> | Property:
</br>allowToggle</br></br><KanbanComponent></br><ColumnsDirective></br><ColumnDirective</br>allowToggle={true}></br></ColumnsDirective></br></KanbanComponent></br> |
```

| Toggle | **Method:** `toggleColumn` *(headerText or \$div)* `</EJ.Kanban>` `</EJ.Kanban>` `var kanbanObj = $(".e-kanban").ejKanban("instance");` `kanbanObj.toggleColumn("Backlog");` | **Not Applicable** |

Dialog Editing

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

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

| Fields | **Property:** `editItems` `</EJ.Kanban>` `editSettings-` `editItems={editItems}` `</EJ.Kanban>` `var editItems = []` | **Property:** `fields` `<KanbanComponent>` `dialogSettings={{fields:` `this.fields}}` `</KanbanComponent>` `private fields:DialogFieldsModel[] = []` |

| Dialog Model | **Not Available** | **Property:** `model` `<KanbanComponent>` `dialogSettings={{model:` `{model}}` `</KanbanComponent>` `private model:DialogModel[] = { width: 250 }` |

| Template | **Property:** `dialogTemplate` `</EJ.Kanban>` `editSettings-` `dialogTemplate="#template"` `</EJ.Kanban>` | **Property:** `template` `<KanbanComponent>` `dialogSettings={{template:` `this.dialogTemplate}}` `</KanbanComponent>` |

| Enable Editing | **Property:** `allowEditing` `</EJ.Kanban>` `editSettings-` `allowEditing={true}` `</EJ.Kanban>` | **In default allowed for editing.** |

| Enable Adding | **Property:** `allowAdding` `</EJ.Kanban>` `editSettings-` `allowAdding={true}` `</EJ.Kanban>` | **Adding applicable using column** `show add button` `or` `public method.` |

| Edit Mode | **Property:** `editMode` `</EJ.Kanban>` `editSettings-` `editMode="dialogtemplate"` `</EJ.Kanban>` | **Not Applicable** |

| External Form template |

Property: `externalFormTemplate` `</EJ.Kanban>` `editSettings-` `externalFormTemplate="#template"` `</EJ.Kanban>` | **Not Applicable** |

| External Form Position | **Property:**

`externalFormPosition` `</EJ.Kanban>` `editSettings-` `externalFormPosition="bottom"` `</EJ.Kanban>` | **Not Applicable** |

| Add Card | **Method:** `KanbanEdit.addCard` `([primaryKey], [card])` `</EJ.Kanban>` `</EJ.Kanban>` `var kanbanObj = $(".e-kanban").ejKanban("instance");` `kanbanObj.KanbanEdit.addCard("2",{ Id: 2, Status: Open});` | **Method:** `addCard` `(cardData)` `<KanbanComponent>` `ref={ kanban =>this.kanbanInstance}` `kanban}` `</KanbanComponent>` `this.kanbanInstance.addCard({ Id: 2, Status: Open});` |

| Update Card | **Method:** `updateCard(key, data)`

```

<EJ.Kanban>
  </EJ.Kanban>
  var kanbanObj = $(".e-kanban").ejKanban("instance");
  kanbanObj.KanbanEdit.updateCard(2, { Id: 2, Status: Open});
  | Method:
  updateCard(cardData)
  <KanbanComponent
    ref={ kanban
  =>
    this.kanbanInstance
  kanban}
  </KanbanComponent>
  this.kanbanInstance.updateCard({ Id: 2, Status: Open});
  |
  | Delete Card | Method:
  <KanbanEdit.deleteCard(key)
  <EJ.Kanban>
  </EJ.Kanban>
  var kanbanObj = $(".e-kanban").ejKanban("instance");
  kanbanObj.KanbanEdit.deleteCard(2);
  | Method: deleteCard()
  <KanbanComponent
    ref={ kanban
  =>
    this.kanbanInstance
  kanban}
  </KanbanComponent>
  this.kanbanInstance.deleteCard(2);
  |
  | Cancel Edit | Method:
  <KanbanEdit.cancelEdit()
  <EJ.Kanban>
  </EJ.Kanban>
  var kanbanObj = $(".e-kanban").ejKanban("instance");
  kanbanObj.KanbanEdit.cancelEdit();
  | Not Available |
  | End Edit | Method:
  <KanbanEdit.endEdit()
  <EJ.Kanban>
  </EJ.Kanban>
  var kanbanObj = $(".e-kanban").ejKanban("instance");
  kanbanObj.KanbanEdit.endEdit();
  | Not Available |
  | Start Edit | Method:
  <KanbanEdit.startEdit>($div or key)
  <EJ.Kanban>
  </EJ.Kanban>
  var kanbanObj = $(".e-kanban").ejKanban("instance");
  kanbanObj.KanbanEdit.startEdit(2);
  | Method:
  <openDialog(action, data)
  <KanbanComponent
    ref={ kanban
  =>
    this.kanbanInstance
  kanban}
  </KanbanComponent>
  this.kanbanInstance.openDialog("Add");
  |
  | Set Validation | Method:
  KanbanEdit.setValidationToField(name, rules)
  <EJ.Kanban>
  </EJ.Kanban>
  var kanbanObj = $(".e-kanban").ejKanban("instance");
  kanbanObj.KanbanEdit.setValidationToField("Summary", { required: true });
  | Not Available |
  | Close Dialog | Not Applicable | Method:
  closeDialog()
  <KanbanComponent
    ref={ kanban
  =>
    this.kanbanInstance
  kanban}
  </KanbanComponent>
  this.kanbanInstance.closeDialog();
  |
  | Triggers before dialog Open | Not Applicable | Event:
  dialogOpen
  <KanbanComponent
    dialogOpen={this.dialogOpen.bind(this)}
  </KanbanComponent>
  private dialogOpen(event) {}
  |
  | Triggers when dialog close | Not Applicable | Event:
  dialogClose
  <KanbanComponent
    dialogClose={this.dialogClose.bind(this)}
  </KanbanComponent>
  private dialogClose(event) {}
  |

```


| Triggers after the card is edited | **Event:**

`endEdit`
`endEdit(args)` | **Not Applicable** |

| Triggers after the card is deleted | **Event:**

`endDelete`
`endDelete(args)` | **Not Applicable** |

| Triggers before task is edited | **Event:** `beginEdit`

`beginEdit(args)` | **Not Applicable** |

Dialog Editing Fields

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

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

| Fields | **Property:** `editItems`
`editItems` | **Property:**

`fields`
`this.fields`
`private fields: DialogFieldsModel[] = []` |

| Mapping key | **Property:** `field`
`editItems` | **Property:**

`key`
`this.fields`
`private fields: DialogFieldsModel[] = [{key: "Id"}]` |

| Label | **Not Applicable** | **Property:**

`text`
`this.fields`
`private fields: DialogFieldsModel[] = [{text: "ID", key: "Id"}]` |

| Type | **Property:** `editType`
`editItems` | **Property:**

`ej.Kanban.EditingType.String`
`type`
`this.fields`
`private fields: DialogFieldsModel[] = [{type: "TextBox", key: "Id"}]` |

| Validation Rules | **Property:** `validationRules`
`editItems` | **Property:**

`required: true`
`validationRules`
`this.fields`
`private fields: DialogFieldsModel[] = [{validationRules: {required: true}}]` |

| Params | **Property:** `editParams`
`editItems` | **Property:**

`decimalPlaces: 2` | **Not Applicable** |

| Default value | **Property:** `defaultValue`
`<EJ.Kanban editSettings-
editItems={editItems}></EJ.Kanban>`
`var editItems = [{
"Open"}]` | **Not Applicable** |

Tooltip

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

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

| Default | **Property:** `tooltipSettings.enable`
`<EJ.Kanban tooltipSettings-
enable={true}></EJ.Kanban>` |

Property: `enableTooltip`
`<KanbanComponent enableTooltip={true}></KanbanComponent>` |

| Template | **Property:** `tooltipSettings.template`
`<EJ.Kanban tooltipSettings-
template=#tooltipTemplate></EJ.Kanban>` | **Property:**
`tooltipTemplate`
`<KanbanComponent tooltipTemplate={this.template.bind(thi
s)}></KanbanComponent>` |

Context Menu

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

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

| Default | **Property:** `enable`
`<EJ.Kanban contextMenuSettings-
enable={true}></EJ.Kanban>` | **Not Applicable** |

| Menu Items | **Property:** `menuItems`
`<EJ.Kanban contextMenuSettings-
enable={true} contextMenuSettings-
menuItems={menuItem}></EJ.Kanban>`
`var menuItem = ["Move Right"]; | Not Applicable |`

| Disable default Items | **Property:**
`disableDefaultItems`
`<EJ.Kanban contextMenuSettings-
enable={true} contextMenuSettings-
disableDefaultItems={disableDefaultItems}></EJ.Kanban>`
`var disableDefaultItems
=<ej.Kanban.MenuItem.MoveLeft]; | Not Applicable |`

| Custom Menu Items | **Property:**
`customMenuItems`
`<EJ.Kanban contextMenuSettings-
enable={true} contextMenuSettings-
customMenuItems={customMenuItems}></EJ.Kanban>`
`var customMenuItems
=<[{ text: "Menu1"}]></EJ.Kanban contextMenuSettings-
enable={true} contextMenuSettings-
customMenuItems={customMenuItems}></EJ.Kanban>`
`var customMenuItems
=<[{ target: ej.Kanban.Target.Header
}></EJ.Kanban contextMenuSettings-
enable={true} contextMenuSettings-
customMenuItems={customMenuItems}></EJ.Kanban>`
`var customMenuItems
=<[{ text: "Hide Column", template: "#template"}]></EJ.Kanban>` | **Not Applicable** |

| Triggers when context menu item click | **Event:**

`contextClick` | `EJ.Kanban` | `contextClick={contextClick}` | `EJ.Kanban` | `function contextClick(args){}` | **Not Applicable** |

| Triggers when context menu open | **Event:**

`contextOpen` | `EJ.Kanban` | `contextOpen={contextOpen}` | `EJ.Kanban` | `function contextOpen(args){}` | **Not Applicable** |

WorkFlows

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

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

| Default | **Property:**

`workFlows` | `EJ.Kanban` | `workflows={workflow}` | `EJ.Kanban` | `var workflow = []` | **Not Applicable** |

| Key | **Property:**

`key` | `EJ.Kanban` | `workflows={workflow}` | `EJ.Kanban` | `var workflow = [{key: "Order"}]` | **Not Applicable** |

| Allowed Transition | **Property:**

`allowedTransition` | `EJ.Kanban` | `workflows={workflow}` | `EJ.Kanban` | `var workflow = [{key: "Order", allowedTransitions: "Served"}]` | **Not Applicable** |

Filtering

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

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

| Default | **Property:**

`filterSettings` | `EJ.Kanban` | `filterSettings={filter}` | `EJ.Kanban` | `var filter = []` | **Not Applicable** |

| Enable | **Property:**

`allowFiltering` | `EJ.Kanban` | `allowFiltering={true}` | `EJ.Kanban` | **Not Applicable** |

| Text | **Property:**

`text` | `EJ.Kanban` | `filterSettings={filter}` | `EJ.Kanban` | `var filter = [{text: "Janet Issues"}]` | **Not Applicable** |

| Query | **Property:**

`query` | `EJ.Kanban` | `filterSettings={filter}` | `EJ.Kanban` | `var filter = [{query: new ej.Query().where("Assignee", "equal", "Janet")}]` | **Not Applicable** |

| Description | **Property:**

`description` | `EJ.Kanban` | `filterSettings={filter}` | `EJ.Kanban` | `var filter = [{description: "Display Issues"}]` | **Not Applicable** |

| Filter Cards | **Method:** `filterCards(query)` | `EJ.Kanban` | `EJ.Kanban` | `var kanbanObj = $(".e-kanban").ejKanban("instance"); kanbanObj.KanbanFilter.filterCards(new ej.Query().where("Assignee", "equal", "Janet"));` | **Not Applicable** |

| Clear | **Method:** `clearFilter()`
`<EJ.Kanban></EJ.Kanban>`
`var kanbanObj =`
`$(".e-kanban").ejKanban("instance");`
`kanbanObj.KanbanFilter.clearFilter();` | **Not Applicable** |

Searching

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

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

| Default | **Property:**

`</searchSettings></EJ.Kanban>`
`searchSettings={searchSettings}/></EJ.Kanban>`
`var searchSettings = []` | **Not Applicable** |

| Enable | **Property:**

`</allowSearching></EJ.Kanban>`
`allowSearching={true}/></EJ.Kanban>` | **Not Applicable** |

| Fields | **Property:**

`fields</EJ.Kanban>`
`searchSettings={searchSettings}/></EJ.Kanban>`
`var searchSettings = [{fields: ["Summary"]}]` | **Not Applicable** |

| Key | **Property:**

`key</EJ.Kanban>`
`searchSettings={searchSettings}/></EJ.Kanban>`
`var searchSettings = [{key: "Task 1"}]` | **Not Applicable** |

| Operator | **Property:**

`operator</EJ.Kanban>`
`searchSettings={searchSettings}/></EJ.Kanban>`
`var searchSettings = [{operator: "contains"}]` | **Not Applicable** |

| Ignore Case | **Property:**

`ignoreCase</EJ.Kanban>`
`searchSettings={searchSettings}/></EJ.Kanban>`
`var searchSettings = [{ignoreCase: true}]` | **Not Applicable** |

| Search Cards | **Method:**

`</searchCards(searchString)></EJ.Kanban></EJ.Kanban>`
`var kanbanObj =`
`$(".e-kanban").ejKanban("instance");`
`kanbanObj.KanbanFilter.searchCards("Analyze");` | **Not Applicable** |

| Clear | **Method:** `clearSearch()`
`<EJ.Kanban></EJ.Kanban>`
`var kanbanObj =`
`$(".e-kanban").ejKanban("instance");`
`kanbanObj.KanbanFilter.clearSearch();` | **Not Applicable** |

External Drag And Drop

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

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

| Default | **Property:**

`</allowExternalDragAndDrop></EJ.Kanban>`
`allowExternalDragAndDrop={true}/></EJ.Kanban>` | **Not Applicable** |

| Target | **Property:** `</externalDropTarget></EJ.Kanban>`

`cardSettings-externalDropTarget="#DroppedKanban"></EJ.Kanban>` | **Not Applicable** |

Scrolling

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

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

| Default | **Property:**

`allowScrolling`

`EJ.Kanban` `allowScrolling={true}` | **Not Applicable** |

| height | **Property:**

`height`

`EJ.Kanban` `allowScrolling={true}` `scrollSettings={scrollSetting}` | **Property:** `height: 400` | **Property:** `KanbanComponent` `height="400"` |

| width | **Property:**

`width`

`EJ.Kanban` `allowScrolling={true}` `scrollSettings={scrollSetting}` | **Property:** `width: 400` | **Property:** `KanbanComponent` `width="400"` |

| Freeze Swimlane | **Property:**

`allowFreezeSwimlane`

`EJ.Kanban` `allowScrolling={true}` `scrollSettings={scrollSetting}` | **Property:** `allowFreezeSwimlane: true` | **Not Applicable** |

| Get Scroll Object | **Method:**

`getScrollObject()`

`EJ.Kanban` | `kanbanObj = $.e-kanban` | `ejKanban("instance");` `kanbanObj.getScrollObject();` | **Not Applicable** |

Card Selection and Hover

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

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

| Enable | **Property:**

`allowSelection`

`EJ.Kanban` `allowSelection={true}` | **Property:**

`cardSettings.selectionType`

`KanbanComponent` `cardSettings={{selectionType: "Single"}}` |

| Type | **Property:**

`selectionType`

`EJ.Kanban` `selectionType="single"` | It is covered under `selectionType` property. |

| Hover | **Property:**

`allowHover`

`EJ.Kanban` `allowHover={true}` | **Not Applicable** |

| Clear | **Method:** `clear()`

`EJ.Kanban` | `kanbanObj = $.e-kanban` | `ejKanban("instance");` `kanbanObj.KanbanSelection.clear();` | **Not Applicable** |

| Triggers before card selected | **Event:**

`beforeCardSelect`

`EJ.Kanban` `beforeCardSelect={beforeCardSelect}` | **Event:** `cardSelecting`

`</br><EJ.Kanban</br>cardSelecting={cardSelecting}></br></EJ.Kanban></br>function
cardSelecting(args){</br> | Not Applicable |`

| Triggers after</br>card selected | **Event:**

`cardSelect</br></br><EJ.Kanban</br>cardSelect={cardSelect}></br></EJ.Kanban></br>function
cardSelect(args){</br> | Not Applicable |`

Toolbar

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

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

| Custom Toolbar | **Property:**

`</br>customToolbarItems.template</br></br><EJ.Kanban</br>customToolbarItems-
template="#Delete"/></br></EJ.Kanban></br> | Not Applicable |`

| Triggers toolbar</br>item click | **Event:**

`toolbarClick</br></br><EJ.Kanban</br>toolbarClick={toolbarClick}></br></EJ.Kanban></br>functio
n toolbarClick(args){</br> | Not Applicable |`

Responsive

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

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

| Default | **Property:**

`isResponsive</br></br><EJ.Kanban</br>isResponsive={true}/></br></EJ.Kanban></br> | Not
Applicable |`

| Minimum width | **Property:**

`minWidth</br></br><EJ.Kanban</br>isResponsive={true}</br>minWidth='400'/></br></EJ.Kanba
n></br> | Not Applicable |`

State Persistence

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

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

| Persistence | **Not Applicable** |

Property:`</br>enablePersistence</br></br><KanbanComponent</br>enablePersistence={true}</br>
></KanbanComponent></br> |`

Right to Left - RTL

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

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

| default | **Property:**

`enableRTL</br></br><EJ.Kanban</br>enableRTL={true}/></br></EJ.Kanban></br> | Property: Link
to the
Video</br></br><KanbanComponent</br>enableRtl={true}</br></KanbanComponent></br> |`

{% endraw %}

Linear Gauge

Getting Started with React Linear Gauge

<!-- markdownlint-disable MD013 -->

This section explains the steps required to create a Linear Gauge and demonstrates the basic usage of the Linear Gauge component.

You can explore some useful features in the Linear Gauge component using the following video.

Dependencies

Following is the list of minimum dependencies required to use the Linear Gauge.

```
`javascript
+-- @syncfusion/ej2-react-lineargauge
|-- @syncfusion/ej2-lineargauge
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-svg-base
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-react-base
`
```

Installation and configuration

To get started with the React application, [create-react-app](#) can be used to setup the application. To install **create-react-app** run the following command.

```
`
npm install -g create-react-app
`
```

To create basic React application, run the following command.

```
`
create-react-app quickstart
`
```

Now, the application is created in the **quickstart** folder. Run the following command to navigate to the **quickstart** folder, and install the required **npm** packages.

```
`
cd quickstart
`
```

In the **quickstart** application, the Syncfusion component is added in the JavaScript file.

Creating a React application with TypeScript

To create React application with TypeScript, use the following command.

```
`
```

```
create-react-app quickstart --template typescript
```

,

Now, the application is created in the **quickstart** folder. Run the following command to navigate to the **quickstart** folder, and install the required **npm** packages.

,

```
cd quickstart
```

,

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. To install Linear Gauge package, use the following command.

,

```
npm install @syncfusion/ej2-react-lineargauge --save
```

,

Adding Linear Gauge component to the Project

Now, the Linear Gauge component can be added in the application. To initialize the Linear Gauge control in the React application, import the Linear Gauge control in the **src/App.js** or **src/App.tsx** as per the application. Please use the below code to include the Linear Gauge component in the application.

```
`ts
```

```
import React from 'react';
```

```
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
```

```
export function App() {
```

```
  return (<LinearGaugeComponent></LinearGaugeComponent>);
```

```
}
```

```
export default App;
```

,

Run the application

The Linear Gauge control is now included in the **quickstart** application. Use the following command to run the application.

,

```
npm start
```

,

Module Injection

Linear Gauge component are segregated into individual feature-wise modules. In order to use a particular feature,

inject its feature service in the **AppModule**. Please find the feature service name and description as follows.

- **Annotations** - Inject this module in to **services** to use annotation feature.
- **GaugeTooltip** - Inject this module in to **services** to use tooltip feature.

These modules should be injected into the **services** section as follows,

```
`ts
```

```
import * as React from "react";
import * as ReactDOM from "react-dom";

import { LinearGaugeComponent, Annotations, GaugeTooltip } from '@syncfusion/ ej2-react-
lineargauge';

export function App(){
  return(<LinearGaugeComponent>
    <Inject services={[Annotations, GaugeTooltip]}/>
  </LinearGaugeComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
```

Adding the Linear Gauge Title

The title can be added to the Linear Gauge component using the [title](#) property in the Linear Gauge.

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (<LinearGaugeComponent title='Linear
Gauge'></LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (<LinearGaugeComponent title='Linear
Gauge'></LinearGaugeComponent>);
}
```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Axis Range

The range of the axis can be set using the [minimum](#) and [maximum](#) properties in the Linear Gauge.

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  LinearGaugeComponent,
  AxesDirective,
  AxisDirective,
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={200}></AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  LinearGaugeComponent,
  AxesDirective,
  AxisDirective,
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={200}></AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

To denote the axis values with temperature units, add the °C as suffix to each label. This can be achieved by setting the `{value}°C` to the `format` property in the `labelStyle` object of the axis. Here, `{value}` acts as a placeholder for each axis label.

To change the pointer value from the default value of the gauge, set the `value` property in `pointers` object of the axis.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, RangeDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={200} labelStyle={ {
format: '{value}°C' } }>
          <PointersDirective>
            <PointerDirective value={140}>
            </PointerDirective>
          </PointersDirective>
          <RangesDirective>
            <RangeDirective start={0} end={80} startWidth={15}
endWidth={15}>
            </RangeDirective>
            <RangeDirective start={80} end={120} startWidth={15}
endWidth={15}>
            </RangeDirective>
            <RangeDirective start={120} end={140} startWidth={15}
endWidth={15}>
            </RangeDirective>
            <RangeDirective start={140} end={200} startWidth={15}
endWidth={15}>
            </RangeDirective>
          </RangesDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, RangeDirective } from
'@syncfusion/ej2-react-lineargauge';
```

```

export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={200} labelStyle={ {
format: '{value}°C' } }>
          <PointersDirective>
            <PointerDirective value={140}>
              </PointerDirective>
            </PointersDirective>
          <RangesDirective>
            <RangeDirective start={0} end={80} startWidth={15}
endWidth={15}>
              </RangeDirective>
            <RangeDirective start={80} end={120} startWidth={15}
endWidth={15}>
              </RangeDirective>
            <RangeDirective start={120} end={140} startWidth={15}
endWidth={15}>
              </RangeDirective>
            <RangeDirective start={140} end={200} startWidth={15}
endWidth={15}>
              </RangeDirective>
          </RangesDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Setting the value of the pointer

The pointer value is changed in the below sample using the [value](#) property in [pointers](#) object of the axis.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={200}>
          <PointersDirective>
            <PointerDirective value={140} color='green'>
              </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>
    );
  }

```

```

    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={200}>
          <PointersDirective>
            <PointerDirective value={140} color='green'>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Dimensions in React Linear gauge component

<!-- markdownlint-disable MD013 -->

Size for Linear Gauge

The height and width of the Linear Gauge can be set using the [width](#) and [height](#) properties in [LinearGaugeComponent](#).

<!-- markdownlint-disable MD036 -->

In Pixel

The size of the Linear Gauge can be set in pixel as demonstrated below.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent width='650px' height='350px'>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));

```

```
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent width='650px' height='350px'>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

In Percentage

By setting value in percentage, Linear Gauge receives its dimension matching to its parent. For example, when the height is set as **50%**, Linear Gauge renders to half of the parent height. The Linear Gauge will be responsive when the width is set as **100%**.

,

```
<div id='container'>
```

```
<div id='element' style="width:1000px; height:600px;"></div>
```

```
</div>
```

,

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent width="100%" height="50%"></LinearGaugeComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
```

```

    <LinearGaugeComponent width="100%" height="50%"></LinearGaugeComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Note: When the component's size is not specified, the height will be **450px** and the width will be the same as the parent element's width.

Axis in React Linear gauge component

<!-- markdownlint-disable MD013 -->

Axis is used to indicate the numeric values in the linear scale. The Linear Gauge component can have any number of axes. The sub-elements of an axis are line, ticks, labels, ranges, and pointers.

Setting the start value and end value of the axis

The start value and end value for the Linear Gauge can be set using the [minimum](#) and [maximum](#) properties in the [AxisDirective](#) respectively. By default, the start value of the axis is **0** and the end value of the axis is **100**.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={20} maximum={200}>>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={20} maximum={200}>>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

```

    </LinearGaugeComponent>);
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

Line Customization

The following properties in the [line](#) can be used to customize the axis line in the Linear Gauge.

- [height](#) - To set the length of the axis line.
- [width](#) - To set the thickness of the axis line.
- [color](#) - To set the color of the axis line.
- [offset](#) - To render the axis line with the specified distance from the Linear Gauge.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return(
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective line={ { height:150, width:2, color:'#4286f4',
offset:2 } }>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return(
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective line={ { height:150, width:2, color:'#4286f4',
offset:2 } }>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);

```



```
{% endraw %}
```

Ticks Customization

Ticks are used to specify the interval in the axis. Ticks are of two types, major ticks and minor ticks. The following properties in the [majorTicks](#) and [minorTicks](#) can be used to customize the major ticks and minor ticks respectively.

- [height](#) - To set the length of the major and minor ticks in pixel values.
- [color](#) - To set the color of the major and minor ticks of the Linear Gauge.
- [width](#) - To set the thickness of the major and minor ticks in pixel values.
- [interval](#) - To set the interval for the major ticks and minor ticks in the Linear Gauge.

<!-- markdownlint-disable MD036 -->

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={20} maximum={140} majorTicks={ {
interval:20 } } minorTicks={ { interval:5, color:'red' } } >
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={20} maximum={140} majorTicks={ {
interval:20 } } minorTicks={ { interval:5, color:'red' } } >
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
```

```
{% endraw %}
```

Positioning the ticks

The minor and major ticks can be positioned by using the [offset](#) and [position](#) properties. The [offset](#) is used to render the ticks with the specified distance from the axis. By default, the offset value is **0**. The possible values of the [position](#) property are **Inside**, **Outside**, **Cross**, and **Auto**. By default, the ticks will be placed inside the axis.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective majorTicks={ { interval:20, position: "Outside",
color: "green" } } minorTicks={ { interval:5, position: "Cross", color:
"red" } } >
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective majorTicks={ { interval:20, position: "Outside",
color: "green" } } minorTicks={ { interval:5, position: "Cross", color:
"red" } } >
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Labels Customization

The style of the labels can be customized using the following properties in the [font](#) property in [labelStyle](#).

- [color](#) - To set the color of the axis label.
- [fontFamily](#) - To set the font family of the axis label.
- [fontStyle](#) - To set the font style of the axis label.
- [fontWeight](#) - To set the font weight of the axis label.
- [opacity](#) - To set the opacity of the axis label.
- [size](#) - To set the size of the axis label.

<!-- markdownlint-disable MD036 -->

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return(
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective labelStyle={ { font:{color:'red' } } }>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return(
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective labelStyle={ { font:{color:'red' } } }>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Positioning the axis label

Labels can be positioned by using [offset](#) and [position](#) properties in the [labelStyle](#). The [offset](#) defines the distance between the labels and ticks. By default, the offset value is 0. The possible values of the [position](#) property are **Inside**, **Outside**, **Cross**, and **Auto**. By default, labels will be placed inside the axis.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective labelStyle={ { position: "Cross" } }>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective labelStyle={ { position: "Cross" } }>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Customizing the display of the last label

If the last label is not in the visible range, it will be hidden by default. The last label can be made visible by setting the [showLastLabel](#) property as **true** in the [AxisDirective](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
```

```
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={20} maximum={170} showLastLabel={true}
majorTicks={ { interval:20 } } minorTicks={ { interval:5 } }>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective minimum={20} maximum={170} showLastLabel={true}
majorTicks={ { interval:20 } } minorTicks={ { interval:5 } }>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

<!-- markdownlint-disable MD036 -->

Label Format

Axis labels in the Linear Gauge control can be formatted using the [format](#) property in the [labelStyle](#). It is used to render the axis labels in a certain format or to add a user-defined unit in the label. It works with the help of placeholder like **{value}°C**, where **value** represents the axis value. For example, 20°C.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective labelStyle={ { format: "{value}°C" } }>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```

    </LinearGaugeComponent>);
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
"@syncfusion/ej2-react-lineargauge";
export function App() {
  return(
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective labelStyle={ { format: "{value}°C" } }>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Displaying numeric format in labels

The numeric formats such as currency, percentage, and so on can be displayed in the labels of the Linear Gauge using the [format](#) property in the [LinearGaugeComponent](#). 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.
------	----	------------	---

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent format='c'>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent format='c'>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Orientation

By default, the Linear Gauge is rendered vertically. To change its orientation, the [orientation](#) property must be set to **Horizontal**.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent orientation='Horizontal'>
      <AxesDirective>
        <AxisDirective minimum={20} maximum={140}>>
      </AxisDirective>
    </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent orientation='Horizontal'>
      <AxesDirective>
        <AxisDirective minimum={20} maximum={140}>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Inverted Axis

The axis of the Linear Gauge component can be inverted by setting the [isInversed](#) property to **true** in the [AxisDirective](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective isInversed={true}>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
```



```

return (
  <LinearGaugeComponent>
    <AxesDirective>
      <AxisDirective isInversed={true}>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Opposed Axis

To place an axis opposite from its original position, [opposedPosition](#) property in the [AxisDirective](#) must be set as **true**.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective opposedPosition={true}>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective opposedPosition={true}>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Multiple Axes

Multiple axes can be added to the Linear Gauge by adding multiple [AxisDirective](#) in the [AxesDirective](#) and customization can be done with the [AxisDirective](#). Each axis can be customized separately as shown in the following example.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective labelStyle={ { format:'{value}°C' } }>
        </AxisDirective>
        <AxisDirective opposedPosition={true} labelStyle={ {
format:'{value}°F' } }>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective labelStyle={ { format:'{value}°C' } }>
        </AxisDirective>
        <AxisDirective opposedPosition={true} labelStyle={ {
format:'{value}°F' } }>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Ranges in React Linear gauge component

<!-- markdownlint-disable MD013 -->

Range is the set of values in the axis. The range can be defined using the [start](#) and [end](#) properties in the [RangeDirective](#). Any number of ranges can be added to the Linear Gauge using the [RangesDirective](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
RangesDirective, RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <AxesDirective>
        <AxisDirective>
          <RangesDirective>
            <RangeDirective start={50} end={80}>
          </RangeDirective>
        </RangesDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
RangesDirective, RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <AxesDirective>
        <AxisDirective>
          <RangesDirective>
            <RangeDirective start={50} end={80}>
          </RangeDirective>
        </RangesDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Customizing the range

Ranges can be customized using the following properties in [RangeDirective](#).

- [startWidth](#) - Customize the range thickness at the start axis value.

- [endWidth](#) - Customize the range thickness at the end axis value.
- [color](#) - Customize the range color.
- [position](#) - To place the range. By default, the range is placed outside of the axis. To change the position, this property can be set as "Inside", "Outside", "Cross", or "Auto".
- [offset](#) - To place the range with specified distance from the axis.
- [border](#) - Customize color and width of range border.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
RangesDirective, RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <AxesDirective>
        <AxisDirective>
          <RangesDirective>
            <RangeDirective start={50} end={80} startWidth={10}
endWidth={20} color='red'>
          </RangeDirective>
        </RangesDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
RangesDirective, RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <AxesDirective>
        <AxisDirective>
          <RangesDirective>
            <RangeDirective start={50} end={80} startWidth={10}
endWidth={20} color='red'>
          </RangeDirective>
        </RangesDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```
{% enddraw %}
```

Setting the range color for the labels

To set the color of the labels like the range color, set the [useRangeColor](#) property as **true** in the [labelStyle](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
RangesDirective, RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <AxesDirective>
        <AxisDirective labelStyle={{ useRangeColor: true }}>
          <RangesDirective>
            <RangeDirective start={50} end={80} startWidth={10}
endWidth={10} color='red'>
              </RangeDirective>
            </RangesDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>;
    )
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
RangesDirective, RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <AxesDirective>
        <AxisDirective labelStyle={{ useRangeColor: true }}>
          <RangesDirective>
            <RangeDirective start={50} end={80} startWidth={10}
endWidth={10} color='red'>
              </RangeDirective>
            </RangesDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>;
    )
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Multiple ranges

Multiple ranges can be added to the Linear Gauge by adding collections of [RangeDirective](#) in the [RangesDirective](#) and customization of ranges can be done with [RangeDirective](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
RangesDirective, RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <AxesDirective>
        <AxisDirective>
          <RangesDirective>
            <RangeDirective start={0} end={30} startWidth={10}
endWidth={10} color='#41f47f'>
            </RangeDirective>
            <RangeDirective start={30} end={50} startWidth={10}
endWidth={10} color='#f49441'>
            </RangeDirective>
            <RangeDirective start={50} end={100} startWidth={10}
endWidth={10} color='#cd41f4'>
            </RangeDirective>
          </RangesDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
RangesDirective, RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <AxesDirective>
        <AxisDirective>
          <RangesDirective>
            <RangeDirective start={0} end={30} startWidth={10}
endWidth={10} color='#41f47f'>
            </RangeDirective>
            <RangeDirective start={30} end={50} startWidth={10}
endWidth={10} color='#f49441'>
            </RangeDirective>
          </RangesDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```

        <RangeDirective start={50} end={100} startWidth={10}
endWidth={10} color='#cd41f4'>
        </RangeDirective>
    </RangesDirective>
</AxisDirective>
</AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Gradient Color

Gradient support allows the addition of multiple colors in the range. The following gradient types are supported in the Linear Gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear-gradient, colors will be applied in a linear progression. The start value of the linear gradient can be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as [color](#), [opacity](#), and [offset](#) to be defined in [colorStop](#).

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, Gradient, Inject,
RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
    return (
        <LinearGaugeComponent container={{width: 30, offset: 30}}
orientation="Horizontal">
            <Inject services={[Gradient]}/>
            <AxesDirective>
                <AxisDirective minimum={0} maximum={100} line={{width: 0}}
majorTicks={{width: 0, interval:25}}
                minorTicks={{width: 0}} labelStyle={{font: {color:
'#424242'}, offset: 70}}>
                    <PointersDirective>
                        <PointerDirective value={80} height={25} width={35}
placement="Near" offset={-44} markerType="Triangle" color= '#f54ea2'>
                        </PointerDirective>
                    </PointersDirective>
                    <RangesDirective>
                        <RangeDirective start={0} end={80} startWidth={30}
endWidth={30} color="#f54ea2" offset={30}
                        linearGradient = {{
                            startValue: '0%',
                            endValue: '100%',
                            colorStop: [

```

```

        { color: '#fef3f9', offset: '0%', opacity: 1 },
        { color: '#f54ea2', offset: '100%', opacity: 1
    }}
    </RangeDirective>
  </RangesDirective>
</AxisDirective>
</AxesDirective>
</LinearGaugeComponent>;
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, Gradient, Inject,
RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent container={{width: 30, offset: 30}}
orientation="Horizontal">
      <Inject services={[Gradient]}/>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={100} line={{width: 0}}
majorTicks={{width: 0, interval:25}}
        minorTicks={{width: 0}} labelStyle={{font: {color:
'#424242'}, offset: 70}}>
          <PointersDirective>
            <PointerDirective value={80} height={25} width={35}
placement="Near" offset={-44} markerType="Triangle" color= '#f54ea2'>
              </PointerDirective>
            </PointersDirective>
            <RangesDirective>
              <RangeDirective start={0} end={80} startWidth={30}
endWidth={30} color="#f54ea2" offset={30}
                linearGradient = {{
                  startValue: '0%',
                  endValue: '100%',
                  colorStop: [
                    { color: '#fef3f9', offset: '0%', opacity: 1 },
                    { color: '#f54ea2', offset: '100%', opacity: 1
                }}
              </RangeDirective>
            </RangesDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>;
    )
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
}

```



```
{% enddraw %}
```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as [color](#), [opacity](#) and [offset](#) to be defined in [colorStop](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, Gradient, Inject,
RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent container={{width: 30, offset: 30}}
orientation="Horizontal">
      <Inject services={[Gradient]}/>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={100} line={{width: 0}}
majorTicks={{width: 0, interval: 25}}
        minorTicks={{width: 0}} labelStyle={{font: {color:
'#424242'}, offset: 70}}>
          <PointersDirective>
            <PointerDirective value={80} height={25} width={35}
placement="Near" offset={-44} markerType="Triangle" color= '#f54ea2'>
          </PointerDirective>
        </PointersDirective>
        <RangesDirective>
          <RangeDirective start={0} end={80} startWidth={30}
endWidth={30} color="#f54ea2" offset={30}
            radialGradient = {{
              radius: '65%',
              outerPosition: { x: '50%', y: '70%' },
              innerPosition: { x: '60%', y: '60%' },
              colorStop: [
                { color: '#fff5f5', offset: '5%', opacity: 0.9
},
                { color: '#f54ea2', offset: '100%', opacity: 0.9
}}
            </RangeDirective>
          </RangesDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, Gradient, Inject,
RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent container={{width: 30, offset: 30}}
orientation="Horizontal">
      <Inject services={[Gradient]}/>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={100} line={{width: 0}}
majorTicks={{width: 0, interval: 25}}
        minorTicks={{width: 0}} labelStyle={{font: {color:
'#424242'}, offset: 70}}>
          <PointersDirective>
            <PointerDirective value={80} height={25} width={35}
placement="Near" offset={-44} markerType="Triangle" color= '#f54ea2'>
              </PointerDirective>
            </PointersDirective>
            <RangesDirective>
              <RangeDirective start={0} end={80} startWidth={30}
endWidth={30} color="#f54ea2" offset={30}>
                radialGradient = {{
                  radius: '65%',
                  outerPosition: { x: '50%', y: '70%' },
                  innerPosition: { x: '60%', y: '60%' },
                  colorStop: [
                    { color: '#fff5f5', offset: '5%', opacity: 0.9
},
                    { color: '#f54ea2', offset: '100%', opacity: 0.9
}
                ]
              }>
            </RangeDirective>
          </RangesDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

If we set both gradients for the range, only the linear gradient gets rendered. If we set the [startValue](#) and [endValue](#) of the [linearGradient](#) as empty strings, then the radial gradient gets rendered in the range of the Linear Gauge.

Pointers in React Linear Gauge Component

<!-- markdownlint-disable MD013 -->

Pointers are used to indicate values on an axis. The value of the pointer can be modified using the [value](#) property in [PointerDirective](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80}>>
          </PointerDirective>
        </PointersDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80}>>
          </PointerDirective>
        </PointersDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Types of pointer

The Linear Gauge supports the following types of pointers:

- Bar
- Marker

The type of pointer can be modified by using the [type](#) property in [PointerDirective](#).

Marker pointer

A marker pointer is a shape that can be used to mark the pointer value in the Linear Gauge.

Types of marker shapes

By default, the marker shape for the pointer is **InvertedTriangle**. To change the shape of the pointer, use the [markerType](#) property in [PointerDirective](#). The following marker types are available in Linear Gauge.

- Circle
- Rectangle
- Triangle
- InvertedTriangle
- Diamond
- Image
- Text

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80} type='Marker'
markerType='Circle'>
          </PointerDirective>
        </PointersDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
```

```

        <AxesDirective>
            <AxisDirective>
                <PointersDirective>
                    <PointerDirective value={80} type='Marker'
markerType='Circle'>
                </PointerDirective>
            </PointersDirective>
        </AxisDirective>
    </AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Image can be rendered instead of rendering a shape as a pointer. It can be achieved by setting the [markerType](#) property to **Image** and setting the source URL of image to [imageUrl](#) property in [PointerDirective](#).

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    LinearGaugeComponent,
    AxesDirective,
    AxisDirective,
    PointersDirective,
    PointerDirective,
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
    return (
        <LinearGaugeComponent id="gauge" orientation="Horizontal" width="650px"
height="400px">
            <AxesDirective>
                <AxisDirective>
                    minorTicks={{ position: 'Outside' }}
                    majorTicks={{ interval: 20, position: 'Outside' }}
                    labelStyle={{ position: 'Outside' }}
                >
                <PointersDirective>
                    <PointerDirective
                        value={60}
                        markerType="Image"
                        imageUrl="https://ej2.syncfusion.com/demos/src/linear-
gauge/images/step-count.png"
                        offset="-27"
                        height={40}
                        width={40}
                    ></PointerDirective>
                </PointersDirective>
            </AxisDirective>
        </AxesDirective>
    </LinearGaugeComponent>);
}

```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent id="gauge" orientation="Horizontal" width="650px"
height="400px">
      <AxesDirective>
        <AxisDirective
          minorTicks={{ position: 'Outside' }}
          majorTicks={{ interval: 20, position: 'Outside' }}
          labelStyle={{ position: 'Outside' }}
        >
          <PointersDirective>
            <PointerDirective
              value={60}
              markerType="Image"
              imageUrl="https://ej2.syncfusion.com/demos/src/linear-
gauge/images/step-count.png"
              offset="-27"
              height={40}
              width={40}
            ></PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Text can be added instead of rendering a shape as a pointer. It can be achieved by setting the [markerType](#) property to **Text**, and the text content can be set using the [text](#) property in [PointerDirective](#).

The following properties in the [textStyle](#) property can be used to set the text style for the text pointer.

- [fontFamily](#) - It is used to set the font family for the text pointer.
- [fontStyle](#) - It is used to set the font style for the text pointer.
- [fontWeight](#) - It is used to set the font weight for the text pointer.
- [size](#) - It is used to set the font size for the text pointer.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  LinearGaugeComponent,
  AxesDirective,
  AxisDirective,
  RangesDirective,
  RangeDirective,
  PointersDirective,
  PointerDirective
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent orientation="Horizontal" width="650px"
height="400px">
      <AxesDirective>
        <AxisDirective
          line={{ width: 5 }}
          minorTicks={{
            interval: 10,
            height: 3,
            position: 'Inside',
          }}
          majorTicks={{
            interval: 20,
            height: 7,
            width: 1,
            position: 'Inside',
          }}
          labelStyle={{ position: 'Inside', font: { fontFamily: 'inherit' }
        }}

        minimum={0}
        maximum={100}
        opposedPosition={true}
      >
        <PointersDirective>
          <PointerDirective
            value={13}
            markerType="Text"
            text="Low"
            color="black"
            offset={-50}
            textStyle={{
              size: '18px',
              fontWeight: 'bold',
            }}
          ></PointerDirective>
          <PointerDirective
            value={48}
            markerType="Text"
            text="Moderate"
            color="black"
            offset={-50}
            textStyle={{
              size: '18px',
              fontWeight: 'bold',

```

```

    }}
  </PointerDirective>
  <PointerDirective
    value={83}
    markerType="Text"
    text="High"
    color="black"
    offset={-50}
    textStyle={{
      size: '18px',
      fontWeight: 'bold',
    }}
  </PointerDirective>
</PointersDirective>
<RangesDirective>
  <RangeDirective
    position="Inside"
    start={0}
    end={30}
    color="#FB7D55"
    startWidth={50}
    endWidth={50}
  />
  <RangeDirective
    position="Inside"
    start={30}
    end={65}
    color="#ECC85B"
    startWidth={50}
    endWidth={50}
  />
  <RangeDirective
    position="Inside"
    start={65}
    end={100}
    color="#6FC78A"
    startWidth={50}
    endWidth={50}
  />
</RangesDirective>
</AxisDirective>
</AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  LinearGaugeComponent,
  AxesDirective,

```



```

AxisDirective,
RangesDirective,
RangeDirective,
PointersDirective,
PointerDirective,
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent orientation="Horizontal" width="650px"
height="400px">
      <AxesDirective>
        <AxisDirective
          line={{ width: 5 }}
          minorTicks={{
            interval: 10,
            height: 3,
            position: 'Inside',
          }}
          majorTicks={{
            interval: 20,
            height: 7,
            width: 1,
            position: 'Inside',
          }}
          labelStyle={{ position: 'Inside', font: { fontFamily: 'inherit' }
        }}

        minimum={0}
        maximum={100}
        opposedPosition={true}
      >
        <PointersDirective>
          <PointerDirective
            value={13}
            markerType="Text"
            text="Low"
            color="black"
            offset={-50}
            textStyle={{
              size: '18px',
              fontWeight: 'bold',
            }}
          ></PointerDirective>
          <PointerDirective
            value={48}
            markerType="Text"
            text="Moderate"
            color="black"
            offset={-50}
            textStyle={{
              size: '18px',
              fontWeight: 'bold',
            }}
          ></PointerDirective>
          <PointerDirective
            value={83}
            markerType="Text"
            text="High"

```

```

        color="black"
        offset={-50}
        textStyle={{
          size: '18px',
          fontWeight: 'bold',
        }}
      </PointerDirective>
    </PointersDirective>
    <RangesDirective>
      <RangeDirective
        position="Inside"
        start={0}
        end={30}
        color="#FB7D55"
        startWidth={50}
        endWidth={50}
      />
      <RangeDirective
        position="Inside"
        start={30}
        end={65}
        color="#ECC85B"
        startWidth={50}
        endWidth={50}
      />
      <RangeDirective
        position="Inside"
        start={65}
        end={100}
        color="#6FC78A"
        startWidth={50}
        endWidth={50}
      />
    </RangesDirective>
  </AxisDirective>
</AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

<!-- markdownlint-disable MD036 -->

Marker Pointer Customization

The marker pointer can be customized using the following properties.

- [height](#) - To set the height of the pointer.
- [position](#) - The position of the pointer can be changed by setting the value as **Inside**, **Outside**, **Cross**, or **Auto**.
- [width](#) - To set the width of the pointer.
- [color](#) - To set the color of the pointer.
- [placement](#) - To place the pointer in the specified position. By default, the pointer is placed **Far** from the axis. To change the placement, set the [placement](#) property as **Near**, **Center**, or **None**.

- [offset](#) - To place the pointer with specified distance from the axis.
- [opacity](#) - To set the opacity of the pointer.
- [animationDuration](#) - To specify the duration of the animation in pointer.
- [border](#) - To set the color and width for the border of the pointer.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80} type='Marker'
markerType='Circle' height={15} width={15} color='#cd41f4'>
          </PointerDirective>
        </PointersDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80} type='Marker'
markerType='Circle' height={15} width={15} color='#cd41f4'>
          </PointerDirective>
        </PointersDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```
{% endraw %}
```

Bar Pointer

The bar pointer is used to track the axis value. The bar pointer starts from the beginning of the gauge and ends at the pointer value. To enable bar pointer set the [type](#) property in [PointerDirective](#) as **Bar**.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
  PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={60} type='Bar'>
          </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
  PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={60} type='Bar'>
          </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

<!-- markdownlint-disable MD036 -->

Bar pointer customization

The bar pointer can be customized using following properties.

- [width](#) - To set the thickness of the bar pointer.
- [color](#) - To set the color of the bar pointer.
- [offset](#) - To place the bar pointer with the specified distance from it's default position.
- [opacity](#) - To set the opacity of the bar pointer.
- [roundedCornerRadius](#) - To set the corner radius for the bar pointer.
- [border](#) - To set the color and width for the border of the pointer.
- [animationDuration](#) - To set the duration of the animation in bar pointer.

The placement property is not applicable for the bar pointer.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={60} type='Bar' width={20}
color='#f44141'>
          </PointerDirective>
        </PointersDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
```

```

        <AxesDirective>
          <AxisDirective>
            <PointersDirective>
              <PointerDirective value={60} type='Bar' width={20}
color='#f44141'>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>;
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

Multiple pointers

Multiple pointers can be added to the Linear Gauge by adding multiple [PointerDirective](#) in the [PointersDirective](#) and customization for the pointers can be done with [PointerDirective](#).

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80}>
            </PointerDirective>
            <PointerDirective value={60} markerType='Circle'>
            </PointerDirective>
            <PointerDirective value={30} markerType='Diamond'>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>;
  )
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80}>
            </PointerDirective>
            <PointerDirective value={60} markerType='Circle'>
            </PointerDirective>
            <PointerDirective value={30} markerType='Diamond'>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}
```

Pointer animation

Pointer is animated on loading the gauge. This can be handled using the [animationDuration](#) property. The duration of the animation can be specified in milliseconds.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={60} animationDuration={1000}>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={60} animationDuration={1000}>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Gradient Color

Gradient support allows the addition of multiple colors in the pointers of the Linear Gauge. The following gradient types are supported in the Linear Gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient can be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as [color](#), [opacity](#), and [offset](#) are set using [colorStop](#) property.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, Gradient, Inject,
RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent container={{width: 30, offset: 30}}
orientation="Horizontal">
      <Inject services={[Gradient]}/>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={100} line={{width: 0}}
majorTicks={{width: 0, interval:25}}
          minorTicks={{width: 0}} labelStyle={{font: {color:
'#424242'}, offset: 70}}>

```



```

        <PointersDirective>
          <PointerDirective value={80} height={25} width={35}
placement="Near" offset={-44} markerType="Triangle"
          linearGradient={{
            startValue: '0%',
            endValue: '100%',
            colorStop: [
              { color: '#fef3f9', offset: '0%', opacity: 1 },
              { color: '#f54ea2', offset: '100%', opacity: 1 }]
          }}>
        </PointerDirective>
      </PointersDirective>
      <RangesDirective>
        <RangeDirective start={0} end={80} startWidth={30}
endWidth={30} color="#f54ea2" offset={30}>
      </RangeDirective>
    </RangesDirective>
  </AxisDirective>
</AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, Gradient, Inject,
RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent container={{width: 30, offset: 30}}
orientation="Horizontal">
      <Inject services={[Gradient]} />
      <AxesDirective>
        <AxisDirective minimum={0} maximum={100} line={{width: 0}}
majorTicks={{width: 0, interval: 25}}
        minorTicks={{width: 0}} labelStyle={{font: {color:
'#424242'}, offset: 70}}>
          <PointersDirective>
            <PointerDirective value={80} height={25} width={35}
placement="Near" offset={-44} markerType="Triangle"
            linearGradient={{
              startValue: '0%',
              endValue: '100%',
              colorStop: [
                { color: '#fef3f9', offset: '0%', opacity: 1 },
                { color: '#f54ea2', offset: '100%', opacity: 1 }]
            }}>
          </PointerDirective>
        </PointersDirective>
      </RangesDirective>
    </AxisDirective>
  </AxesDirective>
</LinearGaugeComponent>
  );
}
{% endraw %}

```

```

        <RangeDirective start={0} end={80} startWidth={30}
endWidth={30} color="#f54ea2" offset={30}>
        </RangeDirective>
    </RangesDirective>
</AxisDirective>
</AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as [color](#), [opacity](#), and [offset](#) are set using [colorStop](#) property.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, Gradient, Inject,
RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
    return (
        <LinearGaugeComponent container={{width: 30, offset: 30}}
orientation="Horizontal">
            <Inject services={[Gradient]}/>
            <AxesDirective>
                <AxisDirective minimum={0} maximum={100} line={{width: 0}}
majorTicks={{width: 0, interval: 25}}
                minorTicks={{width: 0}} labelStyle={{font: {color:
'#424242'}, offset: 70}}>
                    <PointersDirective>
                        <PointerDirective value={80} height={25} width={35}
placement="Near" offset={-44} markerType="Triangle"
                        radialGradient={{
                            radius: '60%',
                            outerPosition: { x: '50%', y: '50%' },
                            innerPosition: { x: '50%', y: '50%' },
                            colorStop: [
                                { color: '#fff5f5', offset: '0%', opacity: 0.9 },
                                { color: '#f54ea2', offset: '100%', opacity: 0.8 }
                            ]
                        }}>
                        </PointerDirective>
                    </PointersDirective>
                    <RangesDirective>
                        <RangeDirective start={0} end={80} startWidth={30}
endWidth={30} color="#f54ea2" offset={30}>
                        </RangeDirective>
                    </RangesDirective>
                </AxisDirective>
            </AxesDirective>

```

```

    </LinearGaugeComponent>);
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, Gradient, Inject,
RangeDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent container={{width: 30, offset: 30}}
orientation="Horizontal">
      <Inject services={[Gradient]}/>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={100} line={{width: 0}}
majorTicks={{width: 0, interval: 25}}
        minorTicks={{width: 0}} labelStyle={{font: {color:
'#424242'}, offset: 70}}>
          <PointersDirective>
            <PointerDirective value={80} height={25} width={35}
placement="Near" offset={-44} markerType="Triangle"
            radialGradient={{
              radius: '60%',
              outerPosition: { x: '50%', y: '50%' },
              innerPosition: { x: '50%', y: '50%' },
              colorStop: [
                { color: '#fff5f5', offset: '0%', opacity: 0.9 },
                { color: '#f54ea2', offset: '100%', opacity: 0.8 } ]
            }}>
          </PointerDirective>
        </PointersDirective>
        <RangesDirective>
          <RangeDirective start={0} end={80} startWidth={30}
endWidth={30} color="#f54ea2" offset={30}>
        </RangeDirective>
      </RangesDirective>
    </AxisDirective>
  </AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

If we set both gradients, only the linear gradient gets rendered. If we set the [startValue](#) and [endValue](#) of the [linearGradient](#) as empty strings, then the radial gradient gets rendered in the pointer of the Linear Gauge.

Annotations in React Linear gauge component

<!-- markdownlint-disable MD013 -->

Annotations are used to mark the specific area of interest in the Linear Gauge with text, HTML elements, or images. Any number of annotations can be added to the Linear Gauge component.

Adding annotation

To render the custom HTML elements in the Linear Gauge component, use the [content](#) property in the [AnnotationDirective](#). The annotation can be rendered either by specifying the id of the element or specifying the code to create a new element that needs to be displayed in the gauge area.

<!-- markdownlint-disable MD036 -->

```
,
<head>
<script id='fruits' type='text/x-template'>
<div id='apple'>
<img src='src/lineargauge/images/apple.png'>
</div>
</script>
</head>
,
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";

import { LinearGaugeComponent, AnnotationsDirective, AnnotationDirective, Annotations, Inject } from
 '@syncfusion/ej2-react-lineargauge';

export function App() {
return(
<LinearGaugeComponent>
<Inject services={[Annotations]}/>
<AnnotationsDirective>
<AnnotationDirective content="#fruits" x={100} zIndex='1' y={100}>
</AnnotationDirective>
</AnnotationsDirective>
</LinearGaugeComponent>);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
```

Customization

The following properties are used to customize the annotation.

- [zIndex](#) - Bring the annotation to the front or back, when annotation overlaps with another element.
- [axisValue](#) - To place the annotation in the specified axis value with respect to the provided axis index.
- [axisIndex](#) - To place the annotation in the specified axis with respect to the provided axis value.
- [horizontalAlignment](#) - To place the annotation horizontally.
- [verticalAlignment](#) - To place the annotation vertically.
- [x](#), [y](#) - To place the annotation in the specified location.

Changing the z-index

To change the stack order of an annotation element, the [zIndex](#) property of the [AnnotationDirective](#) can be used.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AnnotationsDirective, AnnotationDirective,
Annotations, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Gauge</h1></div>' zIndex='1' >
        </AnnotationDirective>
      </AnnotationsDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AnnotationsDirective, AnnotationDirective,
Annotations, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Gauge</h1></div>' zIndex='1' >

```

```

        </AnnotationDirective>
      </AnnotationsDirective>
    </LinearGaugeComponent>);
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

Positioning an annotation

The annotation can be placed anywhere in the Linear Gauge by setting the pixel value to the [x](#) and [y](#) properties in the [AnnotationDirective](#).

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AnnotationsDirective, AnnotationDirective,
Annotations, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Gauge</h1></div>' x={100} y={100} zIndex='1' >
          </AnnotationDirective>
        </AnnotationsDirective>
      </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AnnotationsDirective, AnnotationDirective,
Annotations, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Gauge</h1></div>' x={100} y={100} zIndex='1' >
          </AnnotationDirective>
        </AnnotationsDirective>
      </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

<!-- markdownlint-disable MD036 -->

Alignment of annotation

The annotation can be aligned horizontally and vertically by using the [horizontalAlignment](#) and [verticalAlignment](#) properties respectively. The possible values can be **Center**, **Far**, **Near**, and **None**. The [horizontalAlignment](#) and [verticalAlignment](#) properties are not applicable when the [x](#) and [y](#) properties are set in the [AnnotationDirective](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AnnotationsDirective, AnnotationDirective,
Annotations, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Gauge</h1></div>' verticalAlignment="Center"
horizontalAlignment="Center" zIndex='1' >
          </AnnotationDirective>
        </AnnotationsDirective>
      </LinearGaugeComponent>;
    )
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AnnotationsDirective, AnnotationDirective,
Annotations, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Gauge</h1></div>' verticalAlignment="Center"
horizontalAlignment="Center" zIndex='1' >
          </AnnotationDirective>
        </AnnotationsDirective>
      </LinearGaugeComponent>;
    )
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Multiple annotations

Multiple annotations can be added to the Linear Gauge component by adding the multiple [AnnotationDirective](#) in the [AnnotationsDirective](#) and customization for the annotation can be done with the [AnnotationDirective](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AnnotationsDirective, AnnotationDirective,
Annotations, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div><h1
style="color:red;">Speed</h1></div>' verticalAlignment='Near'
horizontalAlignment='Center' zIndex='1'>
          </AnnotationDirective>
        <AnnotationDirective content='<div><h1
style="color:blue;">Meter</h1></div>' zIndex='1' x={150} y={400}>
          </AnnotationDirective>
        </AnnotationsDirective>
      </LinearGaugeComponent>;
    )
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AnnotationsDirective, AnnotationDirective,
Annotations, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div><h1
style="color:red;">Speed</h1></div>' verticalAlignment='Near'
horizontalAlignment='Center' zIndex='1'>
          </AnnotationDirective>
        <AnnotationDirective content='<div><h1
style="color:blue;">Meter</h1></div>' zIndex='1' x={150} y={400}>
          </AnnotationDirective>
        </AnnotationsDirective>
      </LinearGaugeComponent>;
    )
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}
```


Animation in React Linear Gauge component

All of the elements in the Linear Gauge, such as the axis lines, ticks, labels, ranges, pointers, and annotations, can be animated sequentially by using the [animationDuration](#) property. The animation for the Linear Gauge is enabled when the `animationDuration` property is set to an appropriate value in milliseconds, providing a smooth rendering effect for the component. If the `animationDuration` property is set to `0`, which is the default value, the animation effect is disabled. If the animation is enabled, the component will behave in the following order.

1. The axis line, ticks, labels, and ranges will all be animated at the same time.
2. If available, pointers will be animated in the same way as [pointer animation](#).
3. If available, annotations will be animated.

The animation of the Linear Gauge is demonstrated in the following example.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  LinearGaugeComponent,
  AxesDirective,
  AxisDirective,
  Inject,
  PointersDirective,
  PointerDirective,
  AnnotationDirective,
  Annotations,
  AnnotationsDirective,
  RangesDirective,
  RangeDirective,
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent
      background="transparent"
      id="gauge"
      orientation="Horizontal"
      width="650px" height="400px"
      animationDuration={3000}
    >
      <Inject services={[Annotations]} />
      <AxesDirective>
        <AxisDirective
          minorTicks={{ interval: 2, height: 10, color: '#9E9E9E' }}
          majorTicks={{ interval: 10, height: 20, color: '#9E9E9E' }}
          labelStyle={{ offset: 48, font: { fontFamily: 'inherit' } }}
        >
          <PointersDirective>
            <PointerDirective
              value={10}
              placement="Near"
              height={15}
            >
              </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>
    </App>
  );
}
```

```

        width={15}
        offset={-40}
        markerType="Triangle"
      ></PointerDirective>
    </PointersDirective>
    <RangesDirective>
      <RangeDirective
        start={0}
        end={50}
        startWidth={10}
        endWidth={10}
        color="#F45656"
        offset="35"
      ></RangeDirective>
    </RangesDirective>
  </AxisDirective>
</AxesDirective>
<AnnotationsDirective>
  <AnnotationDirective
    content='<div style="width: 70px;margin-top: 25%;font-size:
16px;">10 MPH</div>'
    axisIndex={0}
    axisValue={10}
    x={10}
    zIndex="1"
    y={-70}
  />
</AnnotationsDirective>
</LinearGaugeComponent>;
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  LinearGaugeComponent,
  AxesDirective,
  AxisDirective,
  Inject,
  PointersDirective,
  PointerDirective,
  AnnotationDirective,
  Annotations,
  AnnotationsDirective,
  RangesDirective,
  RangeDirective,
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent
      background="transparent"

```

```

    id="gauge"
    orientation="Horizontal"
    width="650px" height="400px"
    animationDuration={3000}
  >
  <Inject services={[Annotations]} />
  <AxesDirective>
    <AxisDirective>
      minorTicks={{ interval: 2, height: 10, color: '#9E9E9E' }}
      majorTicks={{ interval: 10, height: 20, color: '#9E9E9E' }}
      labelStyle={{ offset: 48, font: { fontFamily: 'inherit' } }}
    >
    <PointersDirective>
      <PointerDirective>
        value={10}
        placement="Near"
        height={15}
        width={15}
        offset={-40}
        markerType="Triangle"
      </PointerDirective>
    </PointersDirective>
    <RangesDirective>
      <RangeDirective>
        start={0}
        end={50}
        startWidth={10}
        endWidth={10}
        color="#F45656"
        offset="35"
      </RangeDirective>
    </RangesDirective>
  </AxisDirective>
</AxesDirective>
<AnnotationsDirective>
  <AnnotationDirective>
    content='<div style="width: 70px;margin-top: 25%;font-size:
16px;">10 MPH</div>'
    axisIndex={0}
    axisValue={10}
    x={10}
    zIndex="1"
    y={-70}
  />
</AnnotationDirective>
</AnnotationsDirective>
</LinearGaugeComponent>;
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

Only the pointer of the Linear Gauge can be animated individually, not the axis lines, ticks, labels, ranges, and annotations. You can refer this [link](#) to enable only pointer animation.

User interaction in React Linear gauge component

<!-- markdownlint-disable MD013 -->

Tooltip

<!-- markdownlint-disable MD036 -->

Linear Gauge displays the details about a pointer value through [tooltip](#), when the mouse hovers over the pointer. To enable the tooltip, set [enable](#) property as **true** and inject **GaugeTooltip** into **services**.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true } }>
      <Inject services={[GaugeTooltip]} />
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80}>
              </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true } }>
      <Inject services={[GaugeTooltip]} />
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80}>
              </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>);
}
```

```

}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

<!-- markdownlint-disable MD013 -->

Tooltip format

<!-- markdownlint-disable MD013 -->

Tooltip in the Linear Gauge control can be formatted using the [format](#) property in [tooltip](#). It is used to render the tooltip in certain format or to add a user-defined unit in the tooltip. By default, the tooltip shows the pointer value only. In addition to that, more information can be added in the tooltip. For example, the format **{value}km** shows pointer value with kilometer unit in the tooltip.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
"@syncfusion/ej2-react-lineargauge";
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true, format: '{value}km' } }>
      <Inject services={[GaugeTooltip]} />
        <AxesDirective>
          <AxisDirective>
            <PointersDirective>
              <PointerDirective value={80}>
            </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
"@syncfusion/ej2-react-lineargauge";
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true, format: '{value}km' } }>
      <Inject services={[GaugeTooltip]} />
        <AxesDirective>
          <AxisDirective>
            <PointersDirective>

```

```

        <PointerDirective value={80}>
        </PointerDirective>
      </PointersDirective>
    </AxisDirective>
  </AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Tooltip Template

The HTML elements rendered in the tooltip of the Linear Gauge using the [template](#) property of the [tooltip](#). The `${value}` can be used as placeholders in the HTML element to display the pointer values of the corresponding axis.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
"@syncfusion/ej2-react-linear-gauge";
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true, template: '<div>Pointer:
80 </div>' } }>
      <Inject services={[GaugeTooltip]}>
        <AxesDirective>
          <AxisDirective>
            <PointersDirective>
              <PointerDirective value={80}>
              </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>);
    }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
"@syncfusion/ej2-react-linear-gauge";
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true, template: '<div>Pointer:
80 </div>' } }>

```

```

    <Inject services={[GaugeTooltip]}/>
    <AxesDirective>
      <AxisDirective>
        <PointersDirective>
          <PointerDirective value={80}>
        </PointerDirective>
        </PointersDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>;
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Customize the appearance of the tooltip

The tooltip can be customized using the following properties in [tooltip](#).

- [fill](#) - To fill the color for tooltip.
- [enableAnimation](#) - To enable or disable the tooltip animation.
- [border](#) - To set the border color and width of the tooltip.
- [textStyle](#) - To customize the style of the text in tooltip.
- [showAtMousePosition](#) - To show the tooltip at the mouse position.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true, fill: '#e5bcbc', border:
{ color: '#000000' } } }>
      <Inject services={[GaugeTooltip]}/>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80}>
          </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>;
  )
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}

```

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true, fill: '#e5bcbc', border:
{ color: '#000000' } } }>
      <Inject services={[GaugeTooltip]}/>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80}>
              </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Positioning the tooltip

The tooltip is positioned at the **End** of the pointer. To change the position of the tooltip at the start, or center of the pointer, set the [position](#) property to **Start** or **Center**.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true, position: "Center" } }>
      <Inject services={[GaugeTooltip]}/>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80} type="Bar">
              </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX


```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, GaugeTooltip, Inject } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent tooltip={ { enable: true, position: "Center" } }>
      <Inject services={[GaugeTooltip]}/>
        <AxesDirective>
          <AxisDirective>
            <PointersDirective>
              <PointerDirective value={80} type="Bar">
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Pointer Drag

To drag either marker or bar pointer to the desired axis value, set the [enableDrag](#) property as **true** in the [PointerDirective](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80} enableDrag={true}>
          </PointerDirective>
        </PointersDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={80} enableDrag={true}>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Print and export in React Linear gauge component

Print

The rendered Linear Gauge can be printed directly from the browser by calling the [print](#) method. To use the print functionality, set the [allowPrint](#) property as **true** and inject the **Print** module into **services**.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, Print, Inject } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  var gaugeInstance;
  function clickHandler() {
    gaugeInstance.print();
  }
  return (<div>
    <ButtonComponent onClick= {clickHandler}>print</ButtonComponent>
    <LinearGaugeComponent allowPrint={true} ref={g => gaugeInstance = g}>
      <Inject services={[Print]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, Print, Inject } from '@syncfusion/ej2-react-linear-gauge';
export function App() {
  let gaugeInstance : LinearGaugeComponent;
  function clickHandler() {
    gaugeInstance.print();
  }
  return (<div>
    <ButtonComponent onClick= {clickHandler}>print</ButtonComponent>
    <LinearGaugeComponent allowPrint={true} ref={g => gaugeInstance = g}>
      <Inject services={[Print]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Export

Image Export

To use the image export functionality, set the [allowImageExport](#) property as **true** and inject the **ImageExport** module into **services**. The rendered Linear Gauge can be exported as an image using the [export](#) method. This method requires two parameters: export type and file name. The Linear Gauge can be exported as an image with the following formats.

- JPEG
- PNG
- SVG

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, ImageExport, Inject } from '@syncfusion/ej2-react-linear-gauge';
export function App() {
  function clickHandler() {
    gaugeInstance.export('PNG', 'Gauge');
  }
  var gaugeInstance;
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <LinearGaugeComponent allowImageExport={true} ref={g => gaugeInstance = g}>
      <Inject services={[ImageExport]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, ImageExport, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function clickHandler() {
    gaugeInstance.export('PNG', 'Gauge');
  }
  let gaugeInstance : LinearGaugeComponent;
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <LinearGaugeComponent allowImageExport={true} ref={g => gaugeInstance = g}>
      <Inject services={[ImageExport]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

PDF Export

To use the PDF export functionality, set the [allowPdfExport](#) property as **true** and inject the **PdfExport** module into **services**. The rendered Linear Gauge 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 of the PDF document can be set as **Portrait** or **Landscape**.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, PdfExport, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function clickHandler() {
    gaugeInstance.export('PDF', 'Gauge');
  }
  var gaugeInstance;
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <LinearGaugeComponent allowPdfExport={true} ref={g => gaugeInstance = g}>
      <Inject services={[PdfExport]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, PdfExport, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function clickHandler() {
    gaugeInstance.export('PDF', 'Gauge');
  }
  let gaugeInstance: LinearGaugeComponent;
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <LinearGaugeComponent allowPdfExport={true} ref={g => gaugeInstance =
g}>
      <Inject services={[PdfExport]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Exporting Linear Gauge as base64 string of the file

The Linear Gauge can be exported as base64 string for the JPEG, PNG and PDF formats. The rendered Linear Gauge can be exported as base64 string of the exported image or PDF document used in the [export](#) method. The arguments that are required for this method is export type, file name, orientation of the exported PDF document and **allowDownload** boolean value that is set as **false** to return base64 string. The value for the orientation of the exported PDF document is set as **null** for image export and **Portrait** or **Landscape** for the PDF document.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, ImageExport, Inject } from '@syncfusion/ej2-react-lineargauge';
import {
  PdfPageOrientation
} from '@syncfusion/ej2-pdf-export';
export function App() {
  function clickHandler() {
    gaugeInstance.export('PNG', 'Gauge', PdfPageOrientation.Landscape,
false).then((data)=>{
      document.writeln(data);
    })
  }
  var gaugeInstance;
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <LinearGaugeComponent allowImageExport={true} ref={g => gaugeInstance =
g}>
      <Inject services={[ImageExport]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```

    </LinearGaugeComponent></div>);
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, ImageExport, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function clickHandler() {
    gaugeInstance.export('PNG', 'Gauge', null, false).then((data) => {
      document.writeln(data);
    })
  }
  let gaugeInstance : LinearGaugeComponent;
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <LinearGaugeComponent allowImageExport={true} ref={g => gaugeInstance = g}>
      <Inject services={[ImageExport]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

The exporting of the Linear Gauge as base64 string is not applicable for the SVG format.

Appearance in React Linear gauge component

<!-- markdownlint-disable MD013 -->

Customizing the Linear Gauge area

The following properties are available in the [LinearGaugeComponent](#) to customize the Linear Gauge area.

- [background](#) - Applies the background color for the Linear Gauge.
- [border](#) - To customize the color and width of the border in Linear Gauge.
- [margin](#) - To customize the margins of the Linear Gauge.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (

```

```

    <LinearGaugeComponent background='skyblue' border= { { color: "#FF0000",
width: 2 } }
      margin= { { left: 40, right: 40, top: 40, bottom: 40 } }>
    </LinearGaugeComponent>;
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent background='skyblue' border= { { color: "#FF0000",
width: 2 } }
      margin= { { left: 40, right: 40, top: 40, bottom: 40 } }>
    </LinearGaugeComponent>;
  )
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Setting up the Linear Gauge title

The title for the Linear Gauge can be set using [title](#) property in [LinearGaugeComponent](#). Its appearance can be customized using the [titleStyle](#) with the below properties.

- [color](#) - Specifies the text color of the title.
- [fontStyle](#) - Specifies the font style of the title.
- [fontWeight](#) - Specifies the font weight of the title.
- [size](#) - Specifies the font size of the title.
- [opacity](#) - Specifies the opacity of the title.
- [fontFamily](#) - Specifies the font family of the title.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent title='linear gauge'
      titleStyle={ { fontFamily:'Arial', fontStyle:'italic',
fontWeight:'regular', color:'#E27F2D', size:'23px' } }>
    </LinearGaugeComponent>;
  )
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent title='linear gauge'
      titleStyle={ { fontFamily:'Arial', fontStyle:'italic',
fontWeight:'regular', color:'#E27F2D', size:'23px' } }>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Customizing the Linear Gauge container

The area used to render the ranges and pointers at the center position of the gauge is called container. The following types of container to be applicable for Linear Gauge.

- Normal
- Rounded Rectangle
- Thermometer

The type of the container can be modified by using the [type](#) property in [container](#). The container can be customized by using the following properties in [container](#).

- [offset](#) - To place the container with the specified distance from the axis of the Linear Gauge.
- [width](#) - To set the thickness of the container.
- [height](#) - To set the length of the container.
- [backgroundColor](#) - To set the background color of the container.
- [border](#) - To set the color and width for the border of the container.

Normal

The **Normal** type will render the container as a rectangle and this is the default container type.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective ,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent container={ { width:30 } }>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
```



```

        <PointerDirective value={50} width={15} type='Bar'>
        </PointerDirective>
    </PointersDirective>
</AxisDirective>
</AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective ,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
    return (
        <LinearGaugeComponent container={ { width:30 } }>
            <AxesDirective>
                <AxisDirective>
                    <PointersDirective>
                        <PointerDirective value={50} width={15} type='Bar'>
                        </PointerDirective>
                    </PointersDirective>
                </AxisDirective>
            </AxesDirective>
        </LinearGaugeComponent>);
    }
    const root = ReactDOM.createRoot(document.getElementById('container'));
    root.render(<App />);
    {% endraw %}

```

Rounded Rectangle

The **RoundedRectangle** type will render the container as a rectangle with rounded corner radius. The rounded corner radius of the container can be customized using the [roundedCornerRadius](#) property in [container](#).

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective ,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
    return (
        <LinearGaugeComponent container={ { width:30, type:'RoundedRectangle' } }
    >>
        <AxesDirective>
            <AxisDirective>

```

```

        <PointersDirective>
            <PointerDirective value={50} width={15} type='Bar'>
            </PointerDirective>
        </PointersDirective>
    </AxisDirective>
</AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
    return (
        <LinearGaugeComponent container={ { width:30, type:'RoundedRectangle' } }>
        <AxesDirective>
            <AxisDirective>
                <PointersDirective>
                    <PointerDirective value={50} width={15} type='Bar'>
                    </PointerDirective>
                </PointersDirective>
            </AxisDirective>
        </AxesDirective>
        </LinearGaugeComponent>);
    }
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Thermometer

The **Thermometer** type will render the container similar to the appearance of thermometer.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
    return (
        <LinearGaugeComponent container={ { width:30, type:'Thermometer' } }>
        <AxesDirective>
            <AxisDirective>
                <PointersDirective>
                    <PointerDirective value={50} width={15} type='Bar'>

```

```

        </PointerDirective>
      </PointersDirective>
    </AxisDirective>
  </AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  return (
    <LinearGaugeComponent container={ { width:30, type:'Thermometer' } }>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective value={50} width={15} type='Bar'>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Fitting the Linear Gauge to the control

The Linear Gauge component is rendered with margin by default. To remove the margin around the Linear Gauge, the [allowMargin](#) property in [LinearGaugeComponent](#) is set as **false**.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent allowMargin={false} height="100%" width="100%"
orientation='Horizontal' margin= {{ left: 0, right: 0, top: 0, bottom: 0 }}
border= {{ width: 2, color: "red" }}>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  return (
    <LinearGaugeComponent allowMargin={false} height="100%" width="100%"
orientation='Horizontal' margin= {{ left: 0, right: 0, top: 0, bottom: 0 }}
border= {{ width: 2, color: "red" }}>
    </LinearGaugeComponent>;
  )
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

To use this feature, set the [allowMargin](#) property to **false**, the [width](#) property to **100%** and the properties of [margin](#) to **0**.

Accessibility in React Linear Gauge component

The Linear Gauge component follows commonly used accessibility guidelines and standards, such as [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#).

The accessibility compliance for the Linear Gauge component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device 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 Linear Gauge component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Linear Gauge component:

| Attributes | Purpose |

| --- | --- |

| **role=region** | It is specified in the title and pointer. The pointer supports the interactive drag-and-drop function to update the pointer value. |

| **aria-label** | Provides an accessible name for the title, axis labels, text pointer and annotation. |

Screen reading in Linear Gauge

Accessibility in the Linear Gauge component ensures that all users, regardless of ability or disability, can use screen reading. The following Linear Gauge elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |

| --- | --- |

| Title | Reads the title of the Linear Gauge. |

| Axis labels | Reads the axis labels of the Linear Gauge. |

| Text pointer | Reads the text content shown as a pointer in Linear Gauge. |

| Annotation | Reads the content specified in the annotation. |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Internationalization in React Linear gauge component

Globalization is the process of designing and developing a component that works in different cultures. Internationalization is used to globalize the number content in Linear Gauge component using [format](#) property in [LinearGaugeComponent](#). It has static text on some features such as

- Axis label
- Tooltip

The static text on above features can be changed to any culture such as Arabic, Deutsch and French. To know more about the globalization in React components, refer [here](#).

Numeric Format

The text in axis labels and tooltip can be displayed in the numeric format such as currency, percentage and so on. To know more about the numeric formats in axis labels, refer [here](#). In the below example, the axis label is displayed in the currency format.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
import { setCulture, setCurrencyCode } from '@syncfusion/ej2-base';
setCulture('de');
setCurrencyCode('EUR');
export function App() {
  return (
    <LinearGaugeComponent format='c'>
      <AxesDirective>
        <AxisDirective minimum={0} maximum={120} majorTicks={ {
interval:10, height:10 } } minorTicks={ { interval:5, height:5 } } >
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
'@syncfusion/ej2-react-lineargauge';
import { setCulture, setCurrencyCode } from '@syncfusion/ej2-base';
setCulture('de');
setCurrencyCode('EUR');
export function App() {
  return (
    <LinearGaugeComponent format='c'>
      <AxesDirective>
```

```

        <AxisDirective minimum={0} maximum={120} majorTicks={ {
interval:10, height:10 } } minorTicks={ { interval:5, height:5 } } >
        </AxisDirective>
    </AxesDirective>
</LinearGaugeComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Events in React Linear gauge component

This section describes the Linear Gauge component's event that gets triggered when corresponding operations are performed.

animationComplete

When the pointer animation is completed, the [animationComplete](#) event will be triggered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
    function animationComplete(args) {
    }
    return (<div>
    <LinearGaugeComponent animationComplete={animationComplete}>
        <AxesDirective>
            <AxisDirective>
                <PointersDirective>
                    <PointerDirective value={50} animationDuration={1000}>
                    </PointerDirective>
                </PointersDirective>
            </AxisDirective>
        </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, IAnimationCompleteEventArgs } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
    function animationComplete(args: IAnimationCompleteEventArgs) {

```

```

    }
    return (<div>
      <LinearGaugeComponent animationComplete={animationComplete}>
        <AxesDirective>
          <AxisDirective>
            <PointersDirective>
              <PointerDirective value={50} animationDuration={1000}>
            </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent></div>);
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

annotationRender

Before the annotation is rendered in the Linear Gauge, the [annotationRender](#) event will be triggered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Annotations, AnnotationsDirective, AnnotationDirective,
LinearGaugeComponent, Inject } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function annotationRender(args) {
  }
  return (<div>
    <LinearGaugeComponent annotationRender={annotationRender}>
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Gauge</h1></div>' axisValue={0} zIndex='1'>
        </AnnotationDirective>
      </AnnotationsDirective>
    </LinearGaugeComponent></div>);
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Annotations, AnnotationsDirective, AnnotationDirective,
LinearGaugeComponent, Inject, IAnnotationRenderEventArgs } from
 '@syncfusion/ej2-react-lineargauge';
export function App() {
  function annotationRender(args: IAnnotationRenderEventArgs) {

```



```

    }
    return (<div>
      <LinearGaugeComponent annotationRender={annotationRender}>
        <Inject services={[Annotations]} />
        <AnnotationsDirective>
          <AnnotationDirective content='<div
id="first"><h1>Gauge</h1></div>' axisValue={0} zIndex='1'>
            </AnnotationDirective>
          </AnnotationsDirective>
        </LinearGaugeComponent></div>);
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

axisLabelRender

Before each axis label is rendered in the Linear Gauge, the [axisLabelRender](#) event is fired. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  function axisLabelRender(args) {
  }
  return (<div>
    <LinearGaugeComponent axisLabelRender={axisLabelRender}>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, IAxisLabelRenderEventArgs } from
'@syncfusion/ej2-react-lineargauge';
export function App() {

```

```
function axisLabelRender(args: IAxisLabelRenderEventArgs) {
}
return (<div>
  <LinearGaugeComponent axisLabelRender={axisLabelRender}>
    <AxesDirective>
      <AxisDirective>
        <PointersDirective>
          <PointerDirective>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

beforePrint

The [beforePrint](#) event is fired before the print begins. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, Print, Inject } from '@syncfusion/ej2-react-linear-gauge';
export function App() {
  var gaugeInstance;
  function clickHandler() {
    gaugeInstance.print();
  }
  function beforePrint(args) {
  }
  return (<div>
    <ButtonComponent onClick= { clickHandler }>print</ButtonComponent>
    <LinearGaugeComponent allowPrint={true} ref={g => gaugeInstance = g}
    beforePrint={beforePrint}>
      <Inject services={[Print]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
```

```
import { LinearGaugeComponent, Print, Inject, IPrintEventArgs } from
 '@syncfusion/ej2-react-lineargauge';
export function App() {
  let gaugeInstance : LinearGaugeComponent;
  function clickHandler(){
    gaugeInstance.print();
  }
  function beforePrint(args: IPrintEventArgs){
  }
  return (<div>
    <ButtonComponent onClick= { clickHandler }>print</ButtonComponent>
    <LinearGaugeComponent allowPrint={true} ref={g => gaugeInstance = g}
beforePrint={beforePrint}>
      <Inject services={[Print]} />
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

dragEnd

The [dragEnd](#) event will be fired before the pointer drag is completed. To know more about the argument of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App(){
  function dragEnd(args){
  }
  return (<div>
    <LinearGaugeComponent dragEnd={dragEnd}>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective enableDrag={true}>
              </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, IPointerDragEventArgs } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  function dragEnd(args: IPointerDragEventArgs) {
  }
  return (<div>
    <LinearGaugeComponent dragEnd={dragEnd}>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective enableDrag={true}>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

dragMove

The [dragMove](#) event will be fired when the pointer is dragged. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  function dragMove(args) {
  }
  return (<div>
    <LinearGaugeComponent dragMove={dragMove}>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective enableDrag={true}>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, IPointerDragEventArgs } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  function dragMove(args: IPointerDragEventArgs) {
  }
  return (<div>
    <LinearGaugeComponent dragMove={dragMove}>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective enableDrag={true}>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

dragStart

When the pointer drag begins, the [dragStart](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  function dragStart(args) {
  }
  return (<div>
    <LinearGaugeComponent dragStart={dragStart}>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective enableDrag={true}>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, IPointerDragEventArgs } from
'@syncfusion/ej2-react-lineargauge';
export function App() {
  function dragStart(args: IPointerDragEventArgs) {
  }
  return (<div>
    <LinearGaugeComponent dragStart={dragStart}>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective enableDrag={true}>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

gaugeMouseDown

When mouse is pressed down on the gauge, the [gaugeMouseDown](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function gaugeMouseDown(args) {
  }
  return (<div>
    <LinearGaugeComponent gaugeMouseDown={gaugeMouseDown}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, IMouseEventArgs } from '@syncfusion/ej2-
react-lineargauge';
export function App() {
```

```
function gaugeMouseDown(args: IMouseEventArgs) {
}
return (<div>
<LinearGaugeComponent gaugeMouseDown={gaugeMouseDown}>
</LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

gaugeMouseLeave

When mouse pointer moves over the gauge, the [gaugemouseleave](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function gaugeMouseLeave(args) {
  }
  return (<div>
<LinearGaugeComponent gaugeMouseLeave={gaugeMouseLeave}>
</LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, IMouseEventArgs } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function gaugeMouseLeave(args: IMouseEventArgs) {
  }
  return (<div>
<LinearGaugeComponent gaugeMouseLeave={gaugeMouseLeave}>
</LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

gaugeMouseMove

When mouse pointer leaves the gauge, the [gaugeMouseMove](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function gaugeMouseMove(args) {
  }
  return (<div>
    <LinearGaugeComponent gaugeMouseMove={gaugeMouseMove}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, IMouseEventArgs } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function gaugeMouseMove(args: IMouseEventArgs) {
  }
  return (<div>
    <LinearGaugeComponent gaugeMouseMove={gaugeMouseMove}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

gaugeMouseUp

When the mouse pointer is released over the Linear Gauge, the [gaugeMouseUp](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function gaugeMouseUp(args) {
  }
  return (<div>
    <LinearGaugeComponent gaugeMouseUp={gaugeMouseUp}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```


INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, IMouseEventArgs } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function gaugeMouseUp(args: IMouseEventArgs){
  }
  return (<div>
    <LinearGaugeComponent gaugeMouseUp={gaugeMouseUp}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

load

Before the Linear Gauge is loaded, the [load](#) event is fired. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function load(args){
  }
  return (<div>
    <LinearGaugeComponent load={load}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, ILoadEventArgs } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function load(args: ILoadEventArgs){
  }
  return (<div>
    <LinearGaugeComponent load={load}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

loaded

After the Linear Gauge has been loaded, the [loaded](#) event will be triggered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function loaded(args) {
  }
  return (<div>
    <LinearGaugeComponent loaded={loaded}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, ILoadedEventArgs } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function loaded(args: ILoadedEventArgs) {
  }
  return (<div>
    <LinearGaugeComponent loaded={loaded}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

resized

After the window resizing, the [resized](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function resized(args) {
  }
  return (<div>
    <LinearGaugeComponent resized={resized}>

```

```

    </LinearGaugeComponent></div>);
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, IResizeEventArgs } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function resized(args: IResizeEventArgs){
  }
  return (<div>
    <LinearGaugeComponent resized={resized}>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

tooltipRender

The [tooltipRender](#) event is fired before the tooltip is rendered. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
  PointersDirective, PointerDirective,
  GaugeTooltip, Inject
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function tooltipRender(args) {
  }
  return (<div>
    <LinearGaugeComponent tooltipRender={tooltipRender} tooltip={{ enable:
true }}>
    <Inject services={[GaugeTooltip]} />
    <AxesDirective>
      <AxisDirective>
        <PointersDirective>
          <PointerDirective>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));

```

```
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
  PointersDirective, PointerDirective,
  ITooltipRenderEventArgs, GaugeTooltip, Inject
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function tooltipRender(args: ITooltipRenderEventArgs) {
  }
  return (<div>
    <LinearGaugeComponent tooltipRender={tooltipRender} tooltip={{ enable:
true }}>
    <Inject services={[GaugeTooltip]} />
    <AxesDirective>
      <AxisDirective>
        <PointersDirective>
          <PointerDirective>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

valueChange

The [valueChange](#) event is triggered when the pointer is dragged from one value to another. To know more about the arguments of this event, refer [here](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
  PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
  function valueChange(args) {
  }
  return (<div>
    <LinearGaugeComponent valueChange={valueChange}>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective enableDrag={true}>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```

        </PointersDirective>
      </AxisDirective>
    </AxesDirective>
  </LinearGaugeComponent></div>;
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, IValueChangeEventArgs } from
"@syncfusion/ej2-react-lineargauge";
export function App() {
  function valueChange(args: IValueChangeEventArgs) {
  }
  return (<div>
    <LinearGaugeComponent valueChange={valueChange}>
      <AxesDirective>
        <AxisDirective>
          <PointersDirective>
            <PointerDirective enableDrag={true}>>
            </PointerDirective>
          </PointersDirective>
        </AxisDirective>
      </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Methods in React Linear gauge component

The following methods are available in the Linear Gauge component.

setPointerValue

To change the pointer value dynamically, use the [setPointerValue](#) method in the Linear Gauge component. The following are the arguments for this method.

Argument name	Description
axisIndex	Specifies the index of the axis in which the pointer value is to be updated.
pointerIndex	Specifies the index of the pointer to be updated.
pointerValue	Specifies the value of the pointer to be updated.

INDEX.JSX

```

{% raw %}
import * as React from "react";

```

```

import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
    function clickHandler() {
        gaugeInstance.setPointerValue(0, 0, 30);
    }
    var gaugeInstance;
    return (<div>
        <ButtonComponent onClick= { clickHandler }>Click</ButtonComponent>
        <LinearGaugeComponent ref={g => gaugeInstance = g}>
            <AxesDirective>
                <AxisDirective>
                    <PointersDirective>
                        <PointerDirective value={80}>
                    </PointerDirective>
                </PointersDirective>
            </AxisDirective>
        </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective } from '@syncfusion/ej2-react-
lineargauge';
export function App() {
    function clickHandler() {
        gaugeInstance.setPointerValue(0, 0, 30);
    }
    let gaugeInstance : LinearGaugeComponent;
    return (<div>
        <ButtonComponent onClick= { clickHandler }>Click</ButtonComponent>
        <LinearGaugeComponent ref={g => gaugeInstance = g}>
            <AxesDirective>
                <AxisDirective>
                    <PointersDirective>
                        <PointerDirective value={80}>
                    </PointerDirective>
                </PointersDirective>
            </AxisDirective>
        </AxesDirective>
    </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);

```

```
{% endraw %}
```

setAnnotationValue

To change the annotation content dynamically, use the [setAnnotationValue](#) method in the Linear Gauge component. The following are the arguments for this method.

Argument name	Description
-----	-----
annotationIndex	Specifies the index number of the annotation to be updated.
content	Specifies the text for the annotation to be updated.
axisValue	Specifies the value of the axis where the annotation is to be placed.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { Annotations, AnnotationsDirective, AnnotationDirective,
LinearGaugeComponent, AxesDirective, AxisDirective, PointersDirective,
Inject, PointerDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function clickHandler() {
    gaugeInstance.setAnnotationValue(0, '50', 50);
  }
  var gaugeInstance;
  return (<div>
    <ButtonComponent onClick= { clickHandler }>Click</ButtonComponent>
    <LinearGaugeComponent ref={g => gaugeInstance = g}>
      <Inject services={[Annotations]}>
        <AnnotationsDirective>
          <AnnotationDirective content='10' axisValue={0} zIndex='1'>
            </AnnotationDirective>
        </AnnotationsDirective>
        <AxesDirective>
          <AxisDirective>
            <PointersDirective>
              <PointerDirective value={10}>
                </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
```

```

import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { Annotations, AnnotationsDirective, AnnotationDirective,
LinearGaugeComponent, AxesDirective, AxisDirective, PointersDirective,
Inject, PointerDirective } from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function clickHandler() {
    gaugeInstance.setAnnotationValue(0, '50', 50);
  }
  let gaugeInstance : LinearGaugeComponent;
  return (<div>
    <ButtonComponent onClick= { clickHandler }>Click</ButtonComponent>
    <LinearGaugeComponent ref={g => gaugeInstance = g}>
      <Inject services=[Annotations]>/>
      <AnnotationsDirective>
        <AnnotationDirective content='10' axisValue={0} zIndex='1'>
          </AnnotationDirective>
        </AnnotationsDirective>
        <AxesDirective>
          <AxisDirective>
            <PointersDirective>
              <PointerDirective value={10}>
                </PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

refresh

The [refresh](#) method can be used to change the state of the component and render it again.

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import {
  LinearGaugeComponent,
  AxesDirective,
  AxisDirective,
  PointersDirective,
  PointerDirective,
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function clickHandler() {
    gaugeInstance.axes[0].pointers[0].value = 50;
    gaugeInstance.refresh();
  }
  var gaugeInstance;
  return (
    <div>
      <ButtonComponent onClick={clickHandler}>

```



```

        Click
      </ButtonComponent>
      <LinearGaugeComponent ref={ (g) => (gaugeInstance = g) }>
        <AxesDirective>
          <AxisDirective>
            <PointersDirective>
              <PointerDirective value={10}></PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>
    </div>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import {
  LinearGaugeComponent,
  AxesDirective,
  AxisDirective,
  PointersDirective,
  PointerDirective,
} from '@syncfusion/ej2-react-lineargauge';
export function App() {
  function clickHandler() {
    gaugeInstance.axes[0].pointers[0].value = 50;
    gaugeInstance.refresh();
  }
  let gaugeInstance : LinearGaugeComponent;
  return (
    <div>
      <ButtonComponent onClick={clickHandler}>
        Click
      </ButtonComponent>
      <LinearGaugeComponent ref={ (g) => (gaugeInstance = g) }>
        <AxesDirective>
          <AxisDirective>
            <PointersDirective>
              <PointerDirective value={10}></PointerDirective>
            </PointersDirective>
          </AxisDirective>
        </AxesDirective>
      </LinearGaugeComponent>
    </div>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);

```

```
{% endraw %}
```

Ej1 api migration in React Linear gauge component

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

Linear gauge dimensions

```
{% raw %}
```

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

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

| **Height** | **Property:** *height*
 ReactDOM.render(<EJ.LinearGauge id= "gauge" height= '100px'> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *height*
 ReactDOM.render(<LinearGaugeComponent id='gauge' height= '100px'> </LinearGaugeComponent>, document.getElementById('gauge')); |

| **Width** | **Property:** *width*
 ReactDOM.render(<EJ.LinearGauge id= "gauge" width= '200px'> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *width*
 ReactDOM.render(<LinearGaugeComponent id='gauge' width= '200px'> </LinearGaugeComponent>, document.getElementById('gauge')); |

| **Height(In Percentage)** | Not Applicable | **Property:** *height*
 ReactDOM.render(<LinearGaugeComponent id='gauge' height= '70%'> </LinearGaugeComponent>, document.getElementById('gauge')); |

| **Width(In Percentage)** | Not Applicable | **Property:** *width*
 ReactDOM.render(<LinearGaugeComponent id='gauge' width= '80%'> </LinearGaugeComponent>, document.getElementById('gauge')); |

Line customization

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

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

| **Height** | **Property:** *scales.length*
 var scale = [{ length: 300 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.line.height*
 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <AxisDirective line= {height= '150'}> </AxisDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |

| **Width** | **Property:** *scales.width*
 var scale = [{ width: 300 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.line.width*
 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <AxisDirective line= {width= 2}> </AxisDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |

| **Color** | **Property:** *scales.backgroundColor*
 var scale = [{ backgroundColor: "blue" }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.line.color*
 ReactDOM.render(

```
<LinearGaugeComponent id='gauge'> <br/> <AxesDirective> <AxisDirective line= {color=
'#4286f4'}/> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge')));|
```

```
| Offset| Not Applicable | Property: axes.line.offset<br/><br/> ReactDOM.render(
<LinearGaugeComponent id='gauge'> <br/> <AxesDirective> <AxisDirective line= {offset= 2}/>
</AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')));|
```

```
| Opacity| Property: scales.opacity<br/><br/> var scale = [{ opacity: 0.2 }] <br/> ReactDOM.render(
<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
document.getElementById('gauge')));| Not Applicable |
```

```
| DashArray| Not Applicable | Property: axes.line.dashArray<br/><br/> ReactDOM.render(
<LinearGaugeComponent id='gauge'> <br/> <AxesDirective> <AxisDirective line= {dashArray=
1}/> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')));|
```

Ticks

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

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

```
| Type of Ticks| Property: scales.ticks.type<br/><br/> var scale = [{ <br/> &#160; ticks: [{ type:
"majorinterval" }] <br/> }] <br/> ReactDOM.render( <EJ.LinearGauge id= "gauge" scales={scale}>
</EJ.LinearGauge>, document.getElementById('gauge')));| Property:
axes.majorTicks.height<br/><br/> ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/>
<AxesDirective> <AxisDirective majorTicks= { }/> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge')));|
```

```
| Height of Major Ticks| Property: scales.ticks.height<br/><br/> var scale = [{ <br/> &#160; ticks: [{
type: "majorinterval", height:8 }] <br/> }] <br/> ReactDOM.render( <EJ.LinearGauge id= "gauge"
scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')));| Property:
axes.majorTicks.height<br/><br/> ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/>
<AxesDirective> <AxisDirective majorTicks= { height: 10 }/> </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge')));|
```

```
| Width of Major Ticks| Property: scales.ticks.width<br/><br/> var scale = [{ <br/> &#160; ticks: [{
type: "majorinterval", width: 5 }] <br/> }] <br/> ReactDOM.render( <EJ.LinearGauge id= "gauge"
scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')));| Property:
axes.majorTicks.width<br/><br/> ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/>
<AxesDirective> <AxisDirective majorTicks= { width: 2 }/> </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge')));|
```

```
| Color of Major Ticks| Property: scales.ticks.color<br/><br/> var scale = [{ <br/> &#160; ticks: [{ color:
"blue" }] <br/> }] <br/> ReactDOM.render( <EJ.LinearGauge id= "gauge" scales={scale}>
</EJ.LinearGauge>, document.getElementById('gauge')));| Property:
axes.majorTicks.color<br/><br/> ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/>
<AxesDirective> <AxisDirective majorTicks= { color: "Blue" }/> </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge')));|
```

|Offset for Major Ticks| **Property:** *scales.ticks.distanceFromScale*
 var scale = [{ ticks: [{ distanceFromScale: {x: 5, y: 5} }] }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'))
Property: *axes.majorTicks.offset*
 ReactDOM.render(<LinearGaugeComponent id='gauge' >
 <AxesDirective> <AxisDirective majorTicks= { offset: 1 }/> </AxesDirective>
 </LinearGaugeComponent>, document.getElementById('gauge'));

|Interval of Major Ticks| **Property:** *scales.majorIntervalValue*
 var scale = [{ majorIntervalValue: 15 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'))
Property: *axes.majorTicks.interval*
 ReactDOM.render(<LinearGaugeComponent id='gauge' >
 <AxesDirective> <AxisDirective majorTicks= { interval: 20 }/> </AxesDirective>
 </LinearGaugeComponent>, document.getElementById('gauge'));

|Angle of Major Ticks| **Property:** *scales.ticks.angle*
 var scale = [{ ticks: [{ angle: 30 }] }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'))
 | Not Applicable |

|Opcity of Major Ticks| **Property:** *scales.ticks.opacity*
 var scale = [{ ticks: [{ opacity: 0.5 }] }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'))
 | Not Applicable |

|Height of Minor Ticks| **Property:** *scales.ticks.height*
 var scale = [{ ticks: [{ type: "minorinterval", height:8 }] }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'))
Property: *axes.minorTicks.height*
 ReactDOM.render(<LinearGaugeComponent id='gauge' >
 <AxesDirective> <AxisDirective minorTicks= { height: 10 }/> </AxesDirective>
 </LinearGaugeComponent>, document.getElementById('gauge'));

|Width of Minor Ticks| **Property:** *scales.ticks.width*
 var scale = [{ ticks: [{ type: "minorinterval", width: 5 }] }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'))
Property: *axes.minorTicks.width*
 ReactDOM.render(<LinearGaugeComponent id='gauge' >
 <AxesDirective> <AxisDirective minorTicks= { width: 2 }/> </AxesDirective>
 </LinearGaugeComponent>, document.getElementById('gauge'));

|Color of Minor Ticks| **Property:** *scales.ticks.color*
 var scale = [{ ticks: [{ type: "minorinterval", color: "blue" }] }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'))
Property: *axes.minorTicks.color*
 ReactDOM.render(<LinearGaugeComponent id='gauge' >
 <AxesDirective> <AxisDirective minorTicks= { color: "Blue" }/> </AxesDirective>
 </LinearGaugeComponent>, document.getElementById('gauge'));

|Offset for Minor Ticks| **Property:** *scales.ticks.distanceFromScale*
 var scale = [{ ticks: [{ type: "minorinterval", distanceFromScale: {x: 5, y: 5} }] }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'))
Property: *axes.minorTicks.offset*
 ReactDOM.render(<LinearGaugeComponent id='gauge' >
 <AxesDirective> <AxisDirective

```
minorTicks= { offset: 1 }></AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));
```

| Interval of Minor Ticks | **Property:** *scales.minorIntervalValue*

 var scale = [{

minorIntervalValue: 8
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"
scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:**
axes.minorTicks.interval

 ReactDOM.render(<LinearGaugeComponent id='gauge'>

<AxesDirective> <AxisDirective minorTicks= { interval: 5 }> </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge')); |

| Angle of Minor Ticks | **Property:** *scales.ticks.angle*

 var scale = [{
 ticks: [{ type:
"minorinterval",
 angle: 30 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id=
"gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | Not
Applicable |

| Opacity of Minor Ticks | **Property:** *scales.ticks.opacity*

 var scale = [{
 ticks: [{
type: "minorinterval",
 opacity: 0.5 }]
 }]
 ReactDOM.render(
<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
document.getElementById('gauge')); | Not Applicable |

Labels

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

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

| Angle | **Property:** *scales.labels.angle*

 var scale = [{
 labels: [{ angle: 15 }]

 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}>
</EJ.LinearGauge>, document.getElementById('gauge')); | Not Applicable |

| Offset | **Property:** *scales.labels.distanceFromScale*

 var scale = [{
 labels:
[{distanceFromScale:{x: -5, y: 10} }]
 }]
 ReactDOM.render(<EJ.LinearGauge id=
"gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:**
axes.labelStyle.offset

 ReactDOM.render(<LinearGaugeComponent id='gauge'>

<AxesDirective> <AxisDirective labelStyle= { offset: 3 }> </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge')); |

| Format | **Property:** *scales.labels.unitText*

 var scale = [{
 labels: [{ unitText:
"F" }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}>
</EJ.LinearGauge>, document.getElementById('gauge')); | **Property:**
axes.labelStyle.format

 ReactDOM.render(<LinearGaugeComponent id='gauge'>

<AxesDirective> <AxisDirective labelStyle= { format: 'c' }> </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge')); |

| Unit Text Placement | **Property:** *scales.labels.unitTextPlacement*

 var scale = [{

labels: [{ unitTextPlacement: "front" }]
 }]
 ReactDOM.render(<EJ.LinearGauge id=
"gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | Not
Applicable |

| Label Range Color | Not Applicable | **Property:** *axes.labelStyle.useRangeColor*

ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <AxisDirective

```
labelStyle= { useRangeColor: true }/ > </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));
```

| Opacity | **Property:** *scales.labels.opacity*

 Property:

```
scales.labels.unitTextPlacement <br/> <br/> var scale = [{ <br/> &#160; labels: [{ opacity: 0.5 }] <br/>
}] <br/> ReactDOM.render( <EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
document.getElementById('gauge')); | Property: axes.labelStyle.font.opacity <br/> <br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AxesDirective> <AxisDirective
labelStyle= { font: { opacity: 0.5 } }/ > </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));
```

| Label Text Color | **Property:** *scales.labels.textColor*

 var scale = [{
 labels: [{
textColor: "red" }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}>
</EJ.LinearGauge>, document.getElementById('gauge')); | **Property:**
axes.labelStyle.font.color

 ReactDOM.render(<LinearGaugeComponent id='gauge'>

 <AxesDirective> <AxisDirective labelStyle= { font: { color: "red" } }/ > </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge'));

| Label Font Family | **Property:** *scales.labels.font.fontFamily*

 var scale = [{

labels: [{font:{fontFamily:"Segoe UI"}}]
 }]
 ReactDOM.render(<EJ.LinearGauge id=
"gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:**
axes.labelStyle.font.fontFamily

 ReactDOM.render(<LinearGaugeComponent
id='gauge'>
 <AxesDirective> <AxisDirective labelStyle= { font: {fontFamily:"Segoe UI"} }/ >
</AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge'));

| Label Font Style | **Property:** *scales.labels.font.fontStyle*

 var scale = [{
 labels: [{
font:{fontStyle:"Bold"} }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"
scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:**
axes.labelStyle.font.fontStyle

 ReactDOM.render(<LinearGaugeComponent id='gauge'>

 <AxesDirective> <AxisDirective labelStyle= { font: { fontStyle: "Bold" } }/ > </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge'));

| Label Size | **Property:** *scales.labels.font.size*

 var scale = [{
 labels: [{ font: {
size: "20px" }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}>
</EJ.LinearGauge>, document.getElementById('gauge')); | **Property:**
axes.labelStyle.font.size

 ReactDOM.render(<LinearGaugeComponent id='gauge'>

<AxesDirective> <AxisDirective labelStyle= { font: { size: "15px" } }/ > </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge'));

| Label Font Weight | Not Applicable | **Property:** *axes.labelStyle.font.fontWeight*

ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <AxisDirective
labelStyle= { font: { fontWeight: '4' } }/ > </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));

Axis

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

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

| Minimum Value | **Property:** *scales.minimum*
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.minimum*
 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <AxisDirective minimum= 20 /> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |

| Maximum Value | **Property:** *scales.maximum*
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.maximum*
 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <AxisDirective maximum= 120 /> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |

| Inverted Position | Not Applicable | **Property:** *axes.isInversed*
 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <AxisDirective isInversed= true /> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |

| Opposed Position | Not Applicable | **Property:** *axes.opposedPosition*
 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <AxisDirective opposedPosition= true /> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |

Ranges

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

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

| Start Value | **Property:** *scales.ranges.startValue*
 var scale = [{
 ranges: [{ startValue: 20 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.ranges.start*
 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <RangesDirective>
 <RangeDirective start={20} /> </RangesDirective> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |

| End Value | **Property:** *scales.ranges.endValue*
 var scale = [{
 ranges: [{ endValue: 20 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.ranges.end*
 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <RangesDirective>
 <RangeDirective end={20} /> </RangesDirective> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |

| Start Width | **Property:** *scales.ranges.startWidth*
 var scale = [{
 ranges: [{ startWidth: 10 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.ranges.startWidth*
 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <RangesDirective>
 <RangeDirective startWidth={20} /> </RangesDirective> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |

| End Width | **Property:** *scales.ranges.endWidth*
 var scale = [{
 ranges: [{ endWidth: 15 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}>

```

</EJ.LinearGauge>, document.getElementById('gauge'));| Property:
axes.ranges.endWidth<br/><br/> ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/>
<AxesDirective> <RangesDirective> <br/> <RangeDirective endWidth={15} />
</RangesDirective> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));|

| Color| Property: scales.ranges.backgroundColor<br/><br/> var scale = [{ <br/> &#160; ranges: [{
backgroundColor: "red" }]} <br/> ]<br/> ReactDOM.render( <EJ.LinearGauge id= "gauge"
scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'));| Property:
axes.ranges.color<br/><br/> ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/>
<AxesDirective> <RangesDirective> <br/> <RangeDirective color="red" /> </RangesDirective>
</AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge'));|

| Offset| Property: scales.ranges<br/><br/> var scale = [{ <br/> &#160; ranges: [{ distanceFromScale:
{10} }]} <br/> ]<br/> ReactDOM.render( <EJ.LinearGauge id= "gauge" scales={scale}>
</EJ.LinearGauge>, document.getElementById('gauge'));| Property: axes.ranges.offset<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AxesDirective>
<RangesDirective> <br/> <RangeDirective offset= 5 /> </RangesDirective> </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge'));|

| Range Position| Property: scales.ranges.placement<br/><br/> var scale = [{ <br/> &#160; ranges: [{
placement: "Near" }]} <br/> ]<br/> ReactDOM.render( <EJ.LinearGauge id= "gauge"
scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'));| Property:
axes.ranges.position<br/><br/> ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/>
<AxesDirective> <RangesDirective> <br/> <RangeDirective position= "Inside" />
</RangesDirective> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));|

| Opacity| Property: scales.ranges.opacity<br/><br/> var scale = [{ <br/> &#160; ranges: [{ opacity:
{0.3} }]} <br/> ]<br/> ReactDOM.render( <EJ.LinearGauge id= "gauge" scales={scale}>
</EJ.LinearGauge>, document.getElementById('gauge'));| Not Applicable|

| Border Customization| Property: scales.ranges.border<br/><br/> var scale = [{ <br/> &#160; ranges:
[{ border: { color: 'green', <br/> &#160; width: 2 } }]} <br/> ]<br/> ReactDOM.render(
<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
document.getElementById('gauge'));| Property: axes.ranges.border<br/><br/> ReactDOM.render(
<LinearGaugeComponent id='gauge'> <br/> <AxesDirective> <RangesDirective> <br/>
<RangeDirective border= {color: 'blue', width: 2}/> </RangesDirective> </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge'));|

```

Bar Pointer

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

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

```

| Bar Pointer| Property: scales.barPointers.value<br/><br/> var scale = [{ <br/> &#160; barPointers: [{
value: 20 }]} <br/> ]<br/> ReactDOM.render( <EJ.LinearGauge id= "gauge" scales={scale}>
</EJ.LinearGauge>, document.getElementById('gauge'));| Property: axes.pointers.value<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AxesDirective>

```



```
<PointersDirective> <br/> <PointerDirective type= 'Bar', value= {20} /> </PointersDirective>
</AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge'));|
```

| Color of Bar Pointer | **Property:** *scales.barPointers.backgroundColor*

 var scale = [{
 barPointers: [{ value: 20,
 backgroundColor: "red" }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.pointers.color*

 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <PointersDirective>
 <PointerDirective type= 'Bar',value= {20},color= 'red' /> </PointersDirective> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge'));|

| Offset of Bar Pointer | **Property:** *scales.barPointers.distanceFromScale*

 var scale = [{
 barPointers: [{ distanceFromScale: 20 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.pointers.offset*

 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <PointersDirective>
 <PointerDirective type= 'Bar',value= {20},offset= {20} /> </PointersDirective> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge'));|

| Opacity of Bar Pointer | **Property:** *scales.barPointers.opacity*

 var scale = [{
 barPointers: [{ value: 40, opacity: 0.5 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.pointers.opacity*

 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <PointersDirective>
 <PointerDirective type= 'Bar',value= {20},opacity= {0.5} /> </PointersDirective> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge'));|

| Width of Bar Pointer | **Property:** *scales.barPointers.width*

 var scale = [{
 barPointers: [{ value: 40, width: 25 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.pointers.width*

 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <PointersDirective>
 <PointerDirective type= 'Bar',value= {20},width= {25} /> </PointersDirective> </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge'));|

| Gradients of Bar Pointer | **Property:** *scales.barPointers.gradients*

 var scale = [{
 barPointers: [{ value: 40, gradients: {
 colorInfo:[{ colorStop : 0,
 color:"#FFFFFF"}] } }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'));| Not Applicable|

| Border of Bar Pointer | **Property:** *scales.barPointers.border*

 var scale = [{
 barPointers: [{ value: 40,
 border: { color: "red", width: 2 } }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | **Property:** *axes.pointers.border*

 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective> <PointersDirective>
 <PointerDirective type= 'Bar', value= {20}, border={ color: 'red',

```
width: 2.5 } /> </PointersDirective> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));
```

| Animation of Bar Pointer | **Property:** *enableAnimation*

 ReactDOM.render(
 <EJ.LinearGauge id= "gauge" enableAnimation= true> </EJ.LinearGauge>,
 document.getElementById('gauge'));

| **Property:** *axes.pointers.animationDuration*

 ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AxesDirective>
 <PointersDirective>
 <PointerDirective type= 'Bar',value= {20}, animationDuration= {2500}
 /> </PointersDirective> </AxesDirective> </LinearGaugeComponent>,
 document.getElementById('gauge'));

| Rounded Corner of Bar Pointer | Not Applicable | **Property:**
axes.pointers.roundedCornerRadius

 ReactDOM.render(<LinearGaugeComponent
 id='gauge'>
 <AxesDirective> <PointersDirective>
 <PointerDirective type= 'Bar',
 value= {20}, roundedCornerRadius= {15} /> </PointersDirective> </AxesDirective>
 </LinearGaugeComponent>, document.getElementById('gauge'));

Marker Pointer

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

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

| Marker Pointer | **Property:** *scales.markerPointers.value*

 var scale = [{

 markerPointers: [{ value: 20 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"
 scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge'));

| **Property:** *axes.pointers.value*

 ReactDOM.render(<LinearGaugeComponent id='gauge'>

 <AxesDirective> <PointersDirective>
 <PointerDirective type= 'Marker', value= {20} />
 </PointersDirective> </AxesDirective> </LinearGaugeComponent>,
 document.getElementById('gauge'));

| Color of Marker Pointer | **Property:** *scales.markerPointers.backgroundColor*

 var scale = [{

 markerPointers: [{
 value: 20, backgroundColor: "blue" }]
 }]

 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
 document.getElementById('gauge'));

| **Property:** *axes.pointers.color*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'>
 <AxesDirective> <PointersDirective>

 <PointerDirective type= 'Marker', value= {20}, color= 'red' /> </PointersDirective>
 </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge'));

| Offset of Marker Pointer | **Property:** *scales.markerPointers.distanceFromScale*

 var scale = [{

 markerPointers: [{
 value: 40, distanceFromScale: 10 }]
 }]

 ReactDOM.render(<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
 document.getElementById('gauge'));

| **Property:** *axes.pointers.offset*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'>
 <AxesDirective> <PointersDirective>

 <PointerDirective type= 'Marker', value= {20}, offset= {10} /> </PointersDirective>
 </AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge'));

| Opacity of Marker Pointer | **Property:** *scales.markerPointers.opacity*

 var scale = [{

 markerPointers: [{
 value: 40, opacity: 1 }]
 }]
 ReactDOM.render(
 <EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,

```
document.getElementById('gauge')); | Property: axes.pointers.opacity<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AxesDirective>
<PointersDirective> <br/> <PointerDirective type= 'Marker', value= {20}, opacity= {1} />
</PointersDirective> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge')); |
```

```
| Width of Marker Pointer | Property: scales.markerPointers.width<br/><br/> var scale = [{ <br/>
&#160; markerPointers: [{ <br/> &#160; value: 40, width: 20 }] <br/> }] <br/> ReactDOM.render(
<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
document.getElementById('gauge')); | Property: axes.pointers.width<br/><br/> ReactDOM.render(
<LinearGaugeComponent id='gauge'> <br/> <AxesDirective> <PointersDirective> <br/>
<PointerDirective type= 'Marker', value= {20}, width= {25} /> </PointersDirective>
</AxesDirective> </LinearGaugeComponent>, document.getElementById('gauge')); |
```

```
| Gradients of Marker Pointer | Property: scales.markerPointers.gradients<br/><br/> var scale = [{ <br/>
&#160; markerPointers: [{ value: 40, gradients: { <br/> &#160; colorInfo: [{ colorStop : 0, <br/>
&#160; color: "#FFFFFF" }] } } ] <br/> }] <br/> ReactDOM.render( <EJ.LinearGauge id= "gauge"
scales={scale}> </EJ.LinearGauge>, document.getElementById('gauge')); | Not Applicable |
```

```
| Border of Bar Pointer | Property: scales.markerPointers.border<br/><br/> var scale = [{ <br/> &#160;
markerPointers: [{ value: 40, <br/> &#160; border: { color: "red", width: 2 } } ] <br/> }] <br/>
ReactDOM.render( <EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
document.getElementById('gauge')); | Property: axes.pointers.border<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AxesDirective>
<PointersDirective> <br/> <PointerDirective type= 'Marker', value= {20}, border={ color: 'red',
width: 2.5 } /> </PointersDirective> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge')); |
```

```
| Animation of Marker Pointer | Property: enableMarkerPointerAnimation<br/><br/>
ReactDOM.render( <EJ.LinearGauge id= "gauge" enableMarkerPointerAnimation= true>
</EJ.LinearGauge>, document.getElementById('gauge')); | Property:
axes.pointers.animationDuration<br/><br/> ReactDOM.render( <LinearGaugeComponent
id='gauge'> <br/> <AxesDirective> <PointersDirective> <br/> <PointerDirective type=
'Marker', value= {20}, animationDuration= {2500} /> </PointersDirective> </AxesDirective>
</LinearGaugeComponent>, document.getElementById('gauge')); |
```

```
| Type of Marker Pointer | Property: scales.markerPointers.type<br/><br/> var scale = [{ <br/> &#160;
markerPointers: [{ value: 40, <br/> &#160; type: "Diamond" } ] <br/> }] <br/> ReactDOM.render(
<EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
document.getElementById('gauge')); | Property: axes.pointers.markerType<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AxesDirective>
<PointersDirective> <br/> <PointerDirective type= 'Marker', value= {20}, markerType=
'Diamond' /> </PointersDirective> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge')); |
```

```
| Placement of Marker Pointer | Property: scales.markerPointers.placement<br/><br/> var scale = [{
<br/> &#160; markerPointers: [{ value: 40, <br/> &#160; placement: "near" } ] <br/> }] <br/>
```

```
ReactDOM.render( <EJ.LinearGauge id= "gauge" scales={scale}> </EJ.LinearGauge>,
document.getElementById('gauge'));| Property: axes.pointers.placement<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AxesDirective>
<PointersDirective> <br/> <PointerDirective type= 'Marker', value= {20}, placement= "Center"
/> </PointersDirective> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));|
```

```
| Drag of Marker Pointer | Property: readOnly<br/><br/> ReactDOM.render( <EJ.LinearGauge id=
"gauge" readOnly= false> </EJ.LinearGauge>, document.getElementById('gauge'));| Property:
axes.pointers.enableDrag<br/><br/> ReactDOM.render( <LinearGaugeComponent id='gauge'>
<br/> <AxesDirective> <PointersDirective> <br/> <PointerDirective type= 'Marker', value= {20},
enableDrag= true /> </PointersDirective> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));|
```

```
| Image Marker Pointer | Not Applicable | Property: axes.pointers.imageUrl<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AxesDirective>
<PointersDirective> <br/> <PointerDirective type= 'Marker', value= {20}, imageUrl= "image.png"
/> </PointersDirective> </AxesDirective> </LinearGaugeComponent>,
document.getElementById('gauge'));|
```

Annotation

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

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

```
| Content | Property: scales.customLabels.value<br/><br/> var scale = [{ <br/> &#160; customLabels:
[{ showCustomLabels: true, <br/> &#160; value: 'LinearGauge' }] <br/> }] <br/> ReactDOM.render(
<EJ.LinearGauge id= "gauge" <br/> &#160; scales={scale}> </EJ.LinearGauge>, <br/>
document.getElementById('gauge'));| Property: annotations.content<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AnnotationsDirective>
<AnnotationDirective <br/> &#160; content= "Annotation"/> <br/> </AnnotationsDirective>
</LinearGaugeComponent>, <br/> document.getElementById('gauge'));|
```

```
| Horizontal Alignment | Not Applicable | Property: annotations.horizontalAlignment<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AnnotationsDirective>
<AnnotationDirective <br/> &#160; horizontalAlignment='Center'/> <br/>
</AnnotationsDirective> </LinearGaugeComponent>, <br/>
document.getElementById('gauge'));|
```

```
| Vertical Alignment | Not Applicable | Property: annotations.verticalAlignment<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AnnotationsDirective>
<AnnotationDirective <br/> &#160; verticalAlignment='Far'/> <br/> </AnnotationsDirective>
</LinearGaugeComponent>, <br/> document.getElementById('gauge'));|
```

```
| Position of X | Property: scales.customLabels.position.x<br/><br/> var scale = [{ <br/> &#160;
customLabels: [{ showCustomLabels: true, <br/> &#160; value: 'LinearGauge', position: {x: 20} }]
<br/> }] <br/> ReactDOM.render( <EJ.LinearGauge id= "gauge" <br/> &#160; scales={scale}>
</EJ.LinearGauge>, <br/> document.getElementById('gauge'));| Property: annotations.x<br/><br/>
ReactDOM.render( <LinearGaugeComponent id='gauge'> <br/> <AnnotationsDirective>
```

```
<AnnotationDirective <br/> &#160; x={35}/> <br/> </AnnotationsDirective>
</LinearGaugeComponent>, <br/> document.getElementById('gauge'));|
```

| Position of Y | **Property:** *scales.customLabels.position.y*

 var scale = [{

customLabels: [{ showCustomLabels: true,
 value: 'LinearGauge', position: {y: 30} }}

]]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"
 scales={scale}>
</EJ.LinearGauge>,
 document.getElementById('gauge'));| **Property:** *annotations.y*

ReactDOM.render(<LinearGaugeComponent id='gauge'>
 <AnnotationsDirective>
<AnnotationDirective
 y={40}/>
 </AnnotationsDirective>
</LinearGaugeComponent>,
 document.getElementById('gauge'));|

| Z Index | Not Applicable | **Property:** *annotations.zIndex*

 ReactDOM.render(
<LinearGaugeComponent id='gauge'>
 <AnnotationsDirective> <AnnotationDirective

 zIndex='1'/>
 </AnnotationsDirective> </LinearGaugeComponent>,

document.getElementById('gauge'));|

| Axis Index | Not Applicable | **Property:** *annotations.axisIndex*

 ReactDOM.render(
<LinearGaugeComponent id='gauge'>
 <AnnotationsDirective> <AnnotationDirective

 content= 'Annotation' axisIndex='0'/>
 </AnnotationsDirective>
</LinearGaugeComponent>,
 document.getElementById('gauge'));|

| Axis Value | Not Applicable | **Property:** *annotations.axisValue*

 ReactDOM.render(
<LinearGaugeComponent id='gauge'>
 <AnnotationsDirective> <AnnotationDirective

 content= 'Annotation' axisValue='35'/>
 </AnnotationsDirective>
</LinearGaugeComponent>,
 document.getElementById('gauge'));|

| Font customization | **Property:** *scales.customLabels.font*

 var scale = [{

customLabels: [{ showCustomLabels: true,
 value: 'LinearGauge', font: { size: '30px'}
}]
]]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"
 scales={scale}>
</EJ.LinearGauge>,
 document.getElementById('gauge'));| **Property:**
annotations.font

 ReactDOM.render(<LinearGaugeComponent id='gauge'>

<AnnotationsDirective> <AnnotationDirective
 content= 'Annotation' font= { size:
'15px' }>
 </AnnotationsDirective> </LinearGaugeComponent>,

document.getElementById('gauge'));|

| Annotation Color | **Property:** *scales.customLabels.color*

 var scale = [{

customLabels: [{ showCustomLabels: true,
 value: 'LinearGauge', font: { color: 'red'}
}]
]]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"
 scales={scale}>
</EJ.LinearGauge>,
 document.getElementById('gauge'));| **Property:**
annotations.font

 ReactDOM.render(<LinearGaugeComponent id='gauge'>

<AnnotationsDirective> <AnnotationDirective
 content= 'Annotation' font= { color:
'red' }>
 </AnnotationsDirective> </LinearGaugeComponent>,

document.getElementById('gauge'));|

| Opacity of Annotation | **Property:** *scales.customLabels.opacity*

 var scale = [{

customLabels: [{ showCustomLabels: true,
 value: 'LinearGauge', font: { opacity:
0.5} }}
]]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"

scales={scale}> </EJ.LinearGauge>,
 document.getElementById('gauge'));| **Property:**

```

annotations.font<br><br> ReactDOM.render( <LinearGaugeComponent id='gauge'> <br>
<AnnotationsDirective> <AnnotationDirective <br> &#160; content= 'Annotation' font= {
opacity: 0.7 }/> <br> </AnnotationsDirective> </LinearGaugeComponent>, <br>
document.getElementById('gauge')));|

```

|Position Type| **Property:** *scales.customLabels.positionType*

 var scale = [{

customLabels: [{ showCustomLabels: true,
 value: 'LinearGauge', positionType:
"outer" }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"

scales={scale}> </EJ.LinearGauge>,
 document.getElementById('gauge')));| Not applicable|

|TextAngle of Annotation| **Property:** *scales.customLabels.textAngle*

 var scale = [{

 customLabels: [{ showCustomLabels: true,
 value: 'LinearGauge', textAngle:
25 }]
 }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"
 scales={scale}>
</EJ.LinearGauge>,
 document.getElementById('gauge')));| Not applicable|

Tooltip

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

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

|Tooltip for Pointer| Not Applicable| **Property:** *tooltip.enable*

 ReactDOM.render(
<LinearGaugeComponent id='gauge'
 tooltip= { enable: true } />

document.getElementById('gauge')));|

|Tooltip for Label| **Property:** *tooltip.showLabelTooltip*

 var tooltip = [{

showLabelTooltip: true }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"

tooltip={tooltip}> </EJ.LinearGauge>,
 document.getElementById('gauge')));| Not
Applicable|

|Tooltip Format| Not Applicable| **Property:** *tooltip.format*

 ReactDOM.render(
<LinearGaugeComponent id='gauge'
 tooltip= { enable: true, format: '{\$value}' } />

 document.getElementById('gauge')));|

|Tooltip Color| Not Applicable| **Property:** *tooltip.fill*

 ReactDOM.render(
<LinearGaugeComponent id='gauge'
 tooltip= { enable: true, fill: 'gray' } />

document.getElementById('gauge')));|

|Tooltip Template| **Property:** *tooltip.templateID*

 var tooltip = [{
 templateID:
true }]
 ReactDOM.render(<EJ.LinearGauge id= "gauge"
 tooltip={tooltip}>
</EJ.LinearGauge>,
 document.getElementById('gauge')));| **Property:**
tooltip.template

 ReactDOM.render(<LinearGaugeComponent id='gauge'

tooltip= { enable: true, template: 'Template' } />
 document.getElementById('gauge')));|

|Tooltip Animation| Not Applicable| **Property:** *tooltip.enableAnimation*

 ReactDOM.render(
<LinearGaugeComponent id='gauge'
 tooltip= { enableAnimation: true } />

document.getElementById('gauge')));|

|Tooltip Border| Not Applicable| **Property:** *tooltip.border*

 ReactDOM.render(
<LinearGaugeComponent id='gauge'
 tooltip= { enable: true,

border: { width: 2, color: 'red' } } />
 document.getElementById('gauge')));|

| Tooltip TextStyle| Not Applicable| **Property:** *tooltip.textStyle*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 tooltip= { enable: true,

 textStyle: { size: '10px', color: 'white' } } />
 document.getElementById('gauge'));|

Appearance of Linear Gauge

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

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

| Background Color| **Property:** *backgroundColor*

 ReactDOM.render(<EJ.LinearGauge

 id= "gauge" backgroundColor= 'red'>
</EJ.LinearGauge>,
 document.getElementById('gauge'));| **Property:** *background*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 background=
 'blue' />
 document.getElementById('gauge'));|

| Border Color| **Property:** *borderColor*

 ReactDOM.render(<EJ.LinearGauge

 id= "gauge" borderColor= 'Black'>
</EJ.LinearGauge>,
 document.getElementById('gauge'));| **Property:** *border.color*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 border= { color: 'red' }
 />
 document.getElementById('gauge'));|

| Margin| Not Applicable| **Property:** *margin*

 ReactDOM.render(<LinearGaugeComponent
 id='gauge'
 margin= { left: 40, right: 40, top: 40, bottom: 40
 } />
 document.getElementById('gauge'));|

| Orientation| **Property:** *orientation*

 ReactDOM.render(<EJ.LinearGauge
 id=
 "gauge" orientation= 'Vertical'>
 </EJ.LinearGauge>,
 document.getElementById('gauge'));| **Property:** *orientation*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 orientation= 'Horizontal' />

 document.getElementById('gauge'));|

| Locale| **Property:** *locale*

 ReactDOM.render(<EJ.LinearGauge
 id= "gauge"
 locale= 'en-US'>
 </EJ.LinearGauge> , document.getElementById('gauge'));| **Property:**
locale

 ReactDOM.render(<LinearGaugeComponent id='gauge'
 locale=
 'en-US' />
 document.getElementById('gauge'));|

| Theme| **Property:** *theme*

 ReactDOM.render(<EJ.LinearGauge
 id= "gauge"
 theme= 'Highcontrast'>
 </EJ.LinearGauge> , document.getElementById('gauge'));|
Property: *theme*

 ReactDOM.render(<LinearGaugeComponent id='gauge'

 theme= 'Highcontrast' />
 document.getElementById('gauge'));|

| Gauge Title| Not Applicable| **Property:** *title*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 title= 'Linear Gauge' />

 document.getElementById('gauge'));|

Gauge Container type

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

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

| Container Type | **Property:** *scales.type*

 var scale = [{
 type: 'therometer' }]

 ReactDOM.render(<EJ.LinearGauge id= "gauge"
 scales={scale}>
 </EJ.LinearGauge>,
 document.getElementById('gauge')); | **Property:**
container.type

 ReactDOM.render(<LinearGaugeComponent id='gauge'

 container= { type: 'Thermometer' } />
 document.getElementById('gauge')); |

| Container Height | Not Applicable | **Property:** *container.height*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 container= { height: 20 } />

 document.getElementById('gauge')); |

| Container Width | Not Applicable | **Property:** *container.width*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 container= { width: 10 } />

 document.getElementById('gauge')); |

| Container Offset | Not Applicable | **Property:** *container.offset*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 container= { offset: 5 } />

 document.getElementById('gauge')); |

Events

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

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

| Animation Complete Event | Not Applicable | **Event:** *animationComplete*

 ReactDOM.render(<LinearGaugeComponent id='gauge'

 animationComplete={this.animationComplete} />
 document.getElementById('gauge'));

 public animationComplete(args): void { } |

| Annotation Render Event | **Event:** *drawCustomLabel*

 ReactDOM.render(<EJ.LinearGauge
 id= "gauge"
 drawCustomLabel={this.drawCustomLabel}> </EJ.LinearGauge>,

 document.getElementById('gauge'));
 public drawCustomLabel(args): void { } | **Event:**
annotationRender

 ReactDOM.render(<LinearGaugeComponent id='gauge'

 annotationRender={this.annotationRender} />
 document.getElementById('gauge'));

 public annotationRender(args): void { } |

| AxisLabel Render Event | **Event:** *drawLabels*

 ReactDOM.render(<EJ.LinearGauge id=
 "gauge"
 drawLabels={this.drawLabels}> </EJ.LinearGauge>,

 document.getElementById('gauge'));
 public drawLabels(args): void { } | **Event:**
axisLabelRender

 ReactDOM.render(<LinearGaugeComponent id='gauge'

 axisLabelRender={this.axisLabelRender} />
 document.getElementById('gauge'));

 public axisLabelRender(args): void { } |

| Load Event | **Event:** *load*

 ReactDOM.render(<EJ.LinearGauge id= "gauge"

 load={this.load}> </EJ.LinearGauge>,
 document.getElementById('gauge'));
 public
 load(args): void { } | **Event:** *load*

 ReactDOM.render(<LinearGaugeComponent
 id='gauge'
 load={this.load} />
 document.getElementById('gauge'));

 public load(args): void { } |

|Loaded Event| Not Applicable| **Event:** *loaded*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 loaded={this.loaded}/>

 document.getElementById('gauge'));
 public loaded(args): void { }|

|Resize Event| Not Applicable| **Event:** *resized*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 resized={this.resized}/>

 document.getElementById('gauge'));
 public resized(args): void { }|

|Tooltip Render Event| Not Applicable| **Event:** *tooltipRender*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 tooltipRender={this.tooltipRender}/>

 document.getElementById('gauge'));
 public tooltipRender(args): void { }|

|Value Change Event| Not Applicable| **Event:** *valueChange*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'
 valueChange={this.valueChange}/>

 document.getElementById('gauge'));
 public valueChange(args): void { }|

|Mouse Move Event| **Event:** *mouseClickMove*

 ReactDOM.render(<EJ.LinearGauge id=
 "gauge"
 mouseClickMove={this.mouseClickMove}> </EJ.LinearGauge>,

 document.getElementById('gauge'));
 public mouseClickMove(args): void { }| **Event:**
gaugeMouseMove

 ReactDOM.render(<LinearGaugeComponent id='gauge'

 gaugeMouseMove={this.gaugeMouseMove}/>

 document.getElementById('gauge'));
 public gaugeMouseMove(args): void { }|

|Mouse Up Event| **Event:** *mouseClickUp*

 ReactDOM.render(<EJ.LinearGauge id= "gauge"

 mouseClickUp={this.mouseClickUp}> </EJ.LinearGauge>,

 document.getElementById('gauge'));
 public mouseClickUp(args): void { }| **Event:**
gaugeMouseUp

 ReactDOM.render(<LinearGaugeComponent id='gauge'

 gaugeMouseUp={this.gaugeMouseUp}/>
 document.getElementById('gauge'));
 public
 gaugeMouseUp(args): void { }|

|Mouse Down Event| Not Applicable| **Event:** *gaugeMouseDown*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'

 gaugeMouseDown={this.gaugeMouseDown}/>
 document.getElementById('gauge'));

 public gaugeMouseDown(args): void { }|

|Mouse Leave Event| Not Applicable| **Event:** *gaugeMouseLeave*

 ReactDOM.render(
 <LinearGaugeComponent id='gauge'

 gaugeMouseLeave={this.gaugeMouseLeave}/>
 document.getElementById('gauge'));

 public gaugeMouseLeave(args): void { }|

|Mouse Click Event| **Event:** *mouseClick*

 ReactDOM.render(<EJ.LinearGauge id= "gauge"

 mouseClick={this.mouseClick}> </EJ.LinearGauge>,

 document.getElementById('gauge'));
 public mouseClick(args): void { }| Not Applicable|

|Render Complete Event| **Event:** *renderComplete*

 ReactDOM.render(<EJ.LinearGauge id=
 "gauge"
 renderComplete={this.renderComplete}> </EJ.LinearGauge>,

 document.getElementById('gauge'));
 public renderComplete(args): void { }| Not Applicable|

```
| Double Click Event| Event: doubleClick<br/><br/> ReactDOM.render( <EJ.LinearGauge id= "gauge"
<br/> &#160; doubleClick={this.doubleClick}> </EJ.LinearGauge>, <br/>
document.getElementById('gauge'));<br/> public doubleClick(args): void { }| Not Applicable|

| Right Click Event| Event: rightClick<br/><br/> ReactDOM.render( <EJ.LinearGauge id= "gauge"
<br/> &#160; rightClick={this.rightClick}> </EJ.LinearGauge>, <br/>
document.getElementById('gauge'));<br/> public rightClick(args): void { }| Not Applicable|

| BarPointers Event| Event: drawBarPointers<br/><br/> ReactDOM.render( <EJ.LinearGauge id=
"gauge" <br/> &#160; drawBarPointers={this.drawBarPointers}> </EJ.LinearGauge>, <br/>
document.getElementById('gauge'));<br/> public drawBarPointers(args): void { }| Not Applicable|

| Indicators Event| Event: drawIndicators<br/><br/> ReactDOM.render( <EJ.LinearGauge id=
"gauge" <br/> &#160; drawIndicators={this.drawIndicators}> </EJ.LinearGauge>, <br/>
document.getElementById('gauge'));<br/> public drawIndicators(args): void { }| Not Applicable|

| MarkerPointer Event| Event: drawMarkerPointers<br/><br/> ReactDOM.render( <EJ.LinearGauge
id= "gauge" <br/> &#160; drawMarkerPointers={this.drawMarkerPointers}> </EJ.LinearGauge>,
<br/> document.getElementById('gauge'));<br/> public drawMarkerPointers(args): void { }| Not
Applicable|

| Ranges Event| Event: drawRange<br/><br/> ReactDOM.render( <EJ.LinearGauge id= "gauge"
<br/> &#160; drawRange={this.drawRange}> </EJ.LinearGauge>, <br/>
document.getElementById('gauge'));<br/> public drawRange(args): void { }| Not Applicable|

| Gauge Initialized Event| Event: init<br/><br/> ReactDOM.render( <EJ.LinearGauge id= "gauge"
<br/> &#160; init={this.init}> </EJ.LinearGauge>, <br/> document.getElementById('gauge'));<br/>
public init(args): void { }| Not Applicable|

{% endraw %}
```

ListBox

Getting Started

This section briefly explains how to create a simple **ListBox** component and configure its available functionalities in React.

Dependencies

The following list of dependencies are required to use the **ListBox** component in your application.

```
`javascript
|-- @syncfusion/ej2-react-dropdowns
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
```

```
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
\
```

Installation and configuration

You can use [Create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

```
`bash
npm install -g create-react-app
\
```

To set-up a React application in TypeScript environment, run the following command.

```
`bash
npx create-react-app my-app --template typescript
cd my-app
npm start
\
```

To set-up a React application in JavaScript environment, run the following command.

```
`bash
npx create-react-app my-app
cd my-app
npm start
\
```

Adding syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. You can choose the component that you want to install.

To install ListBox component, use the following command

```
`bash
npm install @syncfusion/ej2-react-dropdowns --save
\
```

Adding CSS reference

Import the ListBox component required CSS references as follows in `src/App.css`.

```
`css
/ import the ListBox dependency styles /
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

```
@import "../node_modules/@syncfusion/ej2-react-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-dropdowns/styles/material.css";
,
```

Adding ListBox component

Now, you can start adding ListBox component in the application. For getting started, add the ListBox component in `src/App.tsx` file using following code.

Add the below code in the `src/App.tsx` to initialize the ListBox.

```
`ts
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import './App.css';
function App() {
  return (
    // specifies the tag for render the ListBox component
    <ListBoxComponent id='listbox'></ListBoxComponent>
  );
}
export default App;
,
```

Binding data source

After initialization, populate the ListBox with data using the [dataSource](#) property. Here, an array of object is passed to the ListBox component.

```
`ts
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import './App.css';
function App() {
  // define the array of object
  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' },
];
return (
// specifies the tag for render the ListBox component
<ListBoxComponent dataSource={data} />
);
}
export default App;
`
```

Run the application

After completing the configuration required to render a basic ListBox, run the following command to display the output in your default browser.

```
`
npm start
`
```

The following example shows a basic Listbox component.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  // define the array of object
  let data = [
    { 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' },
  ];
  return (
```

```
// specifies the tag for render the ListBox component
<ListBoxComponent dataSource={data}/>;
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  // define the array of object
  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' },
  ];
  return (
    // specifies the tag for render the ListBox component
    <ListBoxComponent dataSource={data}/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Accessibility in React ListBox component

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

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the ListBox component wrapper element as **listbox**, the **UL** element as **presentation**, and its list item as **option**. |

| **aria-label** | Provides an accessible name for the ListBox component. |

| **aria-multiselectable** | Applied to the element with the ListBox role, tells assistive technologies that the list supports multiple selection. The default value is true. |

| **aria-selected** | Applied to elements with role option that are visually styled as selected to inform assistive technologies that the options are selected. |

Keyboard interaction

The ListBox 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 ListBox component.

Press	To do this
---	---
Up arrow	Moves focus to the previous option.
Down arrow	Moves focus to the next option.
Home	Moves focus to first option.
End	Moves focus to last option.
Space	Changes the selection state of the focused option.
Ctrl + A	Selects all options in the list.
Ctrl + Shift + Home	Selects the focused option and all options up to the first option.
Ctrl + Shift + End	Selects the focused option and all options down to the last option.
Ctrl + (Up or Down)	Press Ctrl key with up / down arrow or mouse to select multiple items.

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Data binding in React List box component

The ListBox loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports

the data type of `array` or [Link to the Video](#).

To get start quickly with Data Binding, you can check on this video:

Fields	Type	Description
-----	-----	-----
text	<code>string</code>	Specifies the display text of each list item.
value	<code>string</code>	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	<code>string</code>	Specifies the category under which the list item has to be grouped.
iconCss	<code>string</code>	Specifies the iconCss class that needs to be mapped.
htmlAttributes	<code>string</code>	Allows additional attributes to configure the elements in various ways to meet the criteria.

When binding complex data to the ListBox, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Local Data

Local data can be represented by the following ways as described below.

Array of string

The ListBox has support to load array of primitive data such as strings or numbers. Here, both value and text field acts as same.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let data = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis',
    'Basket Ball', 'Base Ball', 'Hockey', 'Volley Ball'];
    return (<ListBoxComponent dataSource={data}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let data: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
    'Tennis', 'Basket Ball', 'Base Ball', 'Hockey', 'Volley Ball'];
    return (
        <ListBoxComponent dataSource={data} />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Array of object

The ListBox can generate its list items through an array of object data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `id` and `sports` column from complex data have been mapped to the `value` field and `text` field, respectively.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    // define the array of object
    let data = [
        { 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' },
    ];
    return (<ListBoxComponent dataSource={data}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    // define the array of object
    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' },
    ];
    return (
        <ListBoxComponent dataSource={data} />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Array of complex object

The ListBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `sports.Name` column from complex data have been mapped to the `text` field.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    // define the array of object
    let data = [
        { id: 'game1', sports: { Name: 'Badminton' } },
        { id: 'game2', sports: { Name: 'Cricket' } },
        { id: 'game3', sports: { Name: 'Football' } },
        { id: 'game4', sports: { Name: 'Golf' } },
        { id: 'game5', sports: { Name: 'Tennis' } },
        { id: 'game6', sports: { Name: 'Basket Ball' } },
    ];

```

```

        { id: 'game7', sports: { Name: 'Base Ball' } },
        { id: 'game8', sports: { Name: 'Hockey' } },
        { id: 'game9', sports: { Name: 'Volley Ball' } }
    ];
    let fields = { text: "sports.Name", value: "id" };
    return (<ListBoxComponent dataSource={data} fields={fields}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    // define the array of object
    let data: { [key: string]: Object }[] = [
        { id: 'game1', sports: { Name: 'Badminton' } },
        { id: 'game2', sports: { Name: 'Cricket' } },
        { id: 'game3', sports: { Name: 'Football' } },
        { id: 'game4', sports: { Name: 'Golf' } },
        { id: 'game5', sports: { Name: 'Tennis' } },
        { id: 'game6', sports: { Name: 'Basket Ball' } },
        { id: 'game7', sports: { Name: 'Base Ball' } },
        { id: 'game8', sports: { Name: 'Hockey' } },
        { id: 'game9', sports: { Name: 'Volley Ball' } }
    ];
    let fields: object = { text:"sports.Name", value:"id"};
    return (
        <ListBoxComponent dataSource={data} fields={fields} />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Remote Data

The ListBox supports retrieval of data from remote data services with the help of [DataManager](#) component.

The following sample displays the employee names from **Employee** table.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
function App() {
    let fields = { text: 'FirstName', value: 'EmployeeID' };
    let data = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://ej2services.syncfusion.com/production/web-services/api/Employees'
    });
}

```

```

    return (
      // specifies the tag for render the ListBox component
      <ListBoxComponent dataSource={data} fields={fields}/>;
    )
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
function App() {

  let fields: object = { text: 'FirstName', value: 'EmployeeID' };
  let data: DataManager = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Employees'
  });
  return (
    // specifies the tag for render the ListBox component
    <ListBoxComponent dataSource={data} fields={fields} />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Drag and drop in React List box component

The ListBox has support to drag an item or a group of selected items and drop it within the same list box or into another list box.

The elements can be customized on drag and drop by using the following events,

Events	Description
--------	-------------

-----	-----
-------	-------

dragStart	Triggers when the selected element is being dragged.
---------------------------	--

drag	Triggers when the selected element is being dragged.
----------------------	--

drop	Triggers when the selected element is being dropped.
----------------------	--

Single listbox

To drag and drop an item or group of item within the list box can be achieved by setting

[allowDragAndDrop](#) property as `true`.

The following sample illustrates how to drag and drop an item within the same list box by enabling `allowDragAndDrop` property.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';

```

```
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data = [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },
    { "Name": "Finland", "Code": "FI" },
    { "Name": "Germany", "Code": "DE" },
    { "Name": "Hong Kong", "Code": "HK" }
  ];
  let fields = { text: "Name" };
  return (<ListBoxComponent dataSource={data} allowDragAndDrop="true"
fields={fields}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data: { [key: string]: Object }[] = [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },
    { "Name": "Finland", "Code": "FI" },
    { "Name": "Germany", "Code": "DE" },
    { "Name": "Hong Kong", "Code": "HK" }
  ];
  let fields:object = { text:"Name" }
  return (
    <ListBoxComponent dataSource={data} allowDragAndDrop="true"
fields={fields} />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Multiple listbox

To drag and drop an item or group of item between two list boxes can be achieved by setting `allowDragAndDrop` property as `true` and `scope` property should be set to both the list boxes.

In the following sample, the `allowDragAndDrop` property is set as `true` and `scope` is set as `combined-list` to enable drop and drop in both list boxes.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let groupA = [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },
    { "Name": "Finland", "Code": "FI" },
    { "Name": "Germany", "Code": "DE" },
    { "Name": "Hong Kong", "Code": "HK" }
  ];
  let groupB = [
    { "Name": "India", "Code": "IN" },
    { "Name": "Italy", "Code": "IT" },
    { "Name": "Japan", "Code": "JP" },
    { "Name": "Mexico", "Code": "MX" },
    { "Name": "Norway", "Code": "NO" },
    { "Name": "Poland", "Code": "PL" },
    { "Name": "Switzerland", "Code": "CH" },
    { "Name": "United Kingdom", "Code": "GB" },
    { "Name": "United States", "Code": "US" }
  ];
  let fields = { text: "Name" };
  return (
    <div className="wrapper">
      <div className="listbox1">
        <h4>Group A</h4>
        <ListBoxComponent dataSource={groupA} allowDragAndDrop="true"
height="330px" scope="combined-list" fields={fields}/></div>
        <div className="listbox2">
          <h4>Group B</h4>
          <ListBoxComponent dataSource={groupB} allowDragAndDrop="true"
height="330px" scope="combined-list" fields={fields}/></div>
        </div>);
    }
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let groupA: { [key: string]: Object }[] = [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },
    { "Name": "Finland", "Code": "FI" },
    { "Name": "Germany", "Code": "DE" },
  ];
```

```

    { "Name": "Hong Kong", "Code": "HK" }
  ];
  let groupB: { [key: string]: Object }[] = [
    { "Name": "India", "Code": "IN" },
    { "Name": "Italy", "Code": "IT" },
    { "Name": "Japan", "Code": "JP" },
    { "Name": "Mexico", "Code": "MX" },
    { "Name": "Norway", "Code": "NO" },
    { "Name": "Poland", "Code": "PL" },
    { "Name": "Switzerland", "Code": "CH" },
    { "Name": "United Kingdom", "Code": "GB" },
    { "Name": "United States", "Code": "US" }
  ];
  let fields:object = { text:"Name"};
  return (
    <div className="wrapper">
      <div className="listbox1">
        <h4>Group A</h4>
        <ListBoxComponent dataSource={groupA} allowDragAndDrop="true"
height="330px" scope="combined-list" fields={fields}/></div>
        <div className="listbox2">
          <h4>Group B</h4>
          <ListBoxComponent dataSource={groupB} allowDragAndDrop="true"
height="330px" scope="combined-list" fields={fields} /></div>
        </div>
      );
    }
  export default App;
  ReactDOM.render(<App />, document.getElementById('sample'));

```

Dual list box in React List box component

The dual list box allows the user to move items between two list boxes by clicking the toolbar buttons. Dual list box can be created by listing items in the

[toolbarSettings](#) along with the `scope` property.

The following operations can be performed in dual list box,

Options	Description
----- -----	
moveUp	Move the selected item in the upward direction within the list box.
moveDown	Move the selected item in the downward direction within the list box.
moveTo	Move the selected item to the another list box.
moveFrom	Move the selected item from one list box to the another list box.
moveAllTo	Move all the items to the another list box.
moveAllFrom	Move all the items from one list box to the another list box.

The following example illustrates how to move items from **Group A** to **Group B** list box.

INDEX.JSX

```
import * as React from 'react';
```

```
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let groupA = [
        { "Name": "Australia", "Code": "AU" },
        { "Name": "Bermuda", "Code": "BM" },
        { "Name": "Canada", "Code": "CA" },
        { "Name": "Cameroon", "Code": "CM" },
        { "Name": "Denmark", "Code": "DK" },
        { "Name": "France", "Code": "FR" },
        { "Name": "Finland", "Code": "FI" },
        { "Name": "Germany", "Code": "DE" },
        { "Name": "Hong Kong", "Code": "HK" }
    ];
    let groupB = [
        { "Name": "India", "Code": "IN" },
        { "Name": "Italy", "Code": "IT" },
        { "Name": "Japan", "Code": "JP" },
        { "Name": "Mexico", "Code": "MX" },
        { "Name": "Norway", "Code": "NO" },
        { "Name": "Poland", "Code": "PL" },
        { "Name": "Switzerland", "Code": "CH" },
        { "Name": "United Kingdom", "Code": "GB" },
        { "Name": "United States", "Code": "US" }
    ];
    let fields = { text: "Name" };
    let toolbar = { items: ["moveUp", "moveDown", "moveTo", "moveFrom",
        "moveAllTo", "moveAllFrom"] };
    return (<div className="wrapper">
        <div className="listbox1">
            <ListBoxComponent dataSource={groupA} fields={fields} scope="#listbox"
                toolbarSettings={toolbar}/></div>
        <div className="listbox2">
            <ListBoxComponent id="listbox" dataSource={groupB} fields={fields}/>
        </div>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let groupA: { [key: string]: Object }[] = [
        { "Name": "Australia", "Code": "AU" },
        { "Name": "Bermuda", "Code": "BM" },
        { "Name": "Canada", "Code": "CA" },
        { "Name": "Cameroon", "Code": "CM" },
        { "Name": "Denmark", "Code": "DK" },
        { "Name": "France", "Code": "FR" },
        { "Name": "Finland", "Code": "FI" },
        { "Name": "Germany", "Code": "DE" },
        { "Name": "Hong Kong", "Code": "HK" }
    ]
```



```

];
let groupB: { [key: string]: Object }[] = [
  { "Name": "India", "Code": "IN" },
  { "Name": "Italy", "Code": "IT" },
  { "Name": "Japan", "Code": "JP" },
  { "Name": "Mexico", "Code": "MX" },
  { "Name": "Norway", "Code": "NO" },
  { "Name": "Poland", "Code": "PL" },
  { "Name": "Switzerland", "Code": "CH" },
  { "Name": "United Kingdom", "Code": "GB" },
  { "Name": "United States", "Code": "US" }
];
let fields:object = { text:"Name" }
let toolbar:object = { items:["moveUp", "moveDown", "moveTo", "moveFrom",
"moveAllTo", "moveAllFrom"]}
return (
  <div className="wrapper">
    <div className="listbox1">
      <ListBoxComponent dataSource={groupA} fields={fields} scope="#listbox"
toolbarSettings={toolbar}/></div>
    <div className="listbox2">
      <ListBoxComponent id="listbox" dataSource={groupB} fields={fields} />
    </div>
  </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Icons and templates in React List box component

Icons

To place the icon on a list box, set the [iconCss](#) property to **e-icons** with the required icon CSS. By default, the icon is positioned to the left side of the list.

In the following sample, icon classes are mapped with **iconCss** field.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data = [
    { text: 'Account Settings', iconCss: 'e-list-icons e-list-user-settings' },
    { text: 'Address Book', iconCss: 'e-list-icons e-list-address-book' },
    { text: 'Delete', iconCss: 'e-list-icons e-list-delete' },
    { text: 'Forward', iconCss: 'e-list-icons e-list-forward' },
    { text: 'Reply', iconCss: 'e-list-icons e-list-reply' },
    { text: 'Reply All', iconCss: 'e-list-icons e-list-reply-all' },
    { text: 'Save All Attachments', iconCss: 'e-list-icons e-list-save-all-attachments' },
    { text: 'Save As', iconCss: 'e-list-icons e-list-icon-save-as' },
    { text: 'Touch/Mouse Mode', iconCss: 'e-list-icons e-list-touch' },
  ];

```

```

    { text: 'Undo', iconCss: ' e-list-icons e-list-undo' }
  ];
  let fields = { text: "text", iconCss: "iconCss" };
  return (<ListBoxComponent dataSource={data} fields={fields}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data: { [key: string]: Object }[] = [
    { text: 'Account Settings', iconCss: 'e-list-icons e-list-user-settings' },
    { text: 'Address Book', iconCss: 'e-list-icons e-list-address-book' },
    { text: 'Delete', iconCss: 'e-list-icons e-list-delete' },
    { text: 'Forward', iconCss: 'e-list-icons e-list-forward' },
    { text: 'Reply', iconCss: 'e-list-icons e-list-reply' },
    { text: 'Reply All', iconCss: 'e-list-icons e-list-reply-all' },
    { text: 'Save All Attachments', iconCss: 'e-list-icons e-list-save-all-attachments' },
    { text: 'Save As', iconCss: 'e-list-icons e-list-icon-save-as' },
    { text: 'Touch/Mouse Mode', iconCss: 'e-list-icons e-list-touch' },
    { text: 'Undo', iconCss: ' e-list-icons e-list-undo' }
  ];
  let fields:object = { text:"text", iconCss:"iconCss"}
  return (
    <ListBoxComponent dataSource={data} fields={fields}/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Templates

ListBox items can be customized according to the requirement using [itemTemplate](#) property.

In the following sample, the items in the cart are displayed using list box template,

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data = [
    { text: 'JavaScript', pic: 'javascript', description: 'It is a lightweight interpreted or JIT-compiled programming language.' },
    { text: 'TypeScript', pic: 'typescript', description: 'It is a typed superset of Javascript that compiles to plain JavaScript.' },
    { text: 'Angular', pic: 'angular', description: 'It is a TypeScript-based open-source web application framework.' },
  ];

```

```

    { text: 'React', pic: 'react', description: 'A JavaScript library
for building user interfaces. It can also render on the server using Node.'
},
    { text: 'Vue', pic: 'vue', description: 'A progressive framework for
building user interfaces. it is incrementally adoptable.' }
];
return (<div>
  <ListBoxComponent dataSource={data} itemTemplate='<div class="list-
wrapper"><span class="{pic} e-avatar e-avatar-xlarge e-avatar-
circle"></span><span class="text">{text}</span><span
class="description">{description}</span></div>' />
  </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data: { [key: string]: Object }[] = [
    { text: 'JavaScript', pic: 'javascript', description: 'It is a
lightweight interpreted or JIT-compiled programming language.' },
    { text: 'TypeScript', pic: 'typescript', description: 'It is a typed
superset of Javascript that compiles to plain JavaScript.' },
    { text: 'Angular', pic: 'angular', description: 'It is a TypeScript-
based open-source web application framework.' },
    { text: 'React', pic: 'react', description: 'A JavaScript library for
building user interfaces. It can also render on the server using Node.' },
    { text: 'Vue', pic: 'vue', description: 'A progressive framework for
building user interfaces. it is incrementally adoptable.' }
  ];
  return (
    <div>
      <ListBoxComponent dataSource={data} itemTemplate='<div class="list-
wrapper"><span class="{pic} e-avatar e-avatar-xlarge e-avatar-
circle"></span><span class="text">{text}</span><span
class="description">{description}</span></div>' />
      </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Selection in React List box component

The ListBox provides support to select an item or a group of item by mouse or keyboard action. There are two selection modes available in list box,

- Single - To select single item in the list box.
- Multiple - To select multiple items in the list box.

On selection of each list box item, [change](#) event is triggered.

Single selection

To enable single selection in the list box, [mode](#) should be set as `single` in [selectionSettings](#) property.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let data = [
        { 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 selection = { mode: "single" };
    return (<ListBoxComponent dataSource={data}
    selectionSettings={selection}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    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 selection:object = { mode:"single" }
    return (
        <ListBoxComponent dataSource={data} selectionSettings={selection}/>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Multiple selection

To enable multiple selection in the list box, **mode** should be set as **Multiple** in **selectionSettings** property.

To select multiple items, use the SHIFT, CTRL, and arrow keys to make selections.

By default, the selection mode is set as **Multiple**.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let data = [
        { 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 selection = { mode: "multiple" };
    return (<ListBoxComponent dataSource={data}
selectionSettings={selection}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    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 selection:object = { mode:"multiple" }
    return (
        <ListBoxComponent dataSource={data} selectionSettings={selection}/>
    );
}
```

```
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Sorting and grouping in React List box component

Sorting

The ListBox supports sorting of available items in the alphabetical order that can be either ascending or descending. This can be achieved using

[sortOrder](#) property. Sort order can be **None**, **Ascending** or **Descending**.

In the following example, the **SortOrder** is set as **Ascending**.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let data = [
        { "Name": "Australia", "Code": "AU" },
        { "Name": "Bermuda", "Code": "BM" },
        { "Name": "Canada", "Code": "CA" },
        { "Name": "Cameroon", "Code": "CM" },
        { "Name": "Denmark", "Code": "DK" },
        { "Name": "France", "Code": "FR" },
        { "Name": "Finland", "Code": "FI" },
        { "Name": "Germany", "Code": "DE" },
        { "Name": "Hong Kong", "Code": "HK" }
    ];
    let fields = { text: "Name" };
    return (<ListBoxComponent dataSource={data} sortOrder="Ascending"
fields={fields}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let data: { [key: string]: Object }[] = [
        { "Name": "Australia", "Code": "AU" },
        { "Name": "Bermuda", "Code": "BM" },
        { "Name": "Canada", "Code": "CA" },
        { "Name": "Cameroon", "Code": "CM" },
        { "Name": "Denmark", "Code": "DK" },
        { "Name": "France", "Code": "FR" },
        { "Name": "Finland", "Code": "FI" },
        { "Name": "Germany", "Code": "DE" },
        { "Name": "Hong Kong", "Code": "HK" }
    ];
    let fields:object = { text:"Name" };
    return (
```

```

    <ListBoxComponent dataSource={data} sortOrder="Ascending"
    fields={fields}/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Grouping

The ListBox supports to wrap the nested element into a group based on its category. The category of each list item can be mapped with

[groupBy](#) field in the data table.

In the following example, vegetables are grouped based on its category.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data = [
    { "Vegetable": "Cabbage", "Category": "Leafy and Salad", "Id": "item1" },
    { "Vegetable": "Spinach", "Category": "Leafy and Salad", "Id": "item2" },
    { "Vegetable": "Wheat grass", "Category": "Leafy and Salad", "Id": "item3" },
    { "Vegetable": "Yarrow", "Category": "Leafy and Salad", "Id": "item4" },
    { "Vegetable": "Pumpkins", "Category": "Leafy and Salad", "Id": "item5" },
    { "Vegetable": "Chickpea", "Category": "Beans", "Id": "item6" },
    { "Vegetable": "Green bean", "Category": "Beans", "Id": "item7" },
    { "Vegetable": "Horse gram", "Category": "Beans", "Id": "item8" },
    { "Vegetable": "Garlic", "Category": "Bulb and Stem", "Id": "item9" },
    { "Vegetable": "Nopal", "Category": "Bulb and Stem", "Id": "item10" },
    { "Vegetable": "Onion", "Category": "Bulb and Stem", "Id": "item11" }
  ];
  let fields = { text: "Vegetable", groupBy: "Category", value: "Id" };
  return (<ListBoxComponent dataSource={data} fields={fields}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data: { [key: string]: Object }[] = [

```

```

    { "Vegetable": "Cabbage", "Category": "Leafy and Salad", "Id": "item1"
  },
  { "Vegetable": "Spinach", "Category": "Leafy and Salad", "Id": "item2"
  },
  { "Vegetable": "Wheat grass", "Category": "Leafy and Salad", "Id":
"item3" },
  { "Vegetable": "Yarrow", "Category": "Leafy and Salad", "Id": "item4" },
  { "Vegetable": "Pumpkins", "Category": "Leafy and Salad", "Id": "item5"
  },
  { "Vegetable": "Chickpea", "Category": "Beans", "Id": "item6" },
  { "Vegetable": "Green bean", "Category": "Beans", "Id": "item7" },
  { "Vegetable": "Horse gram", "Category": "Beans", "Id": "item8" },
  { "Vegetable": "Garlic", "Category": "Bulb and Stem", "Id": "item9" },
  { "Vegetable": "Nopal", "Category": "Bulb and Stem", "Id": "item10" },
  { "Vegetable": "Onion", "Category": "Bulb and Stem", "Id": "item11" }
];
let fields:object = { text:"Vegetable", groupBy:"Category", value:"Id" };
return (
  <ListBoxComponent dataSource={data} fields={fields}/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Style and appearance in React List box component

To modify the ListBox appearance, you need to override the default CSS of ListBox component. Please find the list of CSS classes and its corresponding section in ListBox component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

- |.e-listbox-wrapper|To customize the listbox wrapper
- |.e-list-parent .e-list-item|To customize the listbox list items
- |.e-list-parent .e-list-item:hover|To customize the listbox list items on hover
- |.e-list-parent .e-list-item.e-selected|To customize the listbox selected list item
- |.e-listboxtool-wrapper .e-listbox-tool|To customize the listbox toolbar
- |.e-listboxtool-wrapper .e-listbox-tool .e-btn|To customize the listbox toolbar button
- |.e-listboxtool-wrapper .e-listbox-tool .e-btn .e-btn-icon.e-icons::before|To customize the listbox toolbar icon

Horizontal ListBox

You can use [cssClass](#) property to display the Listbox horizontally.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  // define the array of object
  let data = [
    { 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' },
  ];
  return (
    // specifies the tag for render the ListBox component
    <ListBoxComponent dataSource={data} cssClass='e-horizontal-listbox'/>
  )
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  // define the array of object
  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' },
  ];
  return (
    // specifies the tag for render the ListBox component
    <ListBoxComponent dataSource={data} cssClass='e-horizontal-listbox'/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React ListBox</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components"
/>
  <meta name="author" content="Syncfusion" />

```

```
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
inputs/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
dropdowns/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/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>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    /* Custom css for horizontal listbox */
    .e-horizontal-listbox .e-list-parent {
        display: inline-flex;
        align-items: center;
    }
    .e-horizontal-listbox {
        overflow-y: hidden;
        height: 100px;
    }
    .e-horizontal-listbox .e-list-parent .e-list-item {
        width: max-content;
        line-height: 100px;
        height: 100px;
    }
</style>
</head>
<body>
    <div id='sample' style="margin: 20px auto 0; width:250px;">
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>
```

How To

Add items in React List box component

To add an item or multiple items, [addItem](#) method can be used. In the following example, the **Bugatti Veyron Super Sport** and **SSC Ultimate Aero** items will be added while clicking **Add Items** button.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
```

```
function App() {
  let data = [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { 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 listboxobj;
  function btnClick() {
    if (!listboxobj.getDataByValue("Bugatti Veyron Super Sport")) {
      listboxobj.addItem({ text: 'Bugatti Veyron Super Sport', id:
'list-03' }, { text: 'SSC Ultimate Aero', id: 'list-04' });
    }
  }
  return (<div>
    <ListBoxComponent dataSource={data} ref={(scope) => listboxobj =
scope}/>
    <ButtonComponent onClick={btnClick}>Add Items</ButtonComponent>
  </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
  let data: { [key: string]: Object }[] = [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { 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 listboxobj:ListBoxComponent;
  function btnClick(): void {
    if(!listboxobj.getDataByValue("Bugatti Veyron Super Sport")){
      listboxobj.addItem({ text: 'Bugatti Veyron Super Sport', id:
'list-03' }, { text: 'SSC Ultimate Aero', id: 'list-04' });
    }
  }
  return (
    <div>
      <ListBoxComponent dataSource={data} ref={(scope) => listboxobj = scope
as ListBoxComponent} />
      <ButtonComponent onClick={btnClick}>Add Items</ButtonComponent>
    </div>
  );
}
```

```

    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Enable or disable items in React List box component

To enable or disable items in the list box, [enableItems](#) method can be used. In the following example, the Bugatti Veyron Super Sport and SSC Ultimate Aero items are disabled by default and by clicking Enable Items buttons, the disabled items will be enabled.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
  let data = [
    { 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 listboxobj;
  function enableitem() {
    listboxobj.enableItems(['Bugatti Veyron Super Sport', 'SSC Ultimate
Aero'], true);
  }
  function created() {
    listboxobj.enableItems(['Bugatti Veyron Super Sport', 'SSC Ultimate
Aero'], false);
  }
  return (<div>
    <ListBoxComponent dataSource={data} ref={(scope) => listboxobj = scope}
created={created}/>
    <ButtonComponent onClick={enableitem}>Enable Items</ButtonComponent>
  </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {

```

```
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 listboxobj:ListBoxComponent;
function enableitem(): void {
  listboxobj.enableItems(['Bugatti Veyron Super Sport', 'SSC Ultimate
Aero'], true);
}
function created():void {
  listboxobj.enableItems(['Bugatti Veyron Super Sport', 'SSC Ultimate
Aero'], false);
}
return (
<div>
  <ListBoxComponent dataSource={data} ref={(scope) => listboxobj = scope
as ListBoxComponent} created={created} />
  <ButtonComponent onClick={enableitem}>Enable Items</ButtonComponent>
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Enable scroller in React List box component

The ListBox supports scrolling and it can be achieved by restricting the height of the list box using [height](#) property.

In the following sample, **height** of the list box is restricted to **290px**.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data = [
    { 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' },
  ];
};
```

```

    return (<ListBoxComponent dataSource={data} height="290px"/>);
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  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' },
  ];
  return (
    <ListBoxComponent dataSource={data} height="290px"/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Form submit in React List box component

In the following code snippet, the value that is in selected state will be sent on form submit.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
  let data = [
    { 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' },
  ];
  return (<form>
    <ListBoxComponent dataSource={data}/>
    <ButtonComponent isPrimary="true" content="Submit"></ButtonComponent>
  </form>);
}

```

```
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
  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' },
  ];
  return (
    <form>
      <ListBoxComponent dataSource={data}/>
      <ButtonComponent isPrimary="true" content="Submit"></ButtonComponent>
    </form>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Select items in React List box component

In the following example, Bugatti Chiron is selected using [selectItems](#) method.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
  let data = [
    { 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 listboxobj;
```

```
function created() {
    listboxobj.selectItems(['Bugatti Chiron']);
}
return (<ListBoxComponent dataSource={data} ref={(scope) => listboxobj =
scope} created={created}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListBoxComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    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 listboxobj:ListBoxComponent;
    function created(): void {
        listboxobj.selectItems(['Bugatti Chiron']);
    }
    return (
        <ListBoxComponent dataSource={data} ref={(scope) =>listboxobj = scope as
        ListBoxComponent} created={created} />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

ListView

Getting Started

The following section explains the required steps to build the simple **ListView** component with its basic usage in step by step procedure.

Dependencies

Install the below required dependent packages to render the ListView component.

```
`javascript
```

```
+-- @syncfusion/ej2-react-lists
```

```
|-- @syncfusion/ej2-react-base
```

```
|-- @syncfusion/ej2-lists
```



```
|-- @syncfusion/ej2-base  
|-- @syncfusion/ej2-data  
,
```

Installation and Configuration

You can use [create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

```
`bash  
npm install -g create-react-app  
,
```

To set-up a React application in TypeScript environment, run the following command.

```
,  
npx create-react-app my-app --template typescript  
cd my-app  
npm start  
,
```

To set-up a React application in JavaScript environment, run the following command.

```
,  
npx create-react-app my-app  
cd my-app  
npm start  
,
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. Now, we are going to render

`ListView` component from these packages.

To install `ListView` component, use the following command.

```
`bash  
npm install @syncfusion/ej2-react-lists --save  
,
```

The above command installs [ListView dependencies](#) which are required to render the component in the `React` environment.

Adding ListView component

Now, you can add `ListView` component in the application. For getting started, add `ListView` component in `src/App.tsx` file using the following code snippet.

```
`ts
```

```
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import './App.css';
function App() {
  return (
    //specifies the tag to render the ListView component
    <ListViewComponent id='list' />
  );
}
export default App;
```

`ts

```
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import './App.css';
function App() {
  return (
    //specifies the tag to render the ListView component
    <ListViewComponent id='list'/>);
}
export default App;
```

Adding CSS Reference

Import **ListView** component required theme references at the top of **src/App.css**.

`css

/ import the ListView dependency styles /

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-lists/styles/material.css";
```

If you are using **CheckList** behavior in ListView, we need to add **Button** component's styles as given below in **src/App.css** file

`css

```
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

`

Bind dataSource

Populate the data in ListView using `dataSource` property. Here, an array of JSON values passed to `ListView` component.

`ts

```
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import './App.css';
function App() {
  // define the array of Json
  let arts: { [key: string]: string }[] = [
    { text: 'Artwork', id: '01' },
    { text: 'Abstract', id: '02' },
    { text: 'Modern Painting', id: '03' },
    { text: 'Ceramics', id: '04' },
    { text: 'Animation Art', id: '05' },
    { text: 'Oil Painting', id: '06' }];
  return (
    // specifies the tag to render the ListView component
    <ListViewComponent id="list" dataSource={arts} />
  );
}
export default App;
```

`

`ts

```
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import './App.css';
function App() {
  // define the array of Json
  let arts = [
    { text: 'Artwork', id: '01' },
    { text: 'Abstract', id: '02' },
    { text: 'Modern Painting', id: '03' },
```

```

{ text: 'Ceramics', id: '04' },
{ text: 'Animation Art', id: '05' },
{ text: 'Oil Painting', id: '06' }
];
return (
  // specifies the tag to render the ListView component
  <ListViewComponent id="list" dataSource={arts}/>
)
export default App;

```

Running the application

Now use the `npm start` command to run the application in the browser.

```

npm start

```

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  // define the array of Json
  let arts = [
    { text: 'Artwork', id: '01' },
    { text: 'Abstract', id: '02' },
    { text: 'Modern Painting', id: '03' },
    { text: 'Ceramics', id: '04' },
    { text: 'Animation Art', id: '05' },
    { text: 'Oil Painting', id: '06' }
  ];
  return (
    // specifies the tag to render the ListView component
    <ListViewComponent id='list' dataSource={arts}></ListViewComponent>
  )
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  // define the array of Json
  let arts: { [key: string]: string }[] = [
    { text: 'Artwork', id: '01' },

```

```

    { text: 'Abstract', id: '02' },
    { text: 'Modern Painting', id: '03' },
    { text: 'Ceramics', id: '04' },
    { text: 'Animation Art', id: '05' },
    { text: 'Oil Painting', id: '06' } ]];
  return (
    // specifies the tag to render the ListView component
    <ListViewComponent id='list' dataSource={arts} ></ListViewComponent>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

You can refer to our [React ListView](#) feature tour page for its groundbreaking feature representations. You can also explore our [React ListView example](#) to know how to render and configure the listview.

See Also

[Data Binding Types](#)

[ListView customization](#)

[Virtualization](#)

Data binding in React Listview component

ListView provides the option to load the data either from local data sources or remote data services. This can be done through dataSource property which supports the data type of array or through DataManager.

ListView supports different kind of data services such as OData, OData V4, Web API and data formats like XML, JSON, JSONP with the help of DataManager Adaptors.

Fields	Type	Description
id	string	Specifies ID attribute of list item, mapped in dataSource.
text	string	Specifies list item display text field.
isChecked	string	Specifies checked status of list item.
isVisible	string	Specifies visibility state of list item.
enabled	string	Specifies enabled state of list item.
iconCss	string	Specifies the icon class of each list item which will be add before to inner text.
child	string	Specifies child dataSource fields.
tooltip	string	Specifies tooltip title text field.
groupBy	string	Specifies category of each list item.
sortBy	string	Specifies sorting field, which is used to sort the listview data.
htmlAttributes	string	Specifies list item html attributes field.

When complex data bind to ListView, you should map the fields properly. Otherwise, the ListView properties remain as undefined or null.

Bind to local data

Local data represents in two ways, which are described below.

- Array of simple data
- Array of JSON data.

Array of simple data

ListView supports to load the array of primitive data like string and numbers. Here, both value and text field act as same.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    let data = ["Artwork", "Abstract", "Modern Painting", "Ceramics",
    "Animation Art", "Oil Painting"];
    return (
        // specifies the tag to render the ListView component
        <ListViewComponent id='list' dataSource={data}></ListViewComponent>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    let data = ["Artwork", "Abstract", "Modern Painting", "Ceramics",
    "Animation Art", "Oil Painting"];
    return (
        // specifies the tag to render the ListView component
        <ListViewComponent id='list' dataSource={data} ></ListViewComponent>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

Array of JSON data

ListView can generate its list items through an array of complex data. To get it work properly, you should map the appropriate columns to field property.

The below example illustrates the concept of binding Array of JSON data.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    //define the array of JSON
    let settings = [
```

```

        {
            Name: "Display",
            id: "list-01"
        },
        {
            Name: "Notification",
            id: "list-02"
        },
        {
            Name: "Sound",
            id: "list-03"
        },
        {
            Name: "Apps",
            id: "list-04"
        },
        {
            Name: "Storage",
            id: "list-05"
        },
        {
            Name: "Battery",
            id: "list-06"
        }
    ];
    let fields = { text: "Name", tooltip: "Name", id: "id" };
    return (
        // specifies the tag to render the ListView component
        <ListViewComponent id="list" dataSource={settings} fields={fields}
        showHeader={true} headerTitle="Device settings"/>);
    }
    export default App;
    ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    //define the array of JSON
    let settings: { [key: string]: Object }[] = [
        {
            Name: "Display",
            id: "list-01"
        },
        {
            Name: "Notification",
            id: "list-02"
        },
        {
            Name: "Sound",
            id: "list-03"
        },
        {
            Name: "Apps",

```

```

        id: "list-04"
      },
      {
        Name: "Storage",
        id: "list-05"
      },
      {
        Name: "Battery",
        id: "list-06"
      }
    ];
    let fields = { text: "Name", tooltip: "Name", id: "id" };
    return (
      // specifies the tag to render the ListView component
      <ListViewComponent
        id="list"
        dataSource={settings}
        fields={fields}
        showHeader={true}
        headerTitle="Device settings"
      />
    );
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('element'));

```

Bind to remote data

ListView supports to retrieve the data from remote data services with the help of DataManager component and Query property allows to fetch data and return it to ListView from the database.

In the below sample, displayed first 6 Products from Product table of NorthWind data service.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
//import DataManager related classes
import { DataManager, Query } from '@syncfusion/ej2-data';
function App() {
  //bind the DataManager instance to dataSource property
  let data = new DataManager({ url:
'https://services.syncfusion.com/react/production/api/',
  crossDomain: true });
  //bind the Query instance to query property
  let query = new
Query().from('ListView').select('EmployeeID,FirstName').take(10);
  //map the appropriate columns to fields property
  let fields = {
    id: 'EmployeeID',
    text: 'FirstName'
  };
};
return (
  // specifies the tag to render the ListView component
  <ListViewComponent id="list" dataSource={data} fields={fields}
  query={query} showHeader={true} headerTitle="Employees"/>);

```



```

}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
//import DataManager related classes
import { DataManager, Query, ODataV4Adaptor } from '@syncfusion/ej2-data';
function App() {
    //bind the DataManager instance to dataSource property
    let data = new DataManager({ url:
'https://services.syncfusion.com/react/production/api/',
    crossDomain: true });
    //bind the Query instance to query property
    let query = new
Query().from('ListView').select('EmployeeID,FirstName').take(10);
    //map the appropriate columns to fields property
    let fields = {
        id: 'EmployeeID',
        text: 'FirstName'
    };
    return (
        // specifies the tag to render the ListView component
        <ListViewComponent
            id="list"
            dataSource={data}
            fields={fields}
            query={query}
            showHeader={true}
            headerTitle="Employees"
        />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

[Grouping in React Listview component](#)

ListView supports to wrap the nested element into a group based on category.

The category of each list item can be mapped with `groupBy` field in the data table, which also supports single-level navigation.

In below sample, Cars are grouped based on its category using `groupBy` field.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    // define the array of Json
    let arts = [
        {

```

```

        'text': 'Audi A4',
        'id': '9bdb',
        'category': 'Audi'
    },
    {
        'text': 'Audi A5',
        'id': '4589',
        'category': 'Audi'
    },
    {
        'text': 'BMW 501',
        'id': 'f849',
        'category': 'BMW'
    },
    {
        'text': 'BMW 502',
        'id': '7aff',
        'category': 'BMW'
    }
];
let fields = { groupBy: 'category', tooltip: 'text' };
return (
    // specifies the tag to render the ListView component
    <ListViewComponent id='list' dataSource={arts}
fields={fields}></ListViewComponent>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    // define the array of Json
    let arts: { [key: string]: string }[] = [
        {
            'text': 'Audi A4',
            'id': '9bdb',
            'category': 'Audi'
        },
        {
            'text': 'Audi A5',
            'id': '4589',
            'category': 'Audi'
        },
        {
            'text': 'BMW 501',
            'id': 'f849',
            'category': 'BMW'
        },
        {
            'text': 'BMW 502',
            'id': '7aff',
            'category': 'BMW'
        }
    ]
}

```

```

    }
  ];
  let fields = { groupBy: 'category', tooltip: 'text' };
  return (
    // specifies the tag to render the ListView component
    <ListViewComponent id='list' dataSource={arts} fields={fields}
  ></ListViewComponent>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Customization

The grouping header can be customized by using groupTemplate property for both inline and fixed group header.

Check list in React Listview component

The ListView supports checkbox in default and group-lists which is used to select multiple items. The checkbox can be enabled by the showCheckBox property.

The Checkbox will be useful in the scenario where we need to select multiple options. For Example, In Shipping cart we can be able to select or unselect the desired items before checkout and also it will be useful in selecting multiple items that belongs to same category using the group list.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  //define the array of JSON
  let data = [
    { text: 'Do Meditation', id: '1' },
    { text: 'Go Jogging', id: '2' },
    { text: 'Buy Groceries', id: '3' },
    { text: 'Pay Phone bill', id: '4' },
    { text: 'Play Football with your friends', id: '5' },
  ];
  return (
    // specifies the tag to render the ListView component
    <ListViewComponent id='list' dataSource={data} showCheckBox={true}
    headerTitle='TO DO List' showHeader={true}></ListViewComponent>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  //define the array of JSON
  let data: { [key: string]: Object }[] = [
    {text: 'Do Meditation', id: '1'},
  ];

```

```

        {text: 'Go Jogging', id: '2'},
        {text: 'Buy Groceries', id: '3'},
        {text: 'Pay Phone bill', id: '4'},
        {text: 'Play Football with your friends', id: '5'},
    ];
    return (
        // specifies the tag to render the ListView component
        <ListViewComponent id='list' dataSource={data} showCheckBox = {true}
        headerTitle='TO DO List' showHeader={true}></ListViewComponent>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Checkbox Position

In ListView the checkbox can be positioned into either **Left** or **Right** side of the list-item text. This can be achieved by **checkboxPosition** property. By default, checkbox will be positioned to **Left** of list-item text.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    //define the array of JSON
    let settings = [
        { 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 fields = { text: "text", id: "id" };
    return (
        // specifies the tag to render the ListView component
        <ListViewComponent id="list" dataSource={settings} fields={fields}
        showCheckBox={true} checkBoxPosition="Right"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    //define the array of JSON
    let settings: { [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 fields = { text: "text", id: "id" };
  return (
    // specifies the tag to render the ListView component
    <ListViewComponent
      id="list"
      dataSource={settings}
      fields={fields}
      showCheckBox={true}
      checkBoxPosition="Right"
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Nested list in React Listview component

ListView component supports Nested list. For that, we should define child property for the nested list in array of JSON.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  // define the array of Json
  let arts = [
    {
      text: "Asia",
      id: "01",
      category: "Continent",
      child: [
        {
          text: "India",
          id: "1",
          category: "Asia",
          child: [
            {
              text: "Delhi",
              id: "1001",
              category: "India"
            },
            {
              text: "Kashmir",
              id: "1002",
              category: "India"
            }
          ]
        }
      ]
    }
  ]
}

```

```

        },
        {
            text: "Goa",
            id: "1003",
            category: "India"
        }
    ]
},
{
    text: "China",
    id: "2",
    category: "Asia",
    child: [
        {
            text: "Zhejiang",
            id: "2001",
            category: "China"
        },
        {
            text: "Hunan",
            id: "2002",
            category: "China"
        },
        {
            text: "Shandong",
            id: "2003",
            category: "China"
        }
    ]
}
]
},
{
    text: "North America",
    id: "02",
    category: "Continent",
    child: [
        {
            text: "USA",
            id: "3",
            category: "North America",
            child: [
                {
                    text: "California",
                    id: "3001",
                    category: "USA"
                },
                {
                    text: "New York",
                    id: "3002",
                    category: "USA"
                },
                {
                    text: "Florida",
                    id: "3003",
                    category: "USA"
                }
            ]
        }
    ]
}
]
}

```

```

    ],
    {
      text: "Canada",
      id: "4",
      category: "North America",
      child: [
        {
          text: "Ontario",
          id: "4001",
          category: "Canada"
        },
        {
          text: "Alberta",
          id: "4002",
          category: "Canada"
        },
        {
          text: "Manitoba",
          id: "4003",
          category: "Canada"
        }
      ]
    }
  ],
  {
    text: "Europe",
    id: "03",
    category: "Continent",
    child: [
      {
        text: "Germany",
        id: "5",
        category: "Europe",
        child: [
          {
            text: "Berlin",
            id: "5001",
            category: "Germany"
          },
          {
            text: "Bavaria",
            id: "5002",
            category: "Germany"
          },
          {
            text: "Hesse",
            id: "5003",
            category: "Germany"
          }
        ]
      },
      {
        text: "France",
        id: "6",
        category: "Europe",

```

```

        child: [
          {
            text: "Paris",
            id: "6001",
            category: "France"
          },
          {
            text: "Lyon",
            id: "6002",
            category: "France"
          },
          {
            text: "Marseille",
            id: "6003",
            category: "France"
          }
        ]
      },
    ],
  },
  {
    text: "Australia",
    id: "04",
    category: "Continent",
    child: [
      {
        text: "Australia",
        id: "7",
        category: "Australia",
        child: [
          {
            text: "Sydney",
            id: "7001",
            category: "Australia"
          },
          {
            text: "Melbourne",
            id: "7002",
            category: "Australia"
          },
          {
            text: "Brisbane",
            id: "7003",
            category: "Australia"
          }
        ]
      },
    ],
  },
  {
    text: "New Zealand",
    id: "8",
    category: "Australia",
    child: [
      {
        text: "Milford Sound",
        id: "8001",
        category: "New Zealand"
      },
    ],
  },

```



```

        {
          text: "Tongariro National Park",
          id: "8002",
          category: "New Zealand"
        },
        {
          text: "Fiordland National Park",
          id: "8003",
          category: "New Zealand"
        }
      ]
    }
  ],
},
{
  text: "Africa",
  id: "05",
  category: "Continent",
  child: [
    {
      text: "Morocco",
      id: "9",
      category: "Africa",
      child: [
        {
          text: "Rabat",
          id: "9001",
          category: "Morocco"
        },
        {
          text: "Toubkal",
          id: "9002",
          category: "Morocco"
        },
        {
          text: "Todgha Gorge",
          id: "9003",
          category: "Morocco"
        }
      ]
    },
    {
      text: "South Africa",
      id: "10",
      category: "Africa",
      child: [
        {
          text: "Cape Town",
          id: "10001",
          category: "South Africa"
        },
        {
          text: "Pretoria",
          id: "10002",
          category: "South Africa"
        }
      ]
    }
  ]
}

```

```

        text: "Bloemfontein",
        id: "10003",
        category: "South Africa"
      }
    ]
  }
]
];
let fields = { tooltip: "text" };
return (
  // specifies the tag to render the ListView component
  <ListViewComponent id="list" dataSource={arts} fields={fields}
  showHeader={true} headerTitle="Continent"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  // define the array of Json
  let arts: any[] = [
    {
      text: "Asia",
      id: "01",
      category: "Continent",
      child: [
        {
          text: "India",
          id: "1",
          category: "Asia",
          child: [
            {
              text: "Delhi",
              id: "1001",
              category: "India"
            },
            {
              text: "Kashmir",
              id: "1002",
              category: "India"
            },
            {
              text: "Goa",
              id: "1003",
              category: "India"
            }
          ]
        }
      ]
    },
    {
      text: "China",
      id: "2",

```

```
        category: "Asia",
        child: [
          {
            text: "Zhejiang",
            id: "2001",
            category: "China"
          },
          {
            text: "Hunan",
            id: "2002",
            category: "China"
          },
          {
            text: "Shandong",
            id: "2003",
            category: "China"
          }
        ]
      },
    ],
    {
      text: "North America",
      id: "02",
      category: "Continent",
      child: [
        {
          text: "USA",
          id: "3",
          category: "North America",
          child: [
            {
              text: "California",
              id: "3001",
              category: "USA"
            },
            {
              text: "New York",
              id: "3002",
              category: "USA"
            },
            {
              text: "Florida",
              id: "3003",
              category: "USA"
            }
          ]
        },
      ],
    },
    {
      text: "Canada",
      id: "4",
      category: "North America",
      child: [
        {
          text: "Ontario",
          id: "4001",
          category: "Canada"
        }
      ]
    }
  ]
}
```

```
    },
    {
      text: "Alberta",
      id: "4002",
      category: "Canada"
    },
    {
      text: "Manitoba",
      id: "4003",
      category: "Canada"
    }
  ]
}
]
},
{
  text: "Europe",
  id: "03",
  category: "Continent",
  child: [
    {
      text: "Germany",
      id: "5",
      category: "Europe",
      child: [
        {
          text: "Berlin",
          id: "5001",
          category: "Germany"
        },
        {
          text: "Bavaria",
          id: "5002",
          category: "Germany"
        },
        {
          text: "Hesse",
          id: "5003",
          category: "Germany"
        }
      ]
    },
    {
      text: "France",
      id: "6",
      category: "Europe",
      child: [
        {
          text: "Paris",
          id: "6001",
          category: "France"
        },
        {
          text: "Lyon",
          id: "6002",
          category: "France"
        }
      ]
    }
  ]
}
```

```
        {
          text: "Marseille",
          id: "6003",
          category: "France"
        }
      ]
    }
  ]
},
{
  text: "Australia",
  id: "04",
  category: "Continent",
  child: [
    {
      text: "Australia",
      id: "7",
      category: "Australia",
      child: [
        {
          text: "Sydney",
          id: "7001",
          category: "Australia"
        },
        {
          text: "Melbourne",
          id: "7002",
          category: "Australia"
        },
        {
          text: "Brisbane",
          id: "7003",
          category: "Australia"
        }
      ]
    }
  ],
},
{
  text: "New Zealand",
  id: "8",
  category: "Australia",
  child: [
    {
      text: "Milford Sound",
      id: "8001",
      category: "New Zealand"
    },
    {
      text: "Tongariro National Park",
      id: "8002",
      category: "New Zealand"
    },
    {
      text: "Fiordland National Park",
      id: "8003",
      category: "New Zealand"
    }
  ]
}
```

```
    }
  ]
},
{
  text: "Africa",
  id: "05",
  category: "Continent",
  child: [
    {
      text: "Morocco",
      id: "9",
      category: "Africa",
      child: [
        {
          text: "Rabat",
          id: "9001",
          category: "Morocco"
        },
        {
          text: "Toubkal",
          id: "9002",
          category: "Morocco"
        },
        {
          text: "Todgha Gorge",
          id: "9003",
          category: "Morocco"
        }
      ]
    }
  ]
},
{
  text: "South Africa",
  id: "10",
  category: "Africa",
  child: [
    {
      text: "Cape Town",
      id: "10001",
      category: "South Africa"
    },
    {
      text: "Pretoria",
      id: "10002",
      category: "South Africa"
    },
    {
      text: "Bloemfontein",
      id: "10003",
      category: "South Africa"
    }
  ]
}
]
}
];
let fields = { tooltip: "text" };
return (
```

```

// specifies the tag to render the ListView component
<ListViewComponent
  id="list"
  dataSource={arts}
  fields={fields}
  showHeader={true}
  headerTitle="Continent"
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Customizing templates in React Listview component

The ListView component is designed to customize list items, group title and header title.

Header template

The ListView header can be customized with the help of [headerTemplate](#) property.

To customize header template in your application, Declare a custom react elements within the function which returns `JSX.Element` and assign it to `headerTemplate` property along with [showHeader](#) property as `true` to display the ListView header.

```

`ts
function headerTemplate(data): JSX.Element {
  return (
    <div className="header-content">
      <span>Header</span>
    </div>
  );
  return (
    <ListViewComponent id='list'
      dataSource={listData}
      headerTemplate= {headerTemplate}
      showHeader = {true} >
    </ListViewComponent>
  )
}
`ts
function headerTemplate(data) {
  return (<div className="header-content">
    <span>Header</span>

```

```

</div>);
}
;
return (<ListViewComponent id='list' dataSource={listData} headerTemplate={headerTemplate}
showHeader={true}>
</ListViewComponent>);
`

```

In the below example, we have rendered Listview with customized header which contains search, add and sort buttons.

INDEX.JSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
    //Define an array of JSON data
    let fruitsData = [
        { text: 'Date', id: '1', imgUrl: './dates.jpg' },
        { text: 'Fig', id: '2', imgUrl: './fig.jpg' },
        { text: 'Apple', id: '3', imgUrl: './apple.png' },
        { text: 'Apricot', id: '4', imgUrl: './apricot.jpg' },
        { text: 'Grape', id: '5', imgUrl: './grape.jpg' },
        { text: 'Strawberry', id: '6', imgUrl: './strawberry.jpg' },
        { text: 'Pineapple', id: '7', imgUrl: './pineapple.jpg' },
        { text: 'Melon', id: '8', imgUrl: './melon.jpg' },
        { text: 'Lemon', id: '9', imgUrl: './lemon.jpg' },
        { text: 'Cherry', id: '10', imgUrl: './cherry.jpg' },
    ];
    function headerTemplate(data) {
        return (<div className="headerContainer">
            <span className="fruitHeader">Fruits</span>
            <ButtonComponent id="search" cssClass="e-small e-round"
isPrimary={true} iconCss="e-icons e-search-icon"/>
            <ButtonComponent id="add" cssClass="e-small e-round"
isPrimary={true} iconCss="e-icons e-add-icon"/>
            <ButtonComponent id="sort" cssClass="e-small e-round"
isPrimary={true} iconCss="e-icons e-sort-icon"/>
            </div>);
    }
    return (<ListViewComponent id="list" dataSource={fruitsData}
showHeader={true} headerTemplate={headerTemplate}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';

```



```

import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
    //Define an array of JSON data
    let fruitsData: { [key: string]: Object }[] = [
        { text: 'Date', id: '1', imgUrl: './dates.jpg' },
        { text: 'Fig', id: '2', imgUrl: './fig.jpg' },
        { text: 'Apple', id: '3', imgUrl: './apple.png' },
        { text: 'Apricot', id: '4', imgUrl: './apricot.jpg' },
        { text: 'Grape', id: '5', imgUrl: './grape.jpg' },
        { text: 'Strawberry', id: '6', imgUrl: './strawberry.jpg' },
        { text: 'Pineapple', id: '7', imgUrl: './pineapple.jpg' },
        { text: 'Melon', id: '8', imgUrl: './melon.jpg' },
        { text: 'Lemon', id: '9', imgUrl: './lemon.jpg' },
        { text: 'Cherry', id: '10', imgUrl: './cherry.jpg' },
    ];
    function headerTemplate(data: any): JSX.Element {
        return (
            <div className="headerContainer">
                <span className="fruitHeader">Fruits</span>
                <ButtonComponent
                    id="search"
                    cssClass="e-small e-round"
                    isPrimary={true}
                    iconCss="e-icons e-search-icon"
                />
                <ButtonComponent
                    id="add"
                    cssClass="e-small e-round"
                    isPrimary={true}
                    iconCss="e-icons e-add-icon"
                />
                <ButtonComponent
                    id="sort"
                    cssClass="e-small e-round"
                    isPrimary={true}
                    iconCss="e-icons e-sort-icon"
                />
            </div>
        );
    }
    return (
        <ListViewComponent
            id="list"
            dataSource={fruitsData}
            showHeader={true}
            headerTemplate={headerTemplate as any}
        />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Template

The ListView items can be customized with the help of [template](#) property.

To customize list items in your application, Declare a custom react elements within the function which returns `JSX.Element` and assign it to `template` property.

```
`ts
function template(data): JSX.Element {
  return (
    <div className="list-tem">
      <span>{data.text}</span>
    </div>
  );
  return (
    <ListViewComponent id='list'
      dataSource={listData}
      template= {template} >
    </ListViewComponent>
  )
  `
  `ts
  function template(data) {
    return (<div className="list-tem">
      <span>{data.text}</span>
    </div>);
  }
  ;
  return (<ListViewComponent id='list' dataSource={listData} template={template}>
    </ListViewComponent>);
  `
```

We provided the following built-in CSS classes to customize the list-items. Refer to the following table.

CSS class	Description
-----	-----

e-list-template, e-list-wrapper	These classes are used to differentiate normal and template rendering, which are mandatory for template rendering. The <code>e-list-template</code> class should be added to the root of the ListView element and <code>e-list-wrapper</code> class should be added to the template element wrapper <code><ListViewComponent
 cssClass="e-list-template"
template={this.template}>
</code>
---------------------------------	---

```
</ListViewComponent><br><br>template(data): JSX.Element {<br> return (<br><div
className="e-list-wrapper"></div><br>});
```

| e-list-content | This class is used to align list content and it should be added to the content element

```
<br><br> template(data): JSX.Element {<br>return ( <br><div className="e-list-
wrapper"><br><span className="e-list-content">ListItem</span><br></div><br>});|
```

| e-list-avatar | This class is used for avatar customization. It should be added to the template element wrapper. After adding it, we can customize our element with [Avatar](#) classes

```
<br> <br>
template(data): JSX.Element {<br> return (<br> <div className="e-list-wrapper e-list-
avatar"><br> <span className="e-avatar e-avatar-circle">MR</span><br> <span
className="e-list-content">ListItem</span><br> </div><br> });
```

| e-list-avatar-right | This class is used to align avatar to right side of the list item. It should be added to the template element wrapper. After adding it, we can customize our element with [Avatar](#) classes

```
<br><br> <div className="e-list-wrappere-list-avatar-right"> <br> <span className="e-list-
content">ListItem</span><br><span className="e-avatar e-avatar-circle">MR</span><br>
</div>|
```

| e-list-badge | This class is used for badge customization .It should be added to the template element wrapper. After adding it, we can customize our element with [Badge](#) classes

```
<br><br>template(data):
JSX.Element {<br> return (<br><div className="e-list-wrapper e-list-avatar-right"><br><span
className="e-list-content">ListItem</span><br><span className="e-avatar e-avatar-
circle">MR</span><br></div><br>});|
```

| e-list-multi-line | This class is used for multi-line customization. It should be added to the template element wrapper. After adding it, we can customize List item's header and description

```
<br><br>template(data): JSX.Element {<br>return (<br><div className="e-list-wrapper e-list-
multi-line"><br><span className="e-list-content">ListItem</span><br></div><br>});|
```

| e-list-item-header | This class is used to align a list header and it should be added to the header element along with the multi-line class

```
<br><br> template(data): JSX.Element {<br> return (<br>
<div className="e-list-wrapper e-list-multi-line"><br> <span className="e-list-item-
header">ListItem Header</span><br> <span className="e-list-content">ListItem</span><br>
</div><br> });
```

- If you are using [Avatar](#) in ListView templates, we need to add [Layouts](#) component's styles as given below in `src/App.css` file

```
`css
```

```
@import "../node_modules/@syncfusion/ej2-layouts/styles/material.css";
```

In the below example, we have customized list items like [Contact](#) app with our avatar component.

INDEX.JSX

```
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
```

```

function App() {
  //Define an array of JSON data
  let data = [
    {
      text: "Jenifer",
      contact: "(206) 555-985774",
      id: "1",
      avatar: "",
      pic: "pic01"
    },
    { text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A",
pic: "" },
    {
      text: "Isabella",
      contact: "(206) 555-8122",
      id: "4",
      avatar: "",
      pic: "pic02"
    },
    {
      text: "William ",
      contact: "(206) 555-9482",
      id: "5",
      avatar: "W",
      pic: ""
    },
    {
      text: "Jacob",
      contact: "(71) 555-4848",
      id: "6",
      avatar: "",
      pic: "pic04"
    },
    { text: "Matthew", contact: "(71) 555-7773", id: "7", avatar: "M",
pic: "" },
    {
      text: "Oliver",
      contact: "(71) 555-5598",
      id: "8",
      avatar: "",
      pic: "pic03"
    },
    {
      text: "Charlotte",
      contact: "(206) 555-1189",
      id: "9",
      avatar: "C",
      pic: ""
    }
  ];
  let fields = { text: "Name" };
  function listTemplate(data) {
    let letterAvatar = <span className='e-avatar e-avatar-
circle'>{data.avatar}</span>;
    let imageAvatar = <span className={` ${data.pic} e-avatar e-avatar-
circle`} ></span>;
  }
}

```

```

    return (<div className='e-list-wrapper e-list-multi-line e-list-
avatar'>
        {data.avatar !== "" ? (letterAvatar) : (imageAvatar)}
        <span className="e-list-item-header">{data.text}</span>
        <span className="e-list-content">{data.contact}</span>
    </div>);
}
return (<div>
    <ListViewComponent id='list' dataSource={data}
headerTitle='Contacts' showHeader={true} sortOrder="Ascending" width='350px'
template={listTemplate} fields={fields} cssClass='e-list-
template'></ListViewComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    //Define an array of JSON data
    let data: { [key: string]: Object }[] = [
        {
            text: "Jenifer",
            contact: "(206) 555-985774",
            id: "1",
            avatar: "",
            pic: "pic01"
        },
        { text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A",
pic: "" },
        {
            text: "Isabella",
            contact: "(206) 555-8122",
            id: "4",
            avatar: "",
            pic: "pic02"
        },
        {
            text: "William ",
            contact: "(206) 555-9482",
            id: "5",
            avatar: "W",
            pic: ""
        },
        {
            text: "Jacob",
            contact: "(71) 555-4848",
            id: "6",
            avatar: "",
            pic: "pic04"
        },
    ],
}

```

```

    { text: "Matthew", contact: "(71) 555-7773", id: "7", avatar: "M",
    pic: "" },
    {
      text: "Oliver",
      contact: "(71) 555-5598",
      id: "8",
      avatar: "",
      pic: "pic03"
    },
    {
      text: "Charlotte",
      contact: "(206) 555-1189",
      id: "9",
      avatar: "C",
      pic: ""
    }
  ];
  let fields: any = { text: "Name" };
  function listTemplate(data: any): JSX.Element {
    let letterAvatar = <span className='e-avatar e-avatar-circle'>{data.avatar}</span>
    let imageAvatar = <span className={` ${data.pic} e-avatar e-avatar-circle`} ></span>
    return (
      <div className='e-list-wrapper e-list-multi-line e-list-avatar'>
        {data.avatar !== "" ? (letterAvatar) : (imageAvatar)}
        <span className="e-list-item-header">{data.text}</span>
        <span className="e-list-content">{data.contact}</span>
      </div>
    );
  }
  return (
    <div>
      <ListViewComponent id='list' dataSource={data}
      headerTitle='Contacts' showHeader={true}
      sortOrder="Ascending" width='350px' template={listTemplate
      as any}
      fields={fields as any} cssClass='e-list-template'></ListViewComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Group template

The ListView group header can be customized with the help of [groupTemplate](#) property.

To customize the group template in your application, Declare a custom react elements within the function which returns `JSX.Element` and assign it to `groupTemplate` property.

`ts

```

function groupTemplate(data: any): JSX.Element {
  return(

```

```

<div>
  <span className='category'>{data.items[0].category}</span>
  <span className="count"> {data.items.length} Item(s)</span>
</div>
);
}
return (
  <ListViewComponent id='list'
    dataSource={listData}
    groupTemplate= {groupTemplate} >
  </ListViewComponent>
)
,
`ts
function groupTemplate(data) {
return (<div>
  <span className='category'>{data.items[0].category}</span>
  <span className="count"> {data.items.length} Item(s)</span>
</div>);
}
return (<ListViewComponent id='list' dataSource={listData} groupTemplate={groupTemplate}>
</ListViewComponent>);
,

```

In the below example, we have grouped Listview based on the category. The category of each list item should be mapped with `[groupBy]` field of the data. We have also displayed grouped list items count in the group list header.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  //Define an array of JSON data
  let data = [
    { Name: 'Nancy', contact: '(206) 555-985774', id: '1', image:
'https://ej2.syncfusion.com/demos/src/grid/images/1.png', category:
'Experience' },

```

```

    { Name: 'Janet', contact: '(206) 555-3412', id: '2', image:
'https://ej2.syncfusion.com/demos/src/grid/images/3.png', category:
'Fresher' },
    { Name: 'Margaret', contact: '(206) 555-8122', id: '4', image:
'https://ej2.syncfusion.com/demos/src/grid/images/4.png', category:
'Experience' },
    { Name: 'Andrew ', contact: '(206) 555-9482', id: '5', image:
'https://ej2.syncfusion.com/demos/src/grid/images/2.png', category:
'Experience' },
    { Name: 'Steven', contact: '(71) 555-4848', id: '6', image:
'https://ej2.syncfusion.com/demos/src/grid/images/5.png', category:
'Fresher' },
    { Name: 'Michael', contact: '(71) 555-7773', id: '7', image:
'https://ej2.syncfusion.com/demos/src/grid/images/6.png', category:
'Experience' },
    { Name: 'Robert', contact: '(71) 555-5598', id: '8', image:
'https://ej2.syncfusion.com/demos/src/grid/images/7.png', category:
'Fresher' },
    { Name: 'Laura', contact: '(206) 555-1189', id: '9', image:
'https://ej2.syncfusion.com/demos/src/grid/images/8.png', category:
'Experience' },
  ];
  let fields = { text: 'Name', groupBy: 'category' };
  //Set customized list template
  function listTemplate(data) {
    return (<div className="e-list-wrapper e-list-multi-line e-list-
avatar">
      <img className="e-avatar e-avatar-circle" src={data.image}/>
      <span className="e-list-item-header">{data.Name}</span>
      <span className="e-list-content">{data.contact}</span>
    </div>);
  }
  //Set customized group-header template
  function groupTemplate(data) {
    return (<div>
      <span className='category'>{data.items[0].category}</span>
      <span className="count"> {data.items.length} Item(s)</span>
    </div>);
  }
  return (<div>
    <ListViewComponent id='list' dataSource={data} fields={fields}
template={listTemplate} groupTemplate={groupTemplate} width='350px'
cssClass='e-list-template'>
      </ListViewComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  //Define an array of JSON data
```



```

    let data: { [key: string]: Object }[] = [
      { Name: 'Nancy', contact: '(206) 555-985774', id: '1', image:
'https://ej2.syncfusion.com/demos/src/grid/images/1.png', category:
'Experience' },
      { Name: 'Janet', contact: '(206) 555-3412', id: '2', image:
'https://ej2.syncfusion.com/demos/src/grid/images/3.png', category:
'Fresher' },
      { Name: 'Margaret', contact: '(206) 555-8122', id: '4', image:
'https://ej2.syncfusion.com/demos/src/grid/images/4.png', category:
'Experience' },
      { Name: 'Andrew ', contact: '(206) 555-9482', id: '5', image:
'https://ej2.syncfusion.com/demos/src/grid/images/2.png', category:
'Experience' },
      { Name: 'Steven', contact: '(71) 555-4848', id: '6', image:
'https://ej2.syncfusion.com/demos/src/grid/images/5.png', category:
'Fresher' },
      { Name: 'Michael', contact: '(71) 555-7773', id: '7', image:
'https://ej2.syncfusion.com/demos/src/grid/images/6.png', category:
'Experience' },
      { Name: 'Robert', contact: '(71) 555-5598', id: '8', image:
'https://ej2.syncfusion.com/demos/src/grid/images/7.png', category:
'Fresher' },
      { Name: 'Laura', contact: '(206) 555-1189', id: '9', image:
'https://ej2.syncfusion.com/demos/src/grid/images/8.png', category:
'Experience' },
    ];
    let fields: object = { text: 'Name', groupBy: 'category' };
    //Set customized list template
    function listTemplate(data: any): JSX.Element {
      return(
        <div className="e-list-wrapper e-list-multi-line e-list-avatar">
          <img className="e-avatar e-avatar-circle" src={data.image}
/>

          <span className="e-list-item-header">{data.Name}</span>
          <span className="e-list-content">{data.contact}</span>
        </div>
      );
    }
    //Set customized group-header template
    function groupTemplate(data: any): JSX.Element {
      return(
        <div>
          <span className="category">{data.items[0].category}</span>
          <span className="count"> {data.items.length} Item(s)</span>
        </div>
      );
    }
    return (
      <div>
        <ListViewComponent id='list' dataSource={data} fields={fields}
template={listTemplate as any}
groupTemplate={groupTemplate as any} width='350px' cssClass='e-
list-template'>
        </ListViewComponent>
      </div>
    );
  }

```

```
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

Virtualization in React Listview component

UI virtualization loads only viewable list items in a view port which will increase ListView performance on loading large number of data.

Module injection

In order to use UI Virtualization, we need to inject its `virtualization` service in the App. This modules should be injected into the ListView using the `Inject` directive as like the below code snippet.

```
`ts
import { ListViewComponent, Inject, Virtualization } from '@syncfusion/ej2-react-lists';
....
....
return (
// specifies the tag to render the ListView component
<ListViewComponent id='ui-list' dataSource={listData} enableVirtualization={true} >
<Inject services={[Virtualization]} />
</ListViewComponent>
);
`ts
return (
// specifies the tag to render the ListView component
<ListViewComponent id='ui-list' dataSource={listData} enableVirtualization={true}>
<Inject services={[Virtualization]} />
</ListViewComponent>);
export {};
`
```

Getting started

UI virtualization can be enabled in ListView by setting [enableVirtualization](#) property to true. It has two type of scroller

Window scroll - This scroller is used in ListView by default.

Container scroll - This will be used, if the height property of ListView was set.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from "react-dom";
```

```

import { ListViewComponent, Inject, Virtualization } from '@syncfusion/ej2-react-lists';
function App() {
    // define the array of Json
    let listData;
    listData = [
        { text: "Nancy", id: "0" },
        { text: "Andrew", id: "1" },
        { text: "Janet", id: "2" },
        { text: "Margaret", id: "3" },
        { text: "Steven", id: "4" },
        { text: "Laura", id: "5" },
        { text: "Robert", id: "6" },
        { text: "Michael", id: "7" },
        { text: "Albert", id: "8" },
        { text: "Nolan", id: "9" }
    ];
    for (let i = 10; i <= 1010; i++) {
        let index = parseInt((Math.random() * 10).toString());
        listData.push({ text: listData[index].text, id: i.toString() });
    }
    return (
        // specifies the tag to render the ListView component
        <ListViewComponent id="ui-list" dataSource={listData}
        enableVirtualization={true}>
            <Inject services={[Virtualization]}/>
        </ListViewComponent>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent, Inject, Virtualization } from '@syncfusion/ej2-react-lists';
function App() {
    // define the array of Json
    let listData: { [key: string]: string | object }[];
    listData = [
        { text: "Nancy", id: "0" },
        { text: "Andrew", id: "1" },
        { text: "Janet", id: "2" },
        { text: "Margaret", id: "3" },
        { text: "Steven", id: "4" },
        { text: "Laura", id: "5" },
        { text: "Robert", id: "6" },
        { text: "Michael", id: "7" },
        { text: "Albert", id: "8" },
        { text: "Nolan", id: "9" }
    ];
    for (let i: number = 10; i <= 1010; i++) {
        let index: number = parseInt((Math.random() * 10).toString());
        listData.push({ text: listData[index].text, id: i.toString() });
    }
}

```

```

return (
  // specifies the tag to render the ListView component
  <ListViewComponent id="ui-list" dataSource={listData}
  enableVirtualization={true}>
    <Inject services={[Virtualization]} />
  </ListViewComponent>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

We can use **template** property to customize list items in UI virtualization.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent, Inject, Virtualization } from '@syncfusion/ej2-react-lists';
function App() {
  // define the array of Json
  let listData;
  listData = [
    { name: 'Nancy', icon: 'N', id: '0' },
    { name: 'Andrew', icon: 'A', id: '1' },
    { name: 'Janet', icon: 'J', id: '2' },
    { name: 'Margaret', imgUrl:
'https://ej2.syncfusion.com/react/demos/src/listview/images/margaret.png',
id: '3' },
    { name: 'Steven', icon: 'S', id: '4' },
    { name: 'Laura', imgUrl:
'https://ej2.syncfusion.com/react/demos/src/listview/images/laura.png', id:
'5' },
    { name: 'Robert', icon: 'R', id: '6' },
    { name: 'Michael', icon: 'M', id: '7' },
    { name: 'Albert', imgUrl:
'https://ej2.syncfusion.com/react/demos/src/listview/images/albert.png', id:
'8' },
    { name: 'Nolan', icon: 'N', id: '9' }
  ];
  for (let i = 10; i <= 1010; i++) {
    let index = parseInt((Math.random() * 10).toString());
    listData.push({ name: listData[index].name, icon:
listData[index].icon, imgUrl: listData[index].imgUrl, id: i.toString() });
  }
  function template(data) {
    return (
      <div className="e-list-wrapper e-list-avatar">
        <span className={`e-avatar e-avatar-small e-avatar-circle
${data.icon} ${data.imgUrl ? 'hideUI' : 'showUI'}}>{data.icon}</span>
        <img className={`e-avatar e-avatar-circle ${data.imgUrl ? 'showUI' :
'hideUI'}}` src={data.imgUrl ? data.imgUrl : ''} alt="" />
        <span className="e-list-content">{data.name}</span>
      </div>
    );
  }
  return (

```

```

    // specifies the tag to render the ListView component
    <ListViewComponent id='ui-list' dataSource={listData}
    enableVirtualization={true} template={template} cssClass='e-list-template'
    height={500} headerTitle="Contacts" showHeader={true}>
        <Inject services={[Virtualization]}/>
    </ListViewComponent>;
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent, Inject, Virtualization } from '@syncfusion/ej2-react-lists';
function App() {
    // define the array of Json
    let listData: { [key: string]: string | object }[];
    listData = [
        { name: 'Nancy', icon: 'N', id: '0', },
        { name: 'Andrew', icon: 'A', id: '1' },
        { name: 'Janet', icon: 'J', id: '2' },
        { name: 'Margaret', imgUrl:
'https://ej2.syncfusion.com/react/demos/src/listview/images/margaret.png',
id: '3' },
        { name: 'Steven', icon: 'S', id: '4' },
        { name: 'Laura', imgUrl:
'https://ej2.syncfusion.com/react/demos/src/listview/images/laura.png', id:
'5' },
        { name: 'Robert', icon: 'R', id: '6' },
        { name: 'Michael', icon: 'M', id: '7' },
        { name: 'Albert', imgUrl:
'https://ej2.syncfusion.com/react/demos/src/listview/images/albert.png', id:
'8' },
        { name: 'Nolan', icon: 'N', id: '9' }
    ];
    for (let i: number = 10; i <= 1010; i++) {
        let index: number = parseInt((Math.random() * 10).toString());
        listData.push({ name: listData[index].name, icon:
listData[index].icon, imgUrl: listData[index].imgUrl, id: i.toString() });
    }
    // Set customized list template
    function template(data: any) {
        return (
            <div className="e-list-wrapper e-list-avatar">
                <span className={`e-avatar e-avatar-circle ${data.icon}
${data.imgUrl ? 'hideUI' : 'showUI'}`}>{data.icon}</span>
                <img className={`e-avatar e-avatar-circle ${data.imgUrl ?
'showUI' : 'hideUI'}`} src={data.imgUrl ? data.imgUrl : ' '} />
                <span className="e-list-content">{data.name}</span>
            </div>
        );
    }
    return (
        // specifies the tag to render the ListView component

```

```

    <ListViewComponent id='ui-list' dataSource={listData}
    enableVirtualization={true} template={template} height={500} cssClass='e-
    list-template' headerTitle="Contacts" showHeader={true}>
        <Inject services={[Virtualization]} />
    </ListViewComponent>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.CSS

```

#ui-list.e-listview {
    margin: auto;
    max-width: 325px;
    line-height: initial;
    border: 1px solid lightgray;
}
/* ListView header alignment */
#ui-list.e-listview .e-list-header {
    height: 50px
}
#ui-list.e-listview .e-list-header .e-text {
    line-height: 18px
}
/* ListView template customization */
#ui-list.e-listview .showUI {
    display: flex;
    align-items: center;
    justify-content: center;
    border-radius: 50%;
}
#ui-list.e-listview .hideUI {
    display: none;
}
#ui-list.e-listview .e-list-item {
    padding: 3px 0;
}
#ui-list.e-listview .R {
    background: lightgrey;
}
#ui-list.e-listview .M {
    background: pink;
}
#ui-list.e-listview .A {
    background: lightgreen;
}
#ui-list.e-listview .S {
    background: lightskyblue;
}
#ui-list.e-listview .J {
    background: orange;
}
#ui-list.e-listview .N {
    background: lightblue;
}

```

Conditional rendering

We have also provided following conditional rendering support for template and groupTemplate.

| Name | Syntax |

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

| conditional class | `<div class="$ { $id % 2 === 0 ? 'even-list' : 'odd-list' }"></div>` |

| conditional attribute | `<div id="$ { $id % 2 === 0 ? 'even-list' : 'odd-list' }"></div>` |

| conditional text content | `<div> $ { $id % 2 === 0 ? 'even-list' : 'odd-list' } </div>` |

In the below sample, we have applied light blue for even list and light coral for odd list based on the conditional class.

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent, Inject, Virtualization } from '@syncfusion/ej2-react-lists';
function App() {
    // define the array of Json
    let listData;
    listData = [
        { text: "Nancy", id: "0" },
        { text: "Andrew", id: "1" },
        { text: "Janet", id: "2" },
        { text: "Margaret", id: "3" },
        { text: "Steven", id: "4" },
        { text: "Laura", id: "5" },
        { text: "Robert", id: "6" },
        { text: "Michael", id: "7" },
        { text: "Albert", id: "8" },
        { text: "Nolan", id: "9" }
    ];
    for (let i = 10; i <= 1010; i++) {
        let index = parseInt((Math.random() * 10).toString());
        listData.push({ text: listData[index].text, id: i.toString() });
    }
    let template = "<div id=\"list-container\" class=\"$ { $id % 2 === 0 ? 'even-list' : 'odd-list' }\" > ${text} </div>";
    return (
        // specifies the tag to render the ListView component
        <ListViewComponent id="ui-list" dataSource={listData}
        enableVirtualization={true} template={template} height={500}>
            <Inject services={[Virtualization]}>
                </ListViewComponent>
            </Inject>
        </ListViewComponent>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import * as React from 'react';
```

```

import * as ReactDOM from "react-dom";
import { ListViewComponent, Inject, Virtualization } from '@syncfusion/ej2-react-lists';
function App() {
    // define the array of Json
    let listData: { [key: string]: string | object }[];
    listData = [
        { text: "Nancy", id: "0" },
        { text: "Andrew", id: "1" },
        { text: "Janet", id: "2" },
        { text: "Margaret", id: "3" },
        { text: "Steven", id: "4" },
        { text: "Laura", id: "5" },
        { text: "Robert", id: "6" },
        { text: "Michael", id: "7" },
        { text: "Albert", id: "8" },
        { text: "Nolan", id: "9" }
    ];
    for (let i: number = 10; i <= 1010; i++) {
        let index: number = parseInt((Math.random() * 10).toString());
        listData.push({ text: listData[index].text, id: i.toString() });
    }
    let template: string =
        "<div id=\"list-container\" class=\"${ $id % 2 === 0 ? 'even-list' : 'odd-list' }\" > ${text} </div>";
    return (
        // specifies the tag to render the ListView component
        <ListViewComponent
            id="ui-list"
            dataSource={listData}
            enableVirtualization={true}
            template={template}
            height={500}
        >
            <Inject services={[Virtualization]} />
        </ListViewComponent>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Scrolling in React Listview component

Scrolling is a technique that allows you to load more items as the user scrolls through a list, providing a seamless and dynamic user experience.

Render the ListView with `dataSource`, and bind the [scroll](#) event. Within the scroll event, you can access information such as the scroll direction, event name and the distance from the scrollbar to the top and bottom ends through the `distanceY` parameter.

In the given example, new data is seamlessly added while scrolling. When you scrolls to the bottom and the distance scrolled is less than 100 pixels, it dynamically loads a batch of items into the list as long as there are more items to render.

INDEX.JSX

```

import * as ReactDOM from 'react-dom';

```



```

import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  let listViewInstance = null;
  let data = [{
    text: "Hi Guys, Good morning! \uD83D\uDE0A, I'm very delighted to share with you the news that our team is going to launch a new mobile application",
    positionClass: 'right',
  }, {
    text: "Oh! That's great \uD83D\uDE42",
    positionClass: 'left',
  }, {
    {
      text: 'That is a good news \uD83D\uDE00',
      templateHeight: '80px',
      positionClass: 'right',
    },
    {
      text: 'What kind of application we are gonna launch? \uD83E\uDD14',
      positionClass: 'left',
    },
    {
      text: 'A kind of "Emergency Broadcast App" like being able to just invite someone to teams without it impacting how many people have official access',
      positionClass: 'right',
    },
    {
      text: 'Who will be the client users for our applications? ',
      positionClass: 'left',
    },
    {
      text: 'Is currently the only way to invite someone through 0365? Just wondering down the road how organization would want to handle that with freelancers, like being able to just invite someone to teams without it impacting how many people have official access \uD83D\uDE14',
      positionClass: 'right',
    },
    {
      text: 'Yes, however, that feature of inviting someone from outside your organization is planned - expected closer to GA next year \uD83D\uDC4D',
      positionClass: 'left',
    },
    {
      text: 'I guess we should switch things over to hear for a while. How long does the trial last? \uD83E\uDD14',
      positionClass: 'right',
    },
    { text: 'I think the trial is 30 days. \uD83D\uDE03', positionClass: 'left' },
    {
      text: 'Only 0365 only members of your organization. They said that they are listening to customer feedback and hinted that guest users would be brought in down the road \uD83D\uDE09',
      positionClass: 'right',
    },
  ],

```

```

    { text: 'Cool thanks! \uD83D\uDC4C', positionClass: 'left' }
  ];
  function listTemplate(data) {
    const wrapperClass = `e-list-wrapper e-list-multi-
line${data.positionClass === 'right' ? ' e-list-wrapper-right' : ''}`;
    return (
      <div className={ wrapperClass }>
        <span className="e-list-item-header text-span">
          {data.text }
        </span>
      </div>
    );
  }
  let itemsRendered = 5;
  let itemPerScroll = 5;
  let result= [];
  const onListScrolled = (args) => {
    if (args.scrollDirection === 'Bottom' && args.distanceY < 100) {
      if (itemsRendered < data.length) {
        const startIndex = itemsRendered;
        const endIndex = Math.min(itemsRendered + itemPerScroll,
data.length);
        result = data.slice(startIndex, endIndex);
        listViewInstance.addItem(result);
        itemsRendered = endIndex;
      }
    }
  }
  return (
    <div className="grid-container"><div><h3>Chat</h3>
      <ListViewComponent id='list' dataSource={data.slice(0, itemsRendered)}
height= "320px"
width= "400px" template={listTemplate.bind(this)} cssClass='e-list-
template' scroll={onListScrolled.bind(this)} ref={scope => {
listViewInstance = scope;}}></ListViewComponent>
    </div></div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App(this: any) {
  let listViewInstance: ListViewComponent | null = null;
  let data: { text: string; positionClass: string; templateHeight?: string }[] = [
    {
      text: "Hi Guys, Good morning! \uD83D\uDE0A, I'm very delighted to share
with you the news that our team is going to launch a new mobile
application",
      positionClass: 'right',
    }, {
      text: "Oh! That's great \uD83D\uDE42",
    }
  ];
}

```

```

    positionClass: 'left',
  },
  {
    text: 'That is a good news \u2013\u2014',
    templateHeight: '80px',
    positionClass: 'right',
  },
  {
    text: 'What kind of application we are gonna launch? \u2013\u2014',
    positionClass: 'left',
  },
  {
    text: 'A kind of "Emergency Broadcast App" like being able to just invite someone to teams without it impacting how many people have official access',
    positionClass: 'right',
  },
  {
    text: 'Who will be the client users for our applications? ',
    positionClass: 'left',
  },
  {
    text: 'Is currently the only way to invite someone through 0365? Just wondering down the road how organization would want to handle that with freelancers, like being able to just invite someone to teams without it impacting how many people have official access \u2013\u2014',
    positionClass: 'right',
  },
  {
    text: 'Yes, however, that feature of inviting someone from outside your organization is planned - expected closer to GA next year \u2013\u2014',
    positionClass: 'left',
  },
  {
    text: 'I guess we should switch things over to hear for a while. How long does the trial last? \u2013\u2014',
    positionClass: 'right',
  },
  { text: 'I think the trial is 30 days. \u2013\u2014', positionClass: 'left' },
  {
    text: 'Only 0365 only members of your organization. They said that they are listening to customer feedback and hinted that guest users would be brought in down the road \u2013\u2014',
    positionClass: 'right',
  },
  { text: 'Cool thanks! \u2013\u2014', positionClass: 'left' }
];

function listTemplate(data: any): JSX.Element {
  const wrapperClass = `e-list-wrapper e-list-multi-line${data.positionClass === 'right' ? ' e-list-wrapper-right' : ''}`;
  return (
    <div className={ wrapperClass }>
      <span className="e-list-item-header text-span">
        {data.text}
      </span>
    </div>
  );
}

```

```

    );
  }
  let itemsRendered = 5;
  let itemPerScroll = 5;
  let result: { text: string; positionClass: string; templateHeight?: string }[] = [];
  const onListScrolled = (args:any) => {
    if (args.scrollDirection === 'Bottom' && args.distanceY < 100) {
      if (itemsRendered < data.length) {
        const startIndex = itemsRendered;
        const endIndex = Math.min(itemsRendered + itemPerScroll,
data.length);
        result = data.slice(startIndex, endIndex) as { text: string;
positionClass: string; templateHeight?: string }[];
        listViewInstance.addItem(result);
        itemsRendered = endIndex;
      }
    }
  }
  return (
    <div className="grid-container"><div><h3>Chat</h3>
      <ListViewComponent id='list' dataSource={data.slice(0, itemsRendered)}
height= "320px"
width= "400px" template={listTemplate as any} cssClass='e-list-template'
scroll={onListScrolled as any} ref={scope => { listViewInstance =
scope;}}></ListViewComponent>
    </div></div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

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%;
}

#element{
  display: block;
  max-width: 280px;
  margin: auto;
  border: 1px solid #dddddd;
  border-radius: 3px;
}

```

```

#list {
  display: block;
  max-width: 400px;
  margin: auto;
  border: 1px solid #dddddd;
  border-radius: 3px;
}
.grid-container{
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 20px;
  align-items: center;
}
.grid-container > div {
  border: 1px solid #ddd;
  padding: 20px;
  border-radius: 5px;
  background-color: #f4f4f4;
}
h3 {
  margin: 0;
}
.e-list-wrapper-right{
  display: flex;
  justify-content: flex-end;
}
.text-span {
  white-space: normal !important;
  max-width: 80%;
  padding: 10px !important;
  background-color: #e0e0e0;
  border-radius: 10px;
  word-wrap: break-word;
}

```

Style in React Listview component

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

Customizing ListView

Use the following CSS to customize the ListView.

```

`css
.e-listview {
border: 5px solid rgb(173, 255, 47);
}
`

```

Customizing the list items

Use the following CSS to customize the items of ListView.

```

`css

```

```
.e-listview .e-list-item {  
text-align: center;  
color: pink;  
background-color: #2fa1ff;  
}  
`css
```

Customizing ListView's header

Use the following CSS to customize the header of ListView control.

```
`css  
.e-listview .e-list-header{  
color: #2fa1ff;  
justify-content: center;  
}  
`css
```

Customizing group header of ListView

Use the following CSS to customize the category of the group items.

```
`css  
.e-listview .e-list-group-item {  
color: rgb(173, 255, 47);  
background-color: maroon;  
text-align: end;  
}  
`css
```

Customizing the hover state of ListView control

Use the following CSS to customize the list item when hovering.

Customizing ListView hover state with the checkbox checked

```
`css  
.e-listview .e-list-item.e-hover.e-active.e-checklist {  
color: rgb(83, 5, 79);  
background-color: rgb(173, 255, 47);  
}  
`css
```

Customizing ListView hover state

```
`css  
.e-listview .e-list-item.e-hover {
```

```
color:red;
background-color: rgb(173, 255, 47);
}
`
```

Customizing selected item of ListView control

Use the following CSS to customize the selected list item.

Customizing ListView's selected item with the checkbox checked

```
`css
.e-listview .e-list-item.e-checklist.e-focused.e-active {
color: rgb(83, 5, 79);
background-color: rgb(0, 15, 100);
}
`
```

Customizing ListView's selected item

```
`css
.e-listview .e-list-item.e-focused {
color: #2fa1ff;
background-color: rgb(0, 15, 100);
}
`
```

Accessibility in React ListView component

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

Attributes	Purpose
---	---
<code>role=list</code>	Specifies the non selectable list container.
<code>role=listitem</code>	Specifies the role of each item in a selectable ListView and its containment within the list.
<code>role=presentation</code>	Specifies the role of non selectable list element.
<code>role=checkbox</code>	Indicates checkbox control along with listitem element.
<code>aria-checked</code>	Indicates the current checked state of checkbox.
<code>aria-label</code>	Provides an accessible name for the ListView Checkbox.
<code>aria-disabled</code>	Indicates element is perceivable but disabled.

Keyboard interaction

The ListView 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 ListView component.

| Keyboard shortcuts | Actions |

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

| Arrow Up | Move to the previous list item |

| Arrow Down | Move to the next list item |

| Space | Check and uncheck the targeted list from the whole list |

| BackSpace | Get back to the previous lists if it is nested list |

| Home | Moves focus to first list item |

| End | Moves focus to last list item |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Ej1 api migration in React Listview component

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

{% raw %}

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

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

| Virtualization | **Property:** *allowVirtualScrolling*

 <EJ.ListView id="default" allowVirtualScrolling={true}
 virtualScrollMode="normal" dataSource={dataSourceItem}>
 </EJ.ListView>
 | **Property:** *enableVirtualization*

 <ListComponent id='ui-list' dataSource={this.dataSource} enableVirtualization={true} >
</Inject services=[[Virtualization]] />
 </ListComponent> |

| Fields | **Property:** *fieldSettings*

 <EJ.ListView id="default"
 enableGroupList={true} dataSource={dataSourceItem}
 </EJ.ListView>
 | **Property:** *fields*

 Inner properties: *child, enabled, groupBy, htmlAttributes, iconsCss, id, isChecked, isVisible, sortBy, tableName, text, tooltip.*

 public fields = { enabled: enable_items,
 groupBy: groupByProp };

 <ListComponent id='list'
 dataSource={this.dataSource}
 fields={this.fields}
 </ListComponent> |

| Template | **Property:** *renderTemplate*

 <EJ.ListView dataSource={this.dataSource}
 renderTemplate={true} >
 <div id="template">
 <div> Template1 </div>
 </div>
 </EJ.ListView>

 | **Property:** *template*

 private

```
listTemplate(data: any): JSX.Element { <br /> return ( <br /> <div
className="template">Template Data</div> <br /> ); <br /> } <br /> <br /> <ListViewComponent
id='list' <br /> dataSource={this.dataSource} <br /> template={this.listTemplate.bind(this)}> <br
/> </ListViewComponent> |
```

```
| Animation | Not Applicable | Property: animation <br /><br /> public animation = { effect:
'SlideLeft', <br /> duration: 400, easing: 'ease' }; <br /><br /> <ListViewComponent id='list'
dataSource={this.dataSource} <br /> animation={this.animation}> <br /> </ListViewComponent>
<br />|
```

```
| Enable | Not Applicable | Property: enable <br /> <br /> <ListViewComponent id='list'
dataSource={this.dataSource} <br /> enable={true}> <br /> </ListViewComponent> |
```

```
| Template for grouping | Not Applicable | Property: groupTemplate <br /> <br /> private
listGroupTemplate(data: any): JSX.Element { <br /> return ( <br /> <div
className="template">Template Data</div> <br /> ); <br /> } <br /> <br /> <ListViewComponent
id='list' <br /> dataSource={this.dataSource} <br />
groupTemplate={this.listGroupTemplate.bind(this)}> <br /> </ListViewComponent> |
```

```
| Template for header | Not Applicable | Property: headerTemplate <br /><br /> private
listHeaderTemplate(data: any): JSX.Element { <br /> return ( <br /> <div
className="template">Template Data</div> <br /> ); <br /> } <br /> <br /> <ListViewComponent
id='list' <br /> dataSource={this.dataSource} <br />
headerTemplate={this.listHeaderTemplate.bind(this)}> <br /> </ListViewComponent> |
```

```
| HTML attributes | Not Applicable | Property: htmlAttributes <br /><br /> public attributes = {id:
'listid', <br /> class: '.listtest'}; <br /> <br /> <ListViewComponent id='list'
dataSource={this.dataSource} <br /> htmlAttributes={this.attributes}> <br />
</ListViewComponent><br /> |
```

```
| Clear | Method: clear() <br /><br /> <EJ.ListView id='listView' dataSource={this.dataSource}><br
/> </EJ.ListView> <br /><br /> var listObj = $("#default").ejListView("instance"); <br />
listObj.clear(); <br />| Property dataSource <br /> <br /> public emptyDataSrc: { [key: string]:
Object }[] = []; <br /> <br /> <ListViewComponent id='list' dataSource={this.emptyDataSrc}> <br
/> </ListViewComponent> <br /> <br /> listObj.destroy(); <br />|
```

```
| Remove CheckMark | Method: removeCheckMark() <br /><br /> <EJ.ListView id="default"
dataSource={dataSourceItem} <br /> enableCheckMark={true}> <br /> </EJ.ListView> <br /><br />
var listObj = $("#default").ejListView("instance"); <br /> listObj.removeCheckMark(2); <br /> |
Method: uncheckItem() <br /><br /> <ListViewComponent id='list' dataSource={this.dataSource}>
<br /> </ListViewComponent> <br /><br /> listObj.uncheckItem ({id:'2'}); <br />|
```

```
| Set Active | Method: setActive() <br /><br /> <EJ.ListView id="default"
dataSource={dataSourceItem} <br /> persistSelection={true}> <br /> </EJ.ListView> <br /><br />
var listObj = $("#default").ejListView("instance"); <br /> listObj.setActive(2); <br />| Method:
selectItem() <br /> <br /> <ListViewComponent id='list' dataSource={this.dataSource}> <br />
</ListViewComponent> <br /><br /> listObj.selectItem({id:'2'}); <br />|
```

```
| Select | Not Applicable | Event: select <br /> <br /><ListViewComponent id='list'  
dataSource={this.dataSource} <br/> select={this.onSelect}> <br/> </ListViewComponent> <br/>  
<br/> private onSelect(args: Event): void { } <br/> |
```

```
{% endraw %}
```

How To

Get selected items from listview in React Listview component

Single or many items can be selected by users in the ListView component. An API is used to get selected items from the

list items. This is called as the

[getSelectedItems](#)

method.

`getSelectedItems` method

This is used to get the details of the currently selected item from the list items. It returns the

[SelectedItem](#) |

[SelectedCollection](#)

The `getSelectedItems` method returns the following items from the selected list items.

| Return type | Purpose |

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

| text | Returns the text of selected item lists |

| data | Returns the whole data of selected list items, i.e., returns the fields data of selected li.|

| item | Returns the collections of list items |

INDEX.JSX

```
{% raw %}  
import * as React from 'react';  
import * as ReactDOM from "react-dom";  
import { ListViewComponent } from '@syncfusion/ej2-react-lists';  
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';  
function App() {  
  let listobj = null;  
  // define the array of Json  
  let data = [  
    { text: "Hennessey Venom", id: "list-01" },  
    { text: "Bugatti Chiron", id: "list-02", isChecked: true },  
    { text: "Bugatti Veyron Super Sport", id: "list-03" },  
    { text: "SSC Ultimate Aero", id: "list-04", isChecked: true },  
    { text: "Koenigsegg CCR", id: "list-05" },  
    { text: "McLaren F1", id: "list-06" },  
    { text: "Aston Martin One- 77", id: "list-07", isChecked: true },  
    { text: "Jaguar XJ220", id: "list-08" }  
  ];  
  const [state, SetState] = React.useState({  
    selectedItemValue: []  
  });  
}
```

```

function getSelectedItems() {
  if (listobj) {
    SetState({
      selectedItemsValue: listobj.getSelectedItems().data
    });
  }
}

return (<div>
  <ListViewComponent id="list" dataSource={data} showCheckBox={true}
ref={scope => {
    listobj = scope;
  }}/>
  <ButtonComponent id="btn" onClick={getSelectedItems.bind(this)}>
    Get Selected Items
  </ButtonComponent>
  <div>
    <table>
      <tbody>
        <tr>
          <th>Text</th>
          <th>Id</th>
        </tr>
        {state.selectedItemsValue.map((item, index) => {
return (<tr key={index}>
          <td>{item.text}</td>
          <td>{item.id}</td>
        </tr>);
        }}}
      </tbody>
    </table>
  </div>
</div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
  let listobj: ListViewComponent | null = null;
  // define the array of Json
  let data: { [key: string]: Object }[] = [
    { text: "Hennessey Venom", id: "list-01" },
    { text: "Bugatti Chiron", id: "list-02", isChecked: true },
    { text: "Bugatti Veyron Super Sport", id: "list-03" },
    { text: "SSC Ultimate Aero", id: "list-04", isChecked: true },
    { text: "Koenigsegg CCR", id: "list-05" },
    { text: "McLaren F1", id: "list-06" },
    { text: "Aston Martin One- 77", id: "list-07", isChecked: true },
    { text: "Jaguar XJ220", id: "list-08" }
  ]
}

```

```

];
const [state, SetState] = React.useState({
  selectedItemsValue: []
});
function getSelectedItems(): void {
  if (listobj) {
    SetState({
      selectedItemsValue: (listobj.getSelectedItems() as any).data
    });
  }
}
return (
  <div>
    <ListViewComponent
      id="list"
      dataSource={data}
      showCheckBox={true}
      ref={scope => {
        listobj = scope;
      }}
    />
    <ButtonComponent id="btn" onClick={getSelectedItems.bind(this)}>
      Get Selected Items
    </ButtonComponent>
    <div>
      <table>
        <tbody>
          <tr>
            <th>Text</th>
            <th>Id</th>
          </tr>
          {state.selectedItemsValue.map((item: any, index: number) => {
            return (
              <tr key={index}>
                <td>{item.text}</td>
                <td>{item.id}</td>
              </tr>
            );
          })}
        </tbody>
      </table>
    </div>
  </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Use dynamic templates in listview based on device in React Listview component

The Syncfusion Essential JS2 components are desktop and mobile-friendly. So, you can use Syncfusion components in both modes. The component templates are not always fixed. Applications may need to load various templates depending upon the device.

Integration

In the ListView component, template support is being used. In some cases, the component wrapper is always responsive across all devices, but the template contents are dynamically changed with unspecified (sample side) dimensions. CSS customization is also needed in sample-side to align template content responsively in both mobile and desktop modes. Here, two templates have been loaded for mobile and desktop modes. To check the device mode, a **browser module** has been imported from the **ej2-base** package.

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { Browser } from '@syncfusion/ej2-base';
function App() {
  function listTemplate(data) {
    return (<div className="settings">
      <div id="postContainer">
        <div id="postImg">
          <img src={data.image}/>
        </div>
        <div id="content">
          <div className="name">{data.Name}</div>
          <div className="description">{data.content}</div>
          <div id="info">
            <div id="logo">
              <div id="share">
                <span className="share"/>{ " " }
              </div>
              <div id="comments">
                { " " }
                <span className="comments"/>{ " " }
              </div>
              <div id="bookmark">
                { " " }
                <span className="bookmark"/>{ " " }
              </div>
            </div>
            <div className="timeStamp">{data.timeStamp} </div>
          </div>
        </div>
      </div>);
  }
  function mobTemplate(data) {
    return (<div className="settings">
      <div id="postContainer">
        <div id="postImg">
          <img src={data.image}/>
        </div>
        <div id="content">
          <div id="info">
            <div id="logo">
              <div id="share">
                <span className="share"/>{ " " }
              </div>
            </div>
          </div>
        </div>
      </div>);
  }
}
```

```

        </div>
        <div id="comments">
            { " " }
            <span className="comments"/>{ " " }
        </div>
        <div id="bookmark">
            { " " }
            <span className="bookmark"/>{ " " }
        </div>
    </div>
</div>
<div className="name">{data.Name}</div>
<div className="description">{data.content}</div>
<div className="timeStamp">{data.timeStamp} </div>
</div>
</div>
</div>);
}
// define the array of Json
let dataSource = [
    {
        Name: "IBM Open-Sources Web Sphere Liberty Code",
        content: "In September, IBM announced that it would be open-
sourcing the code for WebSphere...",
        id: "1",
        image:
"https://ej2.syncfusion.com/demos/src/listview/images/1.png",
        timeStamp: "Syncfusion Blog - October 19, 2017"
    },
    {
        Name: "Must Reads: 5 Big Data E-books to upend your
development",
        content: "Our first e-book was published in May 2012-jQuery
Succinctly was the start of over...",
        id: "2",
        image:
"https://ej2.syncfusion.com/demos/src/listview/images/2.png",
        timeStamp: "Syncfusion Blog - October 18, 2017"
    },
    {
        Name: "The Syncfusion Global License: Your Questions, Answered
",
        content: "Syncfusion recently hosted a webinar to cover the ins
and outs of the Syncfusion global...",
        id: "4",
        image:
"https://ej2.syncfusion.com/demos/src/listview/images/3.png",
        timeStamp: "Syncfusion Blog - October 18, 2017"
    },
    {
        Name: "Know: What is Coming from Microsoft this Fall ",
        content: "On October 17, Microsoft will release its Fall
Creators Update for the Windows 10 platform...",
        id: "5",
        image:
"https://ej2.syncfusion.com/demos/src/listview/images/6.png",
        timeStamp: "Syncfusion Blog - October 17, 2017"
    }
]

```

```

    }
    ];
    let fields = { text: "Name" };
    let wintemplate;
    let list = null;
    if (Browser.isDevice) {
        if (list) {
            list.element.style.width = "350px";
        }
        wintemplate = mobTemplate;
    }
    else {
        wintemplate = listTemplate;
    }
    return (<div>
        <ListViewComponent id="List" ref={l => { list = l; }}
        dataSource={dataSource} fields={fields} headerTitle="Syncfusion Blog"
        showHeader={true} template={wintemplate}/>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { Browser } from '@syncfusion/ej2-base';
function App() {
    function listTemplate(data: any): JSX.Element {
        return (
            <div className="settings">
                <div id="postContainer">
                    <div id="postImg">
                        <img src={data.image} />
                    </div>
                    <div id="content">
                        <div className="name">{data.Name}</div>
                        <div className="description">{data.content}</div>
                        <div id="info">
                            <div id="logo">
                                <div id="share">
                                    <span className="share" />{ " " }
                                </div>
                                <div id="comments">
                                    { " " }
                                    <span className="comments" />{ " " }
                                </div>
                                <div id="bookmark">
                                    { " " }
                                    <span className="bookmark" />{ " " }
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        );
    }
}

```



```

        <div className="timeStamp">{data.timeStamp} </div>
      </div>
    </div>
  </div>
</div>
);
}
function mobTemplate(data: any): JSX.Element {
  return (
    <div className="settings">
      <div id="postContainer">
        <div id="postImg">
          <img src={data.image} />
        </div>
        <div id="content">
          <div id="info">
            <div id="logo">
              <div id="share">
                <span className="share" />{ " "}
              </div>
              <div id="comments">
                { " "}
                <span className="comments" />{ " "}
              </div>
              <div id="bookmark">
                { " "}
                <span className="bookmark" />{ " "}
              </div>
            </div>
          </div>
          <div className="name">{data.Name}</div>
          <div className="description">{data.content}</div>
          <div className="timeStamp">{data.timeStamp} </div>
        </div>
      </div>
    </div>
  );
}
// define the array of Json
let dataSource: any = [
  {
    Name: "IBM Open-Sources Web Sphere Liberty Code",
    content: "In September, IBM announced that it would be open-sourcing the code for WebSphere...",
    id: "1",
    image: "https://ej2.syncfusion.com/demos/src/listview/images/1.png",
    timeStamp: "Syncfusion Blog - October 19, 2017"
  },
  {
    Name: "Must Reads: 5 Big Data E-books to upend your development",
    content: "Our first e-book was published in May 2012-jQuery Succinctly was the start of over...",
    id: "2",
    image: "https://ej2.syncfusion.com/demos/src/listview/images/2.png",
    timeStamp: "Syncfusion Blog - October 18, 2017"
  },
  {

```

```

        Name: "The Syncfusion Global License: Your Questions, Answered ",
        content: "Syncfusion recently hosted a webinar to cover the ins and
outs of the Syncfusion global...",
        id: "4",
        image: "https://ej2.syncfusion.com/demos/src/listview/images/3.png",
        timeStamp: "Syncfusion Blog - October 18, 2017"
    },
    {
        Name: "Know: What is Coming from Microsoft this Fall ",
        content: "On October 17, Microsoft will release its Fall Creators
Update for the Windows 10 platform...",
        id: "5",
        image: "https://ej2.syncfusion.com/demos/src/listview/images/6.png",
        timeStamp: "Syncfusion Blog - October 17, 2017"
    }
];
let fields = { text: "Name" };
let wintemplate: any;
let list: ListViewComponent | null = null;
if (Browser.isDevice) {
    if (list) {
        list.element.style.width = "350px";
    }
    wintemplate = mobTemplate;
} else {
    wintemplate = listTemplate;
}
return (
    <div>
        <ListViewComponent
            id="List"
            ref={l=>{list=l;}}
            dataSource={dataSource}
            fields={fields}
            headerTitle="Syncfusion Blog"
            showHeader={true}
            template={wintemplate as any}
        />
    </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% enddraw %}

```

Create dual list from listview in React Listview component

The dual list contains two ListView. This allows you to move list items from one list to another using the client-side

events. This section explains how to integrate the ListView component to achieve dual list.

Use cases

- Stock exchanges of two different countries
- Job applications (skill sets)

Integration of Dual List

Here, two ListView components have been used to display the list items. An ej2-button is used to transfer data between

the ListView, and a textbox is used to achieve the UI of filtering support.

The dual list supports:

- Moving whole data from one list to another.
- Moving selected data from one list to another.
- Filtering the list by using a client-side typed character.

In the ListView component, sorting is enabled using the

[sortOrder](#) property, and

the [select](#) event is triggered

while selecting an item. Here, the select event is triggered to enable and disable button states.

Manipulating data

Moving whole data from the first list to the second list(>>)

- Here, the whole data can be moved from the first ListView to the second by clicking the first button. When clicking the button,

the whole list items are sliced, and `concat` with the second ListView. This button is enabled only when the data source

of the first ListView is not empty.

Moving whole data from the second list to the first list(<<)

- The functionality of the second button is the same as above, and data is transferred from the second list to the first

list. This button is enabled only when the data source of the second ListView is not empty.

Moving selected item from one list to another list (>) and (<)

- The [Select](#) event is triggered

when selecting a list item in the ListView. The selected items can be transferred between two lists.

These buttons will be

enabled when selecting an item in lists.

Filtering method

- The filtering method is used to filter list items when typing a character in the text box. In this

method, the [dataManager](#) has been

used to fetch data from the data source and display in ListView.

Sorting

- By using the dual list, list items can be sorted in the ListView component using the

[sortOrder](#) property.

You can enable sorting in one ListView; in the same order, data can be transferred to another ListView.

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { DataManager, Query } from '@syncfusion/ej2-data';
import { enableRipple } from "@syncfusion/ej2-base";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
enableRipple(true);
function App() {
  React.useEffect(() => {
    componentDidMount();
  }, []);
  //Define an array of JSON data
  let firstListData = [
    { 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" }
  ];
  let secondListData = [
    { 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 firstInput = null;
  let secondInput = null;
  let listFirstInstance = null;
  let listSecondInstance = null;
  let firstBtnInstance = null;
  let secondBtnInstance = null;
  let thirdBtnInstance = null;
  let fourthBtnInstance = null;
  function componentDidMount() {
    if (listFirstInstance && listSecondInstance) {
      firstListData = listFirstInstance.dataSource.slice();
      secondListData = listSecondInstance.dataSource.slice();
    }
  }
  //Map the appropriate columns to fields property
  let fields = { text: "text", id: "id" };
  function onFirstListSelect() {
    if (secondBtnInstance)
      secondBtnInstance.disabled = false;
  }
}
```

```

function onSecondListSelect() {
    if (thirdBtnInstance)
        thirdBtnInstance.disabled = false;
}
function updateFirstListData() {
    if (listFirstInstance) {
        Array.prototype.forEach.call(listFirstInstance.liCollection,
(list) => {
            firstListData.forEach((data, index) => {
                if (list.innerText.trim() === data.text) {
                    delete firstListData[index];
                }
            });
        });
        if (firstInput) {
            firstInput.value = "";
        }
        let ds = [];
        firstListData.forEach(data => {
            ds.push(data);
        });
        firstListData = ds;
    }
}
function updateSecondListData() {
    Array.prototype.forEach.call(listSecondInstance.liCollection, (list)
=> {
        secondListData.forEach((data, index) => {
            if (list.innerText.trim() === data.text) {
                delete secondListData[index];
            }
        });
    });
    if (secondInput) {
        secondInput.value = "";
    }
    let ds = [];
    secondListData.forEach(data => {
        ds.push(data);
    });
    secondListData = ds;
}
function onFirstKeyUp(e) {
    if (firstInput) {
        let value = firstInput.value;
        var data = new DataManager(firstListData).executeLocal(new
Query().where("text", "startswith", value, true));
        if (listFirstInstance) {
            if (!value) {
                listFirstInstance.dataSource = firstListData.slice();
            }
            else {
                listFirstInstance.dataSource = data;
            }
            listFirstInstance.dataBind();
        }
    }
}

```

```

    }
    function onSecondKeyUp(e) {
        if (secondInput) {
            let value = secondInput.value;
            var data = new DataManager(secondListData).executeLocal(new
Query().where("text", "startswith", value, true));
            if (listSecondInstance) {
                if (!value) {
                    listSecondInstance.dataSource = secondListData.slice();
                }
                else {
                    listSecondInstance.dataSource = data;
                }
                listSecondInstance.dataBind();
            }
        }
    }
    function setButtonState() {
        if (listFirstInstance &&
            firstBtnInstance &&
            secondBtnInstance &&
            listSecondInstance &&
            fourthBtnInstance &&
            thirdBtnInstance) {
            if (listFirstInstance.dataSource.length) {
                firstBtnInstance.disabled = false;
            }
            else {
                firstBtnInstance.disabled = true;
                secondBtnInstance.disabled = true;
            }
            if (listSecondInstance.dataSource.length) {
                fourthBtnInstance.disabled = false;
            }
            else {
                fourthBtnInstance.disabled = true;
                thirdBtnInstance.disabled = true;
            }
        }
    }
    function firstBtnClick() {
        if (listFirstInstance && firstBtnInstance && listSecondInstance) {
            listSecondInstance.dataSource =
Array.prototype.concat.call(listFirstInstance.dataSource,
listSecondInstance.dataSource);
            updateFirstListData();

listFirstInstance.removeMultipleItems(listFirstInstance.liCollection);
            firstListData =
firstListData.concat(listFirstInstance.dataSource);
            secondListData = listSecondInstance.dataSource.slice();
            firstBtnInstance.disabled = true;
            onFirstKeyUp(null);
            setButtonState();
        }
    }
    function secondBtnClick() {

```

```

        if (listFirstInstance && secondBtnInstance && listSecondInstance) {
            let e = listFirstInstance.getSelectedItems();
            if (e) {
                listSecondInstance.dataSource =
Array.prototype.concat.call(listSecondInstance.dataSource, e.data);
                listFirstInstance.removeItem(e.item);
                firstListData = listFirstInstance.dataSource;
                secondListData = listSecondInstance.dataSource.slice();
                onFirstKeyUp(null);
                secondBtnInstance.disabled = true;
                setButtonState();
            }
        }
    }
    function thirdBtnClick() {
        if (listFirstInstance && listSecondInstance && thirdBtnInstance) {
            let e = listSecondInstance.getSelectedItems();
            if (e) {
                listFirstInstance.dataSource =
Array.prototype.concat.call(listFirstInstance.dataSource, e.data);
                listSecondInstance.removeItem(e.item);
                secondListData = listSecondInstance.dataSource;
                firstListData = listFirstInstance.dataSource.slice();
                onSecondKeyUp(null);
                thirdBtnInstance.disabled = true;
                setButtonState();
            }
        }
    }
    function fourthBtnClick() {
        if (listFirstInstance && listSecondInstance && fourthBtnInstance) {
            listFirstInstance.dataSource =
Array.prototype.concat.call(listFirstInstance.dataSource,
listSecondInstance.dataSource);
            updateSecondListData();

listSecondInstance.removeMultipleItems(listSecondInstance.liCollection);
            secondListData =
secondListData.concat(listSecondInstance.dataSource);
            firstListData = listFirstInstance.dataSource.slice();
            onSecondKeyUp(null);
            setButtonState();
        }
    }
    function firstInputKeyUp() {
        if (listFirstInstance &&
            firstBtnInstance &&
            secondBtnInstance &&
            listSecondInstance &&
            fourthBtnInstance &&
            thirdBtnInstance &&
            firstInput) {
            let value = firstInput.value;
            var data = new DataManager(firstListData).executeLocal(new
Query().where("text", "startswith", value, true));
            if (!value) {
                listFirstInstance.dataSource = firstListData.slice();
            }
        }
    }

```

```

    }
    else {
        listFirstInstance.dataSource = data;
    }
    listFirstInstance.dataBind();
}
}
function secondInputKeyUp() {
    if (listFirstInstance &&
        firstBtnInstance &&
        secondBtnInstance &&
        listSecondInstance &&
        fourthBtnInstance &&
        thirdBtnInstance &&
        secondInput) {
        let value = secondInput.value;
        var data = new DataManager(secondListData).executeLocal(new
Query().where("text", "startswith", value, true));
        if (!value) {
            listSecondInstance.dataSource = secondListData.slice();
        }
        else {
            listSecondInstance.dataSource = data;
        }
        listSecondInstance.dataBind();
    }
}
return (<div id="container">
    <div className="col-lg-12 control-section">
        <div>
            <h3>Dual List</h3>
            <div id="text1">
                <input className="e-input" type="text" id="firstInput"
ref={(inputRef) => (firstInput = inputRef)} placeholder="Filter"
onKeyUp={firstInputKeyUp.bind(this)} title="Type in a name"/>
            </div>
            <ListViewComponent id="list-1" dataSource={firstListData}
fields={fields} sortOrder="Ascending" select={onFirstListSelect.bind(this)}
ref={scope => {
                listFirstInstance = scope;
            }}/>
            <div id="btn">
                <ButtonComponent onClick={firstBtnClick.bind(this)} ref={button1
=> {
                    firstBtnInstance = button1;
                }}>
                    { " " }
                    { ">>" }
                </ButtonComponent>
                <ButtonComponent onClick={secondBtnClick.bind(this)}
ref={button2 => {
                    secondBtnInstance = button2;
                }}>
                    { " " }
                    { ">>" }
                </ButtonComponent>

```



```

=> {
    <ButtonComponent onClick={thirdBtnClick.bind(this) } ref={button3}
    thirdBtnInstance = button3;
  }>
    { " " }
    { "<" } { " " }
  </ButtonComponent>
  <ButtonComponent onClick={fourthBtnClick.bind(this) }
ref={button4} => {
    fourthBtnInstance = button4;
  }>
    { " " }
    { "<<" } { " " }
  </ButtonComponent>
</div>
<div id="text2">
  <input className="e-input" type="text"
onKeyUp={secondInputKeyUp.bind(this) } id="secondInput" ref={(inputRef) =>
(secondInput = inputRef)} placeholder=" Filter" title="Type in a name"/>
</div>
<ListViewComponent id="list-2" dataSource={secondListData}
fields={fields} sortOrder="Ascending" select={onSecondListSelect.bind(this) }
ref={list => {
  listSecondInstance = list;
}}/>
</div>
</div>
</div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { DataManager, Query } from '@syncfusion/ej2-data';
import { enableRipple } from "@syncfusion/ej2-base";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
enableRipple(true);
function App() {
  React.useEffect(() => {
    componentDidMount();
  }, []);
  //Define an array of JSON data
  let firstListData: any[] = [
    { 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" }
  ];
}

```

```

let secondListData: any[] = [
  { 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 firstInput: HTMLInputElement | null = null;
let secondInput: HTMLInputElement | null = null;
let listFirstInstance: ListViewComponent | null = null;
let listSecondInstance: ListViewComponent | null = null;
let firstBtnInstance: ButtonComponent | null = null;
let secondBtnInstance: ButtonComponent | null = null;
let thirdBtnInstance: ButtonComponent | null = null;
let fourthBtnInstance: ButtonComponent | null = null;
function componentDidMount() {
  if (listFirstInstance && listSecondInstance) {
    firstListData = (listFirstInstance.dataSource as any[]).slice();
    secondListData = (listSecondInstance.dataSource as any[]).slice();
  }
}
//Map the appropriate columns to fields property
let fields: Object = { text: "text", id: "id" };
function onFirstListSelect() {
  if (secondBtnInstance) secondBtnInstance.disabled = false;
}
function onSecondListSelect() {
  if (thirdBtnInstance) thirdBtnInstance.disabled = false;
}
function updateFirstListData() {
  if (listFirstInstance) {
    Array.prototype.forEach.call(
      (listFirstInstance as any).liCollection as any[],
      (list: any) => {
        firstListData.forEach((data, index) => {
          if (list.innerText.trim() === data.text) {
            delete firstListData[index];
          }
        });
      }
    );
    if (firstInput) {
      firstInput.value = "";
    }
    let ds: any[] = [];
    firstListData.forEach(data => {
      ds.push(data);
    });
    firstListData = ds;
  }
}
function updateSecondListData() {
  Array.prototype.forEach.call((listSecondInstance as any).liCollection,
    (list: any) => {
      secondListData.forEach((data, index) => {
        if (list.innerText.trim() === data.text) {
          delete secondListData[index];
        }
      }
    )
  );
}

```

```

    });
    });
    if (secondInput) {
        secondInput.value = "";
    }
    let ds: any[] = [];
    secondListData.forEach(data => {
        ds.push(data);
    });
    secondListData = ds;
}
function onFirstKeyUp(e: any) {
    if (firstInput) {
        let value = firstInput.value;
        var data = new DataManager(firstListData).executeLocal(
            new Query().where("text", "startswith", value, true)
        );
        if (listFirstInstance) {
            if (!value) {
                listFirstInstance.dataSource = firstListData.slice();
            } else {
                listFirstInstance.dataSource = data as any[];
            }
            listFirstInstance.dataBind();
        }
    }
}
function onSecondKeyUp(e: any) {
    if (secondInput) {
        let value = secondInput.value;
        var data = new DataManager(secondListData).executeLocal(
            new Query().where("text", "startswith", value, true)
        );
        if (listSecondInstance) {
            if (!value) {
                listSecondInstance.dataSource = secondListData.slice();
            } else {
                listSecondInstance.dataSource = data as any[];
            }
            listSecondInstance.dataBind();
        }
    }
}
function setButtonState() {
    if (
        listFirstInstance &&
        firstBtnInstance &&
        secondBtnInstance &&
        listSecondInstance &&
        fourthBtnInstance &&
        thirdBtnInstance
    ) {
        if ((listFirstInstance.dataSource as any[]).length) {
            firstBtnInstance.disabled = false;
        } else {
            firstBtnInstance.disabled = true;
            secondBtnInstance.disabled = true;
        }
    }
}

```

```

    }
    if ((listSecondInstance.dataSource as any[]).length) {
        fourthBtnInstance.disabled = false;
    } else {
        fourthBtnInstance.disabled = true;
        thirdBtnInstance.disabled = true;
    }
}
}
function firstBtnClick() {
    if (listFirstInstance && firstBtnInstance && listSecondInstance) {
        listSecondInstance.dataSource = Array.prototype.concat.call(
            listFirstInstance.dataSource,
            listSecondInstance.dataSource
        );
        updateFirstListData();
        listFirstInstance.removeMultipleItems((listFirstInstance as
any).liCollection);
        firstListData = firstListData.concat(listFirstInstance.dataSource);
        secondListData = listSecondInstance.dataSource.slice();
        firstBtnInstance.disabled = true;
        onFirstKeyUp(null);
        setButtonState();
    }
}
function secondBtnClick() {
    if (listFirstInstance && secondBtnInstance && listSecondInstance) {
        let e = listFirstInstance.getSelectedItems();
        if (e) {
            listSecondInstance.dataSource = Array.prototype.concat.call(
                listSecondInstance.dataSource,
                e.data
            );
            listFirstInstance.removeItem(e.item as any);
            firstListData = listFirstInstance.dataSource as any[];
            secondListData = listSecondInstance.dataSource.slice();
            onFirstKeyUp(null);
            secondBtnInstance.disabled = true;
            setButtonState();
        }
    }
}
function thirdBtnClick() {
    if (listFirstInstance && listSecondInstance && thirdBtnInstance) {
        let e = listSecondInstance.getSelectedItems();
        if (e) {
            listFirstInstance.dataSource = Array.prototype.concat.call(
                listFirstInstance.dataSource,
                e.data
            );
            listSecondInstance.removeItem(e.item as any);
            secondListData = listSecondInstance.dataSource as any[];
            firstListData = listFirstInstance.dataSource.slice();
            onSecondKeyUp(null);
            thirdBtnInstance.disabled = true;
            setButtonState();
        }
    }
}

```

```

    }
  }
  function fourthBtnClick() {
    if (listFirstInstance && listSecondInstance && fourthBtnInstance) {
      listFirstInstance.dataSource = Array.prototype.concat.call(
        listFirstInstance.dataSource,
        listSecondInstance.dataSource
      );
      updateSecondListData();
      listSecondInstance.removeMultipleItems((listSecondInstance as
any).liCollection);
      secondListData = secondListData.concat(listSecondInstance.dataSource);
      firstListData = listFirstInstance.dataSource.slice();
      onSecondKeyUp(null);
      setButtonState();
    }
  }
  function firstInputKeyUp() {
    if (
      listFirstInstance &&
      firstBtnInstance &&
      secondBtnInstance &&
      listSecondInstance &&
      fourthBtnInstance &&
      thirdBtnInstance &&
      firstInput
    ) {
      let value = firstInput.value;
      var data = new DataManager(firstListData).executeLocal(
        new Query().where("text", "startswith", value, true)
      );
      if (!value) {
        listFirstInstance.dataSource = firstListData.slice();
      } else {
        listFirstInstance.dataSource = data as any[];
      }
      listFirstInstance.dataBind();
    }
  }
  function secondInputKeyUp() {
    if (
      listFirstInstance &&
      firstBtnInstance &&
      secondBtnInstance &&
      listSecondInstance &&
      fourthBtnInstance &&
      thirdBtnInstance &&
      secondInput
    ) {
      let value = secondInput.value;
      var data = new DataManager(secondListData).executeLocal(
        new Query().where("text", "startswith", value, true)
      );
      if (!value) {
        listSecondInstance.dataSource = secondListData.slice();
      } else {
        listSecondInstance.dataSource = data as any[];
      }
    }
  }

```

```

    }
    listSecondInstance.dataBind();
  }
}
return (
  <div id="container">
    <div className="col-lg-12 control-section">
      <div>
        <h3>Dual List</h3>
        <div id="text1">
          <input
            className="e-input"
            type="text"
            id="firstInput"
            ref={(inputRef: any) => (firstInput = inputRef)}
            placeholder="Filter"
            onKeyUp={firstInputKeyUp.bind(this)}
            title="Type in a name"
          />
        </div>
        <ListViewComponent
          id="list-1"
          dataSource={firstListData}
          fields={fields}
          sortOrder="Ascending"
          select={onFirstListSelect.bind(this)}
          ref={scope => {
            listFirstInstance = scope;
          }}
        />
        <div id="btn">
          <ButtonComponent
            onClick={firstBtnClick.bind(this)}
            ref={button1 => {
              firstBtnInstance = button1;
            }}
          >
            { " " }
            { ">>" }
          </ButtonComponent>
          <ButtonComponent
            onClick={secondBtnClick.bind(this)}
            ref={button2 => {
              secondBtnInstance = button2;
            }}
          >
            { " " }
            { ">" }
          </ButtonComponent>
          <ButtonComponent
            onClick={thirdBtnClick.bind(this)}
            ref={button3 => {
              thirdBtnInstance = button3;
            }}
          >
            { " " }
            { "<" } { " " }

```

```

        </ButtonComponent>
        <ButtonComponent
            onClick={fourthBtnClick.bind(this) }
            ref={button4 => {
                fourthBtnInstance = button4;
            }}
        >
            { " " }
            { "<<" } { " " }
        </ButtonComponent>
    </div>
    <div id="text2">
        <input
            className="e-input"
            type="text"
            onKeyUp={secondInputKeyUp.bind(this) }
            id="secondInput"
            ref={(inputRef: any) => (secondInput = inputRef)}
            placeholder="  Filter"
            title="Type in a name"
        />
    </div>
    <ListViewComponent
        id="list-2"
        dataSource={secondListData}
        fields={fields}
        sortOrder="Ascending"
        select={onSecondListSelect.bind(this) }
        ref={list => {
            listSecondInstance = list;
        }}
    />
</div>
</div>
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% enddraw %}

```

Load list items in child list dynamically in React Listview component

To load list items in child list dynamically, push the new list item data into the existing [dataSource](#) using the [select](#) event.

Refer to the following steps to load list item into the child list:

1. Initially, render the ListView with the required data source.
2. Bind the [select](#) event that triggers selecting list item in the ListView component. By using the select event, you can push the new list item to the child list of the data source on specifying its item index. Item index can be obtained from the [SelectEventArgs](#) of the select event.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    // define the array of Json
    let data = [
        {
            id: "01",
            text: "Music",
            icon: "folder",
            child: [{ id: "01-01", text: "Gouttes.mp3", icon: "file" }]
        },
        {
            id: "02",
            text: "Videos",
            icon: "folder",
            child: [
                { id: "02-01", text: "Naturals.mp4", icon: "file" },
                { id: "02-02", text: "Wild.mpeg", icon: "file" }
            ]
        },
        {
            id: "03",
            text: "Documents",
            icon: "folder",
            child: [
                { id: "03-01", text: "Environment Pollution.docx", icon:
"file" },
                { id: "03-02", text: "Global Water, Sanitation, &
Hygiene.docx", icon: "file" },
                { id: "03-03", text: "Global Warming.ppt", icon: "file" },
                { id: "03-04", text: "Social Network.pdf", icon: "file" },
                { id: "03-05", text: "Youth Empowerment.pdf", icon: "file" }
            ]
        },
        {
            id: "04",
            text: "Pictures",
            icon: "folder",
            child: [
                {
                    id: "04-01",
                    text: "Camera Roll",
                    icon: "folder",
                    child: [
                        { id: "04-01-01", text: "WIN_20160726_094117.JPG",
icon: "file" },
                        { id: "04-01-02", text: "WIN_20160726_094118.JPG",
icon: "file" },
                        { id: "04-01-03", text: "WIN_20160726_094119.JPG",
icon: "file" }
                    ]
                },
                {
                    id: "04-02",
                    text: "Wind.jpg",
                    icon: "file"
                }
            ]
        }
    ]
}

```



```

        },
        {
            id: "04-02",
            text: "Stone.jpg",
            icon: "file"
        },
        {
            id: "04-02",
            text: "Home.jpg",
            icon: "file"
        },
        {
            id: "04-02",
            text: "Bridge.png",
            icon: "file"
        }
    ]
},
{
    id: "05",
    text: "Downloads",
    icon: "folder",
    child: [
        { id: "05-01", text: "UI-Guide.pdf", icon: "file" },
        { id: "05-02", text: "Tutorials.zip", icon: "file" },
        { id: "05-03", text: "Game.exe", icon: "file" },
        { id: "05-04", text: "TypeScript.7z", icon: "file" }
    ]
}
];
let fields = { iconCss: "icon", tooltip: "text" };
function onSelect(args) {
    data[args.index].child.push({
        id: "01-02",
        text: "Newly Added File",
        icon: "file",
        htmlAttributes: { role: "li", class: "list" }
    });
}
return (<div>
    <ListViewComponent id="listview" dataSource={data} fields={fields}
    headerTitle="Folders" showIcon={true} showHeader={true}
    select={onSelect.bind(this)} />
</div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    // define the array of Json
    let data: { [key: string]: Object }[] = [

```

```

{
  id: "01",
  text: "Music",
  icon: "folder",
  child: [{ id: "01-01", text: "Gouttes.mp3", icon: "file" }]
},
{
  id: "02",
  text: "Videos",
  icon: "folder",
  child: [
    { id: "02-01", text: "Naturals.mp4", icon: "file" },
    { id: "02-02", text: "Wild.mpeg", icon: "file" }
  ]
},
{
  id: "03",
  text: "Documents",
  icon: "folder",
  child: [
    { id: "03-01", text: "Environment Pollution.docx", icon: "file" },
    { id: "03-02", text: "Global Water, Sanitation, & Hygiene.docx",
icon: "file" },
    { id: "03-03", text: "Global Warming.ppt", icon: "file" },
    { id: "03-04", text: "Social Network.pdf", icon: "file" },
    { id: "03-05", text: "Youth Empowerment.pdf", icon: "file" }
  ]
},
{
  id: "04",
  text: "Pictures",
  icon: "folder",
  child: [
    {
      id: "04-01",
      text: "Camera Roll",
      icon: "folder",
      child: [
        { id: "04-01-01", text: "WIN_20160726_094117.JPG", icon: "file" },
        { id: "04-01-02", text: "WIN_20160726_094118.JPG", icon: "file" },
        { id: "04-01-03", text: "WIN_20160726_094119.JPG", icon: "file" }
      ]
    },
    {
      id: "04-02",
      text: "Wind.jpg",
      icon: "file"
    },
    {
      id: "04-02",
      text: "Stone.jpg",
      icon: "file"
    }
  ]
}

```

```

        id: "04-02",
        text: "Home.jpg",
        icon: "file"
    },
    {
        id: "04-02",
        text: "Bridge.png",
        icon: "file"
    }
]
},
{
    id: "05",
    text: "Downloads",
    icon: "folder",
    child: [
        { id: "05-01", text: "UI-Guide.pdf", icon: "file" },
        { id: "05-02", text: "Tutorials.zip", icon: "file" },
        { id: "05-03", text: "Game.exe", icon: "file" },
        { id: "05-04", text: "TypeScript.7z", icon: "file" }
    ]
}
];
let fields: { [key: string]: Object } = { iconCss: "icon", tooltip: "text"
};
function onSelect(args: SelectEventArgs) {
    (data[args.index].child as any[]).push({
        id: "01-02",
        text: "Newly Added File",
        icon: "file",
        htmlAttributes: { role: "li", class: "list" }
    });
}
return (
    <div>
        <ListViewComponent
            id="listview"
            dataSource={data}
            fields={fields}
            headerTitle="Folders"
            showIcon={true}
            showHeader={true}
            select={onSelect.bind(this) as any}
        />
    </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Customize listview with dynamic tags in React Listview component

You can customize the ListView items using the [template](#) property. Here, the dynamic tags are added and removed in the list item from another ListView. Refer to the following steps to achieve this.

- Render the ListView with data source, and add button element with each list item of ListView on [actionComplete](#) event.

Refer to the following code sample of actionComplete event.

```
`ts
// The actionComplete event for first ListView to add the button
function addButton(args) {
  let buttonObj = { obj: ButtonComponent, prop: { iconCss: 'e-icons e-add-icon', cssClass: 'e-small e-round'
  } };
  let ele = document.getElementsByClassName("e-but");
  for (let i: number = 0; i < ele.length; i++) {
    buttonObj.obj = new ButtonComponent(buttonObj.prop);
    buttonObj.obj.appendTo(ele[i]);
  }
}
`ts
// The actionComplete event for first ListView to add the button
function addButton(args) {
  let buttonObj = { obj: ButtonComponent, prop: { iconCss: 'e-icons e-add-icon', cssClass: 'e-small e-round'
  } };
  let ele = document.getElementsByClassName("e-but");
  for (let i = 0; i < ele.length; i++) {
    buttonObj.obj = new ButtonComponent(buttonObj.prop);
    buttonObj.obj.appendTo(ele[i]);
  }
}
`ts
```

- Initialize dynamic ListView with required property that holds the tags of parent ListView, and bind the [select](#) event (triggers when the list item is selected), in which you can get and add the selected item value as tags into parent ListView. Refer to the following code sample.

```
`ts
//Select the event that is rendered inside dialog for ListView
function addTag(e) {
```

```

let listTag = document.createElement('span');
listTag.className = 'advanced-option';
let labelElem = document.createElement('span');
labelElem.className = 'label';
let deleteElem = document.createElement('span');
deleteElem.className = 'delete';
deleteElem.onclick = removeTag;
labelElem.innerHTML = e.target.textContent;
listTag.appendChild(labelElem);
listTag.appendChild(deleteElem);
let tag = document.createElement('span');
tag.className = 'advanced-option-list';
tag.appendChild(listTag);
listviewInstance.element.querySelector('.e-active').appendChild(tag);
}
,

`ts
//Select the event that is rendered inside dialog for ListView
function addTag(e) {
let listTag = document.createElement('span');
listTag.className = 'advanced-option';
let labelElem = document.createElement('span');
labelElem.className = 'label';
let deleteElem = document.createElement('span');
deleteElem.className = 'delete';
deleteElem.onclick = removeTag;
labelElem.innerHTML = e.target.textContent;
listTag.appendChild(labelElem);
listTag.appendChild(deleteElem);
let tag = document.createElement('span');
tag.className = 'advanced-option-list';
tag.appendChild(listTag);
listviewInstance.element.querySelector('.e-active').appendChild(tag);

```

```
}
`ts
```

- Render the dialog component with empty content and append the created dynamic ListView object to the dialog on [created](#) event.
- Bind the click event for button icon (+) to update the ListView data source with tags, and open the dialog with this dynamic ListView. Refer to the following code sample.

```
`ts
//Method to hide/show the dialog and update the ListView data source
function renderDialog(id) {
  if (document.getElementsByClassName('e-popup-open').length !== 0) {
    dialogInstance.hide();
  }
  else {
    let listElem: any = document.getElementById('dialog').querySelector("#list");
    let listIns = document.getElementById('dialog').querySelector("#list") &&
      document.getElementById('dialog').querySelector("#list").ej2instances &&
      document.getElementById('dialog').querySelector("#list").ej2instances[0] ?
      document.getElementById('dialog').querySelector("#list").ej2_instances[0] : undefined;
    if(listIns){
      listIns.dataSource = datasource[id];
      listIns.fields = fields;
      listIns.addEventListener('select', ()=> { addTag(event);});
      listIns.dataBind();
      listIns.appendTo(listElem);
      dialogInstance.position = { X: document.querySelector('.e-add-icon').getBoundingClientRect().left + 50,
        Y: document.querySelector('.e-add-icon').getBoundingClientRect().top - 5 };
      dialogInstance.show();
    }
  }
}
`ts
```

```
//Method to hide/show the dialog and update the ListView data source
function renderDialog(id) {
  if (document.getElementsByClassName('e-popup-open').length !== 0) {
```

```

dialogInstance.hide();
}
else {
let listElem = document.getElementById('dialog').querySelector("#list");
let listIns = document.getElementById('dialog').querySelector("#list") &&
document.getElementById('dialog').querySelector("#list").ej2_instances &&
document.getElementById('dialog').querySelector("#list").ej2_instances[0] ?
document.getElementById('dialog').querySelector("#list").ej2_instances[0] : undefined;
if (listIns) {
listIns.dataSource = datasource[id];
listIns.fields = fields;
listIns.addEventListener('select', () => { addTag(event); });
listIns.dataBind();
listIns.appendTo(listElem);
dialogInstance.position = { X: document.querySelector('.e-add-icon').getBoundingClientRect().left + 50,
Y: document.querySelector('.e-add-icon').getBoundingClientRect().top - 5 };
dialogInstance.show();
}
}
}
`

```

- Bind the click event with added dynamic tags to remove it. Refer to the following code sample.

```

`ts
//Method to remove the list item
function removeTag() {
parentNode.parentNode.remove();
}
`

```

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import { EventHandler } from "@syncfusion/ej2-base";
function App() {

```

```

//Define an array of JSON data
let data = [
  { Id: "Brooke", Name: "Brooke" },
  { Id: "Claire", Name: "Claire" },
  { Id: "Erik", Name: "Erik" },
  { Id: "Grace", Name: "Grace" },
  { Id: "Jacob", Name: "Jacob" }
];
let fields = { text: "Name" };
let position = null;
let animation = { effect: "None" };
let dialogInstance = null;
let listViewInstance = null;
let brookeTag = [
  { id: "list11", Name: "Discover Music" },
  { id: "list12", Name: "Sales and Events" },
  { id: "list13", Name: "Categories" },
  { id: "list14", Name: "MP3 Albums" },
  { id: "list15", Name: "More in Music" }
];
let claireTag = [
  { id: "list21", Name: "Songs" },
  { id: "list22", Name: "Bestselling Albums" },
  { id: "list23", Name: "New Releases" },
  { id: "list24", Name: "Bestselling Songs" }
];
let erikTag = [
  { id: "list31", Name: "Artwork" },
  { id: "list32", Name: "Abstract" },
  { id: "list33", Name: "Acrylic Mediums" },
  { id: "list34", Name: "Creative Acrylic" },
  { id: "list35", Name: "Canvas Art" }
];
let graceTag = [
  { id: "list41", Name: "Rock" },
  { id: "list42", Name: "Gospel" },
  { id: "list43", Name: "Latin Music" },
  { id: "list44", Name: "Jazz" }
];
let jacobTag = [
  { id: "list51", Name: "100 Albums" },
  { id: "list52", Name: "Hip-Hop and R&B Sale" },
  { id: "list53", Name: "CD Deals" }
];
let datasource = {
  Brooke: brookeTag,
  Claire: claireTag,
  Erik: erikTag,
  Grace: graceTag,
  Jacob: jacobTag
};
let dialogListInstance;
function renderDialog(id) {
  if (document.getElementsByClassName("e-popup-open").length !== 0) {
    if (dialogInstance)
      dialogInstance.hide();
  }
}

```



```

        if (dialogInstance) {
            let listElem = dialogInstance.element.querySelector("#list");
            dialogListInstance =
                listElem && listElem.ej2_instances &&
listElem.ej2_instances[0]
                ? listElem.ej2_instances[0]
                : undefined;
            if (dialogListInstance) {
                EventHandler.remove(dialogListInstance, "select", addTag);
                dialogListInstance.dataSource = datasource[id];
                dialogListInstance.fields = fields;
                EventHandler.add(dialogListInstance, "select", addTag,
this);

                dialogListInstance.appendTo(listElem);
                dialogListInstance.dataBind();
                dialogInstance.position = {
                    X: listViewInstance.element.querySelector(".e-add-
icon").getBoundingClientRect()
                        .left + 50,
                    Y: listViewInstance.element.querySelector(".e-add-
icon").getBoundingClientRect()
                        .top - 5
                };
                dialogInstance.show();
            }
        }
    }
    function addTag(e) {
        let listTag = document.createElement("span");
        listTag.className = "advanced-option";
        let labelElem = document.createElement("span");
        labelElem.className = "label";
        let deleteElem = document.createElement("span");
        deleteElem.className = "delete";
        deleteElem.onclick = removeTag;
        labelElem.innerHTML = e.text;
        listTag.appendChild(labelElem);
        listTag.appendChild(deleteElem);
        let tag = document.createElement("span");
        tag.className = "advanced-option-list";
        tag.appendChild(listTag);
        listViewInstance.element.querySelector(".e-
active").appendChild(tag);
    }
    function removeTag() {
        this.parentNode.parentNode.remove();
    }
    function handleClick(e) {
        renderDialog(e.currentTarget.id);
    }
    function listtemplate(data) {
        return (<div>
            <span className="templatetext">{data.Name}</span>
            <span className="designationstyle">
                <ButtonComponent id={data.Id} className="e-but e-small e-round"
iconCss={"e-icons e-add-icon"} onClick={handleClick.bind(this)} />
            </span>

```

```

    </div>);
  }
  function dialogContent() {
    return <ListViewComponent id="list" showHeader={true}
headerTitle="Favorite" width={"200px"}/>;
  }
  return (<div id="sample">
    <ListViewComponent id="template-list" dataSource={data}
fields={fields} template={listtemplate.bind(this)} ref={scope => {
    listviewInstance = scope;
  }}/>
    <DialogComponent id="dialog" animationSettings={animation}
content={dialogContent} visible={false} ref={dialog => (dialogInstance =
dialog)} width={"200px"} showCloseIcon={true} position={position}/>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import { Browser, EventHandler } from '@syncfusion/ej2-base';
function App() {
  //Define an array of JSON data
  let data: any[] = [
    { Id: "Brooke", Name: "Brooke" },
    { Id: "Claire", Name: "Claire" },
    { Id: "Erik", Name: "Erik" },
    { Id: "Grace", Name: "Grace" },
    { Id: "Jacob", Name: "Jacob" }
  ];
  let fields: object = { text: "Name" };
  let position: any = null;
  let animation: object = { effect: "None" };
  let dialogInstance: DialogComponent | null = null;
  let listviewInstance: ListViewComponent | null = null;
  let brookeTag: object[] = [
    { id: "list11", Name: "Discover Music" },
    { id: "list12", Name: "Sales and Events" },
    { id: "list13", Name: "Categories" },
    { id: "list14", Name: "MP3 Albums" },
    { id: "list15", Name: "More in Music" }
  ];
  let claireTag: object[] = [
    { id: "list21", Name: "Songs" },
    { id: "list22", Name: "Bestselling Albums" },
    { id: "list23", Name: "New Releases" },
    { id: "list24", Name: "Bestselling Songs" }
  ];
}

```

```

let erikTag: object[] = [
    { id: "list31", Name: "Artwork" },
    { id: "list32", Name: "Abstract" },
    { id: "list33", Name: "Acrylic Mediums" },
    { id: "list34", Name: "Creative Acrylic" },
    { id: "list35", Name: "Canvas Art" }
];
let graceTag: object[] = [
    { id: "list41", Name: "Rock" },
    { id: "list42", Name: "Gospel" },
    { id: "list43", Name: "Latin Music" },
    { id: "list44", Name: "Jazz" }
];
let jacobTag: object[] = [
    { id: "list51", Name: "100 Albums" },
    { id: "list52", Name: "Hip-Hop and R&B Sale" },
    { id: "list53", Name: "CD Deals" }
];
let datasource: any = {
    Brooke: brookeTag,
    Claire: claireTag,
    Erik: erikTag,
    Grace: graceTag,
    Jacob: jacobTag
};
let dialogListInstance: any;
function renderDialog(id: any) {
    if (document.getElementsByClassName("e-popup-open").length !== 0) {
        if (dialogInstance) dialogInstance.hide();
    }
    if (dialogInstance) {
        let listElem: any = dialogInstance.element.querySelector("#list");
        dialogListInstance =
            listElem && listElem.ej2_instances && listElem.ej2_instances[0]
                ? listElem.ej2_instances[0]
                : undefined;
        if (dialogListInstance) {
            EventHandler.remove(dialogListInstance, "select", addTag);
            dialogListInstance.dataSource = datasource[id];
            dialogListInstance.fields = fields;
            EventHandler.add(dialogListInstance, "select", addTag, this);
            dialogListInstance.appendTo(listElem);
            dialogListInstance.dataBind();
            dialogInstance.position = {
                X:
                    (listviewInstance as any).element.querySelector(".e-add-
icon").getBoundingClientRect()
                        .left + 50,
                Y:
                    (listviewInstance as any).element.querySelector(".e-add-
icon").getBoundingClientRect()
                        .top - 5
            };
            dialogInstance.show();
        }
    }
}

```

```

function addTag(e: any) {
    let listTag = document.createElement("span");
    listTag.className = "advanced-option";
    let labelElem = document.createElement("span");
    labelElem.className = "label";
    let deleteElem = document.createElement("span");
    deleteElem.className = "delete";
    deleteElem.onclick = removeTag;
    labelElem.innerHTML = e.text;
    listTag.appendChild(labelElem);
    listTag.appendChild(deleteElem);
    let tag = document.createElement("span");
    tag.className = "advanced-option-list";
    tag.appendChild(listTag);
    (listviewInstance as any).element.querySelector(".e-
active").appendChild(tag);
}
function removeTag() {
    (this as any).parentNode.parentNode.remove();
}
function handleClick(e: any) {
    renderDialog(e.currentTarget.id);
}
function listtemplate(data: any): JSX.Element {
    return (
        <div>
            <span className="templatetext">{data.Name}</span>
            <span className="designationstyle">
                <ButtonComponent
                    id={data.Id}
                    className="e-but e-small e-round"
                    iconCss={"e-icons e-add-icon"}
                    onClick={handleClick.bind(this)}
                />
            </span>
        </div>
    );
}
function dialogContent(): JSX.Element {
    return <ListViewComponent id="list" showHeader={true}
headerTitle="Favorite" width={"200px"} />;
}
return (
    <div id="sample">
        <ListViewComponent
            id="template-list"
            dataSource={data}
            fields={fields}
            template={listtemplate.bind(this) as any}
            ref={scope => {
                listviewInstance = scope;
            }}
        />
        <DialogComponent
            id="dialog"
            animationSettings={animation as any}
            content={dialogContent as any}

```

```

        visible={false}
        ref={dialog => (dialogInstance = dialog)}
        width={"200px"}
        showCloseIcon={true}
        position={position}
      />
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Show items count in group header in React Listview component

The ListView component supports wrapping list items into a group based on the category. The category of each list item can

be mapped with `groupBy` field of the data source. You can display grouped list items count in the list-header using the group

header template. Refer to the following code sample to display grouped list item count.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  //Define an array of JSON data
  let data = [
    { Name: 'Nancy', contact: '(206) 555-985774', id: '1', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/1.png', category:
    'Experience' },
    { Name: 'Janet', contact: '(206) 555-3412', id: '2', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/3.png', category:
    'Fresher' },
    { Name: 'Margaret', contact: '(206) 555-8122', id: '4', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/4.png', category:
    'Experience' },
    { Name: 'Andrew ', contact: '(206) 555-9482', id: '5', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/2.png', category:
    'Experience' },
    { Name: 'Steven', contact: '(71) 555-4848', id: '6', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/5.png', category:
    'Fresher' },
    { Name: 'Michael', contact: '(71) 555-7773', id: '7', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/6.png', category:
    'Experience' },
    { Name: 'Robert', contact: '(71) 555-5598', id: '8', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/7.png', category:
    'Fresher' },
    { Name: 'Laura', contact: '(206) 555-1189', id: '9', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/8.png', category:
    'Experience' },
  ];
  let fields = { text: 'Name', groupBy: 'category' };

```

```

//Set customized list template
function listTemplate(data) {
  return (<div className="e-list-wrapper e-list-multi-line e-list-
avatar">
    <img className="e-avatar e-avatar-circle" src={data.image}/>
    <span className="e-list-item-header">{data.Name}</span>
    <span className="e-list-content">{data.contact}</span>
  </div>);
}
//Set customized group-header template
function groupTemplate(data) {
  return (<div><span
className='category'>{data.items[0].category}</span><span className="count">
{data.items.length} Item(s)</span></div>);
}
  return (<div>
    <ListViewComponent id='list' dataSource={data} fields={fields}
template={listTemplate} groupTemplate={groupTemplate} cssClass='e-list-
template'></ListViewComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  //Define an array of JSON data
  let data: { [key: string]: Object }[] = [
    { Name: 'Nancy', contact: '(206) 555-985774', id: '1', image:
'https://ej2.syncfusion.com/demos/src/grid/images/1.png', category:
'Experience' },
    { Name: 'Janet', contact: '(206) 555-3412', id: '2', image:
'https://ej2.syncfusion.com/demos/src/grid/images/3.png', category:
'Fresher' },
    { Name: 'Margaret', contact: '(206) 555-8122', id: '4', image:
'https://ej2.syncfusion.com/demos/src/grid/images/4.png', category:
'Experience' },
    { Name: 'Andrew ', contact: '(206) 555-9482', id: '5', image:
'https://ej2.syncfusion.com/demos/src/grid/images/2.png', category:
'Experience' },
    { Name: 'Steven', contact: '(71) 555-4848', id: '6', image:
'https://ej2.syncfusion.com/demos/src/grid/images/5.png', category:
'Fresher' },
    { Name: 'Michael', contact: '(71) 555-7773', id: '7', image:
'https://ej2.syncfusion.com/demos/src/grid/images/6.png', category:
'Experience' },
    { Name: 'Robert', contact: '(71) 555-5598', id: '8', image:
'https://ej2.syncfusion.com/demos/src/grid/images/7.png', category:
'Fresher' },
    { Name: 'Laura', contact: '(206) 555-1189', id: '9', image:
'https://ej2.syncfusion.com/demos/src/grid/images/8.png', category:
'Experience' },

```

```

];
let fields: object = { text: 'Name', groupBy: 'category' };
//Set customized list template
function listTemplate(data: any): JSX.Element {
    return (
        <div className="e-list-wrapper e-list-multi-line e-list-avatar">
            <img className="e-avatar e-avatar-circle" src={data.image}
        />

            <span className="e-list-item-header">{data.Name}</span>
            <span className="e-list-content">{data.contact}</span>
        </div>
    );
}
//Set customized group-header template
function groupTemplate(data: any): JSX.Element {
    return (
        <div><span
className='category'>{data.items[0].category}</span><span className="count">
{data.items.length} Item(s)</span></div>
    );
}
    return (
        <div>
            <ListViewComponent id='list' dataSource={data} fields={fields}
template={listTemplate as any} groupTemplate={groupTemplate as any}
cssClass='e-list-template'></ListViewComponent>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Filter list items in the listview in React Listview component

The filtered data can be displayed in the ListView component depending upon on user inputs using the [DataManager](#). Refer to the

following steps to render the ListView with filtered data.

- Render a textbox to get input for filtering data.
- Render ListView with

[dataSource](#), and set

the [sortOrder](#) property.

- Bind the `keyup` event for textbox to perform filtering operation. To filter list data, pass the list data source to the

`DataManager`, manipulate the data using the

[executeLocal](#) method,

and then update filtered data as ListView dataSource.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DataManager, Query } from "@syncfusion/ej2-data";
import { ListViewComponent } from "@syncfusion/ej2-react-lists";
function App() {
    let list = [
        { 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" }
    ];
    let fields = { text: "text", id: "id" };
    const [state, SetState] = React.useState({ listData: list });
    function onKeyUp(e) {
        let value = e.target.value;
        let data = new DataManager(state.listData).executeLocal(new
Query().where("text", "startswith", value, true));
        if (!value) {
            SetState({
                listData: list
            });
        }
        else {
            SetState({
                listData: data
            });
        }
    }
    return (<div id="sample">
        <input className="e-input" type="text" id="textbox"
placeholder="Filter" onKeyUp={onKeyUp.bind(this)} title="Type in a name"/>
        <ListViewComponent id="list" dataSource={state.listData}
fields={fields} sortOrder="Ascending"/>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById("element"));

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DataManager, Query, ODataV4Adaptor } from "@syncfusion/ej2-data";
import { ListViewComponent } from "@syncfusion/ej2-react-lists";
function App() {
    let list: any[] = [
        { 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" }
    ];

```



```

];
let fields: Object = { text: "text", id: "id" };
const [state, SetState] = React.useState({ listData: list });
function onKeyUp(e: any) {
    let value = e.target.value;
    let data = new DataManager(state.listData).executeLocal(
        new Query().where("text", "startswith", value, true)
    );
    if (!value) {
        SetState({
            listData: list
        });
    } else {
        SetState({
            listData: data
        });
    }
}
return (
    <div id="sample">
        <input
            className="e-input"
            type="text"
            id="textbox"
            placeholder="Filter"
            onKeyUp={onKeyUp.bind(this)}
            title="Type in a name"
        />
        <ListViewComponent
            id="list"
            dataSource={state.listData}
            fields={fields}
            sortOrder="Ascending"
        />
    </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById("element"));

```

In this demo, data has been filtered with starting character of the list items. You can also filter list items with ending

character by passing the `endswith` in

[where](#)

clause instead of `startswith`.

Add and remove list items from listview in React Listview component

You can add or remove list items from the ListView component using the [addItem](#) and [removeItem](#) methods. Refer to the following steps to add or remove a list item.

- Render the ListView with data source, and use the

[template](#) property to append the delete icon

for each list item. Also, bind the click event for the delete icon using the

[actionComplete](#) handler.

- Render the Add Item button, and bind the click event. On the click event handler, pass data with random id to

the [addItem](#) method to add a

new list item on clicking the Add Item button.

- Bind the click handler to the delete icon created in step 1. Within the click event, remove the list item by passing the

delete icon list item to

[removeItem](#) method.

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
    let listViewInstance = null;
    function listTemplate(data) {
        return (<div className="text-content">
            {data.text}
            <span className="delete-icon" onClick={deleteItem.bind(this)} />
        </div>);
    }
    // define the array of Json
    let dataSource = [
        { text: "Hennessey Venom", id: "1", icon: "delete-icon" },
        { text: "Bugatti Chiron", id: "2", icon: "delete-icon" },
        { text: "Bugatti Veyron Super Sport", id: "3", icon: "delete-icon" },
        { text: "Aston Martin One- 77", id: "4", icon: "delete-icon" },
        { text: "Jaguar XJ220", id: "list-5", icon: "delete-icon" },
        { text: "McLaren P1", id: "6", icon: "delete-icon" }
    ],
    let fields = { text: "text", iconCss: "icon" };
    function addItem() {
        let data = {
            text: "Koenigsegg - " + (Math.random() * 1000).toFixed(0),
            id: (Math.random() * 1000).toFixed(0).toString(),
            icon: "delete-icon"
        };
        listViewInstance.addItem([data]);
    }
    function deleteItem(args) {
        args.stopPropagation();
    }
}
```

```

        let liItem = args.target.closest('li');
        listViewInstance.removeItem(liItem);
    }
    return (<div>
        <ListViewComponent id="sample-list" dataSource={dataSource}
        fields={fields} template={listTemplate.bind(this)} ref={listview => {
            listViewInstance = listview;
        }}/>
        <ButtonComponent id="btn" onClick={addItem.bind(this)}>
            Add Item
        </ButtonComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
    let listViewInstance: ListViewComponent = null as any;
    function listTemplate(data: any): JSX.Element {
        return (
            <div className="text-content">
                {data.text}
                <span className="delete-icon" onClick={deleteItem.bind(this)} />
            </div>
        );
    }
    // define the array of Json
    let dataSource: any = [
        { text: "Hennessey Venom", id: "1", icon: "delete-icon" },
        { text: "Bugatti Chiron", id: "2", icon: "delete-icon" },
        { text: "Bugatti Veyron Super Sport", id: "3", icon: "delete-icon" },
        { text: "Aston Martin One- 77", id: "4", icon: "delete-icon" },
        { text: "Jaguar XJ220", id: "list-5", icon: "delete-icon" },
        { text: "McLaren P1", id: "6", icon: "delete-icon" }
    ];
    let fields: Object = { text: "text", iconCss: "icon" };
    function addItem() {
        let data = {
            text: "Koenigsegg - " + (Math.random() * 1000).toFixed(0),
            id: (Math.random() * 1000).toFixed(0).toString(),
            icon: "delete-icon"
        };
        listViewInstance.addItem([data]);
    }
    function deleteItem(args: any) {
        args.stopPropagation();
        let liItem = args.target.closest('li');
        listViewInstance.removeItem(liItem);
    }
}

```

```

    }
    return (
      <div>
        <ListViewComponent
          id="sample-list"
          dataSource={dataSource}
          fields={fields}
          template={listTemplate.bind(this) as any}
          ref={listview => {
            listviewInstance = listview as any;
          }}
        />
        <ButtonComponent id="btn" onClick={addItem.bind(this)}>
          Add Item
        </ButtonComponent>
      </div>
    );
  }
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Trace all events in listview in React ListView component

The ListView component triggers events based on its actions. The events can be used as extension points to perform custom operations. Refer to the following steps to trace the ListView events:

1. Render the ListView with

[dataSource](#), and

bind the [actionBegin](#),

[actionComplete](#),

and [select](#) events.

2. Perform custom operations in [actionBegin](#), [actionComplete](#), and [select](#) events.
3. Provide event log details for [actionBegin](#) and [actionComplete](#) events, and they will be displayed in the event trace panel when the ListView action starts and the dataSource bound successfully.
4. Get the selected item details from the [SelectEventArgs](#) in the select event, and display the selected list item text in the event trace panel while selecting list items.
5. Use clear button to remove event trace information.

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
  //Define an array of JSON data
  let data = [

```

```

    { 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" }
  ];
  const [state, SetState] = React.useState({
    eventData: [],
  });
  function btnClick() {
    SetState({
      eventData: []
    });
  }
  //Handler for actionBegin event trace
  function onActionBegin() {
    setTimeout(() => {
      SetState({
        eventData: [...state.eventData, "actionBegin"]
      });
    }, 0);
  }
  //Handler for select event trace
  function onSelect(args) {
    SetState({
      eventData: [...state.eventData, `${args.text} is selected`]
    });
  }
  //Handler for actionComplete event trace
  function onActionComplete() {
    setTimeout(() => {
      SetState({
        eventData: [...state.eventData, "actionComplete"]
      });
    }, 0);
  }
  //Display event log
  function appendElement(html) {
    let span = document.createElement("span");
    span.innerHTML = html;
    let log = document.getElementById("EventLog");
    log.insertBefore(span, log.firstChild);
  }
  return (<div id="sample">
    <div className="content-wrapper">
      <ListViewComponent id="listview-def" dataSource={data} width="250"
select={onSelect.bind(this)} actionBegin={onActionBegin.bind(this)}
actionComplete={onActionComplete.bind(this)} />
    </div>
    <div id="list_event">
      <h4>
        <b>Event Trace</b>

```

```

    </h4>
    <div id="evt">
      <div className="eventarea">
        /*Event log element */
        <span className="EventLog" id="EventLog">
          <div />
          {state.eventData.map((data, index) => (<div
key={index}>>{data}</div>))}
          </span>
        </div>
        <div className="evtbtn">
          /*clear button element */
          <ButtonComponent id="clear" onClick={btnClick.bind(this)}>
            Clear
          </ButtonComponent>
        </div>
      </div>
    </div>
  </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
  //Define an array of JSON data
  let data: any[] = [
    { 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" }
  ];
  const [state, SetState] = React.useState({
    eventData: [],
  });
  function btnClick() {
    SetState({
      eventData: []
    });
  }
  //Handler for actionBegin event trace
  function onActionBegin(): void {
    setTimeout(() => {
      SetState({
        eventData: [...state.eventData, "actionBegin"]
      });
    });
  }
}

```

```

    });
    }, 0);
}
//Handler for select event trace
function onSelect(args: SelectEventArgs): void {
    SetState({
        eventData: [...state.eventData, `${args.text} is selected`]
    });
}
//Handler for actionComplete event trace
function onActionComplete(): void {
    setTimeout(() => {
        SetState({
            eventData: [...state.eventData, "actionComplete"]
        });
    }, 0);
}
//Display event log
function appendElement(html: string): void {
    let span: HTMLElement = document.createElement("span");
    span.innerHTML = html;
    let log: any = document.getElementById("EventLog");
    log.insertBefore(span, log.firstChild);
}
return (
    <div id="sample">
        <div className="content-wrapper">
            <ListViewComponent
                id="listview-def"
                dataSource={data}
                width="250"
                select={onSelect.bind(this) as any}
                actionBegin={onActionBegin.bind(this) as any}
                actionComplete={onActionComplete.bind(this) as any}
            />
        </div>
        <div id="list_event">
            <h4>
                <b>Event Trace</b>
            </h4>
            <div id="evt">
                <div className="eventarea">
                    { /*Event log element */}
                    <span className="EventLog" id="EventLog">
                        <div />
                        {state.eventData.map((data: string, index: number) => (
                            <div key={index}>{data}</div>
                        ))}
                    </span>
                </div>
                <div className="evtbtn">
                    { /*clear button element */}
                    <ButtonComponent id="clear" onClick={btnClick.bind(this)}>
                        Clear
                    </ButtonComponent>
                </div>
            </div>
        </div>
    </div>

```

```

        </div>
      </div>
    );
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('element'));

```

Create mobile contact layout from listview in React Listview component

You can customize the ListView using the [template](#) property. Refer to the following steps to customize ListView as mobile contact view with our `ej2-avatar`.

- Render the ListView with [dataSource](#) that has avatar data. You can set avatar data as either text or class names. Refer to the following codes.

```

`ts
let dataSource = [
{
text: "Jenifer", contact: "(206) 555-985774", id: "1", avatar: "", pic: "pic01"
},
{
text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A", pic: ""
}
];
`

```

```

`ts
let dataSource = [
{
text: "Jenifer", contact: "(206) 555-985774", id: "1", avatar: "", pic: "pic01"
},
{
text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A", pic: ""
}
];
`

```

- Set `avatar` classes in ListView template to customize contact icon. In the following codes, medium size avatar has been set using the class name `e-avatar e-avatar-circle` from data source.


```

`ts
function listTemplate(data): JSX.Element {
  let letterAvatar = <span className='e-avatar e-avatar-circle'>{data.avatar}</span>
  let imageAvatar = <span className={` ${data.pic} e-avatar e-avatar-circle`}></span>
  return (
    <div className='e-list-wrapper e-list-multi-line e-list-avatar'>
      {data.avatar !== "" ? (letterAvatar) : (imageAvatar)}
      <span className="e-list-content">{data.contact}</span>
    </div>
  );
}
`ts
function listTemplate(data) {
  let letterAvatar = <span className='e-avatar e-avatar-circle'>{data.avatar}</span>;
  let imageAvatar = <span className={` ${data.pic} e-avatar e-avatar-circle`}></span>;
  return (<div className='e-list-wrapper e-list-multi-line e-list-avatar'>
    {data.avatar !== "" ? (letterAvatar) : (imageAvatar)}
    <span className="e-list-content">{data.contact}</span>
  </div>);
}

```

Avatars can be set in different sizes in avatar classes. To know more about avatar classes, refer to [Avatar](#).

- Sort the contact names using the [sortOrder](#) property of ListView.
- Enable the [showHeader](#) property, and set the [headerTitle](#) as `Contacts`.

INDEX.JSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  //Define an array of JSON data
  let data = [
    { id: 's_01', text: 'Robert', avatar: '', pic: 'pic04' },
    { id: 's_02', text: 'Nancy', avatar: 'N', pic: '' },
    { id: 's_05', text: 'Jenifer', pic: 'pic01', avatar: '' },
    { id: 's_03', text: 'Andrew', avatar: 'A', pic: '' },
  ]

```

```

    { id: 's_06', text: 'Margaret', pic: 'pic02', avatar: '' },
    { id: 's_07', text: 'Steven', pic: 'pic03', avatar: '' },
    { id: 's_08', text: 'Michael', avatar: 'M', pic: '' }
  ];
  let fields = { text: "Name" };
  function listTemplate(data) {
    let letterAvatar = <span className='e-avatar e-avatar-circle'>{data.avatar}</span>;
    let imageAvatar = <span className={` ${data.pic} e-avatar e-avatar-circle`}></span>;
    return (<div className='e-list-wrapper e-list-avatar'>
      {data.avatar !== "" ? (letterAvatar) : (imageAvatar)}
      <span className="e-list-content">{data.text}</span>
    </div>);
  }
  return (<div>
    { /* ListView element */ }
    <ListViewComponent id='list' dataSource={data}
    headerTitle='Contacts' showHeader={true} sortOrder="Ascending" width='350px'
    template={listTemplate} fields={fields} cssClass='e-list-template'></ListViewComponent>
  </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  //Define an array of JSON data
  let data: { [key: string]: Object }[] = [
    { id: 's_01', text: 'Robert', avatar: '', pic: 'pic04' },
    { id: 's_02', text: 'Nancy', avatar: 'N', pic: '' },
    { id: 's_05', text: 'Jenifer', pic: 'pic01', avatar: '' },
    { id: 's_03', text: 'Andrew', avatar: 'A', pic: '' },
    { id: 's_06', text: 'Margaret', pic: 'pic02', avatar: '' },
    { id: 's_07', text: 'Steven', pic: 'pic03', avatar: '' },
    { id: 's_08', text: 'Michael', avatar: 'M', pic: '' }
  ];
  let fields: any = { text: "Name" };
  function listTemplate(data: any): JSX.Element {
    let letterAvatar = <span className='e-avatar e-avatar-circle'>{data.avatar}</span>
    let imageAvatar = <span className={` ${data.pic} e-avatar e-avatar-circle`}></span>
    return (
      <div className='e-list-wrapper e-list-avatar'>
        {data.avatar !== "" ? (letterAvatar) : (imageAvatar)}
        <span className="e-list-content">{data.text}</span>
      </div>
    );
  }
  return (

```

```

        <div>
            {/* ListView element */}
            <ListViewComponent id='list' dataSource={data}
headerTitle='Contacts' showHeader={true}
                sortOrder="Ascending" width='350px' template={listTemplate
as any}
                fields={fields as any} cssClass='e-list-
template'></ListViewComponent>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Display spinner until list items are loaded in React ListView component

The features of the ListView component such as remote data-binding take more time to fetch data from corresponding dataSource/remote URL. In this case, you can use EJ2 [Spinner](#) to enhance the appearance of the UI. This section explains how to load a spinner component to groom the appearance.

Refer to the following code sample to render the spinner component.

```

`ts
function createSpinner({
target: spinnerInstance
});
function showSpinner(spinnerInstance);
`

```

Here, the data is fetched from **Northwind** Service URL; it takes a few seconds to load the data. To enhance the UI, the spinner component has been rendered initially. After the data is loaded from remote URL, the spinner component will be hidden in ListView [actionComplete](#) event.

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
//import DataManager related classes
import { DataManager, Query } from '@syncfusion/ej2-data';
import { createSpinner, showSpinner } from '@syncfusion/ej2-react-popups';
function App() {
    React.useEffect(() => {
        componentDidMount();
    }, []);
    //bind the DataManager instance to dataSource property
    let data = new DataManager({
        url: "https://services.syncfusion.com/js/production/api/",
        crossDomain: true
    });
    //Map the appropriate columns to fields property
    let fields = { id: "EmployeeID", text: "FirstName" };
    let spinnerInstance = null;

```

```

//Initialize query with the Query instance to get specified set of data
let query = new Query()
    .from("ListView")
    .select("EmployeeID,FirstName")
    .take(10);
function componentDidMount() {
    if (spinnerInstance) {
        createSpinner({
            target: spinnerInstance
        });
        showSpinner(spinnerInstance);
    }
}
function onActionComplete() {
    if (spinnerInstance)
        spinnerInstance.style.display = "none";
}
return (<div>
    <ListViewComponent id="list" dataSource={data} fields={fields}
    query={query} showHeader={true} headerTitle="Employees"
    actionComplete={onActionComplete.bind(this)} />
    <div ref={spinner => {
        spinnerInstance = spinner;
    }} id="spinner"/>
</div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
//import DataManager related classes
import { DataManager, Query, ODataV4Adaptor } from '@syncfusion/ej2-data';
import { createSpinner, hideSpinner, showSpinner } from '@syncfusion/ej2-react-popups';
function App() {
    React.useEffect(()=>{
        componentDidMount();
    }, [])
    //Bind the DataManager instance to dataSource property
    let data = new DataManager({
        url: "https://services.syncfusion.com/js/production/api/",
        crossDomain: true
    });
    //Map the appropriate columns to fields property
    let fields = { id: "EmployeeID", text: "FirstName" };
    let spinnerInstance: HTMLElement | null = null;
    //Initialize query with the Query instance to get specified set of data
    let query = new Query()
        .from("ListView")
        .select("EmployeeID,FirstName")

```

```

        .take(10);
    function componentDidMount() {
        if (spinnerInstance) {
            createSpinner({
                target: spinnerInstance
            });
            showSpinner(spinnerInstance);
        }
    }
    function onActionComplete() {
        if (spinnerInstance) spinnerInstance.style.display = "none";
    }
    return (
        <div>
            <ListViewComponent
                id="list"
                dataSource={data}
                fields={fields}
                query={query}
                showHeader={true}
                headerTitle="Employees"
                actionComplete={onActionComplete.bind(this)}
            />
            <div>
                <ref={spinner => {
                    spinnerInstance = spinner;
                }}
                id="spinner"
            />
            </div>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Hide checkbox in listview in React Listview component

The checkbox of the any list item can be hidden by using

[htmlAttributes](#) of

[fields](#) object. With

the help of [htmlAttributes](#) we can add unique class to each list item that will be rendered from the data source, from

the CSS class we can hide the checkbox of the list item.

In this sample, we had hidden the multiple leaf node of nested list. The [e-checkbox-hidden](#) class has been added in the data

source where the checkbox needs to be hidden. Refer the below snippet for simple data source.

`ts

{

```

'text': 'New York',
'id': '3002',
'category': 'USA',
'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' }
}
,

```

Even though we have hidden the checkbox the functionality will be same for the list item which might affect the

`getSelectedItems` method. So, to counteract that we will follow certain logic in the `select` event. The Logic here is to

remove the `e-active` class from the other checkbox hidden list item which will be added when we select on that item and

retain `e-active` on currently selected item.

In this process we will exclude the visible checkbox list items and only consider the hidden checkbox items.

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    //Define an array of JSON data
    let dataSource = [
        {
            'text': 'Asia',
            'id': '01',
            'category': 'Continent',
            'child': [{
                'text': 'India',
                'id': '1',
                'category': 'Asia',
                'child': [{
                    'text': 'Delhi',
                    'id': '1001',
                    'category': 'India',
                    'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
                },
                {
                    'text': 'Kashmir',
                    'id': '1002',
                    'category': 'India',
                    'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
                },
                {
                    'text': 'Goa',

```

```

        'id': '1003',
        'category': 'India',
        'htmlAttributes': { 'class': 'e-file' },
    },
    ],
},
{
    'text': 'China',
    'id': '2',
    'category': 'Asia',
    'child': [{
        'text': 'Zhejiang',
        'id': '2001',
        'category': 'China',
        'htmlAttributes': { 'class': 'e-file' },
    },
    {
        'text': 'Hunan',
        'id': '2002',
        'category': 'China',
        'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
    },
    {
        'text': 'Shandong',
        'id': '2003',
        'category': 'China',
        'htmlAttributes': { 'class': 'e-file' },
    }
    ]]
},
{
    'text': 'North America',
    'id': '02',
    'category': 'Continent',
    'child': [{
        'text': 'USA',
        'id': '3',
        'category': 'North America',
        'child': [{
            'text': 'California',
            'id': '3001',
            'category': 'USA',
            'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
        },
        {
            'text': 'New York',
            'id': '3002',
            'category': 'USA',
            'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
        },
        {
            'text': 'Florida',
            'id': '3003',
            'category': 'USA',

```

```

        'htmlAttributes': { 'class': 'e-file' },
    ]]
},
{
    'text': 'Canada',
    'id': '4',
    'category': 'North America',
    'child': [{
        'text': 'Ontario',
        'id': '4001',
        'category': 'Canada',
        'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
    },
    {
        'text': 'Alberta',
        'id': '4002',
        'category': 'Canada',
        'htmlAttributes': { 'class': 'e-file' },
    },
    {
        'text': 'Manitoba',
        'id': '4003',
        'category': 'Canada',
        'htmlAttributes': { 'class': 'e-file' },
    }
    ]]
},
{
    'text': 'Europe',
    'id': '03',
    'category': 'Continent',
    'child': [{
        'text': 'Germany',
        'id': '5',
        'category': 'Europe',
        'child': [{
            'text': 'Berlin',
            'id': '5001',
            'category': 'Germany',
            'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
        },
        {
            'text': 'Bavaria',
            'id': '5002',
            'category': 'Germany',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Hesse',
            'id': '5003',
            'category': 'Germany',
            'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
        }
    ]
    }, {

```



```

        'text': 'France',
        'id': '6',
        'category': 'Europe',
        'child': [{
            'text': 'Paris',
            'id': '6001',
            'category': 'France',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Lyon',
            'id': '6002',
            'category': 'France',
            'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
        },
        {
            'text': 'Marseille',
            'id': '6003',
            'category': 'France',
            'htmlAttributes': { 'class': 'e-file' },
        }
    ]
},
{
    'text': 'Australia',
    'id': '04',
    'category': 'Continent',
    'child': [{
        'text': 'Australia',
        'id': '7',
        'category': 'Australia',
        'child': [{
            'text': 'Sydney',
            'id': '7001',
            'category': 'Australia',
            'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
        },
        {
            'text': 'Melbourne',
            'id': '7002',
            'category': 'Australia',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Brisbane',
            'id': '7003',
            'category': 'Australia',
            'htmlAttributes': { 'class': 'e-file' },
        }
    ]
}, {
    'text': 'New Zealand',
    'id': '8',
    'category': 'Australia',
    'child': [{
        'text': 'Milford Sound',

```

```

        'id': '8001',
        'category': 'New Zealand',
        'htmlAttributes': { 'class': 'e-file' },
    },
    {
        'text': 'Tongariro National Park',
        'id': '8002',
        'category': 'New Zealand',
        'htmlAttributes': { 'class': 'e-file' },
    },
    {
        'text': 'Fiordland National Park',
        'id': '8003',
        'category': 'New Zealand',
        'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
    }
    ]]
},
{
    'text': 'Africa',
    'id': '05',
    'category': 'Continent',
    'child': [{
        'text': 'Morocco',
        'id': '9',
        'category': 'Africa',
        'child': [{
            'text': 'Rabat',
            'id': '9001',
            'category': 'Morocco',
            'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
        },
        {
            'text': 'Toubkal',
            'id': '9002',
            'category': 'Morocco',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Todgha Gorge',
            'id': '9003',
            'category': 'Morocco',
            'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
        }
    ]
}, {
    'text': 'South Africa',
    'id': '10',
    'category': 'Africa',
    'child': [{
        'text': 'Cape Town',
        'id': '10001',
        'category': 'South Africa',
        'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
    }

```

```

        },
        {
            'text': 'Pretoria',
            'id': '10002',
            'category': 'South Africa',
            'htmlAttributes': { 'class': 'e-file e-checkbox-
hidden' },
        },
        {
            'text': 'Bloemfontein',
            'id': '10003',
            'category': 'South Africa',
            'htmlAttributes': { 'class': 'e-file' },
        }
    ]
}

];
let fields = { tooltip: 'text' };
let title = 'Mixed Leaf Checkbox Hidden List ';
let listViewInstance;
function onSelect(args) {
    // Selecting all the e-active elements from the list.
    let normalElements =
Array.prototype.slice.call(listviewInstance.curUL.getElementsByClassName('e-
checkbox-hidden'));
    // Looping through all the selected element and removing e-active
class
    // to avoid behaviour interference with getSelectedItems method
    normalElements.forEach((element) => {
        element.classList.remove('e-active');
    });
    // Finally adding e-active class to currently selected item except
checkbox item.
    // because if it is checkbox item their actions will taken care from
the source side itself.
    if (args.item.classList.contains('e-checkbox-hidden')) {
        args.item.classList.add('e-active');
    }
}
return (<div id="sample">
    <ListViewComponent id='folderCheckbox' dataSource={dataSource}
fields={fields} headerTitle='Mixed Leaf Checkbox Hidden List'
showHeader="true" showCheckBox="true" select={onSelect.bind(this)}
ref={scope => { listViewInstance = scope; }}></ListViewComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';

```

```

function App() {
    //Define an array of JSON data
    let dataSource: { [key: string]: Object }[] = [
        {
            'text': 'Asia',
            'id': '01',
            'category': 'Continent',
            'child': [{
                'text': 'India',
                'id': '1',
                'category': 'Asia',
                'child': [{
                    'text': 'Delhi',
                    'id': '1001',
                    'category': 'India',
                    'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
},
                },
                {
                    'text': 'Kashmir',
                    'id': '1002',
                    'category': 'India',
                    'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
},
                },
                {
                    'text': 'Goa',
                    'id': '1003',
                    'category': 'India',
                    'htmlAttributes': { 'class': 'e-file' },
                },
            ]
        },
        {
            'text': 'China',
            'id': '2',
            'category': 'Asia',
            'child': [{
                'text': 'Zhejiang',
                'id': '2001',
                'category': 'China',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Hunan',
                'id': '2002',
                'category': 'China',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
},
            },
            {
                'text': 'Shandong',
                'id': '2003',
                'category': 'China',
                'htmlAttributes': { 'class': 'e-file' },
            }
        }
    ]
}

```

```

    },
    {
        'text': 'North America',
        'id': '02',
        'category': 'Continent',
        'child': [{
            'text': 'USA',
            'id': '3',
            'category': 'North America',
            'child': [{
                'text': 'California',
                'id': '3001',
                'category': 'USA',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
            },
            },
            {
                'text': 'New York',
                'id': '3002',
                'category': 'USA',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
            },
            },
            {
                'text': 'Florida',
                'id': '3003',
                'category': 'USA',
                'htmlAttributes': { 'class': 'e-file' },
            }
        ]
    },
    {
        'text': 'Canada',
        'id': '4',
        'category': 'North America',
        'child': [{
            'text': 'Ontario',
            'id': '4001',
            'category': 'Canada',
            'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
        },
        },
        {
            'text': 'Alberta',
            'id': '4002',
            'category': 'Canada',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Manitoba',
            'id': '4003',
            'category': 'Canada',
            'htmlAttributes': { 'class': 'e-file' },
        }
    ]
}
],
{
    'text': 'Europe',

```

```

        'id': '03',
        'category': 'Continent',
        'child': [{
            'text': 'Germany',
            'id': '5',
            'category': 'Europe',
            'child': [{
                'text': 'Berlin',
                'id': '5001',
                'category': 'Germany',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
            },
            },
            {
                'text': 'Bavaria',
                'id': '5002',
                'category': 'Germany',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Hesse',
                'id': '5003',
                'category': 'Germany',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
            },
        ]
    }, {
        'text': 'France',
        'id': '6',
        'category': 'Europe',
        'child': [{
            'text': 'Paris',
            'id': '6001',
            'category': 'France',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Lyon',
            'id': '6002',
            'category': 'France',
            'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
        },
        {
            'text': 'Marseille',
            'id': '6003',
            'category': 'France',
            'htmlAttributes': { 'class': 'e-file' },
        }
    ]
}
],
{
    'text': 'Australia',
    'id': '04',
    'category': 'Continent',
    'child': [{
        'text': 'Australia',

```

```

        'id': '7',
        'category': 'Australia',
        'child': [{
            'text': 'Sydney',
            'id': '7001',
            'category': 'Australia',
            'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
        },
        },
        {
            'text': 'Melbourne',
            'id': '7002',
            'category': 'Australia',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Brisbane',
            'id': '7003',
            'category': 'Australia',
            'htmlAttributes': { 'class': 'e-file' },
        }
    ]],
    {
        'text': 'New Zealand',
        'id': '8',
        'category': 'Australia',
        'child': [{
            'text': 'Milford Sound',
            'id': '8001',
            'category': 'New Zealand',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Tongariro National Park',
            'id': '8002',
            'category': 'New Zealand',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Fiordland National Park',
            'id': '8003',
            'category': 'New Zealand',
            'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
        },
        }
    ]],
    {
        'text': 'Africa',
        'id': '05',
        'category': 'Continent',
        'child': [{
            'text': 'Morocco',
            'id': '9',
            'category': 'Africa',
            'child': [{
                'text': 'Rabat',
                'id': '9001',

```

```

        'category': 'Morocco',
        'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
    },
        },
        {
            'text': 'Toubkal',
            'id': '9002',
            'category': 'Morocco',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Todgha Gorge',
            'id': '9003',
            'category': 'Morocco',
            'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
    },
        },
    ]
}, {
    'text': 'South Africa',
    'id': '10',
    'category': 'Africa',
    'child': [{
        'text': 'Cape Town',
        'id': '10001',
        'category': 'South Africa',
        'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
    },
        },
        {
            'text': 'Pretoria',
            'id': '10002',
            'category': 'South Africa',
            'htmlAttributes': { 'class': 'e-file e-checkbox-hidden'
    },
        },
        {
            'text': 'Bloemfontein',
            'id': '10003',
            'category': 'South Africa',
            'htmlAttributes': { 'class': 'e-file' },
        }
    ]
}
    ]
};
let fields: object = { tooltip: 'text' };
let title = 'Mixed Leaf Checkbox Hidden List ';
let listViewInstance: ListViewComponent;
function onSelect(args: SelectEventArgs) {
    // Selecting all the e-active elements from the list.
    let normalElements: HTMLElement[] =
Array.prototype.slice.call((listviewInstance as
any).curUL.getElementsByClassName('e-checkbox-hidden'));
    // Looping through all the selected element and removing e-active
class
    // to avoid behaviour interference with getSelectedItems method
    normalElements.forEach((element) => {
        element.classList.remove('e-active');
    });
}

```



```

    });
    // Finally adding e-active class to currently selected item except
checkbox item.
    // because if it is checkbox item their actions will taken care from
the source side itself.
    if (args.item.classList.contains('e-checkbox-hidden')) {
        args.item.classList.add('e-active');
    }
}
return (
    <div id="sample">
        <ListViewComponent id='folderCheckbox' dataSource={dataSource}
fields={fields} headerTitle='Mixed Leaf Checkbox Hidden List'
showHeader="true" showCheckBox="true" select={onSelect.bind(this)}
ref={(scope) => { listViewInstance = scope; }} ></ListViewComponent>
    </div>
)
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Manipulate listview as grid layout in React Listview component

In ListView, list items can be rendered in grid layout with following data manipulations.

- Add Item
- Remove Item
- Sort Items
- Filter Items

Grid Layout

In this section, we will discuss about rendering of list items in grid layout.

- Initialize and render ListView with dataSource which will render list items in list layout.
- Now, add the below CSS to list item. This will make list items to render in grid layout

`css

```

default-list .e-list-item {
height: 100px;
width: 100px;
float: left;
}
`

```

In the below sample, we have rendered List items in grid layout.

INDEX.JSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';

```

```
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    //Define an array of number
    let data = [1, 2, 3, 4, 5, 6, 7, 8, 9];
    function listtemplate(data) {
        return ();
    }
    ;
    return (<ListViewComponent id='list' dataSource={data}
template={listtemplate}>
        </ListViewComponent>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    //Define an array of number
    let data: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9];
    function listtemplate(data: any): JSX.Element {
        return (
            
        );
    };
    return (
        <ListViewComponent id='list'
            dataSource={data}
            template= {listtemplate as any} >
            </ListViewComponent>
        )
    }
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

Data manipulation

In this section, we will discuss about ListView data manipulations.

Add Item

We can add list item using [addItem](#) API. This will accept array of data as argument.

```
`ts
```

```
listViewInstance.addItem([{'text': 'Apricot', id: '32'}]);
```

```
,
```

In the below sample, you can add new fruit item by clicking add button which will open dialog box with fruit name and image URL text box. After entering the item details, click the add button. This will add your new fruit item.

Remove item

We can remove list item using [removeItem](#) API. This will accept fields with `id` or list item element as argument.

```
`ts
listViewInstance.removeItem({id: '32'});
`
```

In the below sample, you can remove fruit by hovering the fruit item which will show delete button and click that delete button to delete that fruit from your list.

Sort Items

ListView can be sorted either in Ascending or Descending order. To enable sorting in your ListView, set [sortOrder](#) as `Ascending` or `Descending`.

```
`ts
<ListViewComponent id='list' sortOrder= 'Ascending'></ListViewComponent>
`
```

We can also set sorting after component initialization.

```
`ts
listViewInstance.sortOrder = 'Ascending'
`
```

In the below sample, we have sorted fruits in `Ascending` order. To sort it in descending, click on sort order icon and vice versa.

Filter Items

ListView data can be filtered with the help of [dataManager](#). After filtering the data, update ListView [dataSource](#) with filtered data.

```
`ts
let value = filterInput.value; //input text box value
let filteredData = new DataManager(listdata).executeLocal(
  new Query().where("text", "startswith", value, true)
);
listViewInstance.dataSource = filteredData;
`
```

In the below sample, we can filter fruit items with the help of search text box. This will filter fruit items based on your input. Here we used [startswith](#) of input text to filter data in DataManager.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
```

```

import { closest } from '@syncfusion/ej2-base';
import { DataManager, Query } from "@syncfusion/ej2-data";
import { DialogComponent } from '@syncfusion/ej2-react-popups';
function App() {
    let ascClass = "e-sort-icon-ascending";
    let desClass = "e-sort-icon-descending";
    //Define an array of JSON data
    let fruitsdata = [
        { text: "Date", id: "1", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/dates.jpg" },
        { text: "Fig", id: "2", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/fig.jpg" },
        { text: "Apple", id: "3", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/apple.png" },
        { text: "Apricot", id: "4", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/apricot.jpg" },
        { text: "Grape", id: "5", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/grape.jpg" },
        { text: "Strawberry", id: "6", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/strawberry.jpg" },
        { text: "Pineapple", id: "7", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/pineapple.jpg" },
        { text: "Melon", id: "8", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/melon.jpg" },
        { text: "Lemon", id: "9", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/lemon.jpg" },
        { text: "Cherry", id: "10", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/cherry.jpg" }
    ];
    let add = null;
    let search = null;
    let sort = null;
    let imgURL = null;
    let name = null;
    let dialogInstance = null;
    let listViewInstance = null;
    let buttonObj = [
        {
            click: dlgButtonClick.bind(this),
            buttonModel: { content: "Add", isPrimary: true }
        }
    ];
    function addItem() {
        if (name)
            name.value = "";
        if (imgURL)
            imgURL.value = "";
    }
}

```

```

        if (dialogInstance)
            dialogInstance.show();
    }
    //Here we are removing list item
    function onDeleteBtnClick(e) {
        e.stopPropagation();
        let li = closest(e.currentTarget, ".e-list-item");
        let data = listViewInstance.findItem(li);
        listViewInstance.removeItem(data);
        new DataManager(fruitsdata).remove("id", { id: data["id"] });
    }
    //Here we are adding list item
    function dlgButtonClick() {
        if (listviewInstance && dialogInstance) {
            let name = name.value;
            let url = imgUrl.value;
            let id = Math.random() * 10000;
            listViewInstance.addItem([{ text: name, id: id, imgUrl: url }]);
            fruitsdata.push({ text: name, id: id, imgUrl: url });
            dialogInstance.hide();
        }
    }
    //Here we are sorting list item
    function sortItems() {
        if (listviewInstance && sort) {
            let ele = sort.firstElementChild;
            let des = ele.classList.contains(desClass) ? true : false;
            if (des) {
                ele.classList.remove(desClass);
                ele.classList.add(ascClass);
                listViewInstance.sortOrder = "Ascending";
            }
            else {
                ele.classList.remove(ascClass);
                ele.classList.add(desClass);
                listViewInstance.sortOrder = "Descending";
            }
            listViewInstance.dataBind();
        }
    }
    //Here, the list items are filtered using the DataManager instance.
    function onKeyUp() {
        if (listviewInstance) {
            let value = search.value;
            let data = new DataManager(fruitsdata).executeLocal(new
Query().where("text", "startswith", value, true));
            if (!value) {
                listViewInstance.dataSource = fruitsdata.slice();
            }
            else {
                listViewInstance.dataSource = data;
                listViewInstance.dataBind();
            }
        }
    }
    function content(data) {
        return (<div id="listDialog">

```

```

        <div className="input_name">
            <label htmlFor="name">Fruit Name: </label>
            <input id="name" ref={scope => {
                name = scope;
            }} className="e-input" type="text" placeholder="Enter fruit
name"/>
        </div>
        <div>
            <label htmlFor="imgurl">Fruit Image: </label>
            <input id="imgurl" ref={scope => {
                imgURL = scope;
            }} className="e-input" type="text" placeholder="Enter image
url"/>
        </div>
    </div>);
}
function listtemplate(data) {
    return (<div className="fruits">
        <div className="first">
            <img id="listImage" src={data.imgUrl} alt="fruit"/>
            <button className="delete e-control e-btn e-small e-round e-
delete-btn e-primary e-icon-btn" data-ripple="true">
                <span className="e-btn-icon e-icons delete-icon"
onClick={onDeleteBtnClick.bind(this)} />
            </button>
        </div>
        <div className="fruitName">{data.text}</div>
    </div>);
}
return (<div id="container">
    <div className="headerContainer">
        <div className="e-input-group">
            <input id="search" ref={scope => {
                search = scope;
            }} className="e-input" type="text" onKeyUp={onKeyUp.bind(this)}
placeholder="Search fruits"/>
            <span className="e-input-group-icon e-input-search"/>
        </div>
        <button id="sort" className="e-control e-btn e-small e-round e-
primary e-icon-btn" ref={scope => {
            sort = scope;
        }} title="Sort fruits" onClick={sortItems.bind(this)} data-
ripple="true">
            <span className="e-btn-icon e-icons e-sort-icon-ascending"/>
        </button>
        <button id="add" className="e-control e-btn e-small e-round e-
primary e-icon-btn" ref={scope => {
            add = scope;
        }} onClick={addItem.bind(this)} title="Add fruit" data-
ripple="true">
            <span className="e-btn-icon e-icons e-add-icon"/>
        </button>
    </div>
    <ListViewComponent id="list" dataSource={fruitsdata.slice()}
template={listtemplate.bind(this)} sortOrder="Ascending" ref={scope => {
        listViewInstance = scope;
    }} />

```

```

    <DialogComponent id="dialog" header="Add fruit"
    content={content.bind(this)} visible={false} buttons={buttonObj} ref={dialog
=> (dialogInstance = dialog)} width={"300px"} showCloseIcon={true}/>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { closest, enableRipple, MouseEventArgs } from '@syncfusion/ej2-
base';
import { DataManager, Query } from "@syncfusion/ej2-data";
import { DialogComponent } from '@syncfusion/ej2-react-popups';
function App() {
    let ascClass: string = "e-sort-icon-ascending";
    let desClass: string = "e-sort-icon-descending";
    //Define an array of JSON data
    let fruitsdata: { [key: string]: Object }[] = [
        { text: "Date", id: "1", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/dates.jpg" },
        { text: "Fig", id: "2", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/fig.jpg" },
        { text: "Apple", id: "3", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/apple.png" },
        { text: "Apricot", id: "4", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/apricot.jpg" },
        { text: "Grape", id: "5", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/grape.jpg" },
        { text: "Strawberry", id: "6", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/strawberry.jpg" },
        { text: "Pineapple", id: "7", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/pineapple.jpg" },
        { text: "Melon", id: "8", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/melon.jpg" },
        { text: "Lemon", id: "9", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/lemon.jpg" },
        { text: "Cherry", id: "10", imgUrl:
"https://helpej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/cherry.jpg" }
    ];
    let add: HTMLElement | null = null;

```

```

let search: HTMLInputElement | null = null;
let sort: HTMLElement | null = null;
let imgURL: HTMLInputElement | null = null;
let name: HTMLInputElement | null = null;
let dialogInstance: DialogComponent | null = null;
let listViewInstance: ListViewComponent | null = null;
let buttonObj = [
    {
        click: dlgButtonClick.bind(this),
        buttonModel: { content: "Add", isPrimary: true }
    }
];
function addItem() {
    if (name) name.value = "";
    if (imgURL) imgURL.value = "";
    if (dialogInstance) dialogInstance.show();
}
//Here we are removing list item
function onDeleteBtnClick(e: any) {
    e.stopPropagation();
    let li: Element = closest(e.currentTarget, ".e-list-item");
    let data = (listviewInstance as any).findItem(li);
    (listviewInstance as any).removeItem(data);
    new DataManager(fruitsdata).remove("id", { id: data["id"] });
}
//Here we are adding list item
function dlgButtonClick() {
    if (listviewInstance && dialogInstance) {
        let name: string = (name as any).value;
        let url: string = (imgURL as any).value;
        let id: number = Math.random() * 10000;
        listViewInstance.addItem([ { text: name, id: id, imgUrl: url } ]);
        fruitsdata.push({ text: name, id: id, imgUrl: url });
        dialogInstance.hide();
    }
}
//Here we are sorting list item
function sortItems() {
    if (listviewInstance && sort) {
        let ele = sort.firstElementChild as HTMLElement;
        let des = ele.classList.contains(desClass) ? true : false;
        if (des) {
            ele.classList.remove(desClass);
            ele.classList.add(ascClass);
            listViewInstance.sortOrder = "Ascending";
        } else {
            ele.classList.remove(ascClass);
            ele.classList.add(desClass);
            listViewInstance.sortOrder = "Descending";
        }
        listViewInstance.dataBind();
    }
}
//Here, the list items are filtered using the DataManager instance.
function onKeyUp() {
    if (listviewInstance) {
        let value: string = (search as any).value;

```



```

let data: Object[] = new DataManager(fruitsdata).executeLocal(
    new Query().where("text", "startswith", value, true)
);
if (!value) {
    listViewInstance.dataSource = fruitsdata.slice();
} else {
    listViewInstance.dataSource = data as { [key: string]: Object }[];
    listViewInstance.dataBind();
}
}
}
function content(data: any): JSX.Element {
    return (
        <div id="listDialog">
            <div className="input_name">
                <label htmlFor="name">Fruit Name: </label>
                <input
                    id="name"
                    ref={scope => {
                        name = scope;
                    }}
                    className="e-input"
                    type="text"
                    placeholder="Enter fruit name"
                />
            </div>
            <div>
                <label htmlFor="imgurl">Fruit Image: </label>
                <input
                    id="imgurl"
                    ref={scope => {
                        imgURL = scope;
                    }}
                    className="e-input"
                    type="text"
                    placeholder="Enter image url"
                />
            </div>
        </div>
    );
}
function listtemplate(data: any): JSX.Element {
    return (
        <div className="fruits">
            <div className="first">
                <img id="listImage" src={data.imgUrl} alt="fruit" />
                <button
                    className="delete e-control e-btn e-small e-round e-delete-btn
e-primary e-icon-btn"
                    data-ripple="true"
                >
                    <span
                        className="e-btn-icon e-icons delete-icon"
                        onClick={onDeleteBtnClick.bind(this)}
                    />
                </button>
            </div>
        </div>
    );
}

```

```

        <div className="fruitName">{data.text}</div>
    </div>
    );
}
return (
    <div id="container">
        <div className="headerContainer">
            <div className="e-input-group">
                <input
                    id="search"
                    ref={scope => {
                        search = scope;
                    }}
                    className="e-input"
                    type="text"
                    onKeyUp={onKeyUp.bind(this)}
                    placeholder="Search fruits"
                />
                <span className="e-input-group-icon e-input-search" />
            </div>
            <button
                id="sort"
                className="e-control e-btn e-small e-round e-primary e-icon-btn"
                ref={scope => {
                    sort = scope;
                }}
                title="Sort fruits"
                onClick={sortItems.bind(this)}
                data-ripple="true"
            >
                <span className="e-btn-icon e-icons e-sort-icon-ascending" />
            </button>
            <button
                id="add"
                className="e-control e-btn e-small e-round e-primary e-icon-btn"
                ref={scope => {
                    add = scope;
                }}
                onClick={addItem.bind(this)}
                title="Add fruit"
                data-ripple="true"
            >
                <span className="e-btn-icon e-icons e-add-icon" />
            </button>
        </div>
        <ListViewComponent
            id="list"
            dataSource={fruitsdata.slice()}
            template={listtemplate.bind(this) as any}
            sortOrder="Ascending"
            ref={scope => {
                listViewInstance = scope;
            }}
        />
        <DialogComponent
            id="dialog"
            header="Add fruit"

```

```

        content={content.bind(this) as any}
        visible={false}
        buttons={buttonObj}
        ref={dialog => (dialogInstance = dialog)}
        width={"300px"}
        showCloseIcon={true}
      />
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Drag and drop list items in React Listview component

In ListView component, we don't have drag and drop support. But we can achieve this requirement using [TreeView](#) component with ListView appearance.

Drag and Drop in TreeView component was enabled by setting [allowDragAndDrop](#) to **true**.

```

`ts
<TreeViewComponent id='treeview'
dataSource={data}
allowDragAndDrop = {true}
fields= {field}>
</TreeViewComponent>
`

```

The TreeView component is used to represent hierarchical data in a tree like structure. So, list items in TreeView can be dropped to child of target element. we can prevent this behaviour by cancelling the [nodeDragStop](#) and [nodeDragging](#) events.

```

`ts
function onDragStop(args: DragAndDropEventArgs) {
  //Block the Child Drop operation in TreeView
  let draggingItem: HTMLCollection = document.getElementsByClassName("e-drop-in");
  if (draggingItem.length == 1) {
    draggingItem[0].classList.add('e-no-drop');
    args.cancel = true;
  }
}
return (
<TreeViewComponent id='treeview'
dataSource={data}

```

```

allowDragAndDrop = {true}
nodeDragging = {onDragStop.bind(this)}
nodeDragStop = {onDragStop.bind(this)}
fields= {field} >
</TreeViewComponent>
)
,

`ts
function onDragStop(args) {
//Block the Child Drop operation in TreeView
let draggingItem = document.getElementsByClassName("e-drop-in");
if (draggingItem.length == 1) {
draggingItem[0].classList.add('e-no-drop');
args.cancel = true;
}
}

return (<TreeViewComponent id='treeview' dataSource={data} allowDragAndDrop={true}
nodeDragging={onDragStop.bind(this)} nodeDragStop={onDragStop.bind(this)} fields={field}>
</TreeViewComponent>);
,

```

In the below sample, we have rendered draggable list items.

INDEX.JSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { TreeViewComponent } from '@syncfusion/ej2-react-navigations';
function App() {
    //Define an array of JSON data
    let data = [
        { 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 field = { dataSource: data, id: "id", text: "text" };
    function onDragStop(args) {

```

```

    //Block the Child Drop operation in TreeView
    let draggingItem = document.getElementsByClassName("e-drop-in");
    if (draggingItem.length == 1) {
        draggingItem[0].classList.add("e-no-drop");
        args.cancel = true;
    }
}
return (<TreeViewComponent id="treeview" allowDragAndDrop={true}
nodeDragging={onDragStop.bind(this)} nodeDragStop={onDragStop.bind(this)}
fields={field}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { TreeViewComponent } from '@syncfusion/ej2-react-navigations';
import { DragAndDropEventArgs } from '@syncfusion/ej2-navigations';
function App() {
    //Define an array of JSON data
    let data: any[] = [
        { 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 field: object = { dataSource: data, id: "id", text: "text" };
    function onDragStop(args: DragAndDropEventArgs) {
        //Block the Child Drop operation in TreeView
        let draggingItem: HTMLCollection = document.getElementsByClassName("e-drop-in");
        if (draggingItem.length == 1) {
            draggingItem[0].classList.add("e-no-drop");
            args.cancel = true;
        }
    }
    return (
        <TreeViewComponent
            id="treeview"
            allowDragAndDrop={true}
            nodeDragging={onDragStop.bind(this) as any}
            nodeDragStop={onDragStop.bind(this) as any}
            fields={field}
        />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Load html content via ajax in React Listview component

We can set external **HTML** page content as [template](#) for our **ListView** component by making use of **AJAX** call.

```
`ts
ajax = new Ajax('./template.html', 'GET', false);
ajax.onSuccess = (e: string) => {
  template = e;
};
ajax.send();
`
```

In the below sample, we have rendered smartphone settings template from external **HTML** file.

INDEX.JSX

```
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { Ajax } from '@syncfusion/ej2-base';
function App() {
  let template;
  let ajax;
  ajax = new Ajax("./template.html", "GET", false);
  ajax.onSuccess = (e) => {
    template = e;
  };
  ajax.send();
  //Define an array of JSON data
  let data = [
    { name: "Network & Internet", id: "0", description: "Wi-Fi, mobile, data usage, hotspot" },
    { name: "Connected devices", id: "1", description: "Bluetooth, cast, NFC" },
    { name: "Battery", id: "2", description: "18% -4h 12m left" },
    { name: "Display", id: "3", description: "Wallpaper, sleep, font size" },
    { name: "Sound", id: "4", description: "Volume, vibration, Do Not Disturb" },
    { name: "Storage", id: "5", description: "52% used - 15.48 GB free" }
  ];
  return (<ListViewComponent id="list" dataSource={data} headerTitle="Settings" showHeader={true} template={template} cssClass="e-list-template"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { Ajax } from '@syncfusion/ej2-base';
function App() {
    let template: any;
    let ajax: Ajax;
    ajax = new Ajax("./template.html", "GET", false);
    ajax.onSuccess = (e: string) => {
        template = e;
    };
    ajax.send();
    //Define an array of JSON data
    let data: { [key: string]: Object }[] = [
        { name: "Network & Internet", id: "0", description: "Wi-Fi, mobile, data usage, hotspot" },
        { name: "Connected devices", id: "1", description: "Bluetooth, cast, NFC" },
        { name: "Battery", id: "2", description: "18% -4h 12m left" },
        { name: "Display", id: "3", description: "Wallpaper, sleep, font size" },
        { name: "Sound", id: "4", description: "Volume, vibration, Do Not Disturb" },
        { name: "Storage", id: "5", description: "52% used - 15.48 GB free" }
    ];
    return (
        <ListViewComponent
            id="list"
            dataSource={data}
            headerTitle="Settings"
            showHeader={true}
            template={template}
            cssClass="e-list-template"
        />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Render listview with hyper link navigation in React ListView component

We can use `anchor` tag along with `href` attribute in our ListView `template` property for navigation.

In the below sample, we have rendered `ListView` with search engines URL.

INDEX.JSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
    //Define an array of JSON data
    let dataSource = [
        { name: 'Google', url: 'https://www.google.com' },
        { name: 'Bing', url: 'https://www.bing.com' },
        { name: 'Yahoo', url: 'https://www.yahoo.com' },
        { name: 'Ask.com', url: 'https://www.ask.com' },
    ];

```

```

    { name: 'AOL.com', url: 'https://www.aol.com' }
  ];
  function anchor_template(data) {
    return (<a target='_blank' href={data.url}>{data.name}</a>);
  }
  ;
  return (<ListViewComponent id='list' dataSource={dataSource}
headerTitle='Search engines' showHeader={true} template={anchor_template}>
    </ListViewComponent>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
function App() {
  //Define an array of JSON data
  let dataSource: { [key: string]: Object }[] = [
    { name: 'Google', url: 'https://www.google.com' },
    { name: 'Bing', url: 'https://www.bing.com' },
    { name: 'Yahoo', url: 'https://www.yahoo.com' },
    { name: 'Ask.com', url: 'https://www.ask.com' },
    { name: 'AOL.com', url: 'https://www.aol.com' }
  ];
  function anchor_template(data: any): JSX.Element {
    return (
      <a target='_blank' href={data.url}>{data.name}</a>
    );
    return (
      <ListViewComponent id='list'
        dataSource={dataSource}
        headerTitle = 'Search engines'
        showHeader = {true}
        template= {anchor_template as any} >
        </ListViewComponent>
      )
    )
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('element'));
}

```

Customize listview as chat window in React Listview component

ListView can be customizable as chat window. To achieve that, use ListView [template](#) property and [Avatar](#) component.

- Listview template property is used to showcase the ListView as chat window.
- Avatar component is used to design the image of contact person.

Refer the below template code snippet for Template of chat window.

`ts


```

function listTemplate(data: any): JSX.Element {
  let sendertemplate = <div className='settings'><div id="content"><div
  className="name">{data.text}</div><div id="info">{data.contact}</div></div></div>{
  data.avatar !== "" ?
  <div id="image"><span className='e-avatar img1 e-avatar-circle'>{data.avatar}</span></div> : <div
  id="image"><span className={$ {data.pic} img1 e-avatar e-avatar-circle}></span></div>
  }</div>
  let receivertemplate = <div className='settings'>{
  data.avatar !== "" ?
  <div id="image2"><span className='e-avatar img2 e-avatar-circle'>{data.avatar}</span></div> : <div
  id="image2"><span className={$ {data.pic} img2 e-avatar e-avatar-circle}></span></div>
  }<div id="content1"><div className="name1">{data.text}</div><div
  id="info1">{data.contact}</div></div></div>
  return (
  <div>
  {data.chat !== "receiver" ? (sendertemplate) : (receivertemplate)}
  </div>
  );
}
`ts

function listTemplate(data) {
  let sendertemplate = <div className='settings'><div id="content"><div
  className="name">{data.text}</div><div id="info">{data.contact}</div></div>{data.avatar !== "" ?
  <div id="image"><span className='e-avatar img1 e-avatar-circle'>{data.avatar}</span></div> : <div
  id="image"><span className={$ {data.pic} img1 e-avatar e-avatar-circle}></span></div></div>;
  let receivertemplate = <div className='settings'>{data.avatar !== "" ?
  <div id="image2"><span className='e-avatar img2 e-avatar-circle'>{data.avatar}</span></div> : <div
  id="image2"><span className={$ {data.pic} img2 e-avatar e-avatar-circle}></span></div></div><div
  id="content1"><div className="name1">{data.text}</div><div
  id="info1">{data.contact}</div></div></div>;
  return (<div>
  {data.chat !== "receiver" ? (sendertemplate) : (receivertemplate)}
  </div>);
}
`

```

Chat order in template

In ListView template, we have rendered the list items based on receiver and sender information from dataSource of listview.

Adding messages to chat window

- Use textbox to get message from user.
- Add the textbox message to ListView dataSource using [addItem](#) method.

```
`ts
function btnClick() {
  let value = textboxEle.value;
  listObj.addItem([{ text: "Amenda", contact: value, id: "2", avatar: "A", pic: "", chat: "receiver" }]);
  textboxEle.value = "";
}
,

`ts
function btnClick() {
  let value = textboxEle.value;
  listObj.addItem([{ text: "Amenda", contact: value, id: "2", avatar: "A", pic: "", chat: "receiver" }]);
  textboxEle.value = "";
}
,

```

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
  let listObj = null;
  let textboxEle;
  //Define an array of JSON data
  let data = [
    {
      text: "Jenifer",
      contact: "Hi",
      id: "1",
      avatar: "",
      pic: "pic01",
      chat: "sender"
    },
    { text: "Amenda", contact: "Hello", id: "2", avatar: "A", pic: "",
chat: "receiver" },
    {

```

```

        text: "Jenifer",
        contact: "What Knid of application going to launch",
        id: "4",
        avatar: "",
        pic: "pic01",
        chat: "sender"
    },
    {
        text: "Amenda ",
        contact: "A knid of Emergency broadcast App",
        id: "5",
        avatar: "A",
        pic: "",
        chat: "receiver"
    },
    {
        text: "Jacob",
        contact: "Can you please elaborate",
        id: "6",
        avatar: "",
        pic: "pic04",
        chat: "sender"
    }
];
function listTemplate(data) {
    const sendertemplate = (<div className="settings">
        <div id="content">
            <div className="name">{data.text}</div>
            <div id="info">{data.contact}</div>
        </div>
        {data.avatar !== "" ? (<div id="image">
            <span className="e-avatar img1 e-avatar-circle">{data.avatar}</span>
            </div>) : (<div id="image">
            <span className={`img1 e-avatar e-avatar-circle`} />
            </div>)}
        </div>);
    const receivertemplate = (<div className="settings">
        {data.avatar !== "" ? (<div id="image2">
            <span className="e-avatar img2 e-avatar-circle">{data.avatar}</span>
            </div>) : (<div id="image2">
            <span className={`img2 e-avatar e-avatar-circle`} />
            </div>)}
        <div id="content1">
            <div className="name1">{data.text}</div>
            <div id="info1">{data.contact}</div>
        </div>
        </div>);
    return <div>{data.chat !== "receiver" ? sendertemplate :
receivertemplate}</div>;
}
function btnClick() {
    const value = textboxEle.value;
    listObj.addItem([
        { text: "Amenda", contact: value, id: "2", avatar: "A", pic: "",
chat: "receiver" }

```

```

    });
    textboxEle.value = "";
  }
  return (<div>
    /* ListView element */
    <ListViewComponent id="List" dataSource={data} headerTitle="Chat"
    showHeader={true} template={listTemplate} ref={scope => {
      listObj = scope;
    }}/>
    <input id="inputname" className="e-input" ref={textbox => {
      textboxEle = textbox;
    }} type="text" placeholder="Type your message"/>
    <ButtonComponent id="btn" onClick={btnClick.bind(this)}>
      Send
    </ButtonComponent>
  </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
function App() {
  let listObj: ListViewComponent = null as any;
  let textboxEle: any;
  //Define an array of JSON data
  let data: { [key: string]: Object }[] = [
    {
      text: "Jenifer",
      contact: "Hi",
      id: "1",
      avatar: "",
      pic: "pic01",
      chat: "sender"
    },
    { text: "Amenda", contact: "Hello", id: "2", avatar: "A", pic: "", chat:
"receiver" },
    {
      text: "Jenifer",
      contact: "What Knid of application going to launch",
      id: "4",
      avatar: "",
      pic: "pic01",
      chat: "sender"
    },
    {
      text: "Amenda ",
      contact: "A knid of Emergency broadcast App",
      id: "5",
      avatar: "A",

```

```

        pic: "",
        chat: "receiver"
    },
    {
        text: "Jacob",
        contact: "Can you please elaborate",
        id: "6",
        avatar: "",
        pic: "pic04",
        chat: "sender"
    }
];
function listTemplate(data: any): JSX.Element {
    const sendertemplate = (
        <div className="settings">
            <div id="content">
                <div className="name">{data.text}</div>
                <div id="info">{data.contact}</div>
            </div>
            {data.avatar !== "" ? (
                <div id="image">
                    <span className="e-avatar img1 e-avatar-circle">{data.avatar}</span>
                </div>
            ) : (
                <div id="image">
                    <span className={` ${data.pic} img1 e-avatar e-avatar-circle`} />
                </div>
            )}
        </div>
    );
    const receivertemplate = (
        <div className="settings">
            {data.avatar !== "" ? (
                <div id="image2">
                    <span className="e-avatar img2 e-avatar-circle">{data.avatar}</span>
                </div>
            ) : (
                <div id="image2">
                    <span className={` ${data.pic} img2 e-avatar e-avatar-circle`} />
                </div>
            )}
            <div id="content1">
                <div className="name1">{data.text}</div>
                <div id="info1">{data.contact}</div>
            </div>
        </div>
    );
    return <div>{data.chat !== "receiver" ? sendertemplate :
receivertemplate}</div>;
}
function btnClick() {
    const value = textboxEle.value;
    listObj.addItem([
        { text: "Amenda", contact: value, id: "2", avatar: "A", pic: "", chat:
"receiver" }
    ]

```

```

    });
    textboxEle.value = "";
  }
  return (
    <div>
      { /* ListView element */ }
      <ListViewComponent
        id="List"
        dataSource={data}
        headerTitle="Chat"
        showHeader={true}
        template={listTemplate as any}
        ref={scope => {
          listObj = scope as any;
        }}
      />
      <input
        id="inputname"
        className="e-input"
        ref={textbox => {
          textboxEle = textbox;
        }}
        type="text"
        placeholder="Type your message"
      />
      <ButtonComponent id="btn" onClick={btnClick.bind(this)}>
        Send
      </ButtonComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Integrate paging with listview in React Listview component

The first and foremost step is to obtain the **Pager** component from **Grid**. Install the ej2-react-grids package using the following command.

,

```
npm install @syncfusion/ej2-react-grids --save
```

,

Import the Pager to the ListView sample which has been created.

,

```
import { PagerComponent } from "@syncfusion/ej2-react-grids";
```

,

The [totalRecordsCount](#) property of Pager must be specified whenever using this particular component. By using [pageSize](#) property, the number of list items to be displayed is made available. The [pageCount](#) property allows the user to specify the visibility of the page numbers accordingly. Since the paging

sample in the upcoming code snippet uses these three properties, the explanation provided here were minimal and to the point. For further API concerns in Pager component, [click here](#).

With the help of the [query](#) property of ListView, the user can specify the number of records to be displayed in the current page.

The [query](#) property helps in splitting the entire datasource based on our convenience. In the sample provided below, when clicking the next button in pager, it fetches the datasource based on page size and current page of Pager component.

```
`ts
function click(args: any) {
  query = new Query().range((args.currentPage - 1) * pagesize, (args.currentPage * pagesize));
}
`
```

In the above code snippet, the event stores the [currentPage](#) value, and the datasource which is to be displayed in the next page is obtained.

Note: When [Link to the Video](#) isn't mentioned, it defaults to 12 records per page.

Maps

Getting Started

This section explains you the steps required to create a map and demonstrate the basic usage of the maps component.

You can explore some useful features in the Maps component using the following video.

Dependencies

Below is the list of minimum dependencies required to use the Maps.

```
`javascript
|-- @syncfusion/ej2-react-maps
|-- @syncfusion/ej2-maps
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-svg-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-react-base
`
```

Installation and Configuration

To get started with the React application, [create-react-app](#) can be used to setup the application. To install **create-react-app** run the following command.

```
`
```

```
npm install -g create-react-app
```

,

To create basic React application, run the following command.

,

```
create-react-app quickstart
```

,

Now, the application is created in the **quickstart** folder. Run the following command to navigate to the **quickstart** folder, and install the required **npm** packages.

,

```
cd quickstart
```

,

In the **quickstart** application, the Syncfusion component is added in the JavaScript file.

Creating a React application with TypeScript

To create React application with TypeScript, use the following command.

,

```
create-react-app quickstart --template typescript
```

,

Now, the application is created in the **quickstart** folder. Run the following command to navigate to the **quickstart** folder, and install the required **npm** packages.

,

```
cd quickstart
```

,

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. To install Maps package, use the following command.

,

```
npm install @syncfusion/ej2-react-maps --save
```

,

Add Map to the Project

Now, the Maps component can be added in the application. To initialize the Maps component in the React application, import the Maps component in the **src/App.js** or **src/App.tsx** as per the application. Please use the below code to include the Maps component in the application.

```
`ts
```

```
import { usMap } from './usa-map';
```

```
import React from 'react';
```

```
import { MapsComponent, LayersDirective, LayerDirective } from '@syncfusion/ej2-react-maps';
```



```
export function App() {
  return (<MapsComponent>
    <LayersDirective>
    <LayerDirective shapeData={usMap}>
    </LayerDirective>
    </LayersDirective>
  </MapsComponent>);
}
export default App;
```

Run the application

The Maps component is now included in the **quickstart** application. Use the following command to run the application.

```
npm start
```

Module Injection

Maps component are segregated into individual feature-wise modules. In order to use a particular feature,

you need to inject its feature services using **Inject** tag. You can find the modules available in maps and its description as follows.

- Annotations - Inject this provider to use annotations feature.
- Bubble - Inject this provider to use bubble feature.
- DataLabel - Inject this provider to use data label feature.
- Highlight - Inject this provider to use highlight feature.
- Legend - Inject this provider to use legend feature.
- Marker - Inject this provider to use marker feature.
- MapsTooltip - Inject this provider to use tooltip feature.
- NavigationLine - Inject this provider to use navigation lines feature.
- Selection - Inject this provider to use selection feature.
- Zoom - Inject this provider to use zooming and panning feature.

For example, we are going to use tooltip, data label and legend features of the maps.

Now import the MapsTooltip, DataLabel and Legend modules from maps package and inject it into the Maps component using **Inject** tag with required services.

```
`ts
```

```
import { Maps, Legend, DataLabel, MapsTooltip } from '@syncfusion/ej2-maps';
```

```
import * as React from 'react';
import { MapsComponent } from '@syncfusion/ej2-react-maps';
export function App() {
  return (<MapsComponent>
    <Inject services={[DataLabel, Legend, MapsTooltip]} />
    </MapsComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
```

Render shapes from GeoJSON data

This section explains how to bind GeoJSON data to the map.

`ts

let usMap: Object =

```
{
  "type": "FeatureCollection",
  "crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
  "features": [
    { "type": "Feature", "properties": { "iso31662": "MA", "name": "Massachusetts", "admin": "United States of America" }, "geometry": { "type": "MultiPolygon", "coordinates": [ [ [ [ -70.801756294617277, 41.248076234530558 ] ] ] ] }
  ]
};
`
```

Elements in the maps will get rendered in the layers. So add a layer collection to the maps by using [layers](#) property.

Now bind the GeoJSON data to the [shapeData](#) property.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
 '@syncfusion/ej2-react-maps';
export function App() {
  return (<MapsComponent id="maps">
    <LayersDirective>
```

```

        <LayerDirective shapeData={world_map}>
        </LayerDirective>
    </LayersDirective>
</MapsComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return (<MapsComponent id="maps">
        <LayersDirective>
            <LayerDirective shapeData={world_map}>
            </LayerDirective>
        </LayersDirective>
    </MapsComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Bind data source to map

The following properties in layers are used for binding data source to map.

- [dataSource](#)
- [shapeDataPath](#)
- [shapePropertyPath](#)

The **dataSource** property takes collection value as input. For example, the list of objects can be provided as input. This data is further used in tooltip, data label, bubble, legend and in color mapping.

The **shapeDataPath** property used to refer the data ID in dataSource. Where as, the **shapePropertyPath** property is used to refer the column name in shapeData to identify the shape. Both the properties are related to each other. When the values of the shapeDataPath property in the dataSource property and the value of shapePropertyPath in the shapeData property match, then the associated object from the dataSource is bound to the corresponding shape.

The JSON object "electionData" is used as data source below.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```
import { MapsComponent, LayersDirective, LayerDirective } from
 '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
 '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Note: Refer the value of the data source of [world-map.ts](#) and [data.ts](#) here.

Apply Color Mapping

The Color Mapping feature supports customization of shape colors based on the underlying value of shape received from bounded data.

Specify the field name from which the values have to be compared for the shapes in [colorValuePath](#) property in [shapeSettings](#).

Specify color and value in [colorValuePath](#) property. Here '#D84444' is specified for 'Trump' and '#316DB5' is specified for 'Clinton'.

INDEX.JSX

```
{% raw %}
```

```

import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(<MapsComponent >
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
      shapeSettings={ {
        colorValuePath: 'Membership',
        colorMapping: [
          {
            value: 'Permanent', color: '#D84444'
          },
          {
            value: 'Non-Permanent', color: '#316DB5'
          }
        ]
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(<MapsComponent >
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
      shapeSettings={ {
        colorValuePath: 'Membership',
        colorMapping: [
          {
            value: 'Permanent', color: '#D84444'
          },
          {
            value: 'Non-Permanent', color: '#316DB5'
          }
        ]
      } }>
    </LayerDirective>
  </LayersDirective>

```

```

    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Note: Refer the value of the data source of [world-map.ts](#) and [data.ts](#) here.

Add Title for Maps

You can add a title using [titleSettings](#) property to the map to provide quick information to the user about the shapes rendered in the map.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(<MapsComponent titleSettings={ { text: 'World map with
membership' } }>
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
      shapeSettings={ {
        colorValuePath: 'Membership',
        colorMapping: [
          {
            value: 'Permanent', color: '#D84444'
          },
          {
            value: 'Non-Permanent', color: '#316DB5'
          }
        ]
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(<MapsComponent titleSettings={ { text: 'World map with
membership' } }>
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
      shapeSettings={ {
        colorValuePath: 'Membership',
        colorMapping: [
          {
            value: 'Permanent', color: '#D84444'
          },
          {
            value: 'Non-Permanent', color: '#316DB5'
          }
        ]
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Note: Refer the value of the data source of [world-map.ts](#) and [data.ts](#) here.

Enable Legend

You can show legend for the maps by setting true to the [visible](#) property in [legendSettings](#) object and by injecting the [Legend](#)

service using [Inject](#) tag.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return(<MapsComponent legendSettings={ { visible: true } } >
    <Inject services={[Legend]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
      shapeSettings={ {
        colorValuePath: 'Membership',
        colorMapping: [
          {
            value: 'Permanent', color: '#D84444'
          },
          {
            value: 'Non-Permanent', color: '#316DB5'
          }
        ]
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```

        {
            value: 'Non-Permanent', color: '#316DB5'
        }
    ]
} }>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
    return(<MapsComponent legendSettings={ { visible: true } } >
        <Inject services={[Legend]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
                shapeSettings={ {
                    colorValuePath: 'Membership',
                    colorMapping: [
                        {
                            value: 'Permanent', color: '#D84444'
                        },
                        {
                            value: 'Non-Permanent', color: '#316DB5'
                        }
                    ]
                } }>
            </LayerDirective>
        </LayersDirective>
    </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Note: Refer the value of the data source of [world-map.ts](#) and [data.ts](#) here.

Add Data Label

You can add data labels to show additional information of the shapes in map. This can be achieved by setting [visible](#) property to true in the [dataLabelSettings](#) object and by injecting `DataLabel` service using `Inject` tag.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App(){
  return(<MapsComponent >
    <Inject services={[DataLabel]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
        dataLabelSettings={ {
          visible: true,
          labelPath: 'name',
          smartLabelMode: 'Trim'
        } }>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App(){
  return(<MapsComponent >
    <Inject services={[DataLabel]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
        dataLabelSettings={ {
          visible: true,
          labelPath: 'name',
          smartLabelMode: 'Trim'
        } }>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Enable Tooltip

The tooltip is useful when you cannot display information by using the data labels due to space constraints.

You can enable tooltip by setting the [visible](#) property as true in [tooltipSettings](#) object and by injecting `MapsTooltip` service using `Inject` tag.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  return(<MapsComponent >
    <Inject services={[MapsTooltip]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
        tooltipSettings={ {
          visible: true,
          valuePath: 'name'
        } }>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  return(<MapsComponent >
    <Inject services={[MapsTooltip]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
        tooltipSettings={ {
          visible: true,
          valuePath: 'name'
        } }>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
```

```
root.render(<App />);  
{% endraw %}
```

Creating a Next.js Application Using Syncfusion React Components

This section provides a step-by-step guide for setting up a Next.js application and integrating the Syncfusion React Maps component.

What is Next.js?

[Next.js](#) is a React framework that makes it easy to build fast, SEO-friendly, and user-friendly web applications. It provides features such as server-side rendering, automatic code splitting, routing, and API routes, making it an excellent choice for building modern web applications.

Prerequisites

Before getting started with the Next.js application, ensure the following prerequisites are met:

- [Node.js 18.17](#) or later.
- The application is compatible with macOS, Windows, and Linux operating systems.

Create a Next.js application

To create a new [Next.js](#) application, use one of the commands that are specific to either NPM or Yarn.

NPM

```
npm create-next-app@latest
```

YARN

```
yarn create next-app
```

Using one of the above commands will lead you to set up additional configurations for the project as below:

1. Define the project name: Users can specify the name of the project directly. Let's specify the name of the project as `ej2-nextjs-maps`.

CMD

```
√ What is your project named? » ej2-nextjs-maps
```

2. Select the required packages.

CMD

```
√ What is your project named? ... ej2-nextjs-maps  
√ Would you like to use TypeScript? ... No / `Yes`  
√ Would you like to use ESLint? ... No / `Yes`  
√ Would you like to use Tailwind CSS? ... `No` / Yes  
√ Would you like to use `src/` directory? ... No / `Yes`  
√ Would you like to use App Router? (recommended) ... No / `Yes`  
√ Would you like to customize the default import alias? ... `No` / Yes  
Creating a new Next.js app in D:\ej2-nextjs-maps.
```

3. Once complete the above mentioned steps to create `ej2-nextjs-maps`, navigate to the directory using the below command:

CMD

```
cd ej2-nextjs-maps
```

The application is ready to run with default settings. Now, let's add Syncfusion components to the project.

Install Syncfusion React packages

Syncfusion React component packages are available at [npmjs.com](https://www.npmjs.com). To use Syncfusion React components in the project, install the corresponding npm package.

Here, the [React Maps component](#) is used as an example. To install the React Maps component in the project, use the following command:

NPM

```
npm install @syncfusion/ej2-react-maps --save
```

YARN

```
yarn add @syncfusion/ej2-react-maps
```

Add Syncfusion React component

Follow the below steps to add the React Maps component to the Next.js project:

1. Define the Maps component in the `src/app/page.tsx` file, as shown below:

PAGE.TSX

```
'use client'
import { world_map } from './world-map';
import { MapsComponent, LayersDirective, LayerDirective } from
 '@syncfusion/ej2-react-maps';
function App() {
  return (<MapsComponent id="maps">
    <LayersDirective>
      <LayerDirective shapeData={world_map}>
      </LayerDirective>
    </LayersDirective>
    </MapsComponent>);
}
export default App;
```

Note: Refer the value of the data source of [world-map.ts](#).

Run the application

To run the application, use the following command:

NPM

```
npm run dev
```

YARN

```
yarn run dev
```

To learn more about the functionality of the Maps component, refer to the [documentation](#).

[View the NEXT.js Maps sample in the GitHub repository.](#)

Populate data in React Maps component

Geometry types

GeoJSON data contains geometry objects with properties such as geometry types and coordinates. The geometry types are the values present in the geometry objects of the GeoJSON data that specify the type of shape to be rendered, as well as the coordinates that help to draw the shape's boundary line. The supportive geometry types are:

| **Shapes** | **Supported** |

| --- | --- |

| Polygon | Yes |

| MultiPolygon | Yes |

| LineString | Yes |

| MultiLineString | Yes |

| Point | Yes |

| MultiPoint | Yes |

| GeometryCollection | Yes |

Shape data

The shape data collection describes geographical shape information that is available in GeoJSON format. The Map shapes are rendered with this data. The custom shapes such as seat selection in bus, seat selection in a cricket stadium and more useful information can be also added as [shapeData](#) in the layer of the Maps.

```
`ts
```

```
export let usMap = // Paste all the content copied from the JSON file.
```

```
,
```

Data source

The [dataSource](#) property is used to represent statistical data in the Maps component, and it accepts a collection of values as input. For example, a list of objects as input can be provided to the data source. This data source will be used to color the map, display data labels, and display tooltip, among other things.

The data source is populated with JSON data relative to shape data and stored as JSON object. In the below example, **populationData** can be used as data source in Maps.

```
`ts
```

```
export let populationData: object[] = [
```

```
{
  'code': 'AF',
  'value': 53,
  'name': 'Afghanistan',
  'population': 29863010,
  'density': 119
},
{
  'code': 'AL',
  'value': 117,
  'name': 'Albania',
  'population': 3195000,
  'density': 111
},
{
  'code': 'DZ',
  'value': 15,
  'name': 'Algeria',
  'population': 34895000,
  'density': 15
},
{
  'code': 'AO',
  'value': 15,
  'name': 'Angola',
  'population': 18498000,
  'density': 15
},
{
  'code': 'AR',
  'value': 15,
  'name': 'Argentina',
  'population': 40091359,
```

```
'density': 14
},
{
  'code': 'AM',
  'value': 109,
  'name': 'Armenia',
  'population': 3230100,
  'density': 108
}
];
`
```

Data binding

The following properties in the [layers](#) are used for binding data in the Maps component. Both the properties are related to each other.

- `shapePropertyPath`
- `shapeDataPath`

shapePropertyPath

The [shapePropertyPath](#) property is used to refer the field name in the [shapeData](#) property of shape layers to identify the shape. When the values of [shapeDataPath](#) property from the [dataSource](#) property and [shapePropertyPath](#) property from the [shapeData](#) property match, then the associated object from the data source is bound to the corresponding shape.

`world-map.ts` file contains following data and its field **name** value is used to map the corresponding shape with the provided data source.

```
`ts
export let world_map: object = {
  "type": "Feature",
  "properties": {
    "admin": "Afghanistan",
    "name": "Afghanistan",
    "continent": "Asia"
  },
  "geometry": { "type": "Polygon", "coordinates": [[[61.21081709172573,
https://ej2.syncfusion.com/react/documentation. ]],
...
`
```

shapeDataPath

The [shapeDataPath](#) property is similar to the [shapePropertyPath](#) property, but it refers to the field name in the [dataSource](#) property. For example, [populationData](#) contains the **code**, **value**, **name**, **population** and **density** fields. Here, the **name** field is set to the `shapeDataPath` to map the corresponding value of field name in shape data.

In the below example, both **name** fields contain the same value as **Afghanistan**, this value is matched in both shape data and data source, so that the details associated with **Afghanistan** will be mapped to the corresponding shape and used to color the corresponding shape, display data labels, display tooltips, and more.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='name' shapePropertyPath='name'
                      dataSource=[
                        {
                          'code': 'AF',
                          'value': 53,
                          'name': 'Afghanistan',
                          'population': 29863010,
                          'density': 119,
                          'color': '#DEEBAE'
                        },
                        {
                          'code': 'AL',
                          'value': 117,
                          'name': 'Albania',
                          'population': 3195000,
                          'density': 111,
                          'color': '#A4D6AD'
                        },
                        {
                          'code': 'DZ',
                          'value': 15,
                          'name': 'Algeria',
                          'population': 34895000,
                          'density': 15,
                          'color': '#37AFAB'
                        },
                        {
                          'code': 'AO',
                          'value': 15,
                          'name': 'Angola',
                          'population': 18498000,
                          'density': 15,
```



```

        'color': '#547C84'
      },
      {
        'code': 'AR',
        'value': 15,
        'name': 'Argentina',
        'population': 40091359,
        'density': 14,
        'color': '#CEBF93'
      },
      {
        'code': 'AM',
        'value': 109,
        'name': 'Armenia',
        'population': 3230100,
        'density': 108,
        'color': '#a69d70'
      }
    ]
  },
  shapeSettings: {
    colorValuePath: 'color',
    fill: '#E5E5E5'
  }
}
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='name' shapePropertyPath='name'
dataSource=[
          {
            'code': 'AF',
            'value': 53,
            'name': 'Afghanistan',
            'population': 29863010,
            'density': 119,
            'color': '#DEEBAE'
          },
          {

```

```

        'code': 'AL',
        'value': 117,
        'name': 'Albania',
        'population': 3195000,
        'density': 111,
        'color': '#A4D6AD'
      },
      {
        'code': 'DZ',
        'value': 15,
        'name': 'Algeria',
        'population': 34895000,
        'density': 15,
        'color': '#37AFAB'
      },
      {
        'code': 'AO',
        'value': 15,
        'name': 'Angola',
        'population': 18498000,
        'density': 15,
        'color': '#547C84'
      },
      {
        'code': 'AR',
        'value': 15,
        'name': 'Argentina',
        'population': 40091359,
        'density': 14,
        'color': '#CEBF93'
      },
      {
        'code': 'AM',
        'value': 109,
        'name': 'Armenia',
        'population': 3230100,
        'density': 108,
        'color': '#a69d70'
      }
    ]}
    shapeSettings={{
      colorValuePath: 'color',
      fill: '#E5E5E5'
    }}
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

<!-- markdownlint-disable MD010 -->

Binding complex data source

Data from data source can be bind to the Maps in two different ways.

1. Bind the field name directly to the properties as [shapeDataPath](#), [colorValuePath](#), [valuePath](#) and [shapeValuePath](#).
2. Bind the field name as `data.field` to the properties as [shapeDataPath](#), [colorValuePath](#), [valuePath](#) and [shapeValuePath](#).

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { complexData } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, MapsTooltip
} from '@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
import { MarkersDirective, MarkerDirective, Marker } from '@syncfusion/ej2-
react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker, Bubble, MapsTooltip]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='data.continent' shapePropertyPath='continent'
dataSource={complexData}
          shapeSettings={{
            colorValuePath: 'data.color'
          }} tooltipSettings={{
            visible: true,
            valuePath: 'data.continent'
          }}>
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="data.value"
              animationDuration={0} opacity={0.8}
minRadius={20} maxRadius={90}
              dataSource={[
                { 'name': 'India', 'value':
18.89685398845257, 'population': 391292635,
data: { 'color': 'red', 'population':
391292635,
'value': 189685398845257 }
                }
              ]}
              tooltipSettings={{
                visible: true,
                valuePath: 'data.population',
                template:"<div>${data.population}</div>"
              }}/>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
```

```

        </BubblesDirective>
        <MarkersDirective>
            <MarkerDirective visible={true} height={20}
animationDuration={0}
                                width={20} longitudeValuePath= "data.y"
latitudeValuePath= "data.x" shapeValuePath= "data.shape" colorValuePath=
"data.color"
                                tooltipSettings={{
                                    visible: true,
                                    valuePath: 'data.name',
                                    format: "${data.name}: ${data.x} :
${data.y}"
                                }}
                                offset = {{
                                    y: -10,
                                    x: 0
                                }}
                                dataSource=[
                                    { latitude: 37.6276571, longitude: -
122.4276688,
                                    name: 'San Bruno',
                                    data: { x: 37.6276571, y: -122.4276688,
                                    shape: 'Pentagon', color: 'red', imageUrl:
'images/ballon.png' }
                                    },
                                    { latitude: 33.5302186, longitude: -
117.7418381, name: 'Laguna Niguel',
                                    data: { x: 33.5302186, y: -117.7418381,
                                    color: 'blue', shape: 'Pentagon', imageUrl:
'images/ballon.png' }
                                    }
                                ]
            </MarkerDirective>
        </MarkersDirective>
    </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { complexData } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, MapsTooltip
} from '@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';

```

```

import { MarkersDirective, MarkerDirective, Marker } from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
            <Inject services={[Marker, Bubble, MapsTooltip]}/>
            <LayersDirective>
                <LayerDirective shapeData={world_map}
shapeDataPath='data.continent' shapePropertyPath='continent'
dataSource={complexData}
                    shapeSettings={{
                        colorValuePath: 'data.color'
                    }} tooltipSettings={{
                        visible: true,
                        valuePath: 'data.continent'
                    }}>
                <BubblesDirective>
                    <BubbleDirective visible={true}
valuePath="data.value"
                    animationDuration={0} opacity={0.8}
minRadius={20} maxRadius={90}
                    dataSource={[
                        { 'name': 'India', 'value':
18.89685398845257, 'population': 391292635,
                        data: { 'color': 'red', 'population':
391292635,
                            'value': 189685398845257 }
                        }
                    ]}
                    tooltipSettings={{
                        visible: true,
                        valuePath: 'data.population',
                        template: "<div>${data.population}</div>"
                    }}/>
                </BubblesDirective>
                <MarkersDirective>
                    <MarkerDirective visible={true} height={20}
animationDuration={0}
                    width={20} longitudeValuePath= "data.y"
latitudeValuePath= "data.x" shapeValuePath= "data.shape" colorValuePath=
"data.color"
                    tooltipSettings={{
                        visible: true,
                        valuePath: 'data.name',
                        format: "${data.name}: ${data.x} :
${data.y}"
                    }}
                    offset = {{
                        y: -10,
                        x: 0
                    }}
                    dataSource=[
                        { latitude: 37.6276571, longitude: -
122.4276688,
                        name: 'San Bruno',
                        data: { x: 37.6276571, y: -122.4276688,
name: 'San Bruno',

```

```

                                shape: 'Pentagon', color: 'red', imageUrl:
'images/ballon.png' }
                                },
                                { latitude: 33.5302186, longitude: -
117.7418381, name: 'Laguna Niguel',
                                data: { x: 33.5302186, y: -117.7418381,
name: 'Laguna Niguel',
                                color: 'blue', shape: 'Pentagon', imageUrl:
'images/ballon.png' }
                                }}}>
                                </MarkerDirective>
                                </MarkersDirective>
                                </LayerDirective>
                                </LayersDirective>
                                </MapsComponent>
                                );
                                }
                                const root = ReactDOM.createRoot(document.getElementById('container'));
                                root.render(<App />);
                                {% endraw %}

```

Layers in React Maps component

The Maps component is rendered through [layers](#) and any number of layers can be added to the Maps.

Multilayer

The Multilayer support allows loading multiple shape files and map providers in a single container, enabling Maps to display more information. The shape layer or map providers are the main layers of the Maps. Multiple layers can be added as **SubLayer** over the main layers using the [type](#) property of [layers](#).

Sublayer

Sublayer is a type of shape file layer. It allows loading multiple shape files in a single map view. For example, a sublayer can be added over the main layer to view geographic features such as rivers, valleys and cities in a map of a country. Similar to the main layer, elements in the Maps such as markers, bubbles, color mapping and legends can be added to the sub-layer.

In this example, the United States map shape is used as shape data by utilizing **usa.ts** file, and **texas.ts** and **california.ts** files are used as sub-layers in the United States map.

INDEX.JSX

```

{% raw %}
import { texas } from 'texas.ts';
import { usa_map } from 'usa.ts';
import { california } from 'california.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
            <LayersDirective>
                <LayerDirective shapeData={usa_map}
                    shapeSettings={ {
                        fill: '#E5E5E5',

```

```

        border: { width: 0.1, color: 'Black' },
      } } />
      <LayerDirective shapeData={texas} type="SubLayer"
        shapeSettings={ {
          fill: 'rgba(141, 206, 255, 0.6)',
          border: { width: 0.25, color: '#1a9cff' },
        } } />
      <LayerDirective shapeData={california} type="SubLayer"
        shapeSettings={ {
          fill: 'rgba(141, 206, 255, 0.6)',
          border: { width: 0.25, color: '#1a9cff' },
        } } />
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { texas } from 'texas.ts';
import { usa_map } from 'usa.ts';
import { california } from 'california.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings={ {
            fill: '#E5E5E5',
            border: { width: 0.1, color: 'Black' },
          } } />
        <LayerDirective shapeData={texas} type="SubLayer"
          shapeSettings={ {
            fill: 'rgba(141, 206, 255, 0.6)',
            border: { width: 0.25, color: '#1a9cff' },
          } } />
        <LayerDirective shapeData={california} type="SubLayer"
          shapeSettings={ {
            fill: 'rgba(141, 206, 255, 0.6)',
            border: { width: 0.25, color: '#1a9cff' },
          } } />
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Displaying different layer in the view

Multiple shape files and map providers can be loaded simultaneously in Maps. The [baseLayerIndex](#) property is used to determine which layer on the user interface should be displayed. This property is used for the Maps drill-down feature, so when the [baseLayerIndex](#) value is changed, the corresponding shape is loaded. In this example, two layers can be loaded with the World map and the United States map. Based on the given [baseLayerIndex](#) value the corresponding shape will be loaded in the user interface. If the [baseLayerIndex](#) value is set to **0**, then the world map will be loaded.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent baseLayerIndex={1}>
      <LayersDirective>
        <LayerDirective shapeData={world_map} />
        <LayerDirective shapeData={usa_map} />
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent baseLayerIndex={1}>
      <LayersDirective>
        <LayerDirective shapeData={world_map} />
        <LayerDirective shapeData={usa_map} />
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```


Rendering custom shapes

Custom shapes (also known as custom maps) can be rendered in Maps to represent bus seat booking, cricket stadium, basic home plan/sketch, and so on. To accomplish this, a JSON file in GeoJSON format with proper geometries must be created manually or with the assistance of any online map vendor. The GeoJSON file created must be set to the [shapeData](#) in the Maps layer, and the [geometryType](#) must be set as **Normal**.

Please refer this [link](#) for an example GeoJSON file containing information about bus seat selection.

Please refer this [link](#) for more information and a live demonstration.

Providers

Openstreetmap in React Maps component

The OpenStreetMap (OSM) is the online Maps provider built by a community of developers; it is free to use under an open license. It allows to view geographical data in a collaborative way from anywhere on the earth. The OSM Maps provides small tile images based on our requests and combines those images into a single image to display the Maps area in the Maps component.

Adding OpenStreetMap

The OSM Maps can be rendered using the [urlTemplate](#) property.

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent>
      <LayersDirective>
        <LayerDirective
          urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png" />
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
} from '@syncfusion/ej2-react-maps';
```

```
export function App() {
  return (
    <MapsComponent>
      <LayersDirective>
        <LayerDirective
urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png" />
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Changing the tile server of the OpenStreetMap

The OSM tile server can be changed by setting the tile URL in the [urlTemplate](#) property. For more details about the OSM tile server, refer [here](#).

Enabling zooming and panning

The OSM Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
  Inject,
  Zoom,
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent
      zoomSettings={{
        enable: true,
        toolbars: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset'],
      }}
    >
      <Inject services={[Zoom]} />
      <LayersDirective>
        <LayerDirective
urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png" />
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
  Inject,
  Zoom,
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent
      zoomSettings={{
        enable: true,
        toolbars: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset'],
      }}
    >
      <Inject services={[Zoom]} />
      <LayersDirective>
        <LayerDirective
          urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png" />
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Adding markers and navigation line

Markers can be added to the layers of OSM Maps by setting the corresponding location's coordinates of latitude and longitude using [MarkerDirective](#) tag. Navigation lines can be added on top of an OSM Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [NavigationLineDirective](#) tag.

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  MapsComponent,
  LayersDirective,
  NavigationLineDirective,
  LayerDirective,
  Zoom,
  MarkersDirective,
  NavigationLine,
  NavigationLinesDirective,
  MarkerDirective,
  Marker,
  Inject,
  Maps,
} from '@syncfusion/ej2-react-maps';
export function App() {
```

```

return (
  <MapsComponent
    zoomSettings={{ zoomFactor: 4 }}
    centerPosition={{ latitude: 29.394708, longitude: -94.954653 }}
  >
    <Inject services={[Marker, NavigationLine, Zoom]} />
    <LayersDirective>
      <LayerDirective
        urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png">
        <MarkersDirective>
          <MarkerDirective
            visible={true}
            height={25}
            width={15}
            dataSource={[
              {
                latitude: 34.06062,
                longitude: -118.330491,
                name: 'California',
              },
              {
                latitude: 40.724546,
                longitude: -73.850344,
                name: 'New York',
              },
            ]}
          ></MarkerDirective>
        </MarkersDirective>
        <NavigationLinesDirective>
          <NavigationLineDirective
            visible={true}
            latitude={[34.06062, 40.724546]}
            longitude=[-118.330491, -73.850344]}
            color="blue"
            angle={90}
            width={5}
          />
        </NavigationLinesDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  MapsComponent,
  LayersDirective,
  NavigationLineDirective,

```

```

LayerDirective,
Zoom,
MarkersDirective,
NavigationLine,
NavigationLinesDirective,
MarkerDirective,
Marker,
Inject,
Maps,
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent
      zoomSettings={{ zoomFactor: 4 }}
      centerPosition={{ latitude: 29.394708, longitude: -94.954653 }}
    >
      <Inject services={[Marker, NavigationLine, Zoom]} />
      <LayersDirective>
        <LayerDirective
          urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png">
          <MarkersDirective>
            <MarkerDirective
              visible={true}
              height={25}
              width={15}
              dataSource={[
                {
                  latitude: 34.06062,
                  longitude: -118.330491,
                  name: 'California',
                },
                {
                  latitude: 40.724546,
                  longitude: -73.850344,
                  name: 'New York',
                },
              ]}
            ></MarkerDirective>
          </MarkersDirective>
          <NavigationLinesDirective>
            <NavigationLineDirective
              visible={true}
              latitude={[34.06062, 40.724546]}
              longitude=[-118.330491, -73.850344]}
              color="blue"
              angle={90}
              width={5}
            />
          </NavigationLinesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the OSM Maps layer for highlighting a particular continent or country in OSM Maps by adding another layer and specifying the [type](#) property of Maps layer to **SubLayer**.

INDEX.JSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { africa_continent } from 'africa-continent.ts';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent>
      <LayersDirective>
        <LayerDirective
urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png" />
          <LayerDirective
            shapeData={africa_continent}
            type="SubLayer"
            shapeSettings={{
              fill: 'blue',
            }}
          />
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { africa_continent } from 'africa-continent.ts';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent>
      <LayersDirective>
        <LayerDirective
urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png" />
```

```

        <LayerDirective
            shapeData={africa_continent}
            type="SubLayer"
            shapeSettings={{
                fill: 'blue',
            }}
        />
    </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Enabling legend

The legend can be added to the tile Maps by setting the [visible](#) property of [legendSettings](#) to **true**.

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { markerDataSource } from 'markerdata.ts';
import {
    MapsComponent,
    LayersDirective,
    LayerDirective,
    MarkersDirective,
    MarkerDirective,
    Marker,
    Legend,
    Inject,
} from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent
            id="maps"
            legendSettings={{
                visible: true,
                type: 'Markers',
                useMarkerShape: true,
                toggleLegendSettings: {
                    enable: true,
                    applyShapeSettings: false,
                    border: {
                        color: 'green',
                        width: 2,
                    },
                },
            }}
        >
            <Inject services={[Marker, Legend]} />
            <LayersDirective>
                <LayerDirective
                    urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png">

```

```

        <MarkersDirective>
          <MarkerDirective
            visible={true}
            dataSource={markerDataSource}
            colorValuePath="color"
            shapeValuePath="shape"
            legendText="country"
          ></MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { markerDataSource } from 'markerdata.ts';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
  MarkersDirective,
  MarkerDirective,
  Marker,
  Legend,
  Inject,
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent
      id="maps"
      legendSettings={{
        visible: true,
        type: 'Markers',
        useMarkerShape: true,
        toggleLegendSettings: {
          enable: true,
          applyShapeSettings: false,
          border: {
            color: 'green',
            width: 2,
          },
        },
      }}
    >
      <Inject services={[Marker, Legend]} />
      <LayersDirective>
        <LayerDirective
          urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png">

```



```

        <MarkersDirective>
          <MarkerDirective
            visible={true}
            dataSource={markerDataSource}
            colorValuePath="color"
            shapeValuePath="shape"
            legendText="country"
          ></MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Bing maps in React Maps component

Bing Maps is a online Maps provider, owned by Microsoft. As like OSM, it provide Maps tile images based on our requests and combines those images into a single one to display Maps area.

Adding Bing Maps

The Bing Maps can be rendered using the [urlTemplate](#) property, which is based on the URL generated by the [getBingUrlTemplate](#) method in the Maps. The format of the required URL of Bing Maps varies from other online map providers. As a result, a built-in [getBingUrlTemplate](#) method has been included that returns the URL in a generic format. In the meantime, a subscription key is required for Bing Maps. The Bing Maps key can be obtained from [here](#), then append it to the Bing Maps URL before passing it to the [getBingUrlTemplate](#) method. The URL returned by this method must be passed to the [urlTemplate](#) property.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from '@syncfusion/ej2-react-maps';
function load(args) {
  args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
    args.maps.layers[0].urlTemplate= url;
  });
}
export function App() {
  return(
    <MapsComponent load={load}>
    <LayersDirective>

```

```

<LayerDirective />
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

```

Types of Bing Maps

Bing Maps provides different types of Maps and it is supported in the Maps component.

- **Aerial** - Displays satellite images to highlight roads and major landmarks for easy identification.
- **AerialWithLabel** - Displays aerial Maps with labels for the continent, country, ocean, etc.
- **Road** - Displays the default Maps view of roads, buildings, and geography.
- **CanvasDark** - Displays dark version of the road Maps.
- **CanvasLight** - Displays light version of the road Maps.
- **CanvasGray** - Displays grayscale version of the road Maps.

To render the light version of the road Maps, set the **CanvasLight** value is passed via the URL into the [getBingUrlTemplate](#) method demonstrated in the following code sample.

```

{% raw %}
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';

function load(args) {
  args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/CanvasLight?output=json&uriScheme=https&key=?").then(function(url) {
    args.maps.layers[0].urlTemplate= url;
  });
}

export function App() {
  return(
    <MapsComponent zoomSettings= {{ zoomFactor: 4 }}
    centerPosition = {{
      latitude : 38.8951,

```

```

longitude : -77.0364
}} load={load}>
<Inject services={[Zoom]} />
<LayersDirective>
<LayerDirective />
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

```

```
{% endraw %}
```

Enabling zooming and panning

Bing Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

```
{% raw %}
```

```
`ts
```

```

import * as React from "react";
import * as ReactDOM from "react-dom";

import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';

function load(args) {
  args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
    args.maps.layers[0].urlTemplate= url;
  });
}

export function App() {
  return(
    <MapsComponent load={load} zoomSettings= {{
      enable : true,
      toolbars:[ "Zoom", "ZoomIn", "ZoomOut", "Pan", "Reset" ]}}>
    <Inject services={[Zoom]} />

```

```

<LayersDirective>
<LayerDirective />
</LayersDirective>
</MapsComponent>

);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

```

```
{% endraw %}
```

Adding markers and navigation line

Markers can be added to the layers of Bing Maps by setting the corresponding location's coordinates of latitude and longitude using [MarkerDirective](#) tag. Navigation lines can be added on top of an Bing Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [NavigationLineDirective](#) tag.

```

{% raw %}
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";

import { MapsComponent, MarkersDirective, NavigationLineDirective, NavigationLinesDirective,
MarkerDirective, LayersDirective, LayerDirective, Inject, Zoom, Marker, NavigationLine } from
'@syncfusion/ej2-react-maps';

function load(args) {

args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output
=json&uriScheme=https&key=?").then(function(url) {

args.maps.layers[0].urlTemplate= url;

});

}

export function App() {
return (
<MapsComponent load={load} zoomSettings= {{
zoomFactor : 4
}}
centerPosition = {{
latitude: 29.394708,

```

```
longitude: -94.954653
}}>
<Inject services={[Zoom, Marker, NavigationLine]} />
<LayersDirective>
  <LayerDirective>
    <MarkersDirective>
      <MarkerDirective visible={true}
        height={25}
        width={15}
        dataSource={[
          {
            latitude: 34.060620,
            longitude: -118.330491,
            name: "California"
          },
          {
            latitude: 40.724546,
            longitude: -73.850344,
            name: "New York"
          }
        ]}
      />
    </MarkerDirective>
  </MarkersDirective>
  <NavigationLinesDirective>
    <NavigationLineDirective visible={true}
      latitude={[34.060620, 40.724546]}
      longitude=[[-118.330491,-73.850344]]
      color="blue"
      angle={90}
      width={5} />
    </NavigationLinesDirective>
  </LayerDirective>
```

```

</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

```

```
{% endraw %}
```

Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the Bing Maps layer for highlighting a particular continent or country in Bing Maps by adding another layer and specifying the [type](#) property of Maps layer to **SubLayer**.

```

{% raw %}
`ts
import { africa } from 'africa_continent.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from '@syncfusion/ej2-react-maps';

function load(args) {
  args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
    args.maps.layers[0].urlTemplate= url;
  });
}

export function App() {
  return(
    <MapsComponent load={load}>
    <LayersDirective>
    <LayerDirective/>
    <LayerDirective shapeData= {africa_continent}>
    type= 'SubLayer'
    shapeSettings= {{
    fill: 'blue'
    }}
  )

```

```

/>
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

```

```
{% endraw %}
```

Enabling legend

The legend can be added to the tile Maps by setting the [visible](#) property of [legendSettings](#) to **true**.

```
{% raw %}
```

```
`ts
```

```

import { markerDataSource } from 'markerdata.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

import { MapsComponent, LayersDirective, LayerDirective, Marker, MarkersDirective, MarkerDirective,
Legend, Inject } from '@syncfusion/ej2-react-maps';

function load(args) {
  args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output
=json&uriScheme=https&key=?").then(function(url) {
    args.maps.layers[0].urlTemplate= url;
  });
}

export function App() {
  return(
    <MapsComponent
      load={load}
      legendSettings={{
        visible: true,
        type: 'Markers',
        useMarkerShape: true,
        toggleLegendSettings: {
          enable: true,

```

```

    applyShapeSettings: false,
    border: {
      color: 'green',
      width: 2,
    },
  },
}
}}

```

```

<Inject services={[Marker, Legend]} />
<LayersDirective>
  <LayerDirective>
    <MarkersDirective>
      <MarkerDirective
        visible={true}
        dataSource={markerDataSource}
        colorValuePath="color"
        shapeValuePath="shape"
        legendText="country"
      />
    </MarkersDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

{% enddraw %}

```

[Azure maps in React Maps component](#)

Azure Maps is yet another online Maps provider, owned by Microsoft. As like OSM and Bing Maps, it provides Maps tile images based on our requests and combines those images into a single one to display Maps area.

Adding Azure Maps

The Azure Maps can be rendered using the [urlTemplate](#) property with the tile server URL provided by online map providers. In the meantime, a subscription key is required for Azure Maps. Follow the steps in this [link](#) to generate an API key, and then added the key to the URL.

Refer to [Azure Maps Licensing](#).

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent>
    <LayersDirective>
    <LayerDirective urlTemplate='https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key
    &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY' />
    </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
```

Enabling zooming and panning

The Azure Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a map for in-depth analysis. Panning helps to move a map around to focus the targeted area.

```
{% raw %}
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Maps, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent zoomSettings = { { enable: true, toolbars: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset']} }>
    <Inject services={ [Zoom]} />
  );
}
```

```

<LayersDirective>
<LayerDirective urlTemplate='https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key
&api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY' />
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
{% endraw %}

```

Adding markers and navigation line

Markers can be added to the layers of Azure Maps by setting the corresponding location's coordinates of latitude and longitude using [MarkerDirective](#) tag. Navigation lines can be added on top of the Azure Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [NavigationLineDirective](#) tag.

```

{% raw %}
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";

import { MapsComponent, LayersDirective, NavigationLineDirective, LayerDirective, Zoom,
MarkersDirective, NavigationLine, NavigationLinesDirective, MarkerDirective, Marker, Inject, Maps }
from '@syncfusion/ej2-react-maps';

export function App() {
return(
<MapsComponent zoomSettings = { { zoomFactor: 4 } } centerPosition = {{ latitude: 29.394708,
longitude: -94.954653}}>
<Inject services={[Marker, NavigationLine, Zoom]} />
<LayersDirective>
<LayerDirective urlTemplate='https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key
&api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY'>
<MarkersDirective>
<MarkerDirective visible={true}
height={25}
width={15}
dataSource={

```

```
{
latitude: 34.060620,
longitude: -118.330491,
name: "California"
},
{
latitude: 40.724546,
longitude: -73.850344,
name: "New York"
}
]}

</MarkerDirective>
</MarkersDirective>
<NavigationLinesDirective>
  <NavigationLineDirective visible={true}
    latitude={[34.060620, 40.724546]}
    longitude=[[-118.330491,-73.850344]]
    color="blue"
    angle={90}
    width={5} />
</NavigationLinesDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

{% endraw %}
```

Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the Azure Maps layer for highlighting a particular continent or country in Azure Maps by adding another layer and specifying the [type](#) property of Maps layer to **SubLayer**.

```
{% raw %}
`ts
import { africa_continent } from 'africa-continent.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent>
    <LayersDirective>
    <LayerDirective urlTemplate='https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key
    &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY' />
    <LayerDirective shapeData= {africa_continent}
    type= 'SubLayer'
    shapeSettings= {{
    fill: 'blue'
    }}
    />
    </LayersDirective>
    </MapsComponent>
  );
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
```

```
{% endraw %}
```

Enabling legend

The legend can be added to the Azure Maps by setting the [visible](#) property of [legendSettings](#) to **true**.

```
{% raw %}
`ts
import { markerDataSource } from 'markerdata.ts';
```

```
import * as React from "react";
import * as ReactDOM from "react-dom";

import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective, MarkerDirective, Marker,
Legend, Inject } from '@syncfusion/ej2-react-maps';

export function App() {
  return(
    <MapsComponent
      legendSettings={{
        visible: true,
        type: 'Markers',
        useMarkerShape: true,
        toggleLegendSettings: {
          enable: true,
          applyShapeSettings: false,
          border: {
            color: 'green',
            width: 2,
          },
        },
      }}

      <Inject services={[Marker, Legend]} />
      <LayersDirective>
        <LayerDirective urlTemplate="https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-
        Key &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY">
          <MarkersDirective>
            <MarkerDirective
              visible={true}
              dataSource={markerDataSource}
              colorValuePath="color"
              shapeValuePath="shape"
              legendText="country"
            </MarkerDirective>
          </MarkersDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
```

```

</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

```

```
{% endraw %}
```

Other maps in React Maps component

Apart from OpenStreetMap and Bing Maps, you can also render Maps from other online map service providers by specifying the URL provided by those providers in the [urlTemplate](#) property. The URL template concept has been implemented in such a way that any online map service providers using the following template can benefit from previewing their Map in the Syncfusion React Maps component.

```
<!-- markdownlint-disable MD034 -->
```

Sample Template: https://< domain_name >/maps/basic/{z}/{x}/{y}.png

- "\${z}" - It represents zoom factor (level).
- "\${x}" - It indicates tile image x-position (tileX).
- "\${y}" - It indicates tile image y-position (tileY).

In this case, the key generated for those online map service providers can also be appended to the URL. This allows to create personalized Maps with your own content and imagery.

Following is an example of how to add a TomTom map. You can generate an API key by following the steps in this [link](#) and then adding the key to the URL.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent>
    <LayersDirective>
    <LayerDirective urlTemplate=
    "http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key" />
    </LayersDirective>
    </MapsComponent>

```

```
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
```



Enabling zooming and panning

Tile Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

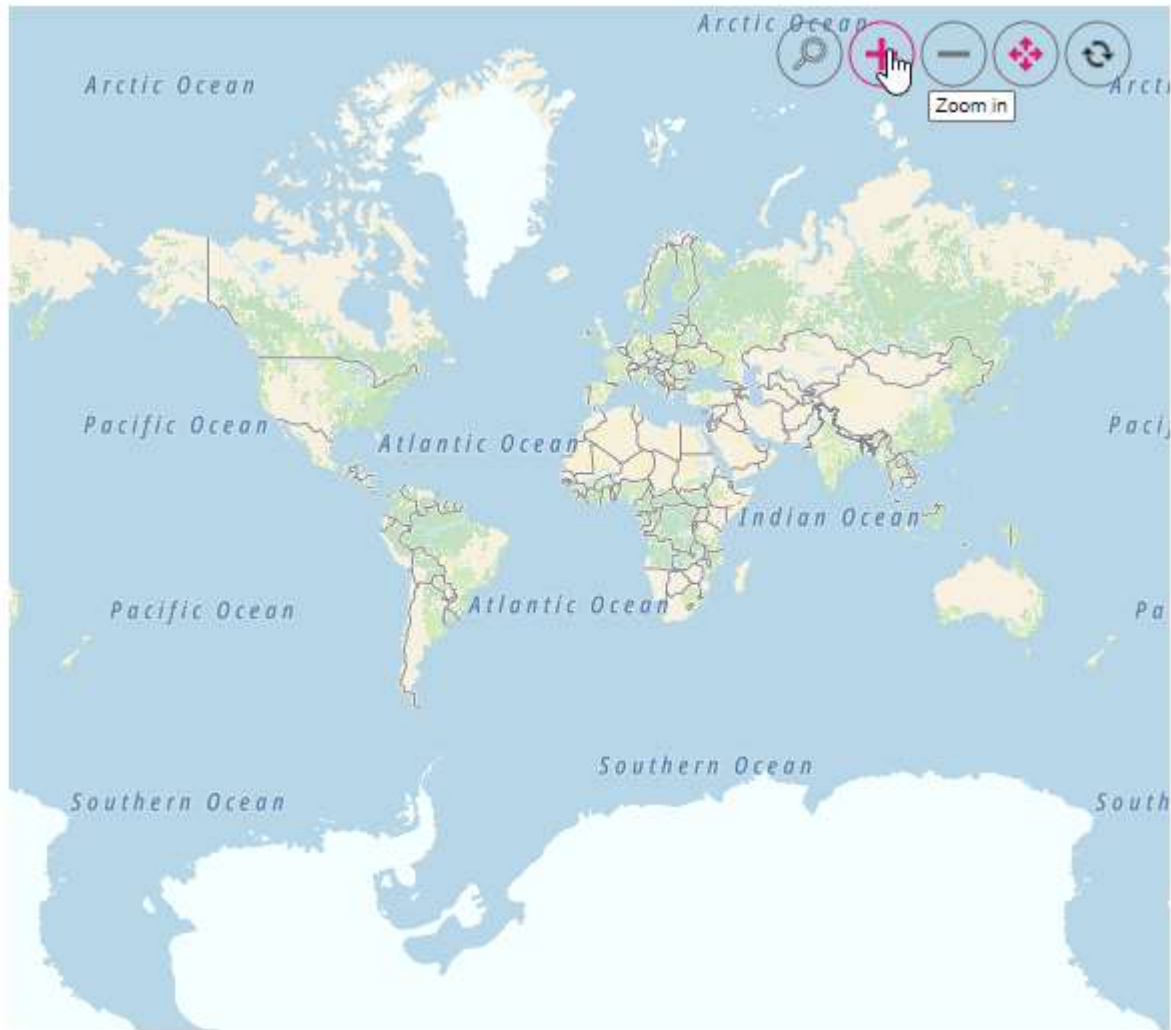
```
{% raw %}
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
```

```
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Maps, Inject } from '@syncfusion/ej2-react-maps';

export function App() {
  return(
    <MapsComponent zoomSettings = { { enable: true, toolbars: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset']} }>
    <Inject services={[[Zoom]]}/>
    <LayersDirective>
    <LayerDirective urlTemplate=
    "http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key" />
    </LayersDirective>
    </MapsComponent>
  );
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

{% endraw %}
```

Adding markers and navigation line

Markers can be added to the layers of tile Maps by setting the corresponding location's coordinates of latitude and longitude using [MarkerDirective](#) tag. Navigation lines can be added on top of an tile Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [NavigationLineDirective](#) tag.

```
{% raw %}
```

```
`ts
```

```
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
```

```
import { MapsComponent, LayersDirective, LayerDirective, Zoom, MarkersDirective, NavigationLine,
NavigationLinesDirective, NavigationLineDirective, MarkerDirective, Marker, Maps, Inject } from
'@syncfusion/ej2-react-maps';
```

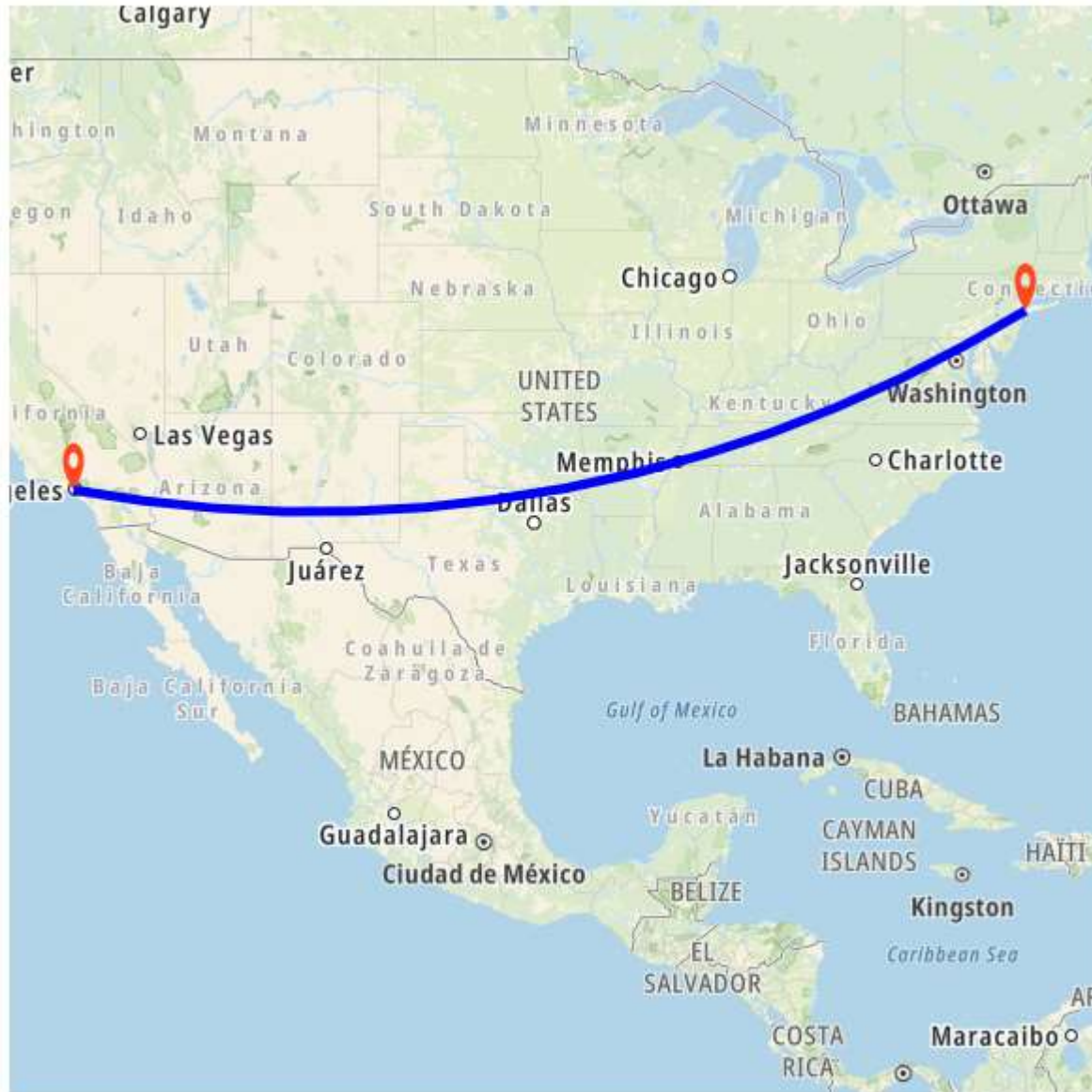
```
export function App() {
```

```
  return(
```

```
<MapsComponent zoomSettings= {{ zoomFactor: 4}}
centerPosition = {{
latitude: 29.394708,
longitude: -94.954653
}}>
<Inject services={[Zoom, NavigationLine, Marker]} />
<LayersDirective>
<LayerDirective urlTemplate=
"http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key">
<MarkersDirective>
<MarkerDirective visible={true}
height={25}
width={15}
dataSource={[
{
latitude: 34.060620,
longitude: -118.330491,
name: "California"
},
{
latitude: 40.724546,
longitude: -73.850344,
name: "New York"
}
]}

</MarkerDirective>
</MarkersDirective>
<NavigationLinesDirective>
<NavigationLineDirective visible={true}
latitude={[34.060620, 40.724546]}
longitude=[[-118.330491,-73.850344]]
color="blue"
```

```
angle={90}
width={5} />
</NavigationLinesDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
、
{% enddraw %}
```



Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the tile Maps layer for highlighting a particular continent or country in tile Maps by adding another layer and specifying the [type](#) property of Maps layer to **SubLayer**.

```
{% raw %}
```

```
`ts
```

```
import { africa_continent } from 'africa-continent.ts';
```

```
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
```

```
import { MapsComponent, LayersDirective, LayerDirective } from '@syncfusion/ej2-react-maps';
```

```
export function App() {
```

```

return(
  <MapsComponent>
    <LayersDirective>
      <LayerDirective urlTemplate=
        "http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key" />
      <LayerDirective shapeData= {africa_continent}
        type= 'SubLayer'
        shapeSettings = {{
          fill: 'blue'
        }}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

{% endraw %}

```



Enabling legend

The legend can be added to the tile Maps by setting the [visible](#) property of [legendSettings](#) to **true**.

```
{% raw %}
```

```
`ts
```

```
import { markerDataSource } from 'markerdata.ts';
```

```
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
```

```
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective, MarkerDirective, Marker,
Legend, Inject } from '@syncfusion/ej2-react-maps';
```

```
export function App() {
```

```
  return(
```

```
    <MapsComponent
```

```
      legendSettings={{
```

```

visible: true,
type: 'Markers',
useMarkerShape: true,
toggleLegendSettings: {
  enable: true,
  applyShapeSettings: false,
  border: {
    color: 'green',
    width: 2,
  },
},
}}

```

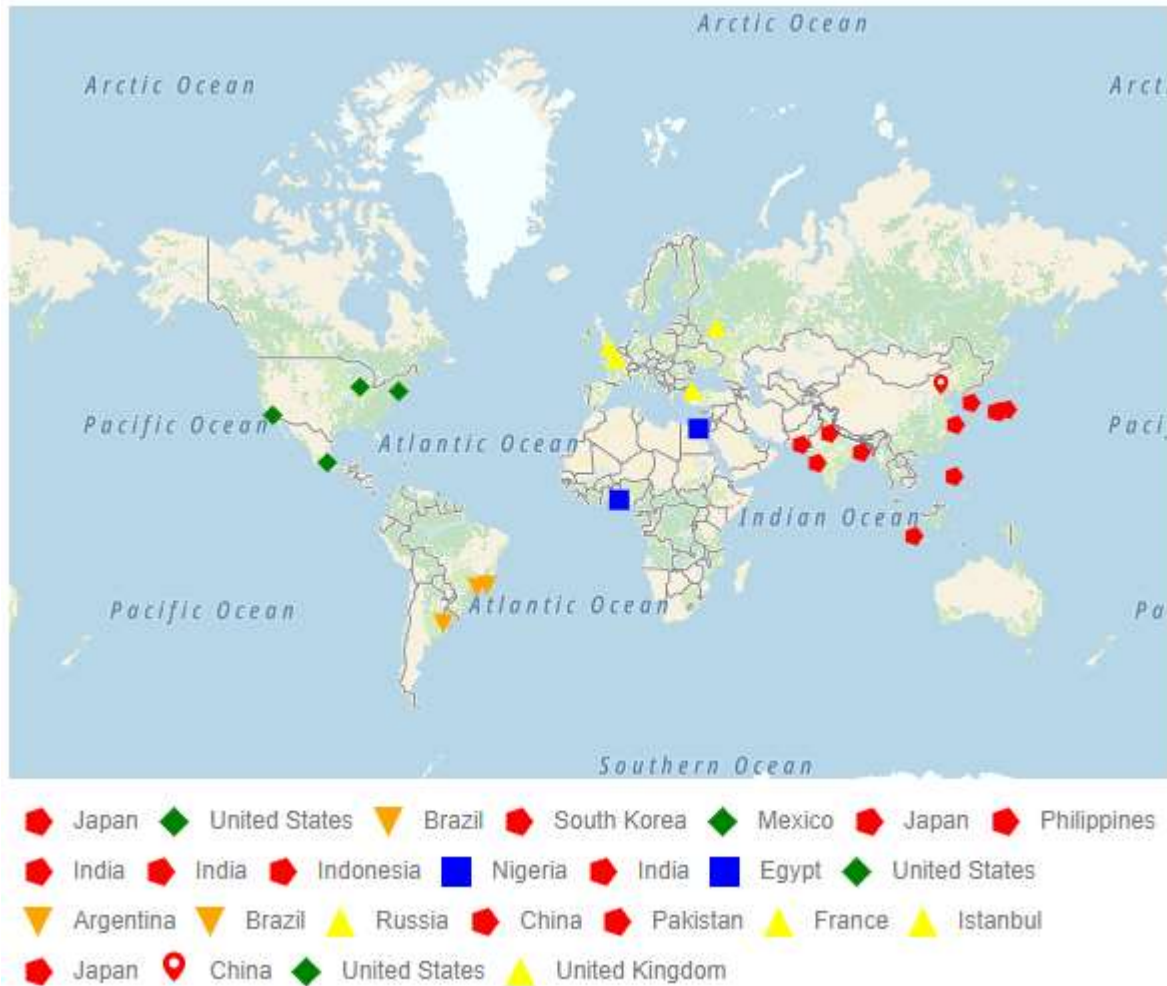
```

<Inject services={[Marker, Legend]} />
<LayersDirective>
  <LayerDirective
    urlTemplate="http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key">
    <MarkersDirective>
      <MarkerDirective
        visible={true}
        dataSource={markerDataSource}
        colorValuePath="color"
        shapeValuePath="shape"
        legendText="country"
      </MarkerDirective>
    </MarkersDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);

```

{% endraw %}



Customization in React Maps component

Setting the size for Maps

The width and height of the Maps can be set using the [width](#) and [height](#) properties in the Maps component. Percentage or pixel values can be used for the height and width values.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent height="200px" width="500px">
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
```



```

                                autofill: true
                                } }>
        </LayerDirective>
    </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent height="200px" width="500px">
            <LayersDirective>
                <LayerDirective shapeData={world_map}
                                shapeSettings={ {
                                    autofill: true
                                } }>
                </LayerDirective>
            </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Maps title

The title for the Maps can be set using the [titleSettings](#). It can be customized using the following properties.

- [alignment](#) - To customize the alignment for the text in the title for the Maps. The possible values are **Center**, **Near** and **Far**.
- [description](#) - To set the description of the title in Maps.
- [text](#) - To set the text for the title in Maps.
- [textStyle](#) - To customize the text of the title in Maps.
- [subtitleSettings](#) - To customize the subtitle for the Maps.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent titleSettings={{
      text: 'Maps Component',
      textStyle: {
        color: 'red',
        fontStyle: 'italic',
        fontWeight: 'regular',
        fontFamily: 'arial',
        size: '14px'
      },
      alignment: 'Center'
    }}>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
            autofill: true
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent titleSettings={{
      text: 'Maps Component',
      textStyle: {
        color: 'red',
        fontStyle: 'italic',
        fontWeight: 'regular',
        fontFamily: 'arial',
        size: '14px'
      },
      alignment: 'Center'
    }}>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
            autofill: true
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```

        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

Setting theme

The Maps component supports following themes.

- Material
- Fabric
- Bootstrap
- Highcontrast
- MaterialDark
- FabricDark
- BootstrapDark
- Bootstrap4
- HighContrastLight
- Tailwind

By default, the Maps are rendered by the **Material** theme. The theme of the Maps component is changed using the [theme](#) property.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent theme="FabricDark">
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
            autofill: true
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}

```

```
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent theme="FabricDark">
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
            autofill: true
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Customizing Maps container

The following properties are available to customize the container in the Maps.

- [background](#) - To apply the background color to the container in the Maps.
- [border](#) - To customize the color, width and opacity of the border of the Maps.
- [margin](#) - To customize the margins of the Maps.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent
      background="#CCD1D1"
      border={{ color: 'green', width: 2}}
      margin={{
        bottom: 10,
        left: 20,
        right: 20,
        top: 10
      }}>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
            autofill: true
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

```

        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent
            background="#CCD1D1"
            border={{ color: 'green', width: 2 }}
            margin={{
                bottom: 10,
                left: 20,
                right: 20,
                top: 10
            }}
        >
            <LayersDirective>
                <LayerDirective shapeData={world_map}
                    shapeSettings={{
                        autofill: true
                    }}
                >
            </LayerDirective>
        </LayersDirective>
    </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Customizing Maps area

By default, the background color of the shape maps is set as **white**. To modify the background color of the Maps area, the [background](#) property in the [mapsArea](#) is used. The border of the Maps area can be customized using the [border](#) property in the [mapsArea](#).

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return (

```

```

    <MapsComponent mapsArea={{
      background: '#CCD1D1',
      border: {
        width: 2,
        color: 'green'
      } }}>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={{
            autofill: true
          }}>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent mapsArea={{
      background: '#CCD1D1',
      border: {
        width: 2,
        color: 'green'
      } }}>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={{
            autofill: true
          }}>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Customizing the shapes

The following properties are available in [shapeSettings](#) to customize the shapes of the Maps.

- [fill](#) - To apply the fill color to the all the shapes.

- [autofill](#) - To apply the palette colors to the shapes if it is set as true.
- [palette](#) - To set the custom palette for the shapes.
- [border](#) - To customize the color, width and opacity of the border of the shapes.
- [dashArray](#) - To define the pattern of dashes and gaps that is applied to the outline of the shapes.
- [opacity](#) - To customize the transparency for the shapes.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
            autofill: true,
            palette: ['#C7DE6C', '#59A076', '#88D0BC',
'#FEA78C', '#FFC557'],
            border: { color: '#FEE1DD', width: 3},
            dashArray: '1',
            opacity: 0.9
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
            autofill: true,
            palette: ['#C7DE6C', '#59A076', '#88D0BC',
'#FEA78C', '#FFC557'],
            border: { color: '#FEE1DD', width: 3},
            dashArray: '1',

```

```

        opacity: 0.9
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Setting color to the shapes from the data source

The color for each shape in the Maps can be set using the [colorValuePath](#) property of [shapeSettings](#). The value for the [colorValuePath](#) property is the field name from the data source of the [shapeSettings](#) which contains the color values.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapePropertyPath="continent" shapeDataPath="continent"
dataSource=[
          { continent: "North America", color: '#71B081' },
          { continent: "South America", color: '#5A9A77' },
          { continent: "Africa", color: '#498770' },
          { continent: "Europe", color: '#39776C' },
          { continent: "Asia", color: '#266665' },
          { continent: "Oceania", color: '#124F5E' } ] ]>
          shapeSettings={ {
            colorValuePath: 'color'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

```



```
import { MapsComponent, LayersDirective, LayerDirective } from
 '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapePropertyPath="continent" shapeDataPath="continent"
  dataSource=[
    { continent: "North America", color: '#71B081' },
    { continent: "South America", color: '#5A9A77' },
    { continent: "Africa", color: '#498770' },
    { continent: "Europe", color: '#39776C' },
    { continent: "Asia", color: '#266665' },
    { continent: "Oceania", color: '#124F5E' }]
    shapeSettings={ {
      colorValuePath: 'color'
    } }>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Applying border to individual shapes

The border of each shape in the Maps can be customized using the [borderColorValuePath](#) and [borderWidthValuePath](#) properties to modify the color and the width of the border respectively. The field name in the data source of the layer which contains the color and the width values must be set in the [borderColorValuePath](#) and [borderWidthValuePath](#) properties respectively. If the values of [borderWidthValuePath](#) and [borderColorValuePath](#) do not match with the field name from the data source, then the color and width of the border will be applied to the shapes using the [border](#) property in the [shapeSettings](#).

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
 '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapePropertyPath="continent" shapeDataPath="continent"
  dataSource=[
    { continent: "North America", color: '#71B081',
borderColor: '#CCFFE5', width: 2 },
    { continent: "South America", color: '#5A9A77',
borderColor: 'red', width: 2 },

```

```

        { continent: "Africa", color: '#498770',
borderColor: '#FFCC99', width: 2 },
        { continent: "Europe", color: '#39776C',
borderColor: '#66B2FF', width: 2 },
        { continent: "Asia", color: '#266665', borderColor:
'#999900', width: 2 },
        { continent: "Oceania", color: '#124F5E',
borderColor: 'blue', width: 2 }
      ]}
      shapeSettings={ {
        borderColorValuePath: 'borderColor',
        borderWidthValuePath: 'width',
        colorValuePath: 'color'
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapePropertyPath="continent" shapeDataPath="continent"
dataSource=[{
  { continent: "North America", color: '#71B081',
borderColor: '#CCFFE5', width: 2 },
  { continent: "South America", color: '#5A9A77',
borderColor: 'red', width: 2 },
  { continent: "Africa", color: '#498770',
borderColor: '#FFCC99', width: 2 },
  { continent: "Europe", color: '#39776C',
borderColor: '#66B2FF', width: 2 },
  { continent: "Asia", color: '#266665', borderColor:
'#999900', width: 2 },
  { continent: "Oceania", color: '#124F5E',
borderColor: 'blue', width: 2 }
}]
        shapeSettings={ {
          borderColorValuePath: 'borderColor',
          borderWidthValuePath: 'width',
          colorValuePath: 'color'
        } }>

```

```

        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Projection type

The Maps component supports the following projection types:

- Mercator
- Equirectangular
- Miller
- Eckert3
- Eckert5
- Eckert6
- Winkel3
- AitOff

By default, the Maps are rendered by the **Mercator** projection type in which the Maps are rendered based on the coordinates. So, the Maps is not stretched. To change the type of projection in the Maps, the [projectionType](#) property is used.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent id="element" projectionType='Miller'>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
            autofill: true,
            palette: ['#33CCFF', '#FF0000', '#800000',
'#FFFF00', '#808000']
          } }>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent id="element" projectionType='Miller'>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings={ {
            autofill: true,
            palette: [ '#33CCFF', '#FF0000', '#800000',
'#FFFF00', '#808000' ]
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Color mapping in React Maps component

Color mapping is used to customize the shape colors based on the given values. It has three types.

1. Range color mapping
2. Equal color mapping
3. Desaturation color mapping.

To add color mapping to the shapes of the Maps, bind the data source to the [dataSource](#) property of [layerSettings](#) and set the field name which contains the color value in the data source to the [colorValuePath](#) property.

Types of color mapping

Range color mapping

Range color mapping applies the color to the shapes of the Maps which matches the numeric values in the data source within the given color mapping ranges. The [from](#) and [to](#) properties in the [colorMapping](#) are used to specify the color mapping ranges in the Maps.

```
`ts
```

```
export let population_density = [
  https://ej2.syncfusion.com/react/documentation.
{
  'code': 'AE',
  'value': 90,
  'name': 'United Arab Emirates',
```

```

'population': 8264070,
'density': 99
},
{
'code': 'GB',
'value': 257,
'name': 'United Kingdom',
'population': 62041708,
'density': 255
},
{
'code': 'US',
'value': 34,
'name': 'United States',
'population': 325020000,
'density': 33
}
...
];
`

```

Bind the **population_density** data to the [dataSource](#) property of [layerSettings](#) and set the [colorValuePath](#) property of [shapeSettings](#) as **density**. The range values can be set using the [from](#) and [to](#) properties of [colorMapping](#).

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { population_density } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath='name '
shapePropertyPath='name '
          dataSource={population_density}
          shapeSettings={ {
            colorValuePath: 'density',
            fill: '#E5E5E5',

```

```

        colorMapping: [
            {
                from: 0.00001, to: 100, color:
'rgb(153,174,214) '
            },
            {
                from: 100, to: 200, color:
'rgb(115,143,199) '
            },
            {
                from: 200, to: 300, color: 'rgb(77,112,184) '
            },
            {
                from: 300, to: 500, color: 'rgb(38,82,168) '
            },
            {
                from: 500, to: 19000, color: 'rgb(0,51,153) '
            }
        ]
    } }>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { population_density } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
            <LayersDirective>
                <LayerDirective shapeData={world_map} shapeDataPath='name'
shapePropertyPath='name'
                    dataSource={population_density}
                    shapeSettings={ {
                        colorValuePath: 'density',
                        fill: '#E5E5E5',
                        colorMapping: [
                            {
                                from: 0.00001, to: 100, color:
'rgb(153,174,214) '
                            },
                            {
                                from: 100, to: 200, color:
'rgb(115,143,199) '

```

```

    },
    {
      from: 200, to: 300, color: 'rgb(77,112,184)'
    },
    {
      from: 300, to: 500, color: 'rgb(38,82,168)'
    },
    {
      from: 500, to: 19000, color: 'rgb(0,51,153)'
    }
  ]
} }>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

Equal color mapping

Equal color mapping applies the color to the shapes of the Maps when the [value](#) property of [colorMapping](#) matches with the values provided in the data source. The following example shows how to apply equal color mapping to the shapes with the data source **unCountries** which illustrates the permanent and non-permanent countries in the UN security council.

`ts

```

export let unCountries: object[] = [
  { Country: 'China', Membership: 'Permanent' },
  { Country: 'France', Membership: 'Permanent' },
  { Country: 'Russia', Membership: 'Permanent' },
  { Country: 'United Kingdom', Membership: 'Permanent' },
  { Country: 'United States', Membership: 'Permanent' },
  { Country: 'Bolivia', Membership: 'Non-Permanent' },
  { Country: 'Eq. Guinea', Membership: 'Non-Permanent' },
  { Country: 'Ethiopia', Membership: 'Non-Permanent' },
  { Country: "Côte d'Ivoire", Membership: 'Permanent' },
  { Country: 'Kazakhstan', Membership: 'Non-Permanent' },
  { Country: 'Kuwait', Membership: 'Non-Permanent' },
  { Country: 'Netherlands', Membership: 'Non-Permanent' },
  { Country: 'Peru', Membership: 'Non-Permanent' },
  { Country: 'Poland', Membership: 'Non-Permanent' },
  { Country: 'Sweden', Membership: 'Non-Permanent' },

```

```
];
`
```

Bind the **unCountries** data to the [dataSource](#) property of [layerSettings](#) and set the [colorValuePath](#) property of [shapeSettings](#) as **Membership**. Set the [value](#) property in the [colorMapping](#) to **Permanent** and **Non-Permanent** in the different set of color mapping properties. If the corresponding value of the [colorValuePath](#) property matches with the corresponding field name in the data source, then the given color will be applied.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name'
          dataSource={uncountries}
          shapeSettings={ {
            fill: '#E5E5E5',
            colorMapping: [
              {
                value: 'Permanent',
                color: '#EDB46F'
              },
              {
                color: '#F1931B',
                value: 'Non-Permanent'
              }
            ],
            colorValuePath: 'Membership'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
```



```
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name'
dataSource={uncountries}
shapeSettings={ {
  fill: '#E5E5E5',
  colorMapping: [
    {
      value: 'Permanent',
      color: '#EDB46F'
    },
    {
      color: '#F1931B',
      value: 'Non-Permanent'
    }
  ],
  colorValuePath: 'Membership'
} } >
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Desaturation color mapping

Desaturation color mapping applies the color to the shapes of the Maps, similar to the range color mapping. The opacity will be applied in this color mapping based on the [minOpacity](#) and [maxOpacity](#) properties in the [colorMapping](#) property.

The following example shows how to apply desaturation color mapping to the shapes with the data source **population_density** that is available in the [Range color mapping](#) section.

Bind the **population_density** data to the [dataSource](#) property of [layerSettings](#) and set the [colorValuePath](#) property of [shapeSettings](#) as **density**. The range values can be set using the [from](#) and [to](#) properties of [colorMapping](#).

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { population_density } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
```

```

        <LayersDirective>
          <LayerDirective shapeData={world_map} shapeDataPath='name'
shapePropertyPath='name'
            dataSource={population_density}
            shapeSettings={ {
              colorValuePath: 'density',
              colorMapping: [
                {
                  from: 0, to: 100, color: 'rgb(153,174,214)',
minOpacity: 0.2, maxOpacity: 1
                },
                {
                  from: 101, to: 260, color:
'rgb(115,143,199)',
minOpacity: 0.3, maxOpacity: 1
                }
              ]
            } }>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { population_density } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath='name'
shapePropertyPath='name'
          dataSource={population_density}
          shapeSettings={ {
            colorValuePath: 'density',
            colorMapping: [
              {
                from: 0, to: 100, color: 'rgb(153,174,214)',
minOpacity: 0.2, maxOpacity: 1
              },
              {
                from: 101, to: 260, color:
'rgb(115,143,199)',
minOpacity: 0.3, maxOpacity: 1
              }
            ]
          } }
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}

```

```

    } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Multiple colors for a single shape

Multiple colors can be added to the color mapping which can be used as gradient effect to a specific shape based on the ranges in the data source. By using the [color](#) property of [colorMapping](#), any number of colors can be set to the shapes as a gradient.

The following example demonstrates how to use multiple colors in color mapping with the data source **population_density** that is available in the [Range color mapping](#) section.

Bind the **population_density** data to the [dataSource](#) property of [layerSettings](#) and set the [colorValuePath](#) property of [shapeSettings](#) as **density**. The range values can be set using the [from](#) and [to](#) properties of [colorMapping](#).

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { population_density } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath='name'
shapePropertyPath='name'
          dataSource={population_density}
          shapeSettings={ {
            colorValuePath: 'density',
            colorMapping: [
              {
                from: 0, to: 100, color: ['red', 'blue']
              },
              {
                from: 101, to: 260, color:
['green', 'yellow']
              }
            ]
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));

```

```
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { population_density } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath='name'
shapePropertyPath='name'
          dataSource={population_density}
          shapeSettings={ {
            colorValuePath: 'density',
            colorMapping: [
              {
                from: 0, to: 100, color: ['red', 'blue']
              },
              {
                from: 101, to: 260, color:
['green', 'yellow']
              }
            ]
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Color for items excluded from color mapping

Color mapping can be applied to the shapes in the Maps which does not match color mapping criteria such as range or equal values using the [color](#) property of [colorMapping](#).

The following example shows how to set the color for items excluded from the color mapping with the data source **population_density** that is available in the [Range color mapping](#) section.

In the following example, color mapping is added for the ranges from 0 to 200. If there are any records in the data source that are outside of this range, the color mapping will not be applied. To apply the color for these excluded items, set the [color](#) property alone in the [colorMapping](#).

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
```

```

import { population_density } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return(
        <MapsComponent >
            <LayersDirective>
                <LayerDirective shapeData={world_map} shapeDataPath='name'
shapePropertyPath='name'
                    dataSource={population_density}
                    shapeSettings={ {
                        colorValuePath: 'density',
                        colorMapping: [
                            {
                                from: 0, to: 100, color: 'skyblue',
                            },
                            {
                                from: 101, to: 200, color: 'blue',
                            },
                            {
                                color: 'green'
                            }
                        ]
                    } }>
            </LayerDirective>
        </LayersDirective>
    </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { population_density } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return(
        <MapsComponent >
            <LayersDirective>
                <LayerDirective shapeData={world_map} shapeDataPath='name'
shapePropertyPath='name'
                    dataSource={population_density}
                    shapeSettings={ {
                        colorValuePath: 'density',
                        colorMapping: [
                            {
                                from: 0, to: 100, color: 'skyblue',
                            },

```

```

        {
            from: 101, to: 200, color: 'blue',
        },
        {
            color: 'green'
        } ]
    } }>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Color mapping for bubbles

The color mapping types such as range color mapping, equal color mapping and desaturation color mapping are applicable for bubbles in the Maps. To add color mapping for bubbles of the Maps, bind the data source to the [dataSource](#) property of [bubbleSettings](#) and set the field name which contains the color value in the data source to the [colorValuePath](#) property. Multiple colors for a single set of bubbles and color for excluded items from [colorMapping](#) are also applicable for bubbles.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
    return(
        <MapsComponent >
            <Inject services={[Bubble]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
                    <BubblesDirective>
                        <BubbleDirective visible={true}
valuePath="population" dataSource=[
                                { name: 'India', population:
'38332521' },
                                { name: 'New Zealand', population:
'19651127' },
                                { name: 'Pakistan', population:
'3090416' } ] ]
                        minRadius={5} colorValuePath=
"population" colorMapping=[
                                { value: '38332521', color:
'#C3E6ED' },
                                { value: '19651127', color:
'#F1931B' },

```

```

        { value: '3090416', color:
'blue' ]}] ]>/>
        </BubblesDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={ [Bubble]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population" dataSource=[
              { name: 'India', population:
'38332521' },
              { name: 'New Zealand', population:
'19651127' },
              { name: 'Pakistan', population:
'3090416' } ] ]
            minRadius={5} colorValuePath=
"population" colorMapping=[ [
              { value: '38332521', color:
'#C3E6ED' },
              { value: '19651127', color:
'#F1931B' },
              { value: '3090416', color:
'blue' } ] ] ]>/>
          </BubblesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);

```

Data label in React Maps component

Data labels provide information to users about the shapes of the Maps component. It can be enabled by setting the [visible](#) property of the [dataLabelSettings](#) to **true**.

Adding data labels

To display data labels in the Maps, the [labelPath](#) property of [dataLabelSettings](#) must be used. The value of the [labelPath](#) property can be taken from the field name in the shape data or data source. In the following example, the value of the [labelPath](#) property is the field name in the shape data of the Maps layer.

INDEX.JSX

```
{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[DataLabel]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings= { {
            autofill: true
          } }
          dataLabelSettings={ {
            visible: true,
            labelPath: 'name'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[DataLabel]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings= { {
            autofill: true
          } }
          dataLabelSettings={ {
            visible: true,
            labelPath: 'name'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```



```

    } }
    dataLabelSettings={ {
      visible: true,
      labelPath: 'name'
    } }>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

In the following example, the value of [labelPath](#) property is set from the field name in the data source of the layer settings.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[DataLabel]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapePropertyPath="name" shapeDataPath="name"
          shapeSettings= { {
            autofill: true
          } }
          dataSource = [
            { "name": "Afghanistan", "value": 53,
"countryCode": "AF", "population": "29863010", "color": "red", "density":
"119", "continent": "Asia" },
            { "name": "Albania", "value": 117,
"countryCode": "AL", "population": "3195000", "color": "Blue", "density":
"111", "continent": "Europe" },
            { "name": "Algeria", "value": 15,
"countryCode": "DZ", "population": "34895000", "color": "Green", "density":
"15", "continent": "Africa" }
          ]
          dataLabelSettings={ {
            visible: true,
            labelPath: 'continent',
            smartLabelMode: 'Trim'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}

```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[DataLabel]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapePropertyPath="name" shapeDataPath="name"
          shapeSettings= { {
            autofill: true
          } }
          dataSource = { [
            { "name": "Afghanistan", "value": 53,
"countryCode": "AF", "population": "29863010", "color": "red", "density":
"119", "continent": "Asia" },
            { "name": "Albania", "value": 117,
"countryCode": "AL", "population": "3195000", "color": "Blue", "density":
"111", "continent": "Europe" },
            { "name": "Algeria", "value": 15,
"countryCode": "DZ", "population": "34895000", "color": "Green", "density":
"15", "continent": "Africa" }
          ] }
          dataLabelSettings={ {
            visible: true,
            labelPath: 'continent',
            smartLabelMode: 'Trim'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Customization

The following properties are available in the [dataLabelSettings](#) to customize the data label of the Maps component.

- [border](#) - To customize the color, width and opacity for the border of the data labels in Maps.
- [fill](#) - To apply the color of the data labels in Maps.
- [opacity](#) - To customize the transparency of the data labels in Maps.

- [textStyle](#) - To customize the text style of the data labels in Maps.

INDEX.JSX

```
{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel,
MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[DataLabel, MapsTooltip]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings= {{
            autofill: true
          }}
          dataLabelSettings={ {
            visible: true,
            smartLabelMode: 'Hide',
            intersectionAction: 'Trim',
            labelPath: 'name',
            border: {
              color: 'green',
              width: 2
            },
            fill: 'transparent',
            opacity: 0.9,
            textStyle: {
              size: '17px',
              fontStyle: 'Sans-serif',
              fontWeight: 'normal'
            }
          } }
          tooltipSettings= {{
            visible: true,
            valuePath: 'name'
          }}
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
```

```
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel,
MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[DataLabel, MapsTooltip]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings= {{
            autofill: true
          }}
          dataLabelSettings={ {
            visible: true,
            smartLabelMode: 'Hide',
            intersectionAction: 'Trim',
            labelPath: 'name',
            border: {
              color: 'green',
              width: 2
            },
            fill: 'transparent',
            opacity: 0.9,
            textStyle: {
              size: '17px',
              fontStyle: 'Sans-serif',
              fontWeight: 'normal'
            }
          } }
          tooltipSettings= {{
            visible: true,
            valuePath: 'name'
          }}
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Label animation

The data labels can be animated during the initial rendering of the Maps. This can be enabled by setting the [animationDuration](#) property in the [dataLabelSettings](#) of the Maps. The duration of the animation is specified in milliseconds.

INDEX.JSX

```
{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel,
MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
```

```

    <MapsComponent >
    <Inject services={[DataLabel, MapsTooltip]} />
    <LayersDirective>
    <LayerDirective shapeData={usa_map}
    shapeSettings= {{
    autofill: true
    }}
    dataLabelSettings={ {
    visible: true,
    smartLabelMode: 'Hide',
    intersectionAction: 'Trim',
    labelPath: 'name',
    animationDuration: 2000
    } }
    tooltipSettings= {{
    visible: true,
    valuePath: 'name'
    }}>
    </LayerDirective>
    </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel,
MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
    <Inject services={[DataLabel, MapsTooltip]} />
    <LayersDirective>
    <LayerDirective shapeData={usa_map}
    shapeSettings= {{
    autofill: true
    }}
    dataLabelSettings={ {
    visible: true,
    smartLabelMode: 'Hide',
    intersectionAction: 'Trim',
    labelPath: 'name',
    animationDuration: 2000
    } }
    tooltipSettings= {{
    visible: true,
    valuePath: 'name'
    }}>
    </LayerDirective>
  );
}

```

```

        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

Smart labels

The Maps component provides an option to handle the labels when they intersect with the corresponding shape borders using the [smartLabelMode](#) property. The following options are available in the [smartLabelMode](#) property.

- None
- Hide
- Trim

INDEX.JSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[DataLabel]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings= { {
            autofill: true
          } }
          dataLabelSettings={ {
            visible: true,
            labelPath: 'name',
            smartLabelMode: 'Hide'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[DataLabel]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings= { {
            autofill: true
          } }
          dataLabelSettings={ {
            visible: true,
            labelPath: 'name',
            smartLabelMode: 'Hide'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Intersect action

The Maps component provides an option to handle the labels when a label intersects with another label using the [intersectionAction](#) property. The following options are available in the [intersectionAction](#) property.

- None
- Hide
- Trim

INDEX.JSX

```
{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[DataLabel]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings= { {
            autofill: true
          } }
          dataLabelSettings= { {
            visible: true,
            labelPath: 'name',

```

```

                                intersectionAction: 'Trim'
                                } }>
        </LayerDirective>
    </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
    return(
        <MapsComponent >
            <Inject services={[DataLabel]} />
            <LayersDirective>
                <LayerDirective shapeData={usa_map}
                                shapeSettings= { {
                                    autofill: true
                                } }
                                dataLabelSettings= { {
                                    visible: true,
                                    labelPath: 'name',
                                    intersectionAction: 'Trim'
                                } }>
                </LayerDirective>
            </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Adding data label as a template

The data label can be added as a template in the Maps component. The [template](#) property of [dataLabelSettings](#) is used to set the data label as a template. Any text or HTML element can be added as the template in data labels.

The properties of data label such as, `smartLabelMode`, `intersectionAction`, `animationDuration`, `border`, `fill`, `opacity` and `textStyle` properties are not applicable to `template` property. The styles can be applied to the label template using the CSS styles of the HTML element.

INDEX.JSX

```

{% raw %}
import { usa_map } from 'usa.ts';

```



```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
    return(
        <MapsComponent >
        <Inject services={[DataLabel]} />
        <LayersDirective>
            <LayerDirective shapeData={usa_map}
shapePropertyPath="name" shapeDataPath="Name"
                dataSource = { [
                    { "Name": "Iowa", "Population":
"29863010" },
                    { "Name": "Utah", "Population":
"1263010" },
                    { "Name": "Texas", "Population":
"963010" }
                ] }
                shapeSettings= {{
                    autofill: true
                }}
                dataLabelSettings={ {
                    visible: true,
                    labelPath: 'Name',
                    template: '<div><div> </div> ${Name}</img></div>'
                } }>
            </LayerDirective>
        </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, DataLabel }
from '@syncfusion/ej2-react-maps';
export function App() {
    return(
        <MapsComponent >
        <Inject services={[DataLabel]} />
        <LayersDirective>
            <LayerDirective shapeData={usa_map}
shapePropertyPath="name" shapeDataPath="Name"
                dataSource = { [
                    { "Name": "Iowa", "Population":
"29863010" },

```

```

    { "Name": "Utah", "Population":
"1263010" },
    { "Name": "Texas", "Population":
"963010" }
  ] }
  shapeSettings= {{
    autofill: true
  }}
  dataLabelSettings={ {
    visible: true,
    labelPath: 'Name',
    template: '<div><div> </div> ${Name}</img></div>'
    } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Polygon shape in React Maps component

The Maps component allows you to add polygon shape to a geometry map or an online map by using the properties in the [polygons](#). This section describes how to add polygon shape to the map and customize them.

Adding polygon shape

To render polygon shape in Maps, **Polygon** module must be injected into the Maps using `<Inject services=[[Polygon]] />` method. The polygon shape can be rendered over the map layer by defining the [points](#) property in the [polygons](#) of the Maps component. The [points](#) property uses a collection of latitude and longitude values to define the polygon shape.

The [polygons](#) provides the following properties for customizing the polygon shape of the Maps component.

- [fill](#) - It is used to change the color of the polygon shape.
- [opacity](#) - It is used to change the opacity of the polygon shape.
- [borderColor](#) - It is used to change the color of the border in the polygon shape.
- [borderWidth](#) - It is used to change the width of the border in the polygon shape.
- [borderOpacity](#) - It is used to change the opacity of the border in the polygon shape.

You can also include “n” polygon shapes using the [polygons](#) property.

The following example shows how to customize the polygon shape over the geometry map.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";

```

```

import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon }
from '@syncfusion/ej2-react-maps';
export function App() {
    var polygonSettings = {
        polygons: [
            {
                points: [
                    { longitude: 34.88539587371454, latitude:
28.181421087099537 },
                    { longitude: 37.50029619722466, latitude:
24.299419888989462 },
                    { longitude: 39.22241423764024, latitude:
22.638529461838658 },
                    { longitude: 38.95650769309776, latitude:
21.424998160017495 },
                    { longitude: 40.19963938650778, latitude:
20.271205391339606 },
                    { longitude: 41.76547269134551, latitude:
18.315451049867193 },
                    { longitude: 42.78452077838921, latitude:
16.097235052947966 },
                    { longitude: 43.36984949591576, latitude:
17.188572054533054 },
                    { longitude: 44.12558191797012, latitude:
17.407258102232234 },
                    { longitude: 46.69027032797584, latitude:
17.33342243475734 },
                    { longitude: 47.09312386141585, latitude:
16.97087769526452 },
                    { longitude: 48.3417299826302, latitude:
18.152700711188004 },
                    { longitude: 49.74762591400318, latitude:
18.81544363931681 },
                    { longitude: 52.41428026336621, latitude:
18.9035706497573 },
                    { longitude: 55.272683129240335, latitude:
20.133861012918544 },
                    { longitude: 55.60121336079203, latitude:
21.92042703112351 },
                    { longitude: 55.08204399107967, latitude:
22.823302662258882 },
                    { longitude: 52.743894337844154, latitude:
22.954463486477437 },
                    { longitude: 51.47035908651375, latitude:
24.35818837668566 },
                    { longitude: 51.122553219055874, latitude:
24.666679732426346 },
                    { longitude: 51.58731671256831, latitude:
25.173806925822717 },
                    { longitude: 51.35950585992913, latitude:
25.84556484481108 },
                    { longitude: 51.088770529661275, latitude:
26.168494193631147 },
                    { longitude: 50.78527056538036, latitude:
25.349051242147596 },

```

```

    { longitude: 50.88330288802666, latitude:
24.779242606720743 },
    { longitude: 50.19702490702369, latitude:
25.66825106363693 },
    { longitude: 50.066461167339924, latitude:
26.268905608606616 },
    { longitude: 49.645896067213215, latitude:
27.15116474192905 },
    { longitude: 48.917371072320805, latitude:
27.55738830340198 },
    { longitude: 48.3984720209192, latitude:
28.566207269716173 },
    { longitude: 47.68851714518985, latitude:
28.5938991332588 },
    { longitude: 47.45059089191449, latitude:
29.009321449856984 },
    { longitude: 44.73549453609391, latitude:
29.157358362696385 },
    { longitude: 41.79487372890989, latitude:
31.23489959729713 },
    { longitude: 40.36977176033773, latitude:
31.9642352513131 },
    { longitude: 39.168270913149826, latitude:
32.18348471414393 },
    { longitude: 37.019253492546454, latitude:
31.47710220862595 },
    { longitude: 37.99644645508337, latitude:
30.4851028633376 },
    { longitude: 37.67756530485232, latitude:
30.3636358598429 },
    { longitude: 37.50181466030105, latitude:
29.960155516804974 },
    { longitude: 36.700288181129594, latitude:
29.882136586478993 },
    { longitude: 36.100009274845206, latitude:
29.15308642012721 },
    { longitude: 34.85774476486728, latitude:
29.3103032832622 },
    { longitude: 34.64498583263142, latitude:
28.135787235699823 },
    { longitude: 34.88539587371454, latitude:
28.181421087099537 }
    ],
    fill: 'red',
    opacity: 0.7,
    borderColor: 'green',
    borderWidth: 2,
    borderOpacity: 0.7
  }
]
};
return (
  <MapsComponent >
    <Inject services={[Polygon]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
polygonSettings={polygonSettings}

```

```

        >
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon }
from '@syncfusion/ej2-react-maps';
export function App() {
  let polygonSettings : object = {
    polygons: [
      {
        points: [
          { longitude: 34.88539587371454, latitude:
28.181421087099537 },
          { longitude: 37.50029619722466, latitude:
24.299419888989462 },
          { longitude: 39.22241423764024, latitude:
22.638529461838658 },
          { longitude: 38.95650769309776, latitude:
21.424998160017495 },
          { longitude: 40.19963938650778, latitude:
20.271205391339606 },
          { longitude: 41.76547269134551, latitude:
18.315451049867193 },
          { longitude: 42.78452077838921, latitude:
16.097235052947966 },
          { longitude: 43.36984949591576, latitude:
17.188572054533054 },
          { longitude: 44.12558191797012, latitude:
17.407258102232234 },
          { longitude: 46.69027032797584, latitude:
17.33342243475734 },
          { longitude: 47.09312386141585, latitude:
16.97087769526452 },
          { longitude: 48.3417299826302, latitude:
18.152700711188004 },
          { longitude: 49.74762591400318, latitude:
18.81544363931681 },
          { longitude: 52.41428026336621, latitude:
18.9035706497573 },
          { longitude: 55.272683129240335, latitude:
20.133861012918544 },
          { longitude: 55.60121336079203, latitude:
21.92042703112351 },

```

```

{ longitude: 55.08204399107967, latitude:
22.823302662258882 },
{ longitude: 52.743894337844154, latitude:
22.954463486477437 },
{ longitude: 51.47035908651375, latitude:
24.35818837668566 },
{ longitude: 51.122553219055874, latitude:
24.666679732426346 },
{ longitude: 51.58731671256831, latitude:
25.173806925822717 },
{ longitude: 51.35950585992913, latitude:
25.84556484481108 },
{ longitude: 51.088770529661275, latitude:
26.168494193631147 },
{ longitude: 50.78527056538036, latitude:
25.349051242147596 },
{ longitude: 50.88330288802666, latitude:
24.779242606720743 },
{ longitude: 50.19702490702369, latitude:
25.66825106363693 },
{ longitude: 50.066461167339924, latitude:
26.268905608606616 },
{ longitude: 49.645896067213215, latitude:
27.15116474192905 },
{ longitude: 48.917371072320805, latitude:
27.55738830340198 },
{ longitude: 48.3984720209192, latitude:
28.566207269716173 },
{ longitude: 47.68851714518985, latitude:
28.5938991332588 },
{ longitude: 47.45059089191449, latitude:
29.009321449856984 },
{ longitude: 44.73549453609391, latitude:
29.157358362696385 },
{ longitude: 41.79487372890989, latitude:
31.23489959729713 },
{ longitude: 40.36977176033773, latitude:
31.9642352513131 },
{ longitude: 39.168270913149826, latitude:
32.18348471414393 },
{ longitude: 37.019253492546454, latitude:
31.47710220862595 },
{ longitude: 37.99644645508337, latitude:
30.4851028633376 },
{ longitude: 37.67756530485232, latitude:
30.3636358598429 },
{ longitude: 37.50181466030105, latitude:
29.960155516804974 },
{ longitude: 36.700288181129594, latitude:
29.882136586478993 },
{ longitude: 36.100009274845206, latitude:
29.15308642012721 },
{ longitude: 34.85774476486728, latitude:
29.3103032832622 },
{ longitude: 34.64498583263142, latitude:
28.135787235699823 },

```

```

        { longitude: 34.88539587371454, latitude:
28.181421087099537 }
      ],
      fill: 'red',
      opacity: 0.7,
      borderColor: 'green',
      borderWidth: 2,
      borderOpacity: 0.7
    }
  ]
};
return (
  <MapsComponent >
    <Inject services={[Polygon]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
        polygonSettings={polygonSettings}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Tooltip

Tooltip is used to display more information about a polygon shape during a mouse or touch interaction. Tooltip and tooltip template can be enabled by setting the [visible](#) property to **true** in the [tooltipSettings](#). Additionally, you need to set the desired content as a value to the [tooltipText](#) property in the `polygons` property to show the tooltip. If you add 'n' numbers of polygon shapes, you can add the `tooltipText` property to each polygon, which will display the tooltip for the associated polygon shape.

Tooltip customization

The following properties are available in the [tooltipSettings](#) to customize the appearance of the tooltip.

- [border](#) - To change the color, width, and opacity of the border of the tooltip in the polygon shape.
- [fill](#) - Applies the color of the tooltip in the polygon shape.
- [textStyle](#) - To change the style of the text in the tooltip of the polygon shape.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon,
MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  var polygonSettings = {
    tooltipSettings: { visible: true, border: { width: 2, color: 'red' }
  },
  polygons: [

```

```

{
  tooltipText: 'Saudi Arabia',
  points: [
    { longitude: 34.88539587371454, latitude: 28.181421087099537 },
    { longitude: 37.50029619722466, latitude: 24.299419888989462 },
    { longitude: 39.22241423764024, latitude: 22.638529461838658 },
    { longitude: 38.95650769309776, latitude: 21.424998160017495 },
    { longitude: 40.19963938650778, latitude: 20.271205391339606 },
    { longitude: 41.76547269134551, latitude: 18.315451049867193 },
    { longitude: 42.78452077838921, latitude: 16.097235052947966 },
    { longitude: 43.36984949591576, latitude: 17.188572054533054 },
    { longitude: 44.12558191797012, latitude: 17.407258102232234 },
    { longitude: 46.69027032797584, latitude: 17.33342243475734 },
    { longitude: 47.09312386141585, latitude: 16.97087769526452 },
    { longitude: 48.3417299826302, latitude: 18.152700711188004 },
    { longitude: 49.74762591400318, latitude: 18.81544363931681 },
    { longitude: 52.41428026336621, latitude: 18.9035706497573 },
    { longitude: 55.272683129240335, latitude: 20.133861012918544 },
    { longitude: 55.60121336079203, latitude: 21.92042703112351 },
    { longitude: 55.08204399107967, latitude: 22.823302662258882 },
    { longitude: 52.743894337844154, latitude: 22.954463486477437 },
    { longitude: 51.47035908651375, latitude: 24.35818837668566 },
    { longitude: 51.122553219055874, latitude: 24.666679732426346 },
    { longitude: 51.58731671256831, latitude: 25.173806925822717 },
    { longitude: 51.35950585992913, latitude: 25.84556484481108 },
    { longitude: 51.088770529661275, latitude: 26.168494193631147 },
    { longitude: 50.78527056538036, latitude: 25.349051242147596 },
    { longitude: 50.88330288802666, latitude: 24.779242606720743 },
    { longitude: 50.19702490702369, latitude: 25.66825106363693 },
    { longitude: 50.066461167339924, latitude: 26.268905608606616 },
  ]
}

```



```

        { longitude: 49.645896067213215, latitude:
27.15116474192905 },
        { longitude: 48.917371072320805, latitude:
27.55738830340198 },
        { longitude: 48.3984720209192, latitude:
28.566207269716173 },
        { longitude: 47.68851714518985, latitude:
28.5938991332588 },
        { longitude: 47.45059089191449, latitude:
29.009321449856984 },
        { longitude: 44.73549453609391, latitude:
29.157358362696385 },
        { longitude: 41.79487372890989, latitude:
31.23489959729713 },
        { longitude: 40.36977176033773, latitude:
31.9642352513131 },
        { longitude: 39.168270913149826, latitude:
32.18348471414393 },
        { longitude: 37.019253492546454, latitude:
31.47710220862595 },
        { longitude: 37.99644645508337, latitude:
30.4851028633376 },
        { longitude: 37.67756530485232, latitude:
30.3636358598429 },
        { longitude: 37.50181466030105, latitude:
29.960155516804974 },
        { longitude: 36.700288181129594, latitude:
29.882136586478993 },
        { longitude: 36.100009274845206, latitude:
29.15308642012721 },
        { longitude: 34.85774476486728, latitude:
29.3103032832622 },
        { longitude: 34.64498583263142, latitude:
28.135787235699823 },
        { longitude: 34.88539587371454, latitude:
28.181421087099537 }
      ],
      fill: 'red',
      opacity: 0.7,
      borderColor: 'green',
      borderWidth: 2,
      borderOpacity: 0.7
    }
  ]
};
return (
  <MapsComponent >
    <Inject services={[Polygon, MapsTooltip]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
polygonSettings={polygonSettings}
      >
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}

```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon,
MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  let polygonSettings : object = {
    tooltipSettings: { visible: true, border: { width: 2, color: 'red' }
  },
  polygons: [
    {
      tooltipText: 'Saudi Arabia',
      points: [
        { longitude: 34.88539587371454, latitude:
28.181421087099537 },
        { longitude: 37.50029619722466, latitude:
24.299419888989462 },
        { longitude: 39.22241423764024, latitude:
22.638529461838658 },
        { longitude: 38.95650769309776, latitude:
21.424998160017495 },
        { longitude: 40.19963938650778, latitude:
20.271205391339606 },
        { longitude: 41.76547269134551, latitude:
18.315451049867193 },
        { longitude: 42.78452077838921, latitude:
16.097235052947966 },
        { longitude: 43.36984949591576, latitude:
17.188572054533054 },
        { longitude: 44.12558191797012, latitude:
17.407258102232234 },
        { longitude: 46.69027032797584, latitude:
17.33342243475734 },
        { longitude: 47.09312386141585, latitude:
16.97087769526452 },
        { longitude: 48.3417299826302, latitude:
18.152700711188004 },
        { longitude: 49.74762591400318, latitude:
18.81544363931681 },
        { longitude: 52.41428026336621, latitude:
18.9035706497573 },
        { longitude: 55.272683129240335, latitude:
20.133861012918544 },
        { longitude: 55.60121336079203, latitude:
21.92042703112351 },
        { longitude: 55.08204399107967, latitude:
22.823302662258882 },
        { longitude: 52.743894337844154, latitude:
22.954463486477437 },

```

```

    { longitude: 51.47035908651375, latitude:
24.35818837668566 },
    { longitude: 51.122553219055874, latitude:
24.666679732426346 },
    { longitude: 51.58731671256831, latitude:
25.173806925822717 },
    { longitude: 51.35950585992913, latitude:
25.84556484481108 },
    { longitude: 51.088770529661275, latitude:
26.168494193631147 },
    { longitude: 50.78527056538036, latitude:
25.349051242147596 },
    { longitude: 50.88330288802666, latitude:
24.779242606720743 },
    { longitude: 50.19702490702369, latitude:
25.66825106363693 },
    { longitude: 50.066461167339924, latitude:
26.268905608606616 },
    { longitude: 49.645896067213215, latitude:
27.15116474192905 },
    { longitude: 48.917371072320805, latitude:
27.55738830340198 },
    { longitude: 48.3984720209192, latitude:
28.566207269716173 },
    { longitude: 47.68851714518985, latitude:
28.5938991332588 },
    { longitude: 47.45059089191449, latitude:
29.009321449856984 },
    { longitude: 44.73549453609391, latitude:
29.157358362696385 },
    { longitude: 41.79487372890989, latitude:
31.23489959729713 },
    { longitude: 40.36977176033773, latitude:
31.9642352513131 },
    { longitude: 39.168270913149826, latitude:
32.18348471414393 },
    { longitude: 37.019253492546454, latitude:
31.47710220862595 },
    { longitude: 37.99644645508337, latitude:
30.4851028633376 },
    { longitude: 37.67756530485232, latitude:
30.3636358598429 },
    { longitude: 37.50181466030105, latitude:
29.960155516804974 },
    { longitude: 36.700288181129594, latitude:
29.882136586478993 },
    { longitude: 36.100009274845206, latitude:
29.15308642012721 },
    { longitude: 34.85774476486728, latitude:
29.3103032832622 },
    { longitude: 34.64498583263142, latitude:
28.135787235699823 },
    { longitude: 34.88539587371454, latitude:
28.181421087099537 }
  ],
  fill: 'red',
  opacity: 0.7,

```

```

        borderColor: 'green',
        borderWidth: 2,
        borderOpacity: 0.7
      }
    ]
  };
  return (
    <MapsComponent >
      <Inject services={[Polygon, MapsTooltip]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          polygonSettings={polygonSettings}>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

Tooltip template

Any HTML element can be rendered in the tooltip of the polygon shapes using the [tooltipTemplate](#) property of the **polygons**. If you add 'n' numbers of polygon shapes, you can add the **tooltipTemplate** property to each polygon, which will display the tooltip for the associated polygon shape.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon,
MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  var polygonSettings = {
    tooltipSettings: { visible: true },
    polygons: [
      {
        tooltipTemplate: '<div style="width:200px;border: 2px solid
#000;padding: 10px;background-color:white;color:black;font-weight:bold;font-
size:15px;"> Country Name : Saudi Arabia</div>',
        points: [
          { longitude: 34.88539587371454, latitude:
28.181421087099537 },
          { longitude: 37.50029619722466, latitude:
24.299419888989462 },
          { longitude: 39.22241423764024, latitude:
22.638529461838658 },
          { longitude: 38.95650769309776, latitude:
21.424998160017495 },
          { longitude: 40.19963938650778, latitude:
20.271205391339606 },

```

```
{ longitude: 41.76547269134551, latitude:
18.315451049867193 },
{ longitude: 42.78452077838921, latitude:
16.097235052947966 },
{ longitude: 43.36984949591576, latitude:
17.188572054533054 },
{ longitude: 44.12558191797012, latitude:
17.407258102232234 },
{ longitude: 46.69027032797584, latitude:
17.33342243475734 },
{ longitude: 47.09312386141585, latitude:
16.97087769526452 },
{ longitude: 48.3417299826302, latitude:
18.152700711188004 },
{ longitude: 49.74762591400318, latitude:
18.81544363931681 },
{ longitude: 52.41428026336621, latitude:
18.9035706497573 },
{ longitude: 55.272683129240335, latitude:
20.133861012918544 },
{ longitude: 55.60121336079203, latitude:
21.92042703112351 },
{ longitude: 55.08204399107967, latitude:
22.823302662258882 },
{ longitude: 52.743894337844154, latitude:
22.954463486477437 },
{ longitude: 51.47035908651375, latitude:
24.35818837668566 },
{ longitude: 51.122553219055874, latitude:
24.666679732426346 },
{ longitude: 51.58731671256831, latitude:
25.173806925822717 },
{ longitude: 51.35950585992913, latitude:
25.84556484481108 },
{ longitude: 51.088770529661275, latitude:
26.168494193631147 },
{ longitude: 50.78527056538036, latitude:
25.349051242147596 },
{ longitude: 50.88330288802666, latitude:
24.779242606720743 },
{ longitude: 50.19702490702369, latitude:
25.66825106363693 },
{ longitude: 50.066461167339924, latitude:
26.268905608606616 },
{ longitude: 49.645896067213215, latitude:
27.15116474192905 },
{ longitude: 48.917371072320805, latitude:
27.55738830340198 },
{ longitude: 48.3984720209192, latitude:
28.566207269716173 },
{ longitude: 47.68851714518985, latitude:
28.5938991332588 },
{ longitude: 47.45059089191449, latitude:
29.009321449856984 },
{ longitude: 44.73549453609391, latitude:
29.157358362696385 },
```

```

        { longitude: 41.79487372890989, latitude:
31.23489959729713 },
        { longitude: 40.36977176033773, latitude:
31.9642352513131 },
        { longitude: 39.168270913149826, latitude:
32.18348471414393 },
        { longitude: 37.019253492546454, latitude:
31.47710220862595 },
        { longitude: 37.99644645508337, latitude:
30.4851028633376 },
        { longitude: 37.67756530485232, latitude:
30.3636358598429 },
        { longitude: 37.50181466030105, latitude:
29.960155516804974 },
        { longitude: 36.700288181129594, latitude:
29.882136586478993 },
        { longitude: 36.100009274845206, latitude:
29.15308642012721 },
        { longitude: 34.85774476486728, latitude:
29.3103032832622 },
        { longitude: 34.64498583263142, latitude:
28.135787235699823 },
        { longitude: 34.88539587371454, latitude:
28.181421087099537 }
    ],
    fill: 'red',
    opacity: 0.7,
    borderColor: 'green',
    borderWidth: 2,
    borderOpacity: 0.7
  }
]
};
return (
  <MapsComponent >
    <Inject services={[Polygon, MapsTooltip]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
polygonSettings={polygonSettings}
      >
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```

import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon,
MapsTooltip } from '@syncfusion/ej2-react-maps';
export function App() {
  let polygonSettings : object = {
    tooltipSettings: { visible: true },
    polygons: [
      {
        tooltipTemplate: '<div style="width:200px;border: 2px solid
#000;padding: 10px;background-color:white;color:black;font-weight:bold;font-
size:15px;"> Country Name : Saudi Arabia</div>',
        points: [
          { longitude: 34.88539587371454, latitude:
28.181421087099537 },
          { longitude: 37.50029619722466, latitude:
24.299419888989462 },
          { longitude: 39.22241423764024, latitude:
22.638529461838658 },
          { longitude: 38.95650769309776, latitude:
21.424998160017495 },
          { longitude: 40.19963938650778, latitude:
20.271205391339606 },
          { longitude: 41.76547269134551, latitude:
18.315451049867193 },
          { longitude: 42.78452077838921, latitude:
16.097235052947966 },
          { longitude: 43.36984949591576, latitude:
17.188572054533054 },
          { longitude: 44.12558191797012, latitude:
17.407258102232234 },
          { longitude: 46.69027032797584, latitude:
17.33342243475734 },
          { longitude: 47.09312386141585, latitude:
16.97087769526452 },
          { longitude: 48.3417299826302, latitude:
18.152700711188004 },
          { longitude: 49.74762591400318, latitude:
18.81544363931681 },
          { longitude: 52.41428026336621, latitude:
18.9035706497573 },
          { longitude: 55.272683129240335, latitude:
20.133861012918544 },
          { longitude: 55.60121336079203, latitude:
21.92042703112351 },
          { longitude: 55.08204399107967, latitude:
22.823302662258882 },
          { longitude: 52.743894337844154, latitude:
22.954463486477437 },
          { longitude: 51.47035908651375, latitude:
24.35818837668566 },
          { longitude: 51.122553219055874, latitude:
24.666679732426346 },
          { longitude: 51.58731671256831, latitude:
25.173806925822717 },
          { longitude: 51.35950585992913, latitude:
25.84556484481108 },
          { longitude: 51.088770529661275, latitude:
26.168494193631147 },

```

```

    { longitude: 50.78527056538036, latitude:
25.349051242147596 },
    { longitude: 50.88330288802666, latitude:
24.779242606720743 },
    { longitude: 50.19702490702369, latitude:
25.66825106363693 },
    { longitude: 50.066461167339924, latitude:
26.268905608606616 },
    { longitude: 49.645896067213215, latitude:
27.15116474192905 },
    { longitude: 48.917371072320805, latitude:
27.55738830340198 },
    { longitude: 48.3984720209192, latitude:
28.566207269716173 },
    { longitude: 47.68851714518985, latitude:
28.5938991332588 },
    { longitude: 47.45059089191449, latitude:
29.009321449856984 },
    { longitude: 44.73549453609391, latitude:
29.157358362696385 },
    { longitude: 41.79487372890989, latitude:
31.23489959729713 },
    { longitude: 40.36977176033773, latitude:
31.9642352513131 },
    { longitude: 39.168270913149826, latitude:
32.18348471414393 },
    { longitude: 37.019253492546454, latitude:
31.47710220862595 },
    { longitude: 37.99644645508337, latitude:
30.4851028633376 },
    { longitude: 37.67756530485232, latitude:
30.3636358598429 },
    { longitude: 37.50181466030105, latitude:
29.960155516804974 },
    { longitude: 36.700288181129594, latitude:
29.882136586478993 },
    { longitude: 36.100009274845206, latitude:
29.15308642012721 },
    { longitude: 34.85774476486728, latitude:
29.3103032832622 },
    { longitude: 34.64498583263142, latitude:
28.135787235699823 },
    { longitude: 34.88539587371454, latitude:
28.181421087099537 }
  ],
  fill: 'red',
  opacity: 0.7,
  borderColor: 'green',
  borderWidth: 2,
  borderOpacity: 0.7
}
]
};
return (
  <MapsComponent >
    <Inject services={[Polygon, MapsTooltip]} />
    <LayersDirective>

```



```

        <LayerDirective shapeData={world_map}
          polygonSettings={polygonSettings}>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Markers in React Maps component

Markers are notes that are used to leave a message on the Maps. It indicates or marks a specific location with desired symbols on the Maps. It can be enabled by setting the [visible](#) property of the [markerSettings](#) to **true**.

Adding marker

To add the markers, the [dataSource](#) property of the [markerSettings](#) has a list of objects that contains the data for markers. Using this property, any number of markers can be added to the layers of the Maps. By default, it displays the markers based on the specified latitude and longitude in the given data source. Each data source object contains the following list of properties.

- latitude - The latitude point which determines the X location of the marker.
- longitude - The longitude point which determines the Y location of the marker.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              height={20}
              width={20}
              animationDuration={0}
              dataSource=[
                { latitude:
49.95121990866204, longitude: 18.468749999999998 },
                { latitude:
59.88893689676585, longitude: -109.3359375 },
                { latitude: -
6.64607562172573, longitude: -55.54687499999999 }
              ]>
            </MarkerDirective>
          </MarkersDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}

```

```

        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              height={20}
              width={20}
              animationDuration={0}
              dataSource=[
                { latitude:
49.95121990866204, longitude: 18.468749999999998 },
                { latitude:
59.88893689676585, longitude: -109.3359375 },
                { latitude: -
6.64607562172573, longitude: -55.54687499999999 }
              ]>
              </MarkerDirective>
            </MarkersDirective>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Adding marker template

The Marker can be added as a template in the Maps component. The [template](#) property of the [markerSettings](#) is used to set the HTML element or id of an element as a template.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';

```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
        <Inject services={[Marker]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}>
                <MarkersDirective>
                    <MarkerDirective visible={true}
                        height={30}
                        width={30}
                        template='<div
id="marker1">Europe</div>'
                        dataSource=[[{ latitude:
49.95121990866204, longitude: 18.468749999999998 }]]
                    >
                    </MarkerDirective>
                    <MarkerDirective visible={true}
                        height={30}
                        width={30}
                        template='<div
id="marker2">North America</div>'
                        dataSource=[[{ latitude:
59.88893689676585, longitude: -109.3359375 }]]
                    >
                    </MarkerDirective>
                    <MarkerDirective visible={true}
                        height={30}
                        width={30}
                        template='<div
id="marker3">South America</div>'
                        dataSource=[[{ latitude: -
6.64607562172573, longitude: -55.54687499999999 }]]
                    >
                    </MarkerDirective>
                </MarkersDirective>
            </LayerDirective>
        </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';

```

```

export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              height={30}
              width={30}
              template='<div
id="marker1">Europe</div>'
              dataSource=[[{ latitude:
49.95121990866204, longitude: 18.468749999999998 }]]
            >
          </MarkerDirective>
            <MarkerDirective visible={true}
              height={30}
              width={30}
              template='<div
id="marker2">North America</div>'
              dataSource=[[{ latitude:
59.88893689676585, longitude: -109.3359375 }]]
            >
          </MarkerDirective>
            <MarkerDirective visible={true}
              height={30}
              width={30}
              template='<div
id="marker3">South America</div>'
              dataSource=[[{ latitude: -
6.64607562172573, longitude: -55.54687499999999 }]]
            >
          </MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Customization

The following properties are available in [markerSettings](#) to customize the Markers of the Maps component.

- [border](#) - To customize the color, width and opacity of the border for the markers in Maps.
- [fill](#) - To apply the color for markers in Maps.
- [dashArray](#) - To define the pattern of dashes and gaps that is applied to the outline of the markers in Maps.
- [height](#) - To customize the height of the markers in Maps.
- [width](#) - To customize the width of the markers in Maps.

- [offset](#) - To customize the position of the markers in Maps.
- [opacity](#) - To customize the transparency of the markers in Maps.
- [animationDelay](#) - To change the time delay in the transition for markers.
- [animationDuration](#) - To change the time duration of animation for markers.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              height={20}
              width={20}
              animationDuration={1000}
              animationDelay={100}
              opacity={0.9}
              shape="Balloon"
              dashArray="1"
              border={{
                color: 'green',
                width: 2
              }}
              dataSource={[
                { latitude: 37.0000,
longitude: -120.0000, city: 'California' },
                { latitude: 40.7127,
longitude: -74.0059, city: 'New York' },
                { latitude: 42, longitude: -
93, city: 'Iowa' }
              ]}>
              </MarkerDirective>
            </MarkersDirective>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
}{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
        <Inject services={[Marker]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}>
                <MarkersDirective>
                    <MarkerDirective visible={true}
                        height={20}
                        width={20}
                        animationDuration={1000}
                        animationDelay={100}
                        opacity={0.9}
                        shape="Balloon"
                        dashArray="1"
                        border={{
                            color: 'green',
                            width: 2
                        }}
                        dataSource=[
longitude: -120.0000, city: 'California' },
                        { latitude: 37.0000,
longitude: -74.0059, city: 'New York' },
                        { latitude: 40.7127,
longitude: 93, city: 'Iowa' }
                        { latitude: 42, longitude: -
                    ]}>
                </MarkerDirective>
            </MarkersDirective>
        </LayerDirective>
    </LayersDirective>
</MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

Marker shapes

The Maps component supports the following marker shapes. To set the shape of the marker, the [shape](#) property in [markerSettings](#) is used.

- Balloon
- Circle
- Cross
- Diamond
- Image
- Rectangle
- Start
- Triangle

- VerticalLine
- HorizontalLine

Rendering marker shape as image

To render a marker as an image in Maps, set the [shape](#) property of [markerSettings](#) as **Image** and specify the path of the image to [imageUrl](#) property. There is another way to render a marker as an image using the [imageUrlValuePath](#) property of the [markerSettings](#). Bind the field name that contains the path of the image in the data source to the [imageUrlValuePath](#) property.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              shape="Image"
              imageUrl="https://ej2.syncfusion.com/react/demos/src/maps/images/ballon.png"
              height={10}
              width={10}
              dataSource=[
                { latitude: 37.0000,
longitude: -120.0000, city: 'California' },
                { latitude: 40.7127,
longitude: -74.0059, city: 'New York' },
                { latitude: 42, longitude: -
93, city: 'Iowa' }
              ]>
          </MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
```

```

import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
        <Inject services={[Marker]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}>
                <MarkersDirective>
                    <MarkerDirective visible={true}
                        shape="Image"
                        imageUrl="https://ej2.syncfusion.com/react/demos/src/maps/images/ballon.png"
                        height={10}
                        width={10}
                        dataSource=[
                            { latitude: 37.0000,
                                longitude: -120.0000, city: 'California' },
                            { latitude: 40.7127,
                                longitude: -74.0059, city: 'New York' },
                            { latitude: 42, longitude: -
                                93, city: 'Iowa' }
                        ]>
                    </MarkerDirective>
                </MarkersDirective>
            </LayerDirective>
        </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Multiple marker groups

Multiple groups of markers can be added to the Maps by adding multiple [MarkerDirective](#) in the [MarkersDirective](#) and the customization for the markers can be done with the [MarkerDirective](#).

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
        <Inject services={[Marker]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}>
                <MarkersDirective>
                    <MarkerDirective visible={true} shape= "Diamond"
                        height={15} fill="green" width={15}
                        dataSource=[

```



```

{ latitude: 37.0000, longitude:
-120.0000, name: 'California'},
{ latitude: 40.7127, longitude:
-74.0059, name: "New York" },
{ latitude: 42, longitude: -93,
name: 'Iowa' }
    ]}>
    </MarkerDirective>
    <MarkerDirective visible={true} height={10}
width={10} fill="blue" shape= "Circle"
    dataSource=[
        { latitude: 19.228825,
longitude: 72.854118, name: "Mumbai"},
        { latitude: 28.610001,
longitude: 77.230003, name: "Delhi"},
        { latitude: 13.067439,
longitude: 80.237617, name: "Chennai"}
    ]}>
    </MarkerDirective>
    </MarkersDirective>
    </LayerDirective>
    </LayersDirective>
    </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
        <Inject services={[Marker]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}>
                <MarkersDirective>
                    <MarkerDirective visible={true} shape= "Diamond"
height={15} fill="green" width={15}
                    dataSource=[
                        { latitude: 37.0000, longitude:
-120.0000, name: 'California'},
                        { latitude: 40.7127, longitude:
-74.0059, name: "New York" },
                        { latitude: 42, longitude: -93,
name: 'Iowa' }
                    ]}>
                </MarkerDirective>

```

```

        <MarkerDirective visible={true} height={10}
width={10} fill="blue" shape= "Circle"
                                dataSource={[
                                { latitude: 19.228825,
longitude: 72.854118, name: "Mumbai"},
                                { latitude: 28.610001,
longitude: 77.230003, name: "Delhi"},
                                { latitude: 13.067439,
longitude: 80.237617, name: "Chennai"}
                                ]}>
        </MarkerDirective>
        </MarkersDirective>
        </LayerDirective>
        </LayersDirective>
    </MapsComponent>

    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Customize marker shapes from data source

Bind different colors and shapes to the marker from data source

Using the [shapeValuePath](#) and [colorValuePath](#) properties, the color and shape of the marker can be applied from the given data source. Bind the data source to the [dataSource](#) property of the [markerSettings](#) and set the field names that contains the shape and color values in the data source to the [shapeValuePath](#) and [colorValuePath](#) properties.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
        <Inject services={[Marker]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}>
                <MarkersDirective>
                    <MarkerDirective visible={true}
                                shape= {'Circle'}
shapeValuePath= {'shape'} colorValuePath= {'color'}
                                dataSource={[
                                { latitude:
49.95121990866204, longitude: 18.468749999999998, name: 'Europe',
color: 'red', shape: 'Triangle' },
                                { latitude:
59.88893689676585, longitude: -109.3359375, name: 'North America',
color: 'blue',
shape: 'Pentagon' },

```

```

        { latitude: -
6.64607562172573, longitude: -55.54687499999999, name: 'South America',
        color: 'green',
shape: 'InvertedTriangle' }
        ]]
      >
    </MarkerDirective>
  </MarkersDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              shape= {'Circle'}
shapeValuePath= {'shape'} colorValuePath= {'color'}
              dataSource=[
                { latitude:
49.95121990866204, longitude: 18.468749999999998, name: 'Europe',
color: 'red', shape: 'Triangle' },
                { latitude:
59.88893689676585, longitude: -109.3359375, name: 'North America',
color: 'blue',
shape: 'Pentagon' },
                { latitude: -
6.64607562172573, longitude: -55.54687499999999, name: 'South America',
color: 'green',
shape: 'InvertedTriangle' }
              ]]
            >
          </MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}

```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Setting value path from the data source

The latitude and longitude values are used to determine the location of each marker in the Maps. The [latitudeValuePath](#) and [longitudeValuePath](#) properties are used to specify the value path that presents in the data source of the marker. In the following example, the field name from the data source is set to the [latitudeValuePath](#) and [longitudeValuePath](#) properties.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              latitudeValuePath = "latitude"
              longitudeValuePath= "longitude"
              dataSource=[
                { latitude:
49.95121990866204, longitude: 18.468749999999998 },
                { latitude:
59.88893689676585, longitude: -109.3359375},
                { latitude: -
6.64607562172573, longitude: -55.54687499999999 }
              ]>
            </MarkerDirective>
          </MarkersDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
```

```

export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              latitudeValuePath = "latitude"
              longitudeValuePath= "longitude"
              dataSource=[
                { latitude:
49.95121990866204, longitude: 18.468749999999998 },
                { latitude:
59.88893689676585, longitude: -109.3359375},
                { latitude: -
6.64607562172573, longitude: -55.54687499999999 }
              ]>
            </MarkerDirective>
          </MarkersDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Repositioning the marker using drag and drop

The markers on the map can be dragged and dropped to change their position. To enable marker drag and drop, set the [enableDrag](#) property to **true** in the [markerSettings](#) property.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MapsTooltip,
Inject, MarkersDirective, MarkerDirective, Marker } from '@syncfusion/ej2-react-maps';
export function App() {
  return (<div><MapsComponent>
    <Inject services={[MapsTooltip, Marker]} />
    <LayersDirective>
      <LayerDirective
        shapeData={world_map}
        shapeSettings={{
          fill: '#C3E6ED',
        }}
      >
      <MarkersDirective>
        <MarkerDirective
          enableDrag={true}
          visible={true}

```

```

        animationDuration={0}
        dataSource={[
          {
            latitude: 49.95121990866204,
            longitude: 18.468749999999998,
            name: 'MarkerOne',
          },
          {
            latitude: 59.88893689676585,
            longitude: -109.3359375,
            name: 'MarkerTwo',
          },
          {
            latitude: -6.64607562172573,
            longitude: -55.54687499999999,
            name: 'MarkerThree',
          },
          {
            latitude: 23.644385824912135,
            longitude: 77.83189239539234,
            name: 'MarkerFour',
          },
          {
            latitude: 63.66569332894224,
            longitude: 98.2225173953924,
            name: 'MarkerFive',
          },
        ]}
        shape="Balloon"
        width={20}
        height={20}
        border={{ width: 2, color: '#285255' }}
        tooltipSettings={{
          visible: true,
          valuePath: 'name',
        }}
      </MarkerDirective>
    </MarkersDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent></div>;
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,

```

```

MapsTooltip,
Inject,
MarkersDirective,
MarkerDirective,
Marker,
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <div><MapsComponent>
      <Inject services={[MapsTooltip, Marker]} />
      <LayersDirective>
        <LayerDirective
          shapeData={world_map}
          shapeSettings={{
            fill: '#C3E6ED',
          }}
        >
          <MarkersDirective>
            <MarkerDirective
              enableDrag={true}
              visible={true}
              animationDuration={0}
              dataSource={[
                {
                  latitude: 49.95121990866204,
                  longitude: 18.468749999999998,
                  name: 'MarkerOne',
                },
                {
                  latitude: 59.88893689676585,
                  longitude: -109.3359375,
                  name: 'MarkerTwo',
                },
                {
                  latitude: -6.64607562172573,
                  longitude: -55.54687499999999,
                  name: 'MarkerThree',
                },
                {
                  latitude: 23.644385824912135,
                  longitude: 77.83189239539234,
                  name: 'MarkerFour',
                },
                {
                  latitude: 63.66569332894224,
                  longitude: 98.2225173953924,
                  name: 'MarkerFive',
                },
              ]}
              shape="Balloon"
              width={20}
              height={20}
              border={{ width: 2, color: '#285255' }}
              tooltipSettings={{
                visible: true,
                valuePath: 'name',
              }}
            >

```

```

        </MarkerDirective>
      </MarkersDirective>
    </LayerDirective>
  </LayersDirective>
</MapsComponent></div>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

The data of the drag and dropped marker in the marker data source can be customized using the [markerDragStart](#) and [markerDragEnd](#) events. When you change the appropriate marker data, the tooltip and legend item text of that marker are automatically updated. The following properties are available in the event argument of the marker drag events.

Argument Name	Description
dataIndex	It represents the index of the data of the dragged marker in the marker data source.
latitude	It represents the latitude coordinate point of the dragged marker.
longitude	It represents the longitude coordinate point for the dragged marker.
markerIndex	It represents the index of the marker setting.
layerIndex	It represents the index of the layer in which the marker belongs.
name	It represents the name of the event.
x	It represents the horizontal location of the mouse pointer on the map when the drag action is performed.
y	It represents the vertical location of the mouse pointer on the map when the drag action is performed.

The following example shows how to use marker drag events to customize the data of the drag and dropped marker in the marker data source.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MapsTooltip,
Inject, MarkersDirective, MarkerDirective, Marker, Legend } from
'@syncfusion/ej2-react-maps';
export function App() {
  var mapInstance;
  function onMarkerDragStart(args) {

```



```

    // When the marker begins to move on the map, the event is
    triggered.
    };
    function onMarkerDragEnd(args) {
        // When the marker on the map stops dragging, the event is
        triggered.

mapInstance.layers[args.layerIndex].markerSettings[args.markerIndex].dataSou
rce[args.dataIndex].name = 'Dragged Marker ' + (args.dataIndex + 1);
        mapInstance.refresh();
    };
    return(<div><MapsComponent ref={m => mapInstance = m}
markerDragStart={onMarkerDragStart.bind(this)}
markerDragEnd={onMarkerDragEnd.bind(this)} legendSettings={{
    visible: true,
    type: 'Markers',
    shape: 'Circle',
    shapeWidth: 10,
    shapeHeight: 10,
    fill: '#FF471A',
    shapeBorder: { width: 2, color: '#285255' },
}}>
    <Inject services={[MapsTooltip, Marker, Legend]} />
    <LayersDirective>
        <LayerDirective
            shapeData={world_map}
            shapeSettings={{
                fill: '#C3E6ED',
            }}
        >
            <MarkersDirective>
                <MarkerDirective
                    enableDrag={true}
                    legendText="name"
                    visible={true}
                    animationDuration={0}
                    dataSource=[
                        {
                            latitude: 49.95121990866204,
                            longitude: 18.468749999999998,
                            name: 'MarkerOne',
                        },
                        {
                            latitude: 59.88893689676585,
                            longitude: -109.3359375,
                            name: 'MarkerTwo',
                        },
                        {
                            latitude: -6.64607562172573,
                            longitude: -55.54687499999999,
                            name: 'MarkerThree',
                        },
                        {
                            latitude: 23.644385824912135,
                            longitude: 77.83189239539234,
                            name: 'MarkerFour',
                        },
                    ],
                </MarkerDirective>
            </MarkersDirective>
        </LayerDirective>
    </LayersDirective>
</div>

```

```

        {
          latitude: 63.66569332894224,
          longitude: 98.2225173953924,
          name: 'MarkerFive',
        },
      ],
      shape: "Balloon"
      width={20}
      height={20}
      border={{ width: 2, color: '#285255' }}
      tooltipSettings={{
        visible: true,
        valuePath: 'name',
      }}
    </MarkerDirective>
  </MarkersDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
  MapsTooltip,
  Inject,
  Legend,
  MarkersDirective,
  MarkerDirective,
  IMarkerDragEvents,
  Marker
} from '@syncfusion/ej2-react-maps';
export function App() {
  let mapInstance: MapsComponent;
  function onMarkerDragStart(args: IMarkerDragEvents): void {
    // When the marker begins to move on the map, the event is
    triggered.
  }
  function onMarkerDragEnd(args: IMarkerDragEvents) : void {
    // When the marker on the map stops dragging, the event is
    triggered.
    mapInstance.layers[args.layerIndex].markerSettings[
      args.markerIndex
    ].dataSource[args.dataIndex].name =
      'Dragged Marker ' + (args.dataIndex + 1);
    mapInstance.refresh();
  }
}

```

```

    }
    return (
      <div><MapsComponent ref={m => mapInstance = m}
        markerDragStart={onMarkerDragStart.bind(this)}
        markerDragEnd={onMarkerDragEnd.bind(this)} legendSettings={{
          visible: true,
          type: 'Markers',
          shape: 'Circle',
          shapeWidth: 10,
          shapeHeight: 10,
          fill: '#FF471A',
          shapeBorder: { width: 2, color: '#285255' },
        }}>
        <Inject services={[MapsTooltip, Marker, Legend]} />
        <LayersDirective>
          <LayerDirective
            shapeData={world_map}
            shapeSettings={{
              fill: '#C3E6ED',
            }}
          >
          <MarkersDirective>
            <MarkerDirective
              enableDrag={true}
              legendText="name"
              visible={true}
              animationDuration={0}
              dataSource={[
                {
                  latitude: 49.95121990866204,
                  longitude: 18.468749999999998,
                  name: 'MarkerOne',
                },
                {
                  latitude: 59.88893689676585,
                  longitude: -109.3359375,
                  name: 'MarkerTwo',
                },
                {
                  latitude: -6.64607562172573,
                  longitude: -55.54687499999999,
                  name: 'MarkerThree',
                },
                {
                  latitude: 23.644385824912135,
                  longitude: 77.83189239539234,
                  name: 'MarkerFour',
                },
                {
                  latitude: 63.66569332894224,
                  longitude: 98.2225173953924,
                  name: 'MarkerFive',
                },
              ]}
              shape="Balloon"
              width={20}
              height={20}
            >

```

```

        border={{ width: 2, color: '#285255' }}
        tooltipSettings={{
          visible: true,
          valuePath: 'name',
        }}
      ></MarkerDirective>
    </MarkersDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent></div>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Marker zooming

The Maps can be initially scaled to the center value based on the marker distance. This can be achieved by setting the [shouldZoomInitially](#) property in [zoomSettings](#) as **true**.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject, Zoom } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings= {{enable: true,
horizontalAlignment:'Near', shouldZoomInitially: true}}>
      <Inject services={[Marker, Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              shape= {'Circle'}
              dataSource={[
                { latitude:
49.95121990866204, longitude: 18.468749999999998, name:'Europe' },
                { latitude:
59.88893689676585, longitude: -109.3359375, name:'North America' },
                { latitude: -
6.64607562172573, longitude: -55.54687499999999, name:'South America'}
              ]}
            >
          </MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);

```

```
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject, Zoom } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings= {{enable: true,
horizontalAlignment:'Near', shouldZoomInitially: true}}>
      <Inject services={[Marker, Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              shape= {'Circle'}
              dataSource=[
                { latitude:
49.95121990866204, longitude: 18.468749999999998, name:'Europe' },
                { latitude:
59.88893689676585, longitude: -109.3359375, name:'North America' },
                { latitude: -
6.64607562172573, longitude: -55.54687499999999, name:'South America' }
              ]
            >
          </MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Marker clustering

Maps provide support to cluster the markers when they overlap each other. The number on a cluster indicates how many overlapped markers it contains. If zooming is performed on any of the cluster locations in Maps, the number on the cluster will decrease, and the individual markers will be seen on the map. When zooming out, the overlapping marker will increase. So that it can cluster again and increase the count over the cluster.

To enable clustering in markers, set the [allowClustering](#) property of [markerClusterSettings](#) as **true** and customization of clustering can be done with the [markerClusterSettings](#).

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { cluster } from 'marker-cluster.ts';
```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject, MapsTooltip, Zoom } from '@syncfusion/ej2-
react-maps';
export function App() {
    return (
        <MapsComponent useGroupingSeparator={true} format='n'
zoomSettings={{ enable: true }} titleSettings={{ text: 'Top 13 largest
cities in the World', textStyle: { size: '16px' } }}>
            <Inject services={[Marker, MapsTooltip, Zoom]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map}
shapeSettings={{ fill: '#C1DFF5' }} markerClusterSettings={{
allowClustering: true, shape: 'Circle', height: 30, width: 30, labelStyle: {
color: 'white' }, }}>
                    <MarkersDirective>
                        <MarkerDirective visible={true}
dataSource={cluster} shape='Balloon' tooltipSettings={{ visible: true,
valuePath: 'area', }} height={15} width={15} animationDuration={0}>
                            </MarkerDirective>
                        </MarkersDirective>
                    </LayerDirective>
                </LayersDirective>
            </MapsComponent>
        );
    }
    const root = ReactDOM.createRoot(document.getElementById('container'));
    root.render(<App />);
    {% endraw %}
}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { cluster } from 'marker-cluster.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject, MapsTooltip, Zoom } from '@syncfusion/ej2-
react-maps';
export function App() {
    return (
        <MapsComponent useGroupingSeparator={true} format='n'
zoomSettings={{ enable: true }} titleSettings={{ text: 'Top 13 largest
cities in the World', textStyle: { size: '16px' } }}>
            <Inject services={[Marker, MapsTooltip, Zoom]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map}
shapeSettings={{ fill: '#C1DFF5' }} markerClusterSettings={{
allowClustering: true, shape: 'Circle', height: 30, width: 30, labelStyle: {
color: 'white' }, }}>
                    <MarkersDirective>
                        <MarkerDirective visible={true}
dataSource={cluster} shape='Balloon' tooltipSettings={{ visible: true,
valuePath: 'area', }} height={15} width={15} animationDuration={0}>

```

```

        </MarkerDirective>
      </MarkersDirective>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>

);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Customization of marker cluster

The following properties are available to customize the marker clustering in the Maps component.

- [border](#) - To customize the color, width and opacity of the border of cluster in Maps.
- [connectorLineSettings](#) - To customize the connector line in cluster separating the markers.
- [dashArray](#) - To customize the dash array for the marker cluster in Maps.
- [fill](#) - Applies the color of the cluster in Maps.
- [height](#) - To customize the height of the marker cluster in Maps.
- [imageUrl](#) - To customize the URL path for the marker cluster when the cluster shape is set as image in Maps.
- [labelStyle](#) - To customize the text in marker cluster.
- [offset](#) - To customize the offset position for the marker cluster in Maps.
- [opacity](#) - To customize the opacity of the marker cluster.
- [shape](#) - To customize the shape for the cluster of markers.
- [width](#) - To customize the width of the marker cluster in Maps.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { cluster } from 'marker-cluster.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
Point, MarkerDirective, Marker, Inject, MapsTooltip, Zoom } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent useGroupingSeparator={true} format='n'
zoomSettings={{ enable: true }} titleSettings={{ text: 'Top 13 largest
cities in the World', textStyle: { size: '16px' } }}>
      <Inject services={[Marker, MapsTooltip, Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeSettings={{ fill: '#C1DFF5' }} markerClusterSettings={{
          allowClustering: true,
          allowClusterExpand: true,
          shape: 'Circle',
          height: 40,
          width: 40,
          labelStyle : { color: 'white' },
          offset: new Point(10, 20),

```

```

        opacity: 0.9,
        fill: 'green',
        connectorLineSettings: {
          color: 'orange',
          opacity: 0.8,
          width: 2
        }
      }>
    <MarkersDirective>
      <MarkerDirective visible={true}
dataSource={cluster} shape='Balloon' tooltipSettings={{ visible: true,
valuePath: 'area', }} height={15} width={15} animationDuration={0}>
        </MarkerDirective>
      </MarkersDirective>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>

);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { cluster } from 'marker-cluster.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
Point, MarkerDirective, Marker, Inject, MapsTooltip, Zoom } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent useGroupingSeparator={true} format='n'
zoomSettings={{ enable: true }} titleSettings={{ text: 'Top 13 largest
cities in the World', textStyle: { size: '16px' } }}>
      <Inject services={[Marker, MapsTooltip, Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeSettings={{ fill: '#C1DFF5' }} markerClusterSettings={{
          allowClustering: true,
          allowClusterExpand: true,
          shape: 'Circle',
          height: 40,
          width: 40,
          labelStyle : { color: 'white' },
          offset: new Point(10, 20),
          opacity: 0.9,
          fill: 'green',
          connectorLineSettings: {
            color: 'orange',
            opacity: 0.8,
            width: 2
          }
        }}>

```



```

    }}>
    <MarkersDirective>
      <MarkerDirective visible={true}
dataSource={cluster} shape='Balloon' tooltipSettings={{ visible: true,
valuePath: 'area', }} height={15} width={15} animationDuration={0}>
      </MarkerDirective>
    </MarkersDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Expanding the marker cluster

The cluster is formed by grouping an identical and non-identical marker from the surrounding area. By clicking on the cluster and setting the [allowClusterExpand](#) property in [markerClusterSettings](#) as **true** to expand the identical markers. If zoom in any of the locations of the cluster, the number on the cluster will decrease and the overlapping marker will be split into an individual marker on the map. When performing zoom out, it will increase the marker count and then cluster it again.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Marker, Inject, Zoom } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings= {{enable: true, mouseWheelZoom
: true }}>
      <Inject services={[Marker, Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
markerClusterSettings={{ allowClustering: true, allowClusterExpand:
true, shape: 'Circle', height: 40, width: 40, labelStyle: { color: 'white' },
}}>
          <MarkersDirective>
            <MarkerDirective visible={true}
dataSource=[
              { latitude:
49.95121990866204, longitude: 18.468749999999998, name: 'Europe' },
              { latitude:
49.95121990866204, longitude: 18.468749999999998, name: 'Europe' },
              { latitude:
49.95121990866204, longitude: 18.468749999999998, name: 'Europe' },
              { latitude:
49.95121990866204, longitude: 18.468749999999998, name: 'Europe' },
              { latitude:
49.95121990866204, longitude: 18.468749999999998, name: 'Europe' },

```

```

    { latitude:
49.95121990866204, longitude: 18.468749999999998, name:'Europe' },
    { latitude:
49.95121990866204, longitude: 18.468749999999998, name:'Europe' },
    { latitude:
59.88893689676585, longitude: -109.3359375, name:'North America' },
    { latitude: -
6.64607562172573, longitude: -55.54687499999999, name:'South America' }
  ]} animationDuration={0}>
    </MarkerDirective>
  </MarkersDirective>
</LayerDirective>
</LayersDirective>
</MapComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX[illegible]

```

        { latitude:
59.88893689676585, longitude: -109.3359375, name: 'North America' },
        { latitude: -
6.64607562172573, longitude: -55.54687499999999, name: 'South America' }
    ]} animationDuration={0}>
        </MarkerDirective>
    </MarkersDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Tooltip for marker

Tooltip is used to display more information about a marker on mouse over or touch end event. This can be enabled separately for marker by setting the [visible](#) property of [tooltipSettings](#) to [Link to the Video](#). The [valuePath](#) property in the [tooltipSettings](#) takes the field name that presents in dataSource and displays that value as tooltip text.

INDEX.JSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, MapsTooltip, Marker, Inject } from '@syncfusion/ej2-react-
maps';
export function App() {
    return (
        <MapsComponent >
        <Inject services={[Marker, MapsTooltip]} />
        <LayersDirective>
            <LayerDirective shapeData={usa_map}>
                <MarkersDirective>
                    <MarkerDirective visible={true}
                        height={20}
                        width={20}
                        animationDuration={0}
                        tooltipSettings={{
                            visible: true,
                            valuePath: 'city'
                        }}
                        dataSource=[
                            { latitude: 40.7424509,
longitude: -74.0081468, city: 'New York' }
                        ]>
                        </MarkerDirective>
                    </MarkersDirective>
                </LayerDirective>
            </LayersDirective>
        </MapsComponent>
    );
}

```

```

}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, MapsTooltip, Marker, Inject } from '@syncfusion/ej2-react-
maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker, MapsTooltip]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              height={20}
              width={20}
              animationDuration={0}
              tooltipSettings={{
                visible: true,
                valuePath: 'city'
              }}
              dataSource={[
                { latitude: 40.7424509,
longitude: -74.0081468, city: 'New York' }
              ]}>
              </MarkerDirective>
            </MarkersDirective>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

Bubble in React Maps component

This section shows how to customize the appearance of the bubbles in the Maps component. The below video demonstrates the same.

Bubbles in the Maps component represent the underlying data values of the Maps. It can be scattered throughout the Maps shapes that contain values in the data source. Bubbles are enabled by setting the [visible](#) property of [bubbleSettings](#) to **true**. To add bubbles to the Maps, bind the data source to the [dataSource](#) property of [BubbleDirective](#) and set the field name, that contains the numerical data, in the data source to the [valuePath](#) property.

```
`ts
```

```
export let world_map = // paste the World map from World.json GeoJSON file.
```

```
,
```

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Bubble]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population" dataSource=[[{color: 'green', name: 'India',
population: '38332521' },
            {color: 'purple', name: 'New Zealand', population: '19651127' },
            {color: 'blue', name: 'Pakistan', population: '3090416' }]]>
minRadius={20} maxRadius={40} />
          </BubblesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Bubble]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">

```

```

        <BubblesDirective>
          <BubbleDirective visible={true}
valuePath="population" dataSource={[{color: 'green', name: 'India',
population: '38332521' },
          {color: 'purple', name: 'New Zealand', population: '19651127' },
          {color: 'blue', name: 'Pakistan', population: '3090416' }]}
minRadius={20} maxRadius={40} />
        </BubblesDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Bubble shapes

The following types of shapes are available to render the bubbles in Maps.

- Circle
- Square

By default, bubbles are rendered in the **Circle** type. To change the type of the bubble, set the [bubbleType](#) property of [BubbleDirective](#) as **Square** to render the square shape bubbles.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Bubble]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population" bubbleType="Square" dataSource={[{ name: 'India',
population: '38332521' },
              { name: 'Pakistan', population: '3090416' }]}
minRadius={20} maxRadius={40} />
          </BubblesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}

```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Bubble]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population" bubbleType="Square" dataSource=[[{ name: 'India',
population: '38332521' },
{ name: 'Pakistan', population: '3090416' }]]
minRadius={20} maxRadius={40} />
          </BubblesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Customization

The following properties are available in [BubbleDirective](#) to customize the bubbles of the Maps component.

- [border](#) – To customize the color, width and opacity of the border of the bubbles in Maps.
- [fill](#) – To apply the color for bubbles in Maps.
- [opacity](#) – To apply opacity to the bubbles in Maps.
- [animationDelay](#) - To change the time delay in the transition for bubbles.
- [animationDuration](#) - To change the time duration of animation for bubbles.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
```

```

import { MapsComponent, LayersDirective, LayerDirective, Inject } from
 '@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[Bubble]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population"
                                dataSource=[
                                  { name: 'India', population:
'38332521' },
                                  { name: 'Pakistan', population:
'3090416' },
                                  { name: 'New Zealand',
population: '19651127' }
                                ]
                                minRadius={5} maxRadius={40}
                                animationDelay={100}
                                border={{color: "blue", width: 2}}
                                />
          </BubblesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
 '@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[Bubble]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>

```



```

        <BubbleDirective visible={true}
valuePath="population"
                                dataSource={[
                                { name: 'India', population:
'38332521' },
                                { name: 'Pakistan', population:
'3090416' },
                                { name: 'New Zealand',
population: '19651127' }
                                ]}
                                minRadius={5} maxRadius={40}
fill="green"
                                animationDelay={100}
                                animationDuration={1000} opacity={1}
                                border={{color: "blue", width: 2}}
/>
    </BubblesDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

Setting colors to the bubbles from the data source

The color for each bubble in the Maps can be set using the [colorValuePath](#) property of [BubbleDirective](#). The value for the [colorValuePath](#) property is the field name from the data source of the [BubbleDirective](#) which contains the color values.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Bubble]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population"
                                dataSource={[
                                { name: 'India', population:
'38332521', color: 'blue' },
                                { name: 'New Zealand',
population: '19651127', color: '#c2d2d6' },

```

```

                                { name: 'Pakistan', population:
'3090416', color: '#09156d' }
                                ]]
                                minRadius={20} maxRadius={40}
colorValuePath="color"/>
                                </BubblesDirective>
                                </LayerDirective>
                                </LayersDirective>
                                </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
    return (
        <MapsComponent >
            <Inject services={[Bubble]}/>
            <LayersDirective>
                <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
                    <BubblesDirective>
                        <BubbleDirective visible={true}
valuePath="population"
                                dataSource=[
                                    { name: 'India', population:
'38332521', color: 'blue' },
                                    { name: 'New Zealand',
population: '19651127', color: '#c2d2d6' },
                                    { name: 'Pakistan', population:
'3090416', color: '#09156d' }
                                ]]
                                minRadius={20} maxRadius={40}
colorValuePath="color"/>
                        </BubblesDirective>
                        </LayerDirective>
                    </LayersDirective>
                </MapsComponent>
            );
        }
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Setting the range of the bubble size

The size of the bubbles is calculated from the values got from the [valuePath](#) property. The range for the radius of the bubbles can be modified using [minRadius](#) and [maxRadius](#) properties.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
 '@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Bubble]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population" dataSource={[{name: 'India', population: '38332521'
},
            {name: 'New Zealand', population: '19651127' },
            {name: 'Pakistan', population: '3090416' }]} minRadius={5}
maxRadius={80} />
          </BubblesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
 '@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble } from '@syncfusion/ej2-
react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Bubble]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
```

```

        <BubbleDirective visible={true}
valuePath="population" dataSource={[{name: 'India', population: '38332521'
},
        {name: 'New Zealand', population: '19651127' },
        {name: 'Pakistan', population: '3090416' }]} minRadius={5}
maxRadius={80} />
    </BubblesDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Multiple bubble groups

Multiple groups of bubbles can be added to the Maps using the [BubbleDirective](#) in which the properties of bubbles are added as an array. The customization for the bubbles can be done with the [BubbleDirective](#). In the following example, the gender-wise population ratio is demonstrated with two different bubble groups.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
  Inject,
} from '@syncfusion/ej2-react-maps';
import {
  BubblesDirective,
  BubbleDirective,
  Bubble,
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent>
      <Inject services={[Bubble]} />
      <LayersDirective>
        <LayerDirective
          shapeData={world_map}
          shapeDataPath="country"
          shapePropertyPath="name"
        >
          <BubblesDirective>
            <BubbleDirective
              visible={true}
              valuePath="femaleRatio"
              colorValuePath="femaleRatioColor"
              dataSource=[
                {

```

```

        country: 'United States',
        femaleRatio: 50.50442726,
        maleRatio: 49.49557274,
        femaleRatioColor: '#99295D',
        maleRatioColor: 'blue',
      },
    {
      country: 'India',
      femaleRatio: 48.18032713,
      maleRatio: 51.81967287,
      femaleRatioColor: '#2E769F',
      maleRatioColor: '#c2d2d6',
    },
    {
      country: 'Oman',
      femaleRatio: 34.15597234,
      maleRatio: 65.84402766,
      femaleRatioColor: '#816F28',
      maleRatioColor: 'orange',
    },
    {
      country: 'United Arab Emirates',
      femaleRatio: 27.59638942,
      maleRatio: 72.40361058,
      femaleRatioColor: '#364A98',
      maleRatioColor: 'orange',
    },
  ]}
  minRadius={5}
  maxRadius={20}
/>
<BubbleDirective
  visible={true}
  bubbleType="Circle"
  valuePath="maleRatio"
  colorValuePath="maleRatioColor"
  dataSource={[
    {
      country: 'France',
      femaleRatio: 50.50442726,
      maleRatio: 49.49557274,
      femaleRatioColor: 'green',
      maleRatioColor: '#c2d2d6',
    },
    {
      country: 'China',
      femaleRatio: 48.18032713,
      maleRatio: 51.81967287,
      femaleRatioColor: 'blue',
      maleRatioColor: '#09156d',
    },
    {
      country: 'Kazakhstan',
      femaleRatio: 34.15597234,
      maleRatio: 65.84402766,
      femaleRatioColor: '#09156d',
      maleRatioColor: 'yellow',
    },
  ]}

```

```

        },
        {
            country: 'Poland',
            femaleRatio: 27.59638942,
            maleRatio: 72.40361058,
            femaleRatioColor: '#09156d',
            maleRatioColor: 'orange',
        },
    ]}
    minRadius={15}
    maxRadius={25}
    opacity={0.4}
  />
</BubblesDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
    MapsComponent,
    LayersDirective,
    LayerDirective,
    Inject,
} from '@syncfusion/ej2-react-maps';
import {
    BubblesDirective,
    BubbleDirective,
    Bubble,
} from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent>
            <Inject services={[Bubble]} />
            <LayersDirective>
                <LayerDirective
                    shapeData={world_map}
                    shapeDataPath="country"
                    shapePropertyPath="name"
                >
                    <BubblesDirective>
                        <BubbleDirective
                            visible={true}
                            valuePath="femaleRatio"
                            colorValuePath="femaleRatioColor"
                            dataSource={ [

```

```

        {
          country: 'United States',
          femaleRatio: 50.50442726,
          maleRatio: 49.49557274,
          femaleRatioColor: '#99295D',
          maleRatioColor: 'blue',
        },
        {
          country: 'India',
          femaleRatio: 48.18032713,
          maleRatio: 51.81967287,
          femaleRatioColor: '#2E769F',
          maleRatioColor: '#c2d2d6',
        },
        {
          country: 'Oman',
          femaleRatio: 34.15597234,
          maleRatio: 65.84402766,
          femaleRatioColor: '#816F28',
          maleRatioColor: 'orange',
        },
        {
          country: 'United Arab Emirates',
          femaleRatio: 27.59638942,
          maleRatio: 72.40361058,
          femaleRatioColor: '#364A98',
          maleRatioColor: 'orange',
        },
      ],
      minRadius={5}
      maxRadius={20}
    />
    <BubbleDirective
      visible={true}
      bubbleType="Circle"
      valuePath="maleRatio"
      colorValuePath="maleRatioColor"
      dataSource={[
        {
          country: 'France',
          femaleRatio: 50.50442726,
          maleRatio: 49.49557274,
          femaleRatioColor: 'green',
          maleRatioColor: '#c2d2d6',
        },
        {
          country: 'China',
          femaleRatio: 48.18032713,
          maleRatio: 51.81967287,
          femaleRatioColor: 'blue',
          maleRatioColor: '#09156d',
        },
        {
          country: 'Kazakhstan',
          femaleRatio: 34.15597234,
          maleRatio: 65.84402766,
          femaleRatioColor: '#09156d',

```

```

        maleRatioColor: 'yellow',
      },
      {
        country: 'Poland',
        femaleRatio: 27.59638942,
        maleRatio: 72.40361058,
        femaleRatioColor: '#09156d',
        maleRatioColor: 'orange',
      },
    ]],
    minRadius={15}
    maxRadius={25}
    opacity={0.4}
  />
</BubblesDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

Enable tooltip for bubble

The tooltip for the bubbles can be enabled by setting the [visible](#) property of the [tooltipSettings](#) as **true**. The content for the tooltip can be set using the [valuePath](#) property in the [tooltipSettings](#) of the [BubbleDirective](#) where the value for the [valuePath](#) property is the field name from the data source of the [BubbleDirective](#). Also added any HTML element as the template in tooltip using the [template](#) property.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble, MapsTooltip } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Bubble, MapsTooltip]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population" dataSource=[
              { name: 'India', population:
'38332521' },
              { name: 'New Zealand', population:
'19651127' },

```



```

        { name: 'Pakistan', population:
'3090416' ]]]
        minRadius={5} maxRadius={80}
        tooltipSettings={{
            visible: true,
            valuePath: 'population'
        }} />
    </BubblesDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble, MapsTooltip } from
'@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
        <Inject services={[Bubble, MapsTooltip]}/>
        <LayersDirective>
            <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
                <BubblesDirective>
                    <BubbleDirective visible={true}
valuePath="population" dataSource=[
                    { name: 'India', population:
'38332521' },
                    { name: 'New Zealand', population:
'19651127' },
                    { name: 'Pakistan', population:
'3090416' }]]
                    minRadius={5} maxRadius={80}
                    tooltipSettings={{
                        visible: true,
                        valuePath: 'population'
                    }} />
                </BubblesDirective>
            </LayerDirective>
        </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);

```

```
{% endraw %}
```

Annotations in React Maps component

<!-- markdownlint-disable MD013 -->

Annotations are used to mark the specific area of interest in the Maps with texts, shapes, or images. Any number of annotations can be added to the Maps component.

Annotation

By using the [content](#) property of [AnnotationsDirective](#), text content or id of an element or an HTML string can be specified to render a new HTML element in Maps.

<!-- markdownlint-disable MD036 -->

,

```
<script id='annotation' type='text/x-template'>
```

```
<div id='template'>
```

```
<img style="width:50px;height:50px"
```

```
src='https://ej2.syncfusion.com/react/demos/src/maps/images/weather-clear.png'>
```

```
</div>
```

```
</script>
```

,

```
`ts
```

```
import { world_map } from 'world-map.ts';
```

```
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
```

```
import { MapsComponent, LayersDirective, LayerDirective, Inject } from '@syncfusion/ej2-react-maps';
```

```
import { AnnotationsDirective, AnnotationDirective, Annotations } from '@syncfusion/ej2-react-maps';
```

```
export function App() {
```

```
  return(
```

```
    <MapsComponent >
```

```
      <Inject services={[Annotations]}/>
```

```
      <AnnotationsDirective>
```

```
        <AnnotationDirective content="#annotation" x="0%" y="50%"/>
```

```
      </AnnotationsDirective>
```

```
      <LayersDirective>
```

```
        <LayerDirective shapeData={world_map}>
```

```
      </LayerDirective>
```

```
    </LayersDirective>
```

```

</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));

root.render(<App />);
`

```

Annotation customization

Changing the z-index

The stack order of an annotation element can be changed using the [zIndex](#) property in the [AnnotationsDirective](#).

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { AnnotationsDirective, AnnotationDirective, Annotations } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Maps</h1></div>' x="0%" y="50%" zIndex="-1"/>
      </AnnotationsDirective>
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
      </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { AnnotationsDirective, AnnotationDirective, Annotations } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (

```

```

    <MapsComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Maps</h1></div>' x="0%" y="50%" zIndex="-1"/>
      </AnnotationsDirective>
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

<!-- markdownlint-disable MD036 -->

Positioning an annotation

Annotations can be placed anywhere in the Maps by specifying pixel or percentage values to the [x](#) and [y](#) properties in the [AnnotationsDirective](#).

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { AnnotationsDirective, AnnotationDirective, Annotations } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Maps</h1></div>' x="20%" y="50%" zIndex="-1"/>
      </AnnotationsDirective>
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';

```

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { AnnotationsDirective, AnnotationDirective, Annotations } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Maps</h1></div>' x="20%" y="50%" zIndex="-1"/>
        </AnnotationsDirective>
        <LayersDirective>
          <LayerDirective shapeData={world_map}>
            </LayerDirective>
          </LayersDirective>
        </MapsComponent>
      );
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}
```

<!-- markdownlint-disable MD036 -->

Alignment of an annotation

Annotations can be aligned using the [horizontalAlignment](#) and [verticalAlignment](#) properties in the [AnnotationsDirective](#). The possible values can be **Center**, **Far**, **Near** and **None**.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { AnnotationsDirective, AnnotationDirective, Annotations } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Maps</h1></div>' x="20%" y="50%" zIndex="-1"
verticalAlignment="Center"
horizontalAlignment="Center"/>
        </AnnotationsDirective>
        <LayersDirective>
          <LayerDirective shapeData={world_map}>
            </LayerDirective>
          </LayersDirective>
        </MapsComponent>
      );
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}
```

```

    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { AnnotationsDirective, AnnotationDirective, Annotations } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div
id="first"><h1>Maps</h1></div>' x="20%" y="50%" zIndex="-1"
verticalAlignment="Center"
horizontalAlignment="Center"/>
      </AnnotationsDirective>
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Multiple Annotation

Multiple annotations can be added to the Maps by adding multiple [AnnotationDirective](#) in the [AnnotationsDirective](#) and customization for the annotations can be done with the [AnnotationDirective](#).

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { AnnotationsDirective, AnnotationDirective, Annotations } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Annotations]} />

```

```

        <AnnotationsDirective>
          <AnnotationDirective content='<div id="first"><h1>Maps-
Annotation</h1></div>' x="50%" y="0%" zIndex="-1"/>
          <AnnotationDirective content='<div
id="first"><h1>Maps</h1></div>' x="20%" y="50%" zIndex="-1"/>
        </AnnotationsDirective>
        <LayersDirective>
          <LayerDirective shapeData={world_map}>
        </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { AnnotationsDirective, AnnotationDirective, Annotations } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Annotations]} />
      <AnnotationsDirective>
        <AnnotationDirective content='<div id="first"><h1>Maps-
Annotation</h1></div>' x="50%" y="0%" zIndex="-1"/>
        <AnnotationDirective content='<div
id="first"><h1>Maps</h1></div>' x="20%" y="50%" zIndex="-1"/>
      </AnnotationsDirective>
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
      </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Legend in React Maps component

A Legend is a visual representation of the symbols used on the Maps. It can be represented in various colors, shapes or other identifiers based on the data and provides valuable information for interpreting what the Maps are displaying. It explains what each symbol in the Maps represents. Legends are enabled by setting the [visible](#) property of [legendSettings](#) to **true**.

Modes of legend

Legend had two types of mode.

1. **Default** mode
2. **Interactive** mode

Default mode

Default mode legends having symbols with legend labels, used to identify the shape or bubble or marker color. To enable this option by setting the [mode](#) property of [legendSettings](#) as **Default**.

Interactive mode

The legends can be made interactive with an arrow mark indicating the exact range color in the legend when the mouse hovers over the corresponding shapes. To enable this type of mode by setting the [mode](#) property of [legendSettings](#) as **Interactive**. The [invertedPointer](#) property is used to enable or disable the visibility of the inverted pointer in interactive legend in Maps.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ {
      visible: true,
      mode: 'Interactive',
      invertedPointer: true
    } } >
      <Inject services={[Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
shapeSettings={ {
          colorValuePath: 'Membership',
          colorMapping: [
            { value: 'Permanent', color: '#D84444' },
            { value: 'Non-Permanent', color: '#316DB5' }
          ]
        } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX


```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ {
      visible: true,
      mode: 'Interactive',
      invertedPointer: true
    } } >
      <Inject services={[Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
        shapeSettings={ {
          colorValuePath: 'Membership',
          colorMapping: [
            { value: 'Permanent', color: '#D84444' },
            { value: 'Non-Permanent', color: '#316DB5' }
          ]
        } } >
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
}{% endraw %}
```

Positioning of the legend

The legend can be positioned in the following two ways:

- Absolute position
- Dock position

Absolute position

The legend of the Maps can be positioned using the [location](#) property in the [legendSettings](#). For positioning the legend based on co-ordinates corresponding to a Maps, the [position](#) property is set as **Float**.

Dock position

Legends are positioned in the following locations within the container. The [position](#) property in [legendSettings](#) is used to set these options in Maps.

- Top
- Left
- Bottom

- Right

The above four positions can be aligned with combination of **Near**, **Center** and **Far** using [alignment](#) property in [legendSettings](#). So, the legend can be aligned to 12 positions.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
return(
    <MapsComponent legendSettings={ {
                                visible: true,
                                position: 'Top',
                                alignment: 'Near'
                                } } >
        <Inject services={[Legend]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
            shapeSettings={ {
                colorValuePath: 'Membership',
                colorMapping: [
                    { value: 'Permanent', color: '#D84444' },
                    { value: 'Non-Permanent', color: '#316DB5' }
                ]
            } }>
        </LayerDirective>
    </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
return(
    <MapsComponent legendSettings={ {
                                visible: true,
                                position: 'Top',
                                alignment: 'Near'
                                } } >
```

```

    } } >
    <Inject services={[Legend]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
      shapeSettings={ {
        colorValuePath: 'Membership',
        colorMapping: [
          { value: 'Permanent', color: '#D84444' },
          { value: 'Non-Permanent', color: '#316DB5' }
        ]
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Legend for shapes

Legend for shapes can be generated from color mapping types such as equal color mapping, range color mapping and desaturation color mapping.

The below code snippet demonstrate the equal color mapping legends for the shapes. To bind the given data source to the [dataSource](#) property of [layerSettings](#). Set the value of [shapePropertyPath](#) to **name** and [shapeDataPath](#) to **Country**. To enable equal color mapping, set the [colorMapping](#) as an array in [shapeSettings](#). Finally, set the [visible](#) property of [legendSettings](#) as **true**. The [label](#) property in [colorMapping](#) is used to set the text name for legend in Maps.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ { visible: true } } >
    <Inject services={[Legend]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
      shapeSettings={ {
        colorValuePath: 'Membership',
        colorMapping: [
          { value: 'Permanent', color: '#D84444' },
          { value: 'Non-Permanent', color: '#316DB5' }
        ]
      } }>
    </LayerDirective>
  )
}

```

```

        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ { visible: true } } >
      <Inject services={[Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
shapeSettings={ {
  colorValuePath: 'Membership',
  colorMapping: [
    { value: 'Permanent', color: '#D84444' },
    { value: 'Non-Permanent', color: '#316DB5' }
  ]
} }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Legend shape

Maps supports the following types of legend shapes. The [shape](#) property in the [legendSettings](#) can be used to change the type of legend shapes.

- Circle
- Rectangle
- Triangle
- Diamond
- Cross
- Star
- HorizontalLine
- VerticalLine
- Pentagon

- `InvertedTriangle`

The shape of legends can be customized by using the [shapeWidth](#), [shapeHeight](#), [shapeBorder](#) and [shapePadding](#) properties.

Customization

The following properties are available in legend to customize the legend shape and legend text in Maps.

- [background](#) - To customize the background color of the Legend.
- [border](#) - To customize the color, width and opacity of the border for the Legend.
- [fill](#) - To apply the color for the Legend.
- [labelDisplayMode](#) - To customize the display mode for the Legend text.
- [labelPosition](#) - To customize the position of the Legend text.
- [orientation](#) - To customize the orientation of the Legend.
- [textStyle](#) - To customize the text style for Legend.
- [title](#) - To apply the title for the Legend.
- [titleStyle](#) - To customize the style of the title for the Legend.
- [height](#) - To customize the height of the Legend.
- [width](#) - To customize the width of the Legend.
- [opacity](#) - To apply the opacity to the Legend.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ {
      visible: true,
      background: 'green',
      border: {
        color: 'blue',
        width: 2
      },
      fill: 'orange',
      labelPosition: 'Before',
      orientation: 'Vertical',
      textStyle: {
        size: '12px',
        color: 'red',
        fontStyle: 'italic'
      },
      title: {
        description: 'Legend title',
        text: 'Legend'
      },
      titleStyle: {
        size: '12px',
        color: '#d6e341',
        fontStyle: 'italic'
      }
    }
    />
  );
}
```

```

    }
    } } >
    <Inject services={[Legend]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={[
        { "Country": "China", "Membership": "Permanent" },
        { "Country": "France", "Membership": "Permanent" },
        { "Country": "Russia", "Membership": "Permanent" },
        { "Country": "Kazakhstan", "Membership": "Non-Permanent"
      },
        { "Country": "Poland", "Membership": "Non-Permanent" },
        { "Country": "Sweden", "Membership": "Non-Permanent" }
      ]}
      shapeSettings={ {
        colorValuePath: 'Membership',
        colorMapping: [
          { value: 'Permanent', color: '#D84444' },
          { value: 'Non-Permanent', color: '#316DB5' } ]
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ {
      visible: true,
      background: 'green',
      border: {
        color: 'blue',
        width: 2
      },
      fill: 'orange',
      labelPosition: 'Before',
      orientation: 'Vertical',
      textStyle: {
        size: '12px',
        color: 'red',
        fontStyle: 'italic'
      },
      title: {
        description: 'Legend title',

```

```

        text: 'Legend'
      },
      titleStyle: {
        size: '12px',
        color: '#d6e341',
        fontStyle: 'italic'
      }
    } } >
    <Inject services={[Legend]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={[
        { "Country": "China", "Membership": "Permanent" },
        { "Country": "France", "Membership": "Permanent" },
        { "Country": "Russia", "Membership": "Permanent" },
        { "Country": "Kazakhstan", "Membership": "Non-Permanent"
      },
        { "Country": "Poland", "Membership": "Non-Permanent" },
        { "Country": "Sweden", "Membership": "Non-Permanent" }
      ]}
      shapeSettings={ {
        colorValuePath: 'Membership',
        colorMapping: [
          { value: 'Permanent', color: '#D84444' },
          { value: 'Non-Permanent', color: '#316DB5' } ]
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Legend for items excluded from color mapping

The legend can be enabled for items excluded from the color mapping using the [color](#) property in [colorMapping](#). Refer to the **population_density** data which demonstrates the population density of some countries.

```
`ts
```

```

export let population_density = [
  https://ej2.syncfusion.com/react/documentation.
  {
    'code': 'AE',
    'value': 90,
    'name': 'United Arab Emirates',
    'population': 8264070,
    'density': 99
  }
]

```

```

},
{
  'code': 'GB',
  'value': 257,
  'name': 'United Kingdom',
  'population': 62041708,
  'density': 255
},
{
  'code': 'US',
  'value': 34,
  'name': 'United States',
  'population': 325020000,
  'density': 33
}
...
];
`

```

In the following example, color mapping is added for the ranges from **0** to **200**. If there are any records in the data source that are outside of this range, the color mapping will not be applied. To apply the color for these excluded items, set the [color](#) property alone in the [colorMapping](#). To enable legend for these items, set the [visible](#) property of [legendSettings](#) to **true**.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ { visible: true } } >
      <Inject services={[Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
          shapeSettings={ {
            colorValuePath: 'Membership',
            colorMapping: [
              { from: 0, to: 100, color: 'rgb(153,174,214)' },

```



```

        { from: 101, to: 200, color: 'rgb(115,143,199)' }
    },
    { color: 'rgb(77,112,184)' }
  ]
} }>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ { visible: true } } >
      <Inject services={[Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
shapeSettings={ {
  colorValuePath: 'Membership',
  colorMapping: [
    { from: 0, to: 100, color: 'rgb(153,174,214)' },
    { from: 101, to: 200, color: 'rgb(115,143,199)' }
  ],
  { color: 'rgb(77,112,184)' }
} }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Hide desired legend items

Use the [showLegend](#) in the [colorMapping](#) property to show or hide the desired legend items in Maps. If the [showLegend](#) property is set to **false**, the legend item will be hidden. otherwise, it will be visible.

INDEX.JSX

```

{% raw %}

```

```

import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent legendSettings={ { visible: true } } >
        <Inject services={[Legend]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
            shapeSettings={ {
                colorValuePath: 'Membership',
                colorMapping: [
                    {
                        from: 0, to: 100, color: 'rgb(153,174,214)',
showLegend: true
                    },
                    {
                        from: 101, to: 200, color:
'rgb(115,143,199)', showLegend: false
                    },
                    {
                        color: 'rgb(77,112,184)', showLegend: false
                    }
                ]
            } }>
            </LayerDirective>
        </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent legendSettings={ { visible: true } } >
        <Inject services={[Legend]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}
shapeDataPath='Country' shapePropertyPath='name' dataSource={uncountries}
            shapeSettings={ {
                colorValuePath: 'Membership',

```

```

        colorMapping: [
            {
                from: 0, to: 100, color: 'rgb(153,174,214)',
showLegend: true
            },
            {
                from: 101, to: 200, color:
'rgb(115,143,199)', showLegend: false
            },
            {
                color: 'rgb(77,112,184)', showLegend: false
            }
        ]
    } }>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Hide legend items based on data source value

Depending on the boolean values provided in the data source, the legend items will be hidden or visible. Bind the field name that contains the visibility state in the data source to the [showLegendPath](#) property of the [legendSettings](#) to achieve this.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { defaultData } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent legendSettings={ { visible: true , showLegendPath:
'visibility' } } >
            <Inject services={[Legend]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map}
shapeDataPath='continent' shapePropertyPath='continent'
dataSource={defaultData}
                shapeSettings={ {
                    colorValuePath: 'color',
                } }>
            </LayerDirective>
        </LayersDirective>
    </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);

```

```
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { defaultData } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent legendSettings={ { visible: true , showLegendPath:
'visibility' } } >
      <Inject services={[Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='continent' shapePropertyPath='continent'
dataSource={defaultData}
          shapeSettings={ {
            colorValuePath: 'color',
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Binding legend item text from data source

To show the legend text based on values provided in the data source, use the [valuePath](#) property in the [legendSettings](#).

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { defaultData } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent legendSettings={ { visible: true , valuePath:
'continent' } } >
      <Inject services={[Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='continent' shapePropertyPath='continent'
dataSource={defaultData}
          shapeSettings={ {

```

```

        colorValuePath: 'color',
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { defaultData } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent legendSettings={ { visible: true , valuePath:
'continent' } } >
      <Inject services={ [Legend] } />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='continent' shapePropertyPath='continent'
dataSource={defaultData}
          shapeSettings={ {
            colorValuePath: 'color',
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Hide duplicate legend items

To hide the duplicate legend items in Maps, set the [removeDuplicateLegend](#) property to **true** in the [legendSettings](#).

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { defaultData } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {

```

```

    return (
      <MapsComponent legendSettings={ { visible: true, valuePath:
'continent', removeDuplicateLegend: true } } >
        <Inject services={[Legend]} />
        <LayersDirective>
          <LayerDirective shapeData={world_map}
shapeDataPath='continent' shapePropertyPath='continent'
dataSource={daefaultData}
            shapeSettings={ {
              colorValuePath: 'color',
            } }>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { daefaultData } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ { visible: true, valuePath:
'continent', removeDuplicateLegend: true } } >
      <Inject services={[Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath='continent' shapePropertyPath='continent'
dataSource={daefaultData}
          shapeSettings={ {
            colorValuePath: 'color',
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Toggle option in legend

The toggle option has been provided for legend. If the legend can be toggled, the given color will be changed to the corresponding Maps shape item. To enable the toggle options in Legend, set the [enable](#) property of the [toggleLegendSettings](#) to **true**.

The following properties are available to customize the toggle option in legend.

- [applyShapeSettings](#) – To apply the [fill](#) property value to the shape of the Maps when toggling the legend items.
- [fill](#) – To apply the color to the shape of the Maps for which legend item is toggled.
- [opacity](#) – To customize the transparency for the shapes for which legend item is toggled.
- [border](#) – To customize the color, width and opacity of the border of the shapes in Maps.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { population_density } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={ { visible: true,
toggleLegendSettings: { enable: true } } } >
      <Inject services={[Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map} shapeDataPath='name'
shapePropertyPath='name'
          dataSource={population_density}
          shapeSettings={ {
            colorValuePath: 'density',
            colorMapping: [
              {
                from: 0, to: 100, color: 'rgb(153,174,214)',
              },
              {
                from: 101, to: 200, color:
'rgb(115,143,199)',
              },
              {
                color: 'rgb(77,112,184)'
              }
            ]
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { population_density } from 'data.ts'
```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Legend }
from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent legendSettings={ { visible: true,
toggleLegendSettings: { enable: true } } } >
            <Inject services={[Legend]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map} shapeDataPath='name'
shapePropertyPath='name'
                    dataSource={population_density}
                    shapeSettings={ {
                        colorValuePath: 'density',
                        colorMapping: [
                            {
                                from: 0, to: 100, color: 'rgb(153,174,214)',
                            },
                            {
                                from: 101, to: 200, color:
'rgb(115,143,199)',
                            },
                            {
                                color: 'rgb(77,112,184)'
                            }
                        ]
                    } } >
            </LayerDirective>
        </LayersDirective>
    </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Enable Legend for bubbles

To enable the legend for the bubble by setting the [visible](#) property of [legendSettings](#) as **true** and [type](#) of [legendSettings](#) as **Bubbles**.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble, Legend } from
'@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent legendSettings= {{visible: true, type: 'Bubbles'}} >
            <Inject services={[Bubble, Legend]} />
            <LayersDirective>

```



```

        <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
                             valuePath="population"
                             colorValuePath="color"
                             dataSource= {[
population: '38332521' },
                             { color: 'green', name: 'India',
Zealand', population: '19651127' },
                             { color: 'purple', name: 'New
'Pakistan', population: '3090416' }
                             ]}
                             minRadius={20} maxRadius={40} />
          </BubblesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble, Legend } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings= {{visible: true, type: 'Bubbles'}}>
    <Inject services={[Bubble, Legend]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map} shapeDataPath="name"
shapePropertyPath="name">
        <BubblesDirective>
          <BubbleDirective visible={true}
                           valuePath="population"
                           colorValuePath="color"
                           dataSource= {[
population: '38332521' },
                           { color: 'green', name: 'India',
Zealand', population: '19651127' },
                           { color: 'purple', name: 'New
'Pakistan', population: '3090416' }
                           ]}
                           minRadius={20} maxRadius={40} />
        </BubblesDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
  );
}

```

```

        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Enable legend for markers

To enable legend for marker by setting the [visible](#) property of [legendSettings](#) as **true** and [type](#) property of [legendSettings](#) as **Markers**. The [legendText](#) property in the [markerSettings](#) can be used to show the legend text based on values provided in the data source.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
Legend, MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings= {{visible: true, type: 'Markers'}}>
      <Inject services=[Marker, Legend] />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective
              visible={true}
              legendText= 'city'
              dataSource=[
                { latitude: 37.0000, longitude: -120.0000,
city: 'California' },
                { latitude: 40.7127, longitude: -74.0059,
city: 'New York' },
                { latitude: 42, longitude: -93, city: 'Iowa'
}
              ]}>
              </MarkerDirective>
            </MarkersDirective>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';

```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
Legend, MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent legendSettings= {{visible: true, type: 'Markers'}}>
        <Inject services={[Marker, Legend]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}>
                <MarkersDirective>
                    <MarkerDirective
                        visible={true}
                        legendText= 'city'
                        dataSource=[
                            { latitude: 37.0000, longitude: -120.0000,
city: 'California' },
                            { latitude: 40.7127, longitude: -74.0059,
city: 'New York' },
                            { latitude: 42, longitude: -93, city: 'Iowa'
}
                        ]>
                    </MarkerDirective>
                </MarkersDirective>
            </LayerDirective>
        </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Imitate/Map marker shape to the legend shape

To imitate or map the marker shape with its legend item shape, set the [useMarkerShape](#) property to **true** in the [legendSettings](#) property.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { markerDataSource } from 'markerdata.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
Legend, MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent
            legendSettings={{
                visible: true,
                type: 'Markers',
                useMarkerShape: true,
                toggleLegendSettings: {
                    enable: true,
                    applyShapeSettings: false,

```

```

        border: {
            color: 'green',
            width: 2,
        },
    },
    {}
}
>
<Inject services={[Marker, Legend]} />
<LayersDirective>
  <LayerDirective
    shapeData={world_map}
    shapeSettings={{
      fill: '#E5E5E5',
    }}
  >
    <MarkersDirective>
      <MarkerDirective
        visible={true}
        dataSource={markerDataSource}
        colorValuePath="color"
        shapeValuePath="shape"
        legendText="country"
      ></MarkerDirective>
    </MarkersDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { markerDataSource } from 'markerdata.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
Legend, MarkerDirective, Marker, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent
      legendSettings={{
        visible: true,
        type: 'Markers',
        useMarkerShape: true,
        toggleLegendSettings: {
          enable: true,
          applyShapeSettings: false,
          border: {
            color: 'green',
            width: 2,
          },

```

```

    },
  }}
>
<Inject services={[Marker, Legend]} />
<LayersDirective>
  <LayerDirective
    shapeData={world_map}
    shapeSettings={{
      fill: '#E5E5E5',
    }}
  >
    <MarkersDirective>
      <MarkerDirective
        visible={true}
        dataSource={markerDataSource}
        colorValuePath="color"
        shapeValuePath="shape"
        legendText="country"
      ></MarkerDirective>
    </MarkersDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Navigation line in React Maps component

The navigation lines are used to denote the path between two locations. This feature can be used to draw flight or sea routes. Navigation lines are enabled by setting the [visible](#) property of the [navigationLineSettings](#) to **true**.

Customization

The following properties are available in [navigationLineSettings](#) to customize the navigation line of the Maps component.

- [color](#) - To apply the color for navigation lines in Maps.
- [dashArray](#) - To define the pattern of dashes and gaps that is applied to the outline of the navigation lines.
- [width](#) - To customize the width of the navigation lines.
- [angle](#) - To customize the angle of the navigation lines.
- [highlightSettings](#) - To customize the highlight settings of the navigation line.
- [selectionSettings](#) - To customize the selection settings of the navigation line.

To navigate the line between two cities on the world map, [latitude](#) and [longitude](#) values are used to indicate the start and end points of navigation lines drawn on Maps.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';

```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject,
NavigationLine, NavigationLinesDirective, NavigationLineDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return(
        <MapsComponent id="element">
            <Inject services={[NavigationLine]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map}>
                    <NavigationLinesDirective>
                        <NavigationLineDirective visible={true}
latitude=[37.6276571,
-122.4276688]]
longitude=[-74.0060, -
117.7418381]]
color="black"
angle={90}
width={2}
dashArray="4"/>
                    </NavigationLinesDirective>
                </LayerDirective>
            </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject,
NavigationLine, NavigationLinesDirective, NavigationLineDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
    return(
        <MapsComponent id="element">
            <Inject services={[NavigationLine]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map}>
                    <NavigationLinesDirective>
                        <NavigationLineDirective visible={true}
latitude=[37.6276571,
-122.4276688]]
longitude=[-74.0060, -
117.7418381]]
color="black"
angle={90}
width={2}
dashArray="4"/>
                    </NavigationLinesDirective>
                </LayerDirective>
            </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

```

        </NavigationLinesDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Enabling the arrows

To enable the arrow in the navigation line, set the [showArrow](#) property of [arrowSettings](#) to **true**. The following properties are available in [arrowSettings](#) to customize the arrow of the navigation lines.

- [color](#) - To apply the color for arrow of the navigation line.
- [offset](#) - To customize the offset position of the arrow of the navigation line.
- [position](#) - To customize the position of the arrow in navigation line. The possible values can be **Start** and **End**.
- [size](#) - To customize the size of the arrow in pixels.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject,
NavigationLine, NavigationLinesDirective, NavigationLineDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent id="element">
      <Inject services={[NavigationLine]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <NavigationLinesDirective>
            <NavigationLineDirective visible={true}
latitude={ [37.6276571,
-122.4276688] }
longitude={ [-74.0060, -
117.7418381] }
color="black"
angle={90}
width={2}
dashArray="4"
arrowSettings={{
  showArrow: true,
  size: 15,
  position: 'Start'
}}/>
          </NavigationLinesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>

```

```

    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject,
NavigationLine, NavigationLinesDirective, NavigationLineDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent id="element">
      <Inject services={[NavigationLine]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <NavigationLinesDirective>
            <NavigationLineDirective visible={true}
latitude={ [37.6276571,
-122.4276688] }
longitude={ [-74.0060, -
117.7418381] }
color="black"
angle={90}
width={2}
dashArray="4"
arrowSettings={{
  showArrow: true,
  size: 15,
  position: 'Start'
}} />
          </NavigationLinesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

User interactions in React Maps component

Zooming

The zooming feature is used to zoom in and out of Maps to show in-depth information. It is controlled by the [zoomFactor](#) property of the [zoomSettings](#) of the Maps. The Map is zoomed in when the [zoomFactor](#) is increased. The [zoomFactor](#) is increased or decrease dynamically based on zoom in and out interaction.

Enable zooming

Zooming of Maps is enabled by setting the [enable](#) property of [zoomSettings](#) to **true**.

Enable panning

To enable the panning feature, set the [enablePanning](#) property of [zoomSettings](#) to **true**.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from
 '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings={ { enable: true, enablePanning:
true } } >
      <Inject services={[Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from
 '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings={ { enable: true, enablePanning:
true } } >
      <Inject services={[Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Various type of zooming

Zooming can be performed in following types:

Zooming toolbar

By default, the toolbar is rendered with **zoom-in**, **zoom-out**, and **reset** options when it is set to **true** in the [enable](#) property of [zoomSettings](#).

The following options are available in toolbar.

1. Zoom - Provides rectangular zoom support.
2. ZoomIn - Zooms in the Maps.
3. ZoomOut - Zooms out the Maps.
4. Pan - Switches to panning if rectangular zoom is activated.
5. Reset - Restores the Maps to the default view.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent zoomSettings={ {
      enable: true,
      color: 'green',
      highlightColor: 'blue',
      selectionColor: 'orange',
      horizontalAlignment: 'Center',
      toolbars: ['ZoomIn', 'ZoomOut',
'Pan', 'Reset']} }>
      <Inject services={[Zoom]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent zoomSettings={ {
      enable: true,
      buttonSettings: {

```

```

        toolbarItems: ['Zoom', 'ZoomIn',
'ZoomOut', 'Pan', 'Reset']
    }
  } }>
  <Inject services={[Zoom]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Pinch zooming

To enable or disable the pinch zooming, use the [pinchZooming](#) property in [zoomSettings](#).

```

{% raw %}
`ts
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent zoomSettings={{ enable: true, pinchZooming: true }}>
    <Inject services={[Zoom]} />
    <LayersDirective>
    <LayerDirective shapeData={world_map}>
    </LayerDirective>
    </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
{% endraw %}

```

Single-click zooming

To enable or disable the single-click zooming, use the [zoomOnClick](#) property in [zoomSettings](#).

```
{% raw %}
`ts
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent zoomSettings={ { enable: true, zoomOnClick: true } }>
    <Inject services={[Zoom]}/>
    <LayersDirective>
    <LayerDirective shapeData={world_map}>
    </LayerDirective>
    </LayersDirective>
    </MapsComponent>
  );
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
```

```
{% endraw %}
```

Double-click zooming

To enable or disable the double-click zooming, use the [doubleClickZoom](#) property in [zoomSettings](#).

```
{% raw %}
`ts
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
```

```

return(
  <MapsComponent zoomSettings={ { enable: true, doubleClickZoom: true } }>
    <Inject services={[Zoom]}/>
    <LayersDirective>
      <LayerDirective shapeData={world_map}>
    </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

```

```
{% endraw %}
```

Mouse wheel zooming

To enable or disable mouse wheel zooming, use the [mouseWheelZoom](#) property in [zoomSettings](#).

```
{% raw %}
```

```
`ts
```

```

import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';

export function App() {
  return(
    <MapsComponent zoomSettings={ { enable: true, mouseWheelZoom: true } }>
      <Inject services={[Zoom]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}

```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
```

```
{% endraw %}
```

Selection zooming

To enable or disable selection zooming, use the [enableSelectionZooming](#) property in [zoomSettings](#). The [enablePanning](#) property must be set to **false** to enable the selection zooming in Maps.

```
{% raw %}
```

```
`ts
```

```
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';

export function App() {
  return(
    <MapsComponent zoomSettings={{ enable: true, enableSelectionZooming: true,
    enablePanning: false }}>
    <Inject services={[Zoom]}/>
    <LayersDirective>
    <LayerDirective shapeData={world_map}>
    </LayerDirective>
    </LayersDirective>
    </MapsComponent>
  );
}
```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
```

```
{% endraw %}
```

Setting minimum and maximum values for zoom factor

The zooming range can be adjusted using the [minZoom](#) and [maxZoom](#) properties in [zoomSettings](#). The minZoom value is set to 1 by default, and the maxZoom value is set to 10.

```
{% raw %}
```

```
`ts
```

```
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent zoomSettings={ { enable: true, minZoom: 2,
    maxZoom: 12 } }>
    <Inject services={[Zoom]}/>
    <LayersDirective>
    <LayerDirective shapeData={world_map}>
    </LayerDirective>
    </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`
```

```
{% endraw %}
```

Zooming with animation

To zoom in or zoom out the Maps with animation, use the [animationDuration](#) property in [layers](#).

```
{% raw %}
```

```
`ts
```

```
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent zoomSettings={ { enable: true } }>
    <Inject services={[Zoom]}/>
    </MapsComponent>
  );
}
```

```

<LayersDirective>
<LayerDirective shapeData={world_map} animationDuration={500}>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

{% endraw %}

```

Customizing the zoom toolbar

The zoom toolbar can be customized by using the [toolbarSettings](#) option in the [zoomSettings](#). The following properties can be used to customize the zoom toolbar.

- [backgroundColor](#) - It is used to customize the background color of the zoom toolbar.
- [borderOpacity](#) - It is used to customize the opacity of the border of the zoom toolbar.
- [borderWidth](#) - It is used to customize the thickness of the border of the zoom toolbar.
- [borderColor](#) - It is used to customize the color of the border of the zoom toolbar.
- [horizontalAlignment](#) - It is used to position the zoom toolbar in near, far, and center positions to customize its horizontal placement.
- [verticalAlignment](#) - It is used to position the zoom toolbar in near, far, and center positions to customize its vertical placement.
- [orientation](#) - It is used to change the orientation (horizontal/vertical) of the zoom toolbar.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Zoom }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings= {{
      enable: true,
      toolbarSettings:{
        orientation:'Vertical',
        backgroundColor:'pink',
        borderWidth:3,
        borderColor:'green',
        verticalAlignment:'Near',
        buttonSettings: {
          toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset']
        }
      }
    }}

```



```

    }
  }>
  <Inject services={[Zoom]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
        shapeSettings= {{
          fill: '#C1DFF5'
        }}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Zoom }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings= {{
      enable: true,
      toolbarSettings:{
        orientation:'Vertical',
        backgroundColor:'pink',
        borderWidth:3,
        borderColor:'green',
        verticalAlignment:'Near',
        buttonSettings: {
          toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan',
'Reset']
        }
      }
    }}>
    <Inject services={[Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings= {{
            fill: '#C1DFF5'
          }}>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

<!-- markdownlint-disable MD036 -->

Customizing the buttons in the zoom toolbar

The appearance of the buttons in the zoom toolbar can be customized by using the [buttonSettings](#) option in the [toolbarSettings](#) of the [zoomSettings](#) property. The following properties can be used to customize the zoom toolbar buttons.

- [fill](#) - It is used to set the background color of the buttons.
- [color](#) - It is used to customize the color of the icons inside the button.
- [borderOpacity](#) - It is used to set the opacity of the border of the zoom toolbar buttons.
- [borderWidth](#) - It is used to set the thickness of the border of the zoom toolbar buttons.
- [borderColor](#) - It is used to set the color of the border of the zoom toolbar buttons.
- [radius](#) - It is used to set the size of the button.
- [selectionColor](#) - It is used to set the color of the icons inside the button when selection is performed.
- [highlightColor](#) - It is used to change the color of the button when the mouse is hovered over it.
- [padding](#) - It is used to change the padding space between each button.
- [opacity](#) - It is used to change the opacity of the button.
- [toolbarItems](#) - It is used to change the items that should be displayed in the zoom toolbar. By default, zoom-in, zoom-out, and reset buttons will be available. Other options include selection zoom and panning.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Zoom } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent zoomSettings= {{
      enable: true,
      toolbarSettings:{
        buttonSettings: {
          fill:'pink',
          padding: 10,
          toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan',
'Reset'],
          color: 'red',
          borderColor:'green',
          radius:35,
          selectionColor:'#d55e5e',
          hightlightColor:'#5ed59a',
          opacity:0.6,
          borderWidth: 2
        }
      }
    }}>
    <Inject services={[Zoom]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
```

```

        shapeSettings= {{
            fill: '#C1DFF5'
        }}>
    </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Zoom }
from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent zoomSettings= {{
            enable: true,
            toolbarSettings:{
                buttonSettings: {
                    fill:'pink',
                    padding: 10,
                    toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan',
'Reset'],

                    color: 'red',
                    borderColor:'green',
                    radius:35,
                    selectionColor:'#d55e5e',
                    highlightColor:'#5ed59a',
                    opacity:0.6,
                    borderWidth: 2
                }
            }
        }}>
        <Inject services={[Zoom]} />
        <LayersDirective>
            <LayerDirective shapeData={world_map}
                shapeSettings= {{
                    fill: '#C1DFF5'
                }}>
            </LayerDirective>
        </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

<!-- markdownlint-disable MD036 -->

Customizing the tooltip of the zoom toolbar

The appearance of the tooltip of the zoom toolbar can be customized by using the [tooltipSettings](#) option in the [toolbarSettings](#) of the [zoomSettings](#) property. The following properties are available to customize the zoom toolbar tooltip.

- [visible](#) - Enables or disables the tooltip of the zoom toolbar.
- [fill](#) - It is used to change the background color of the tooltip of the zoom toolbar.
- [borderOpacity](#) - It is used to change the opacity of the border of the zoom toolbar's tooltip.
- [borderWidth](#) - It is used to change the thickness of the border of the zoom toolbar's tooltip.
- [borderColor](#) - It is used to change the color of the border of the zoom toolbar's tooltip.
- [fontColor](#) - It is used to change the color of the text in the tooltip of the zoom toolbar.
- [fontFamily](#) - It is used to change the font family of the text in the tooltip of the zoom toolbar.
- [fontStyle](#) - It is used to change the font style of the text in the tooltip of the zoom toolbar.
- [fontWeight](#) - It is used to change the font weight of the text in the tooltip of the zoom toolbar.
- [fontSize](#) - It is used to change the size of the text in the tooltip of the zoom toolbar.
- [fontOpacity](#) - It is used to change the opacity of the text in the tooltip of the zoom toolbar.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Zoom }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings= {{
      enable: true,
      toolbarSettings:{
        tooltipSettings:{
          visible:true,
          borderWidth:2,
          borderColor:'green',
          fontColor:'black',
          fill:'violet',
          fontFamily:'Times New Roman',
          fontWeight:200,
          fontSize:'22px',
          fontOpacity:1
        },
        buttonSettings: {
          toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset']
        }
      }
    }}>
    <Inject services={[Zoom]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}>
        shapeSettings= {{
          fill: '#C1DFF5'
        }}
      </LayerDirective>
    </LayersDirective>
  )
}
```

```

        }}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Zoom }
from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings= {{
      enable: true,
      toolbarSettings:{
        tooltipSettings:{
          visible:true,
          borderWidth:2,
          borderColor:'green',
          fontColor:'black',
          fill:'violet',
          fontFamily:'Times New Roman',
          fontWeight:200,
          fontSize:'22px',
          fontOpacity:1
        },
        buttonSettings: {
          toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan',
'Reset']
        }
      }
    }}>
      <Inject services={[Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeSettings= {{
            fill: '#C1DFF5'
          }}>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Selection

Each shape in the Maps can be selected and deselected during interaction with the shapes. Selection is enabled by setting the [enable](#) property of [selectionSettings](#) to **true**.

The following properties are available to customize the selection of Maps elements such as shapes, bubbles, markers and legends.

- [border](#) - To customize the color, width and opacity of the border of which element is selected in Maps.
- [fill](#) - Applies the color for the element that is selected.
- [opacity](#) - To customize the transparency for the element that is selected.
- [enableMultiSelect](#) - To enable or disable the selection for multiple shapes or markers or bubbles in the Maps.

By tapping on the specific legend, the shapes which are bounded to the selected legend is also selected and vice versa.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Selection, Inject,
Legend } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent legendSettings={{visible: true}}>
      <Inject services={[Selection, Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
selectionSettings={ {
  enable: true,
  fill: 'blue',
  border: { color: 'white', width: 2 }
} } dataSource=[
  { "Country": "China", "Membership": "Permanent" },
  { "Country": "France", "Membership": "Permanent" },
  { "Country": "Russia", "Membership": "Permanent" },
  { "Country": "Kazakhstan", "Membership": "Non-
Permanent" },
  { "Country": "Poland", "Membership": "Non-
Permanent" },
  { "Country": "Sweden", "Membership": "Non-Permanent" }
]} shapePropertyPath="name" shapeDataPath="Country"
shapeSettings={{
  colorValuePath: 'Membership',
  colorMapping: [
    { value: 'Permanent', color: '#D84444' },
    { value: 'Non-Permanent', color: '#316DB5' }
  ]
}}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
```

```

}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Selection, Inject,
Legend } from '@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent legendSettings={{visible: true}}>
      <Inject services={[Selection, Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
selectionSettings={ {
          enable: true,
          fill: 'blue',
          border: { color: 'white', width: 2 }
        } } dataSource=[
          { "Country": "China", "Membership": "Permanent" },
          { "Country": "France", "Membership": "Permanent" },
          { "Country": "Russia", "Membership": "Permanent" },
          { "Country": "Kazakhstan", "Membership": "Non-
Permanent" },
          { "Country": "Poland", "Membership": "Non-
Permanent" },
          { "Country": "Sweden", "Membership": "Non-Permanent" }
        ] shapePropertyPath="name" shapeDataPath="Country"
shapeSettings={{
          colorValuePath: 'Membership',
          colorMapping: [
            { value: 'Permanent', color: '#D84444' },
            { value: 'Non-Permanent', color: '#316DB5' }
          ]
        }}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Enable selection for bubbles

To enable the selection for bubbles in Maps, set the [selectionSettings](#) in [bubbleSettings](#) and set the [enable](#) property of [selectionSettings](#) as **true**.

To use the bubble feature, the Bubble module must be injected.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble, Selection } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[Bubble, Selection]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath="name" shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population"
                                dataSource={[
                                  { name: 'India', population:
'38332521' },,
                                  { name: 'Pakistan',
population: '3090416' },,
                                  { name: 'New Zealand',
population: '19651127' }
                                ]}
                                selectionSettings={{
                                  enable: true,
                                  fill: 'green',
                                  border: { color: 'white',
width: 2}
                                }} />
          </BubblesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble, Selection } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <Inject services={[Bubble, Selection]}/>
```



```

        <LayersDirective>
          <LayerDirective shapeData={world_map}
            shapeDataPath="name" shapePropertyPath="name">
            <BubblesDirective>
              <BubbleDirective visible={true}
                valuePath="population"
                width: 2}
                dataSource={[
                  { name: 'India', population:
                    '38332521' },
                  { name: 'Pakistan',
                    population: '3090416' },
                  { name: 'New Zealand',
                    population: '19651127' }
                ]}
                selectionSettings={{
                  enable: true,
                  fill: 'green',
                  border: { color: 'white',
                    }} />
            </BubblesDirective>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}

```

Enable selection for markers

To enable the selection for markers in Maps, set the [selectionSettings](#) in the [markerSettings](#) and set the [enable](#) property of the [selectionSettings](#) as **true**.

To use the marker feature, the Marker module must be injected.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
  MarkerDirective, Selection, Marker, Inject } from '@syncfusion/ej2-react-
  maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker, Selection]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              height={20}
              width={20}
              animationDuration={0}

```

```

        fill="green"
        shape="Balloon"
        dataSource=[
          { latitude:
49.95121990866204, longitude: 18.468749999999998, name:'Europe' },
          { latitude:
59.88893689676585, longitude: -109.3359375, name:'North America'},
          { latitude: -
6.64607562172573, longitude: -55.54687499999999, name:'South America'}
        ]
        selectionSettings={{
          enable: true,
          fill: 'blue',
          border: { color: 'white',
width: 2}
        }}
      </MarkerDirective>
    </MarkersDirective>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Selection, Marker, Inject } from '@syncfusion/ej2-react-
maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker, Selection]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              height={20}
              width={20}
              animationDuration={0}
              fill="green"
              shape="Balloon"
              dataSource=[
                { latitude:
49.95121990866204, longitude: 18.468749999999998, name:'Europe' },
                { latitude:
59.88893689676585, longitude: -109.3359375, name:'North America'},
                { latitude: -
6.64607562172573, longitude: -55.54687499999999, name:'South America'}

```

```

width: 2}

    selectionSettings={{
      enable: true,
      fill: 'blue',
      border: { color: 'white',

    }}>
    </MarkerDirective>
  </MarkersDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>

);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Enable selection for polygons

When the [enable](#) property of [selectionSettings](#) is set to **true**, the polygon shapes can be selected via user interaction. The following properties are available in [selectionSettings](#) to customize the polygon shape when it is selected.

- [enableMultiSelect](#) - It is used to enable multiple selection of polygon shapes.
- [fill](#) - It is used to change the color of the selected polygon shape.
- [opacity](#) - It is used to change the opacity of the selected polygon shape.
- [border](#) - This property is used to change the color, width, and opacity of the border of the selected polygon shape.

To use the polygon feature, the **Polygon** module must be injected, as described in [this link](#).

The following example shows how to select the polygon shape in the geometry map.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon,
Highlight, Selection } from '@syncfusion/ej2-react-maps';
export function App() {
  var polygonSettings = {
    polygons: [
      {
        points: [
          { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
          { longitude: -1.6479633699113947, latitude:
33.58989612266137 },
          { longitude: -1.4201220366858252, latitude:
32.819439646045254 },
          { longitude: -1.197974596225663, latitude:
32.26940895444655 },

```

```
{ longitude: -2.891112397949655, latitude:
32.10303058820031 },
{ longitude: -3.8246984550501963, latitude:
31.34551662687602 },
{ longitude: -3.720166273688733, latitude:
30.758086682848685 },
{ longitude: -5.6571886081189575, latitude:
29.613582597203006 },
{ longitude: -7.423353242214745, latitude:
29.44328441403087 },
{ longitude: -8.6048931685323, latitude:
28.761444633616776 },
{ longitude: -8.695726975465703, latitude:
27.353491085576195 },
{ longitude: 3.837867279970908, latitude:
19.15916564839422 },
{ longitude: 6.0705408799045415, latitude:
19.48749097192868 },
{ longitude: 12.055736352807713, latitude:
23.694596786078293 },
{ longitude: 11.272522332402986, latitude:
24.289329186946034 },
{ longitude: 10.30872578261932, latitude:
24.65419958524693 },
{ longitude: 9.910236690050027, latitude:
25.48943950947175 },
{ longitude: 9.432639882414293, latitude:
26.398372489836902 },
{ longitude: 9.898266456582292, latitude:
26.73489453809293 },
{ longitude: 9.560243026853641, latitude:
30.31040379467153 },
{ longitude: 8.943853847283322, latitude:
32.350324876652195 },
{ longitude: 7.57004059025715, latitude:
33.75071049019398 },
{ longitude: 8.0906322609153, latitude:
34.69043151009983 },
{ longitude: 8.363285449347273, latitude:
35.38654406371319 },
{ longitude: 8.26139549449448, latitude:
36.44751078733985 },
{ longitude: 8.61100824823302, latitude:
36.881913362940196 },
{ longitude: 7.4216488925819135, latitude:
37.021408008916254 },
{ longitude: 6.461182254165351, latitude:
36.99092409199429 },
{ longitude: 5.297178918070159, latitude:
36.69985479014656 },
{ longitude: 3.6718056161224695, latitude:
36.86470546831693 },
{ longitude: 1.2050052555659931, latitude:
36.57658056301722 },
{ longitude: -0.26968570003779746, latitude:
35.806903541813625 },
```

```

        { longitude: -0.995191786435754, latitude:
35.58466127904214 },
        { longitude: -1.8920678947185365, latitude:
35.06195799239681 }
      ],
      fill: 'blue',
      opacity: 0.7,
      borderColor: 'green',
      borderWidth: 2,
      borderOpacity: 0.7
    }
  ],
  highlightSettings: {
    enable: true,
    fill: 'blue',
    opacity: 0.7,
    border: {
      color: 'green',
      width: 2,
      opacity: 0.7
    }
  },
  selectionSettings: {
    enable: true,
    fill: 'violet',
    enableMultiSelect: false,
    opacity: 0.8,
    border: {
      color: 'cyan',
      opacity: 1,
      width: 7
    }
  }
};
return (
  <MapsComponent >
    <Inject services={[Polygon, Highlight, Selection]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
polygonSettings={polygonSettings}
      >
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```
import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon,
Highlight, Selection } from '@syncfusion/ej2-react-maps';
export function App() {
  let polygonSettings : object = {
    polygons: [
      {
        points: [
          { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
          { longitude: -1.6479633699113947, latitude:
33.58989612266137 },
          { longitude: -1.4201220366858252, latitude:
32.819439646045254 },
          { longitude: -1.197974596225663, latitude:
32.26940895444655 },
          { longitude: -2.891112397949655, latitude:
32.10303058820031 },
          { longitude: -3.8246984550501963, latitude:
31.34551662687602 },
          { longitude: -3.720166273688733, latitude:
30.758086682848685 },
          { longitude: -5.6571886081189575, latitude:
29.613582597203006 },
          { longitude: -7.423353242214745, latitude:
29.44328441403087 },
          { longitude: -8.6048931685323, latitude:
28.761444633616776 },
          { longitude: -8.695726975465703, latitude:
27.353491085576195 },
          { longitude: 3.837867279970908, latitude:
19.15916564839422 },
          { longitude: 6.0705408799045415, latitude:
19.48749097192868 },
          { longitude: 12.055736352807713, latitude:
23.694596786078293 },
          { longitude: 11.272522332402986, latitude:
24.289329186946034 },
          { longitude: 10.30872578261932, latitude:
24.65419958524693 },
          { longitude: 9.910236690050027, latitude:
25.48943950947175 },
          { longitude: 9.432639882414293, latitude:
26.398372489836902 },
          { longitude: 9.898266456582292, latitude:
26.73489453809293 },
          { longitude: 9.560243026853641, latitude:
30.31040379467153 },
          { longitude: 8.943853847283322, latitude:
32.350324876652195 },
          { longitude: 7.57004059025715, latitude:
33.75071049019398 },
          { longitude: 8.0906322609153, latitude:
34.69043151009983 },
          { longitude: 8.363285449347273, latitude:
35.38654406371319 },
          { longitude: 8.26139549449448, latitude:
36.44751078733985 },
```

```

    { longitude: 8.61100824823302, latitude:
36.881913362940196 },
    { longitude: 7.4216488925819135, latitude:
37.021408008916254 },
    { longitude: 6.461182254165351, latitude:
36.99092409199429 },
    { longitude: 5.297178918070159, latitude:
36.69985479014656 },
    { longitude: 3.6718056161224695, latitude:
36.86470546831693 },
    { longitude: 1.2050052555659931, latitude:
36.57658056301722 },
    { longitude: -0.26968570003779746, latitude:
35.806903541813625 },
    { longitude: -0.995191786435754, latitude:
35.58466127904214 },
    { longitude: -1.8920678947185365, latitude:
35.06195799239681 }
    ],
    fill: 'blue',
    opacity: 0.7,
    borderColor: 'green',
    borderWidth: 2,
    borderOpacity: 0.7
  }
],
highlightSettings: {
  enable: true,
  fill: 'blue',
  opacity: 0.7,
  border: {
    color: 'green',
    width: 2,
    opacity: 0.7
  }
},
selectionSettings: {
  enable: true,
  fill: 'violet',
  enableMultiSelect: false,
  opacity: 0.8,
  border: {
    color: 'cyan',
    opacity: 1,
    width: 7
  }
}
};
return (
  <MapsComponent >
    <Inject services={[Polygon, Highlight, Selection]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
        polygonSettings={polygonSettings}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>

```

```

    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% enddraw %}

```

Public method for the shape selection

The [shapeSelection](#) method can be used to select each shape in the Maps.

LayerIndex, propertyName, country name, and selected value as a boolean state(true / false) are the input parameters for this method.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Selection, Inject }
from '@syncfusion/ej2-react-maps';
export function App() {
  var mapsInstance;
  function selectclickHandler() {
    mapsInstance.shapeSelection(0, "continent", "Asia", true);
  }
  function unselectclickHandler() {
    mapsInstance.shapeSelection(0, "continent", "Asia", false);
  }
  return (<div>
    <MapsComponent ref={maps => mapsInstance = maps}>
      <Inject services={[Selection]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
selectionSettings={ {
          enable: true,
          fill: 'green',
          border: { color: 'white', width: 2 }
        } }>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
  <button value='select'
onClick={selectclickHandler}>select</button>
  <button value='unselect'
onClick={unselectclickHandler}>unselect</button></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

```



```
import { MapsComponent, LayersDirective, LayerDirective, Selection, Inject }
from '@syncfusion/ej2-react-maps';
export function App() {
  let mapsInstance: MapsComponent;
  function selectclickHandler() {
    mapsInstance.shapeSelection(0, "continent", "Asia", true);
  }
  function unselectclickHandler() {
    mapsInstance.shapeSelection(0, "continent", "Asia", false);
  }
  return (<div>
    <MapsComponent ref={maps => mapsInstance = maps}>
      <Inject services={[Selection]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
selectionSettings={ {
  enable: true,
  fill: 'green',
  border: { color: 'white', width: 2 }
} }>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
  <button value='select'
onClick={selectclickHandler}>select</button>
  <button value='unselect'
onClick={unselectclickHandler}>unselect</button></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Initial shape selection

The shape is initially selected using the [initialShapeSelection](#), and the values are mapped to the [shapePath](#) and [shapeValue](#).

initialShapeSelection is an Array property.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, InitialShapeSelectionsDirective,
InitialShapeSelectionDirective, LayerDirective, Selection, Inject } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Selection]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
selectionSettings={{
  enable: true,
  fill: 'green',
```

```

        border: { color: 'white', width: 2 }
      }}>
      <InitialShapeSelectionsDirective>
        <InitialShapeSelectionDirective shapePath={'continent'}
shapeValue={'Africa'}>
        </InitialShapeSelectionDirective>
        <InitialShapeSelectionDirective shapePath={'name'}
shapeValue={'India'}>
        </InitialShapeSelectionDirective>
      </InitialShapeSelectionsDirective>
    </LayerDirective>
  </LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, InitialShapeSelectionsDirective,
InitialShapeSelectionDirective, LayerDirective, Selection, Inject } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Selection]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
selectionSettings={{
          enable: true,
          fill: 'green',
          border: { color: 'white', width: 2 }
        }}>
        <InitialShapeSelectionsDirective>
          <InitialShapeSelectionDirective shapePath={'continent'}
shapeValue={'Africa'}>
          </InitialShapeSelectionDirective>
          <InitialShapeSelectionDirective shapePath={'name'}
shapeValue={'India'}>
          </InitialShapeSelectionDirective>
        </InitialShapeSelectionsDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Initial marker selection

Using the [initialMarkerSelection](#), the marker shape can be selected initially. Markers render based on the [latitude](#) and [longitude](#) values.

`initialMarkerSelection` is an Array property.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Selection, Marker, Inject } from '@syncfusion/ej2-react-
maps';
export function App() {
  return (
    <MapsComponent >
    <Inject services={[Marker, Selection]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}>
        <MarkersDirective>
          <MarkerDirective visible={true}
            height={20}
            width={20}
            animationDuration={0}
            fill="green"
            shape="Balloon"
            dataSource=[
              { latitude:
49.95121990866204, longitude: 18.468749999999998, name:'Europe' },
              { latitude:
59.88893689676585, longitude: -109.3359375, name:'North America'},
              { latitude: -
6.64607562172573, longitude: -55.54687499999999, name:'South America'}
            ]
            initialMarkerSelection={{
              latitude: -6.64607562172573,
longitude: -55.54687499999999
            }}
            selectionSettings={{
              enable: true,
              fill: 'blue',
              border: { color: 'white',
width: 2}
            }}
          </MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Selection, Marker, Inject } from '@syncfusion/ej2-react-
maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker, Selection]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}
              height={20}
              width={20}
              animationDuration={0}
              fill="green"
              shape="Balloon"
              dataSource=[
                { latitude:
49.95121990866204, longitude: 18.468749999999998, name:'Europe' },
                { latitude:
59.88893689676585, longitude: -109.3359375, name:'North America'},
                { latitude: -
6.64607562172573, longitude: -55.54687499999999, name:'South America'}
              ]}
              initialMarkerSelection={{
                latitude: -6.64607562172573,
longitude: -55.54687499999999
              }}
              selectionSettings={{
                enable: true,
                fill: 'blue',
                border: { color: 'white',
width: 2}
              }}
            </MarkerDirective>
          </MarkersDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Highlight

Each shape in the Maps can be highlighted during mouse hover on the Maps elements such as shapes, bubbles, markers and legends. Highlight is enabled by setting the [enable](#) property of [highlightSettings](#) to **true**.

The following properties are available to customize the highlight of Maps elements such as shapes, bubbles, markers and legends.

- [border](#) - To customize the color, width and opacity of the border of which element is highlighted in Maps.
- [fill](#) - Applies the color for the element that is highlighted.
- [opacity](#) - To customize the transparency for the element that is highlighted.

Hovering on the specific legend, the shapes which are bounded to the selected legend is also highlighted and vice versa.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Highlight, Inject,
Legend } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={{visible: true}}>
      <Inject services={[Highlight, Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
highlightSettings={ {
          enable: true,
          fill: 'green',
          border: { color: 'white', width: 2 }
        } }
        dataSource=[
          { "Country": "China", "Membership":
"Permanent"},
          { "Country": "France", "Membership": "Permanent"
},
          { "Country": "Russia", "Membership":
"Permanent"},
          { "Country": "Kazakhstan", "Membership": "Non-
Permanent"},
          { "Country": "Poland", "Membership": "Non-
Permanent"},
          { "Country": "Sweden", "Membership": "Non-
Permanent"}
        ]} shapePropertyPath="name" shapeDataPath="Country"
        shapeSettings={{
          colorValuePath: 'Membership',
          colorMapping: [
            { value: 'Permanent', color: '#D84444' },
            { value: 'Non-Permanent', color: '#316DB5' }
          ]
        }}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Highlight, Inject,
Legend } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent legendSettings={{visible: true}}>
      <Inject services={[Highlight, Legend]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
highlightSettings={ {
          enable: true,
          fill: 'green',
          border: { color: 'white', width: 2 }
        } }
        dataSource=[
          { "Country": "China", "Membership":
"Permanent" },
          { "Country": "France", "Membership": "Permanent"
},
          { "Country": "Russia", "Membership":
"Permanent" },
          { "Country": "Kazakhstan", "Membership": "Non-
Permanent" },
          { "Country": "Poland", "Membership": "Non-
Permanent" },
          { "Country": "Sweden", "Membership": "Non-
Permanent" }
        ] shapePropertyPath="name" shapeDataPath="Country"
        shapeSettings={{
          colorValuePath: 'Membership',
          colorMapping: [
            { value: 'Permanent', color: '#D84444' },
            { value: 'Non-Permanent', color: '#316DB5' }
          ]
        }}
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Enable highlight for bubbles

To enable the highlight for bubbles in Maps, set the [highlightSettings](#) in [bubbleSettings](#) and set the [enable](#) property of [highlightSettings](#) as **true**.

To use the bubble feature, the Bubble module must be injected.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble, Highlight } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Bubble, Highlight]}/>
      <LayersDirective>
        <LayerDirective shapeData={world_map}
shapeDataPath="name" shapePropertyPath="name">
          <BubblesDirective>
            <BubbleDirective visible={true}
valuePath="population"
                                dataSource=[
                                  { name: 'India', population:
'38332521' },
                                  { name: 'Pakistan',
population: '3090416' },
                                  { name: 'New Zealand',
population: '19651127' }
                                ]
                                highlightSettings={{
                                  enable: true,
                                  fill: 'green',
                                  border: { color: 'white',
width: 2}
                                }} />
          </BubblesDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject } from
'@syncfusion/ej2-react-maps';
import { BubblesDirective, BubbleDirective, Bubble, Highlight } from
'@syncfusion/ej2-react-maps';
export function App() {
```

```

return (
  <MapsComponent >
    <Inject services={[Bubble, Highlight]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
shapeDataPath="name" shapePropertyPath="name">
        <BubblesDirective>
          <BubbleDirective visible={true}
valuePath="population"
width: 2}
          dataSource={[
            { name: 'India', population:
'38332521' },
            { name: 'Pakistan',
population: '3090416' },
            { name: 'New Zealand',
population: '19651127' }
          ]}
          highlightSettings={{
            enable: true,
            fill: 'green',
            border: { color: 'white',
}} />
        </BubblesDirective>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Enable highlight for markers

To enable the highlight for markers in Maps, set the [highlightSettings](#) in [markerSettings](#) and set the [enable](#) property of [highlightSettings](#) as **true**.

To use the marker feature, the Marker module must be injected.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Highlight , Marker, Inject } from '@syncfusion/ej2-react-
maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker, Highlight]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective visible={true}

```



```

height={20}
width={20}
animationDuration={0}
fill="green"
shape="Balloon"
dataSource=[
  { latitude:
49.95121990866204, longitude: 18.468749999999998, name: 'Europe' },
  { latitude:
59.88893689676585, longitude: -109.3359375, name: 'North America'},
  { latitude: -
6.64607562172573, longitude: -55.54687499999999, name: 'South America'}
]
highlightSettings={{
  enable: true,
  fill: 'blue',
  border: { color: 'white',
width: 2}
}}
</MarkerDirective>
</MarkersDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
MarkerDirective, Highlight , Marker, Inject } from '@syncfusion/ej2-react-
maps';
export function App() {
  return (
    <MapsComponent >
    <Inject services={[Marker, Highlight]} />
    <LayersDirective>
    <LayerDirective shapeData={world_map}>
    <MarkersDirective>
    <MarkerDirective visible={true}
height={20}
width={20}
animationDuration={0}
fill="green"
shape="Balloon"
dataSource=[
  { latitude:
49.95121990866204, longitude: 18.468749999999998, name: 'Europe' },

```

```

        { latitude:
59.88893689676585, longitude: -109.3359375, name: 'North America'},
        { latitude: -
6.64607562172573, longitude: -55.54687499999999, name: 'South America'}
    ]}
    highlightSettings={{
        enable: true,
        fill: 'blue',
        border: { color: 'white',
width: 2}
    }}
  </MarkerDirective>
</MarkersDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Enable highlight for polygons

The polygon shapes can be highlighted via user interaction if the [enable](#) property of [highlightSettings](#) is set to **true**. The following properties are available in [highlightSettings](#) to customize the polygon shape when it is highlighted.

- [fill](#) - It is used to change the color of the highlighted polygon shape.
- [opacity](#) - It is used to change the opacity of the highlighted polygon shape.
- [border](#) - This property is used to change the color, width, and opacity of the border of the highlighted polygon shape.

To use the polygon feature, the **Polygon** module must be injected, as described in [this link](#).

The following example shows how to highlight a polygon shape on a geometry map.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon,
Highlight, Selection } from '@syncfusion/ej2-react-maps';
export function App() {
    var polygonSettings = {
        polygons: [
            {
                points: [
                    { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
                    { longitude: -1.6479633699113947, latitude:
33.58989612266137 },
                    { longitude: -1.4201220366858252, latitude:
32.819439646045254 },

```

```
{ longitude: -1.197974596225663, latitude:
32.26940895444655 },
{ longitude: -2.891112397949655, latitude:
32.10303058820031 },
{ longitude: -3.8246984550501963, latitude:
31.34551662687602 },
{ longitude: -3.720166273688733, latitude:
30.758086682848685 },
{ longitude: -5.6571886081189575, latitude:
29.613582597203006 },
{ longitude: -7.423353242214745, latitude:
29.44328441403087 },
{ longitude: -8.6048931685323, latitude:
28.761444633616776 },
{ longitude: -8.695726975465703, latitude:
27.353491085576195 },
{ longitude: 3.837867279970908, latitude:
19.15916564839422 },
{ longitude: 6.0705408799045415, latitude:
19.48749097192868 },
{ longitude: 12.055736352807713, latitude:
23.694596786078293 },
{ longitude: 11.272522332402986, latitude:
24.289329186946034 },
{ longitude: 10.30872578261932, latitude:
24.65419958524693 },
{ longitude: 9.910236690050027, latitude:
25.48943950947175 },
{ longitude: 9.432639882414293, latitude:
26.398372489836902 },
{ longitude: 9.898266456582292, latitude:
26.73489453809293 },
{ longitude: 9.560243026853641, latitude:
30.31040379467153 },
{ longitude: 8.943853847283322, latitude:
32.350324876652195 },
{ longitude: 7.57004059025715, latitude:
33.75071049019398 },
{ longitude: 8.0906322609153, latitude:
34.69043151009983 },
{ longitude: 8.363285449347273, latitude:
35.38654406371319 },
{ longitude: 8.26139549449448, latitude:
36.44751078733985 },
{ longitude: 8.61100824823302, latitude:
36.881913362940196 },
{ longitude: 7.4216488925819135, latitude:
37.021408008916254 },
{ longitude: 6.461182254165351, latitude:
36.99092409199429 },
{ longitude: 5.297178918070159, latitude:
36.69985479014656 },
{ longitude: 3.6718056161224695, latitude:
36.86470546831693 },
{ longitude: 1.2050052555659931, latitude:
36.57658056301722 },
```

```

        { longitude: -0.26968570003779746, latitude:
35.806903541813625 },
        { longitude: -0.995191786435754, latitude:
35.58466127904214 },
        { longitude: -1.8920678947185365, latitude:
35.06195799239681 }
      ],
      fill: 'red',
      opacity: 0.7,
      borderColor: 'green',
      borderWidth: 2,
      borderOpacity: 0.7
    }
  ],
  highlightSettings: {
    enable: true,
    fill: 'yellow',
    opacity: 0.4,
    border: {
      color: 'blue',
      opacity: 0.6,
      width: 4
    }
  },
  selectionSettings: {
    enable: true,
    fill: 'red',
    opacity: 0.7,
    border: {
      color: 'green',
      width: 2,
      opacity: 0.7
    }
  }
};
return (
  <MapsComponent >
    <Inject services={[Polygon, Highlight, Selection]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
polygonSettings={polygonSettings}
      >
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";

```

```
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Inject, Polygon,
Highlight, Selection } from '@syncfusion/ej2-react-maps';
export function App() {
  let polygonSettings: object = {
    polygons: [
      {
        points: [
          { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
          { longitude: -1.6479633699113947, latitude:
33.58989612266137 },
          { longitude: -1.4201220366858252, latitude:
32.819439646045254 },
          { longitude: -1.197974596225663, latitude:
32.26940895444655 },
          { longitude: -2.891112397949655, latitude:
32.10303058820031 },
          { longitude: -3.8246984550501963, latitude:
31.34551662687602 },
          { longitude: -3.720166273688733, latitude:
30.758086682848685 },
          { longitude: -5.6571886081189575, latitude:
29.613582597203006 },
          { longitude: -7.423353242214745, latitude:
29.44328441403087 },
          { longitude: -8.6048931685323, latitude:
28.761444633616776 },
          { longitude: -8.695726975465703, latitude:
27.353491085576195 },
          { longitude: 3.837867279970908, latitude:
19.15916564839422 },
          { longitude: 6.0705408799045415, latitude:
19.48749097192868 },
          { longitude: 12.055736352807713, latitude:
23.694596786078293 },
          { longitude: 11.272522332402986, latitude:
24.289329186946034 },
          { longitude: 10.30872578261932, latitude:
24.65419958524693 },
          { longitude: 9.910236690050027, latitude:
25.48943950947175 },
          { longitude: 9.432639882414293, latitude:
26.398372489836902 },
          { longitude: 9.898266456582292, latitude:
26.73489453809293 },
          { longitude: 9.560243026853641, latitude:
30.31040379467153 },
          { longitude: 8.943853847283322, latitude:
32.350324876652195 },
          { longitude: 7.57004059025715, latitude:
33.75071049019398 },
          { longitude: 8.0906322609153, latitude:
34.69043151009983 },
          { longitude: 8.363285449347273, latitude:
35.38654406371319 },
        ]
      }
    ]
  }
```

```

        { longitude: 8.26139549449448, latitude:
36.44751078733985 },
        { longitude: 8.61100824823302, latitude:
36.881913362940196 },
        { longitude: 7.4216488925819135, latitude:
37.021408008916254 },
        { longitude: 6.461182254165351, latitude:
36.99092409199429 },
        { longitude: 5.297178918070159, latitude:
36.69985479014656 },
        { longitude: 3.6718056161224695, latitude:
36.86470546831693 },
        { longitude: 1.2050052555659931, latitude:
36.57658056301722 },
        { longitude: -0.26968570003779746, latitude:
35.806903541813625 },
        { longitude: -0.995191786435754, latitude:
35.58466127904214 },
        { longitude: -1.8920678947185365, latitude:
35.06195799239681 }
      ],
      fill: 'red',
      opacity: 0.7,
      borderColor: 'green',
      borderWidth: 2,
      borderOpacity: 0.7
    }
  ],
  highlightSettings: {
    enable: true,
    fill: 'yellow',
    opacity: 0.4,
    border: {
      color: 'blue',
      opacity: 0.6,
      width: 4
    }
  },
  selectionSettings: {
    enable: true,
    fill: 'red',
    opacity: 0.7,
    border: {
      color: 'green',
      width: 2,
      opacity: 0.7
    }
  }
};
return (
  <MapsComponent >
    <Inject services={[Polygon, Highlight, Selection]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
        polygonSettings={polygonSettings}>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent >
);

```

```

        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Tooltip

On mouse over or touch end event, the tooltip is used to get more information about the layer, bubble, or marker. To enable tooltip in Maps, the **Tooltip** module must be injected into Maps using **Inject services={{Tooltip}}** tag. It can be enabled separately for layer or bubble or marker by using the [visible](#) property of [tooltipSettings](#) as **true**. The [valuePath](#) property in the tooltip takes the field name that presents in data source and displays that value as tooltip text. The [tooltipDisplayMode](#) property is used to change the display mode of the tooltip in Maps. Following display modes of tooltip are available in the Maps component. By default, [tooltipDisplayMode](#) is set to **MouseMove**.

- MouseMove
- Click
- DoubleClick

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MapsTooltip, Inject
} from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent >
            <Inject services={[MapsTooltip]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map} tooltipSettings={
{
                    visible: true,
                    valuePath: 'name'
                } }>
            </LayerDirective>
        </LayersDirective>
    </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```
import { MapsComponent, LayersDirective, LayerDirective, MapsTooltip, Inject
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[MapsTooltip]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map} tooltipSettings={
{
          visible: true,
          valuePath: 'name'
        } }>
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Customization

The following properties are available in the [tooltipSettings](#) to customize the tooltip of the Maps component.

- [border](#) - To customize the color, width and opacity of the border of the tooltip in layers, markers, and bubbles of Maps.
- [fill](#) - Applies the color of the tooltip in layers, markers, and bubbles of Maps.
- [format](#) - To customize the format of the tooltip in layers, markers, and bubbles of Maps
- [textStyle](#) - To customize the style of the text in the tooltip for layers, markers, and bubbles of Maps.

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { default_data } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MapsTooltip, Inject
} from '@syncfusion/ej2-react-maps';
export function App() {
  return(<div><MapsComponent >
    <Inject services={[MapsTooltip]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
dataSource={default_data} shapeDataPath='continent'
shapePropertyPath='continent'
      tooltipSettings={{
        visible: true,
        valuePath: 'continent',
        format: '${continent}',
        fill: '#D0D0D0',
        textStyle: {

```



```

        color: 'green',
        fontFamily: 'Times New Roman',
        fontStyle: 'Sans-serif'
      }
    }>/>
  </LayersDirective>
</MapsComponent></div>;
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { default_data } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MapsTooltip, Inject }
  from '@syncfusion/ej2-react-maps';
export function App() {
  return(<div><MapsComponent >
    <Inject services={ [MapsTooltip]} />
    <LayersDirective>
      <LayerDirective shapeData={world_map}
dataSource={default_data} shapeDataPath='continent'
shapePropertyPath='continent'
tooltipSettings={{
  visible: true,
  valuePath: 'continent',
  format: '${continent}',
  fill: '#D0D0D0',
  textStyle: {
    color: 'green',
    fontFamily: 'Times New Roman',
    fontStyle: 'Sans-serif'
  }
}} />
    </LayersDirective>
  </MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Tooltip template

The HTML element can be rendered in the tooltip of the Maps using the [template](#) property of the [tooltipSettings](#).

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { default_data } from 'data.ts';

```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MapsTooltip, Inject
} from '@syncfusion/ej2-react-maps';
export function App() {
    return(<div><MapsComponent >
        <Inject services=[MapsTooltip] />
        <LayersDirective>
            <LayerDirective shapeData={world_map}
dataSource={default_data} shapeDataPath='continent'
shapePropertyPath='continent'
tooltipSettings={{
    visible: true,
    valuePath: 'continent',
    template: '<div style="width:60px; text-align:center;
background-color: white; border: 2px solid black; padding-bottom:
10px;padding-top: 10px;padding-left: 10px;padding-right:
10px;"><span>${continent}</span></div>',
    textStyle: {
        color: 'black'
    }
    }} />
        </LayersDirective>
    </MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { default_data } from 'data.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, MapsTooltip, Inject
} from '@syncfusion/ej2-react-maps';
export function App() {
    return(<div><MapsComponent >
        <Inject services=[MapsTooltip] />
        <LayersDirective>
            <LayerDirective shapeData={world_map}
dataSource={default_data} shapeDataPath='continent'
shapePropertyPath='continent'
tooltipSettings={{
    visible: true,
    valuePath: 'continent',
    template: '<div style="width:60px; text-align:center;
background-color: white; border: 2px solid black; padding-bottom:
10px;padding-top: 10px;padding-left: 10px;padding-right:
10px;"><span>${continent}</span></div>',
    textStyle: {
        color: 'black'
    }
    }} />
        </LayersDirective>
    </MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

```

        </LayersDirective>
        </MapsComponent></div>;
    }
    const root = ReactDOM.createRoot(document.getElementById('container'));
    root.render(<App />);
    {% endraw %}

```

Print in React Maps component

Print

The rendered Maps can be printed directly from the browser by calling the [print](#) method. To use the print functionality, the **Print** module must be injected into the Maps using **Inject services={[]}** tag and set the [allowPrint](#) property to **true**.

`ts

```
<MapsComponent id="maps">
```

```
<Inject services={[Print]}/>
```

```
<MapsComponent>
```

,

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, Inject, Print,
DataLabel } from '@syncfusion/ej2-react-maps';
export function App() {
    var mapsInstance;
    function clickHandler() {
        mapsInstance.print();
    }
    return (<div>
        <ButtonComponent onClick= { clickHandler}>print</ButtonComponent>
        <MapsComponent allowPrint={true} ref={g => mapsInstance = g}>
            <Inject services={[DataLabel, Print]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map}
                    dataLabelSettings={ {
                        visible: true,
                        labelPath: 'name',
                        smartLabelMode: 'Trim'
                    } }>
            </LayerDirective>
        </LayersDirective>
    </MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, Inject, Print,
DataLabel } from '@syncfusion/ej2-react-maps';
export function App() {
  let mapsInstance : MapsComponent;
  function clickHandler(){
    mapsInstance.print();
  }
  return (<div>
    <ButtonComponent onClick= { clickHandler}>print</ButtonComponent>
    <MapsComponent allowPrint={true} ref={g => mapsInstance = g}>
      <Inject services={[DataLabel, Print]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          dataLabelSettings={ {
            visible: true,
            labelPath: 'name',
            smartLabelMode: 'Trim'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Export

Image Export

To use the image export functionality in Maps, **ImageExport** module must be injected into the Maps using **Inject services={[ImageExport]}** tag and set the [allowImageExport](#) property to **true**.

```
`ts
```

```
<MapsComponent id="maps">
<Inject services={[ImageExport]}/>
<MapsComponent>
`
```

The rendered Maps can be exported as an image using the [export](#) method. This method requires two parameters: image type and file name. The Maps can be exported as an image in the following formats.

- JPEG
- PNG
- SVG

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, DataLabel, Inject,
ImageExport } from '@syncfusion/ej2-react-maps';
export function App() {
  var mapsInstance;
  function clickHandler() {
    mapsInstance.export('PNG', 'Maps');
  }
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <MapsComponent allowImageExport={true} ref={g => mapsInstance =
g}>
      <Inject services=[DataLabel, ImageExport] />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          dataLabelSettings={ {
            visible: true,
            labelPath: 'name',
            smartLabelMode: 'Trim'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, DataLabel, Inject,
ImageExport } from '@syncfusion/ej2-react-maps';
export function App() {
  let mapsInstance: MapsComponent;
  function clickHandler() {
    mapsInstance.export('PNG', 'Maps');
  }
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <MapsComponent allowImageExport={true} ref={g => mapsInstance =
g}>
      <Inject services=[DataLabel, ImageExport] />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          dataLabelSettings={ {

```

```

        visible: true,
        labelPath: 'name',
        smartLabelMode: 'Trim'
      } }>
    </LayerDirective>
  </LayersDirective>
</MapsComponent></div>;
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Exporting Maps as base 64 string of the file

The image can be exported as base64 string for the JPEG and PNG formats. The rendered Maps 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.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, DataLabel, Inject,
ImageExport } from '@syncfusion/ej2-react-maps';
export function App() {
  var mapsInstance;
  function clickHandler() {
    mapsInstance.export('PNG', 'Maps', null, false).then((data)=>{
      document.getElementById('data').innerHTML = data;
    })
  }
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <MapsComponent allowImageExport={true} ref={g => mapsInstance =
g}>
      <Inject services={[DataLabel, ImageExport]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          dataLabelSettings={ {
            visible: true,
            labelPath: 'name',
            smartLabelMode: 'Trim'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent><div id="data"></div></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, DataLabel, Inject,
ImageExport } from '@syncfusion/ej2-react-maps';
export function App() {
  let mapsInstance: MapsComponent;
  function clickHandler() {
    mapsInstance.export('PNG', 'Maps', null, false).then((data) => {
      document.getElementById('data').innerHTML = data;
    })
  }
  return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <MapsComponent allowImageExport={true} ref={g => mapsInstance =
g}>
      <Inject services=[[DataLabel, ImageExport]] />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          dataLabelSettings={ {
            visible: true,
            labelPath: 'name',
            smartLabelMode: 'Trim'
          } }>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent><div id="data"></div></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

PDF Export

To use the PDF export functionality, **PdfExport** module must be injected into the Maps using **Inject services=[[PdfExport]]** method and set the [allowPdfExport](#) property to **true**.

`ts

```
<MapsComponent id="maps">
<Inject services=[[PdfExport]]/>
<MapsComponent>
`
```

The rendered Maps 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.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
```

```

import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, DataLabel, Inject, PdfExport } from '@syncfusion/ej2-react-maps';
export function App() {
    var mapsInstance;
    function clickHandler() {
        mapsInstance.export('PDF', 'Maps');
    }
    return (<div>
        <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
        <MapsComponent allowPdfExport={true} ref={g => mapsInstance = g}>
            <Inject services=[[DataLabel, PdfExport]] />
            <LayersDirective>
                <LayerDirective shapeData={world_map}
                    dataLabelSettings={ {
                        visible: true,
                        labelPath: 'name',
                        smartLabelMode: 'Trim'
                    } }>
            </LayerDirective>
        </LayersDirective>
    </MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, DataLabel, Inject, PdfExport } from '@syncfusion/ej2-react-maps';
export function App() {
    let mapsInstance : MapsComponent;
    function clickHandler() {
        mapsInstance.export('PDF', 'Maps');
    }
    return (<div>
        <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
        <MapsComponent allowPdfExport={true} ref={g => mapsInstance = g}>
            <Inject services=[[DataLabel, PdfExport]] />
            <LayersDirective>
                <LayerDirective shapeData={world_map}
                    dataLabelSettings={ {
                        visible: true,
                        labelPath: 'name',
                        smartLabelMode: 'Trim'
                    } }>
            </LayerDirective>
        </LayersDirective>
    </MapsComponent></div>);

```



```

}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

The exporting of the Maps as base64 string is not supported for the PDF export.

Export the tile Maps

The rendered Maps with providers such as OSM, Bing and Google static Maps can be exported using the [export](#) method. It supports the following export formats.

- JPEG
- PNG
- PDF

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, Inject, ImageExport } from '@syncfusion/ej2-react-maps';
export function App() {
    var mapsInstance;
    function clickHandler() {
        mapsInstance.export('PNG', 'Maps');
    }
    return (<div>
        <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
        <MapsComponent allowImageExport={true}
            ref={g => mapsInstance = g}
            titleSettings={ { text: 'OSM' } }>
            <Inject services={[ImageExport]} />
            <LayersDirective>
                <LayerDirective
urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png">
                </LayerDirective>
            </LayersDirective>
        </MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MapsComponent, LayersDirective, LayerDirective, Inject, ImageExport } from '@syncfusion/ej2-react-maps';
export function App() {

```

```

let mapsInstance : MapsComponent;
function clickHandler() {
    mapsInstance.export('PNG', 'Maps');
}
return (<div>
    <ButtonComponent onClick= { clickHandler}>Export</ButtonComponent>
    <MapsComponent allowImageExport={true}
                    ref={g => mapsInstance = g}
                    titleSettings={ { text: 'OSM' } }>
        <Inject services={[ImageExport]} />
        <LayersDirective>
            <LayerDirective
urlTemplate="https://tile.openstreetmap.org/level/tileX/tileY.png">
            </LayerDirective>
        </LayersDirective>
    </MapsComponent></div>);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

State persistence in React Maps component

State Persistence

State persistence allows the Maps to retain the current model value in the browser cookies for state maintenance. This action is handled through the [enablePersistence](#) property which is set to **false** by default. When this property is set to **true**, some of the Maps component model values are preserved even after the page is refreshed.

```
{% raw %}
```

```
`ts
```

```
import { world_map } from 'world-map.ts';
```

```
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
```

```
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from '@syncfusion/ej2-react-maps';
```

```
export function App() {
```

```
    return(
```

```
        <MapsComponent enablePersistence={true} zoomSettings={ { enable: true } }>
```

```
        <Inject services={[Zoom]} />
```

```
        <LayersDirective>
```

```
            <LayerDirective shapeData={world_map}>
```

```
            </LayerDirective>
```

```
        </LayersDirective>
```

```
    </MapsComponent>
```

```

);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
`

{% endraw %}

```

Accessibility in React Maps component

The Maps component follows commonly used accessibility guidelines and standards, such as [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#).

The accessibility compliance for the Maps component 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 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

To meet accessibility standards, the Maps component follows to the [WAI-ARIA](#) patterns. In the Maps component, the following ARIA attributes are used:

| Attributes | Purpose |

| --- | --- |

| **role=region** | It specifies the Maps areas that do not support interactive functions like selection and highlight. |

| **role=button** | It specifies the Maps areas where interactive functions such as selection and highlight are available. |

| **aria-label** | Provides an accessible name for Maps elements such as geometric map shapes, title, subtitle, legend title, legend item labels, data labels, and so on. To learn more, see the next topic. |

Screen reading in Maps

Accessibility in the Maps component ensures that all users, regardless of ability or disability, can use screen reading. The following Map elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |

| --- | --- |

| Shapes in the layer | Reads the names of the geographical shapes (such as countries, states, and regions) that appear on the Maps. |

| Title | Reads the title content in the Maps. |

| Subtitle | Reads the title below the main title content in the Maps. |

| Legend title | Reads the contents of the legend's title as specified in Maps. |

| Legend item label | Reads the label of a legend item in Maps. |

| Data label | Reads the label specified for the shapes in the Maps layer. |

| Annotation | Reads the content specified in the annotation. |

| Marker template | Reads the content provided in the marker template. |

| Tooltip template | Reads the content provided in the tooltip template. |

| Data label template | Reads the content provided in the data label template. |

Keyboard Navigation

All the Maps actions can be controlled via keyboard keys. The applicable key combinations and their relative functionalities are listed below for the appropriate UI features available in the component.

Interaction Keys | Description

Tab | Moves to the next focusable element on the map, such as the legend or shape.

Shift + Tab | Moves to the previous focusable element on the map, such as the legend or shape.

+ | When zooming is enabled, zoom in operation can be performed.

- | When zooming is enabled, zoom out operation can be performed.

Left arrow | When zoomed in, the map can be scrolled to the left.

Right arrow | When zoomed in, the map can be scrolled to the right.

Up arrow | When zoomed in, the map can be scrolled upward.

Down arrow | When zoomed in, the map can be scrolled downward.

R | When zooming is enabled, reset operation can be performed.

Enter | The page can be navigated to the next and previous states in legend. Similarly, the selection can be made while navigating over the shape.

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Internationalization in React Maps component

Maps provide support for internationalization for the below elements.

- Data label
- Tooltip

For more information about number and date formatter, refer to the [internationalization](#) section.

<!-- markdownlint-disable MD036 -->

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, time values in Maps component using [format](#) property in the [Maps](#).

Numeric Format

The numeric formats such as currency, percentage and so on can be displayed in the tooltip and data labels of the Maps using the [format](#) property in the [Maps](#). In the below example, the tooltip is globalized to **German** culture. When setting the [useGroupingSeparator](#) property as **true**, the numeric text in the Maps separates with the comma separator.

INDEX.JSX

```
{% raw %}
```

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { world_map } from 'world-map.ts';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
  MapsTooltip,
  Inject,
} from '@syncfusion/ej2-react-maps';
import { setCulture, setCurrencyCode } from '@syncfusion/ej2-base';
setCulture('de');
setCurrencyCode('EUR');
export function App() {
  return (
    <MapsComponent format="c" useGroupingSeparator={true}>
      <Inject services={[MapsTooltip]} />
      <LayersDirective>
        <LayerDirective
          shapeData={world_map}
          shapeDataPath="Country"
          shapePropertyPath="name"
          dataSource=[
            {
              Country: 'China',
              Membership: 'Permanent',
              population: '38332521'
            },
            {
              Country: 'France',
              Membership: 'Permanent',
              population: '19651127'
            },
            {
              Country: 'Russia',
              Membership: 'Permanent',
              population: '3090416'
            },
            {
              Country: 'Kazakhstan',
              Membership: 'Non-Permanent',
              population: '1232521'
            },
            {
              Country: 'Poland',
              Membership: 'Non-Permanent',
              population: '90332521'
            },
            {
              Country: 'Sweden',
              Membership: 'Non-Permanent',
              population: '383521'
            }
          ]
        >
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
  )
  shapeSettings={{
    colorValuePath: 'Membership',
    colorMapping: [

```

```

        { value: 'Permanent', color: '#D84444' },
        { value: 'Non-Permanent', color: '#316DB5' }
      ]
    })
    tooltipSettings={{
      visible: true,
      valuePath: 'population'
    }}
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { world_map } from 'world-map.ts';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
  MapsTooltip,
  Inject,
} from '@syncfusion/ej2-react-maps';
import { setCulture, setCurrencyCode } from '@syncfusion/ej2-base';
setCulture('de');
setCurrencyCode('EUR');
export function App() {
  return (
    <MapsComponent format="c" useGroupingSeparator={true}>
      <Inject services={[MapsTooltip]} />
      <LayersDirective>
        <LayerDirective
          shapeData={world_map}
          shapeDataPath="Country"
          shapePropertyPath="name"
          dataSource=[
            {
              Country: 'China',
              Membership: 'Permanent',
              population: '38332521'
            },
            {
              Country: 'France',
              Membership: 'Permanent',
              population: '19651127'
            },
            {
              Country: 'Russia',
              Membership: 'Permanent',

```

```
        population: '3090416'
      },
      {
        Country: 'Kazakhstan',
        Membership: 'Non-Permanent',
        population: '1232521'
      },
      {
        Country: 'Poland',
        Membership: 'Non-Permanent',
        population: '90332521'
      },
      {
        Country: 'Sweden',
        Membership: 'Non-Permanent',
        population: '383521'
      }
    ]
  },
  shapeSettings: {
    colorValuePath: 'Membership',
    colorMapping: [
      { value: 'Permanent', color: '#D84444' },
      { value: 'Non-Permanent', color: '#316DB5' }
    ]
  },
  tooltipSettings: {
    visible: true,
    valuePath: 'population'
  }
}
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Localization in React Maps component

The localization library allows localizing the default text content of the Maps component. The Maps component has the static text of some features such as tooltip of zoom toolbar, and that can be changed to any other culture(Arabic, Deutsch, French, etc) by defining the locale value and translation object.

<!-- markdownlint-disable MD033 -->

Locale key words	Text to display
Zoom	Zoom
ZoomIn	Zoom In
ZoomOut	Zoom Out
Reset	Reset

Pan	Pan
-----	-----

To load translation object in the application, use `load` function of **L10n** class. For more information about localization, refer [here](#).

INDEX.JSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from
'@syncfusion/ej2-react-maps';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
  'ar-AR': {
    'maps': {
      ZoomIn: 'تكبير',
      ZoomOut: 'تصغير',
      Zoom: 'زوم',
      Pan: 'مقلاة',
      Reset: 'إعادة تعيين'
    },
  },
});
export function App() {
  return (
    <MapsComponent locale="ar-AR" zoomSettings={ { enable: true }
  >
      <Inject services={[Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}
          shapeDataPath='Country'
          shapePropertyPath='name'
          dataSource={uncountries}>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
}{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { world_map } from 'world-map.ts';
import { uncountries } from 'data.ts'
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective, Zoom, Inject } from
'@syncfusion/ej2-react-maps';
import { L10n } from '@syncfusion/ej2-base';
```

```

L10n.load({
  'ar-AR': {
    'maps': {
      ZoomIn: 'تكبير',
      ZoomOut: 'تصغير',
      Zoom: 'زوم',
      Pan: 'مقلاة',
      Reset: 'إعادة تعيين'
    }
  }
});
export function App() {
  return (
    <MapsComponent locale="ar-AR" zoomSettings={ { enable: true } }>
      <Inject services={[Zoom]}>
        <LayersDirective>
          <LayerDirective shapeData={world_map}
            shapeDataPath='Country'
            shapePropertyPath='name'
            dataSource={uncountries}>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Methods in React Maps component

Methods

This section explains the methods used in the Maps component.

[getMinMaxLatitudeLongitude](#)

The `getMinMaxLatitudeLongitude` method returns the minimum and maximum latitude and longitude values of the Maps visible area. This method returns a [IMinMaxLatitudeLongitude](#) object that contains the Maps minimum and maximum latitude and longitude coordinates.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import {
  MapsComponent,
  LayersDirective,
  LayerDirective,
  MarkersDirective,
  MarkerDirective,
  Inject,
  Marker,
  Zoom,

```

```

} from '@syncfusion/ej2-react-maps';
export function App() {
  var mapsInstance;
  function formatKey(key) {
    if (key === 'minLatitude') {
      return 'Minimum Latitude';
    } else if (key === 'maxLatitude') {
      return 'Maximum Latitude';
    } else if (key === 'minLongitude') {
      return 'Minimum Longitude';
    } else if (key === 'maxLongitude') {
      return 'Maximum Longitude';
    }
  }
  function getMinMaxValues() {
    var mapBoundCoordinates;
    mapBoundCoordinates = mapsInstance.getMinMaxLatitudeLongitude();
    const displayDiv = document.getElementById('coordinatesDisplay');
    displayDiv.innerHTML = '';
    if (mapBoundCoordinates) {
      for (const key in mapBoundCoordinates) {
        if (Object.hasOwnProperty.call(mapBoundCoordinates, key)) {
          const p = document.createElement('p');
          const formattedKey = formatKey(key);
          p.textContent = `${formattedKey}: ${mapBoundCoordinates[key]}`;
          displayDiv.appendChild(p);
        }
      }
    } else {
      displayDiv.textContent = 'No coordinates available';
    }
  }
  return (
    <div>
      <ButtonComponent onClick={getMinMaxValues}>
        GetMinMaxLatitudeLongitude
      </ButtonComponent>
      <p id="coordinatesDisplay"></p>
      <MapsComponent
        ref={(g) => (mapsInstance = g)}
        zoomSettings={{ enable: true, zoomFactor: 7 }}
        centerPosition={{
          latitude: 21.815447,
          longitude: 80.1932,
        }}
      >
        <Inject services={[Marker, Zoom]} />
        <LayersDirective>
          <LayerDirective shapeData={world_map}>
            <MarkersDirective>
              <MarkerDirective
                visible={true}
                height={25}
                width={25}
                shape="Circle"
                animationDuration={1500}

```

```

                dataSource=[
                    {
                        latitude: 22.572646,
                        longitude: 88.363895,
                    },
                    {
                        latitude: 25.0700428,
                        longitude: 67.2847875,
                    },
                ]
            <MarkerDirective>
        </MarkersDirective>
    </LayerDirective>
</LayersDirective>
</MapsComponent>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import {
    MapsComponent,
    LayersDirective,
    LayerDirective,
    MarkersDirective,
    MarkerDirective,
    Inject,
    Marker,
    Zoom,
} from '@syncfusion/ej2-react-maps';
export function App() {
    let mapsInstance: MapsComponent;
    function formatKey(key) {
        if (key === 'minLatitude') {
            return 'Minimum Latitude';
        } else if (key === 'maxLatitude') {
            return 'Maximum Latitude';
        } else if (key === 'minLongitude') {
            return 'Minimum Longitude';
        } else if (key === 'maxLongitude') {
            return 'Maximum Longitude';
        }
    }
    function getMinMaxValues() {
        var mapBoundCoordinates;
        mapBoundCoordinates = mapsInstance.getMinMaxLatitudeLongitude();
        const displayDiv = document.getElementById('coordinatesDisplay');
    }
}

```

```

displayDiv.innerHTML = '';
if (mapBoundCoordinates) {
  for (const key in mapBoundCoordinates) {
    if (Object.hasOwnProperty.call(mapBoundCoordinates, key)) {
      const p = document.createElement('p');
      const formattedKey = formatKey(key);
      p.textContent = `${formattedKey}:
${mapBoundCoordinates[key]}`;
      displayDiv.appendChild(p);
    }
  }
} else {
  displayDiv.textContent = 'No coordinates available';
}
}
return (
  <div>
    <ButtonComponent onClick={getMinMaxValues}>
      GetMinMaxLatitudeLongitude
    </ButtonComponent>
    <p id="coordinatesDisplay"></p>
    <MapsComponent
      ref={(g) => (mapsInstance = g)}
      zoomSettings={{ enable: true, zoomFactor: 7 }}
      centerPosition={{
        latitude: 21.815447,
        longitude: 80.1932,
      }}
    >
      <Inject services={[Marker, Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          <MarkersDirective>
            <MarkerDirective
              visible={true}
              height={25}
              width={25}
              shape="Circle"
              animationDuration={1500}
              dataSource={[
                {
                  latitude: 22.572646,
                  longitude: 88.363895,
                },
                {
                  latitude: 25.0700428,
                  longitude: 67.2847875,
                },
              ]}
            ></MarkerDirective>
          </MarkersDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  </div>
);
}

```

```
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

Ej1 api migration in React Maps component

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

Size Customization

{% raw %}

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

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

| Height | Not Applicable | **Property:** *height*
<MapsComponent id='maps' height="150"></MapsComponent>,

document.getElementById('maps');|

| Width | Not Applicable | **Property:** *width*
<MapsComponent id='maps' width="150"></MapsComponent>,

document.getElementById('maps');|

Title and Subtitle Customization

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

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

| Title Text | Not Applicable | **Property:** *title.text*

<MapsComponent id='maps' ref={m => this.mapInstance = m} titleSettings = {{text:'Members of the UN Security Council'}}></MapsComponent>,

document.getElementById('maps');|

| Subtitle Text | Not Applicable | **Property:** *title.subtitle.text*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} titleSettings = {{ subtitleSettings:{ text:'In 2017' }}></MapsComponent>,

document.getElementById('maps');|

| Title Alignment | Not Applicable | **Property:** *title.alignment*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} titleSettings = {{text:'Members of the UN Security Council', alignment: 'Center'}}></MapsComponent>,

document.getElementById('maps');|

| Subtitle Alignment | Not Applicable | **Property:** *title.subtitle.alignment*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} titleSettings = {{ subtitleSettings:{ text:'In 2017', alignment: 'Center' }}></MapsComponent>,

document.getElementById('maps');|

<!-- markdownlint-disable MD033 -->

<!-- markdownlint-disable MD038 -->

Zooming Customization

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

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

| Enable | **Property:** *zoomSettings.enableZoom*

 var zoomSettings = { enableZoom:true};
<EJ.Map id="maps" zoomSettings = {zoomSettings} ></EJ.Map>
document.getElementById('maps'); | **Property:** *zoomSettings.enable*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} zoomSettings={{ enable: true }}>

```

</br> <Inject services={{[Zoom]}} /> </br>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Minimum Zoom | Property: zoomSettings.minValue<br><br> var zoomSettings = {
minValue:2};<br><EJ.Map id="maps" zoomSettings = {zoomSettings} >
</EJ.Map><br>document.getElementById('maps'); | Property: zoomSettings.minZoom<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} zoomSettings={{ minZoom: 2 }}>
<br> <Inject services={{[Zoom]}} /> </br>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Maximum Zoom | Property: zoomSettings.maxValue<br><br> var zoomSettings = {
maxValue:2};<br><EJ.Map id="maps" zoomSettings = {zoomSettings} >
</EJ.Map><br>document.getElementById('maps'); | Property: zoomSettings.maxZoom<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} zoomSettings={{ maxZoom: 2 }}>
<br> <Inject services={{[Zoom]}} /> </br>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Mouse Wheel Zoom | Property: zoomSettings.enableMouseWheelZoom<br><br> var zoomSettings
= { enableMouseWheelZoom:true};<br><EJ.Map id="maps" zoomSettings = {zoomSettings} >
</EJ.Map><br>document.getElementById('maps'); | Property:
zoomSettings.mouseWheelZoom<br><br> <MapsComponent id='maps' ref={m =>
this.mapInstance = m} zoomSettings={{ mouseWheelZoom: true }}> <br> <Inject
services={{[Zoom]}} /> </br> </MapsComponent>,<br><br>document.getElementById('maps');|

| Double Click Zoom | Not Applicable | Property: zoomSettings.doubleClickZoom<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} zoomSettings={{
doubleClickZoom: true }}> <br> <Inject services={{[Zoom]}} /> </br>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Pinch Zoom | Not Applicable | Property: zoomSettings.pinchZooming<br><br> <MapsComponent
id='maps' ref={m => this.mapInstance = m} zoomSettings={ pinchZooming: true }> <br> <Inject
services={{[Zoom]}} /> </br> </MapsComponent>,<br><br>document.getElementById('maps');|

| Single Click Zoom | Property: zoomSettings.enableZoomOnSelection<br><br> var zoomSettings = {
enableZoomOnSelection:true};<br><EJ.Map id="maps" zoomSettings = {zoomSettings} >
</EJ.Map><br>document.getElementById('maps'); | Property:
zoomSettings.zoomOnClick<br><br> <MapsComponent id='maps' ref={m => this.mapInstance =
m} zoomSettings={{ zoomOnClick: true }}><br> <Inject services={{[Zoom]}} />
<br></MapsComponent>,<br><br>document.getElementById('maps');|

| Zoom Factor | Property: zoomSettings.factor<br><br> var zoomSettings = {
factor:2};<br><EJ.Map id="maps" zoomSettings = {zoomSettings} >
</EJ.Map><br>document.getElementById('maps'); | Property: zoomSettings.zoomFactor<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} zoomSettings={{ zoomFactor: 2
}}> <br> <Inject services={{[Zoom]}} /> </br>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Toolbars | Not Applicable | Property: zoomSettings.toolbars<br><br> <MapsComponent
id='maps' ref={m => this.mapInstance = m} zoomSettings={{ toolbars: ['Zoom', 'ZoomIn',

```

```

'ZoomOut', 'Pan', 'Reset'] ]}> <br/> <Inject services={[Zoom]} /> <br/>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Toolbar Orientation | Not Applicable | Property: zoomSettings.toolbarOrientation<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} zoomSettings={{
toolbarOrientation: 'Horizontal' }}><br/> <Inject services={[Zoom]} /> <br/>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Toolbar Vertical Alignment | Not Applicable | Property: zoomSettings.verticalAlignment<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} zoomSettings={{
verticalAlignment: 'Center' }}><br/> <Inject services={[Zoom]} /> <br/>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Toolbar Horizontal Alignment | Not Applicable | Property:
zoomSettings.horizontalAlignment<br><br> <MapsComponent id='maps' ref={m =>
this.mapInstance = m} zoomSettings={{ horizontalAlignment: 'Center' }}><br/> <Inject
services={[Zoom]} /> <br/></MapsComponent>,<br><br>document.getElementById('maps');|

| Toolbar Highlight Color | Not Applicable | Property: zoomSettings.highlightColor<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} zoomSettings={{ highlightColor:
'#e61576' }}><br/> <Inject services={[Zoom]} /> <br/>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Toolbar Selection Color | Not Applicable | Property: zoomSettings.selectionColor<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} zoomSettings={{ selectionColor:
'#e61576' }}> <br/> <Inject services={[Zoom]} /> <br/>
</MapsComponent>,<br><br>document.getElementById('maps');|

| Toolbar Fill Color | Not Applicable | Property: zoomSettings.color<br><br> <MapsComponent
id='maps' ref={m => this.mapInstance = m} zoomSettings={{ color: '#e61576' }}><br/> <Inject
services={[Zoom]} /> <br/></MapsComponent>,<br><br>document.getElementById('maps');|

```

Layer Customization

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

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

```

| Type | Not Applicable | Property: layers.type<br><br> <MapsComponent
id='maps'><br><LayersDirective> <LayerDirective type="Layer"> </LayerDirective>
</LayersDirective><br></MapsComponent>,<br><br>document.getElementById('maps');|

| Layer Type | Property: layers.layerType<br><br> var layers = [{
layerType:'Geometry'}];<br><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br>document.getElementById('maps'); | Property:
layers.layerType<br><br><MapsComponent id='maps'><br><LayersDirective> <LayerDirective
layerType="Geometry"> </LayerDirective>
</LayersDirective><br></MapsComponent>,<br><br>document.getElementById('maps');|

| Visible | Not Applicable | Property: layers.visible<br><br> <MapsComponent
id='maps'><br><LayersDirective> <LayerDirective visible="true"> </LayerDirective>
</LayersDirective><br></MapsComponent>,<br><br>document.getElementById('maps');|

```



```

| Bing Map Type | Property: layers.bingMapType<br/><br/> var layers = [{
bingMapType:'Aerial'}];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property: layers.bingMapType<br/><br/>
<MapsComponent id='maps'><br/><LayersDirective> <LayerDirective bingMapType="Aerial">
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Bing Map Key | Property: layers.key<br/><br/> var layers = [{ key:''}];<br/><EJ.Map id="maps"
layers = {layers} > </EJ.Map><br/>document.getElementById('maps'); | Property:
layers.key<br/><br/><MapsComponent id='maps'><br/><LayersDirective> <LayerDirective
key=""> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| URL Template | Property: layers.urlTemplate<br/><br/> var layers = [{
urlTemplate:'http://a.tile.openstreetmap.org/level/tileX/tileY.png'}];<br/><EJ.Map id="maps"
layers = {layers} > </EJ.Map><br/>document.getElementById('maps'); | Property:
layers.urlTemplate<br/><br/> <MapsComponent id='maps'><br/><LayersDirective>
<LayerDirective urlTemplate='http://a.tile.openstreetmap.org/level/tileX/tileY.png'>
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Shape Data | Property: layers.shapeData<br/><br/>var layers = [{
shapeData:worldmap}];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property: layers.shapeData<br/><br/>
<MapsComponent id='maps'><br/><LayersDirective> <LayerDirective shapeData={worldmap}>
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Data Source | Property: layers.dataSource<br/><br/>var layers = [{
dataSource:populationdensity}];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property: layers.dataSource<br/><br/>
<MapsComponent id='maps'><br/><LayersDirective> <LayerDirective
dataSource={populationdensity}> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Query | Not Applicable | Property: layers.query<br/><br/> <MapsComponent
id='maps'><br/><LayersDirective> <LayerDirective query=""> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Shape Data Path | Property: layers.shapeDataPath<br/><br/> var layers = [{
shapeDataPath:"name"}];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property: layers.shapeDataPath<br/><br/>
<MapsComponent id='maps'><br/><LayersDirective> <LayerDirective shapeDataPath='name'>
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Shape Property Path | Property: layers.shapePropertyPath<br/><br/> var layers = [{
shapePropertyPath:"name"}];<br/><EJ.Map id="maps" layers = {layers} >

```

```

</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.shapePropertyPath<br/><br/> <MapsComponent id='maps'><br/><LayersDirective>
<LayerDirective shapePropertyPath='name'> </LayerDirective>
</LayersDirective><br/></MapsComponent>, <br/><br/>document.getElementById('maps'); |
| Layer Animation | Not Applicable | Property: layers.animationDuration<br/><br/>
<MapsComponent id='maps'><br/><LayersDirective> <LayerDirective
animationDuration={500}> </LayerDirective>
</LayersDirective><br/></MapsComponent>, <br/><br/>document.getElementById('maps'); |

```

Shape Customization

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

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

| Shape Fill | **Property:** layers.shapeSettings.fill

var layers = [{
shapeSettings:{fill:"#626171"}}];
<EJ.Map id="maps" layers = {layers} >
</EJ.Map>
document.getElementById('maps'); | **Property:** layers.shapeSettings.fill

<MapsComponent id='maps'>
<LayersDirective> <LayerDirective shapeSettings={{
fill:'#626171' }}> </LayerDirective>
</LayersDirective>
</MapsComponent>,

document.getElementById('maps'); |

| Shape Palette | **Property:** layers.shapeSettings.colorPalette

 var layers = [{
shapeSettings:{colorPalette:"customPalette", customPalette: ["#E51400", "#A4C400"]
}}];
<EJ.Map id="maps" layers = {layers} >
</EJ.Map>
document.getElementById('maps'); | **Property:**
layers.shapeSettings.palette

 <MapsComponent id='maps'>
<LayersDirective>
<LayerDirective palette=""> </LayerDirective>
</LayersDirective>
</MapsComponent>,

document.getElementById('maps'); |

| Shape Point Radius | Not Applicable | **Property:** layers.shapeSettings.circleRadius

<MapsComponent id='maps'>
<LayersDirective> <LayerDirective shapeSettings={{
circleRadius:'10' }}> </LayerDirective>
</LayersDirective>
</MapsComponent>,

document.getElementById('maps'); |

| Shape Color Value Path | **Property:** layers.shapeSettings.colorValuePath

 var layers = [{
shapeSettings:{colorValuePath:"Candidate"}}];
<EJ.Map id="maps" layers = {layers} >
</EJ.Map>
document.getElementById('maps'); | **Property:**
layers.shapeSettings.colorValuePath

 <MapsComponent
id='maps'>
<LayersDirective> <LayerDirective shapeSettings={{ colorValuePath:'color' }}>
</LayerDirective>
</LayersDirective>
</MapsComponent>,

document.getElementById('maps'); |

| Shape Value Path | **Property:** layers.shapeSettings.valuePath

var layers = [{
shapeSettings:{valuePath:"population"}}];
<EJ.Map id="maps" layers = {layers} >
</EJ.Map>
document.getElementById('maps'); | **Property:**
layers.shapeSettings.valuePath

 <MapsComponent id='maps'>
<LayersDirective>
<LayerDirective shapeSettings={{ valuePath:'population' }}> </LayerDirective>
</LayersDirective>
</MapsComponent>,

document.getElementById('maps'); |

| Shape DashArray | Not Applicable | **Property:** *layers.shapeSettings.dashArray*
`<MapsComponent id='maps'><LayersDirective> <LayerDirective shapeSettings={{ dashArray:'1,2' }}> </LayerDirective>
</LayersDirective></MapsComponent>`,
`document.getElementById('maps');`

| Shape Opacity | Not Applicable | **Property:** *layers.shapeSettings.opacity*
`<MapsComponent id='maps'><LayersDirective> <LayerDirective shapeSettings={{ opacity: 0.5 }}> </LayerDirective>
</LayersDirective></MapsComponent>`,
`document.getElementById('maps');`

| Range Color Mapping | **Property:** *layers.shapeSettings.colorMappings.rangeColorMapping*
`var layers = [{
 shapeSettings:{valuePath:"population", enableGradient: true,
 colorMappings:{rangeColorMapping:[{from:50000,to:10000,gradientColors:["red","green"]}]}}];
<EJ.Map id="maps" layers = {layers} > </EJ.Map>`
`document.getElementById('maps');`
Property: *layers.shapeSettings.colorMapping*
`<MapsComponent
id='maps'><LayersDirective> <LayerDirective shapeSettings={{ colorValuePath: 'density',
colorMapping:[{ from:0.00001, to:100, color:'rgb(153,174,214)' }] }>
</LayerDirective>
</LayersDirective></MapsComponent>`,
`document.getElementById('maps');`

| Equal Color Mapping | **Property:** *layers.shapeSettings.colorMappings.equalColorMapping*
`var layers = [{
 shapeSettings:{colorValuePath:"Candidate", enableGradient: true,
 colorMappings:{equalColorMapping:[{value:"Trump",color:"red"}]}}];
<EJ.Map id="maps" layers = {layers} > </EJ.Map>`
`document.getElementById('maps');`
Property: *layers.shapeSettings.colorMapping*
`<MapsComponent
id='maps'><LayersDirective> <LayerDirective shapeSettings={{ colorValuePath:
'Candidate', colorMapping:[{ value:'Trump', color:'#D84444' }] }> </LayerDirective>
</LayersDirective></MapsComponent>`,
`document.getElementById('maps');`

Marker Customization

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

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

| Marker Data Source | **Property:** *layers.markers*
`var markers = [{ latitude:37.0000,
longitude: -120.000, city:"california"}];
var layers = [{ markers:markers }];
<EJ.Map id="maps" layers = {layers} > </EJ.Map>`
`document.getElementById('maps');`
Property: *layers.markerSettings.dataSource*
`<MapsComponent id='maps'> <Inject
services=[[Marker]] /> <LayersDirective> <LayerDirective> <MarkersDirective>
<MarkerDirective dataSource={ topPopulation }> </MarkerDirective> </MarkersDirective>
</LayerDirective>
</LayersDirective></MapsComponent>`,
`document.getElementById('maps');`

| Marker Template | **Property:** *layers.markerTemplate*
`<div> //... </div>`
`var layers = [{ markerTemplate:'template' }];
<EJ.Map id="maps" layers = {layers} >
</EJ.Map>`
`document.getElementById('maps');`
Property:

```

layers.markerSettings.template<br/><br/> <MapsComponent id='maps'><br/><Inject
services=[[Marker]] /> <LayersDirective> <LayerDirective> </LayerDirective>
<MarkersDirective> <MarkerDirective template='<div id="marker1"></div>'>
</MarkerDirective> </MarkersDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Visible | Not Applicable | Property: layers.markerSettings.visible<br/><br/>
<MapsComponent id='maps'><br/> <Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective visible={ true }> </MarkerDirective>
</MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Fill | Not Applicable | Property: layers.markerSettings.fill<br/><br/> <MapsComponent
id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective> <LayerDirective>
<MarkersDirective> <MarkerDirective fill= 'white'> </MarkerDirective> </MarkersDirective>
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Height | Not Applicable | Property: layers.markerSettings.height<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective height= {20}> </MarkerDirective>
</MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Width | Not Applicable | Property: layers.markerSettings.width<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective width= {20}> </MarkerDirective>
</MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Shape | Not Applicable | Property: layers.markerSettings.shape<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective shape='Circle'> </MarkerDirective>
</MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker ImageURL | Not Applicable | Property: layers.markerSettings.imageUrl<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective
imageUrl='http://js.syncfusion.com/demos/web/Images/map/pin.png'> </MarkerDirective>
</MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Opacity | Not Applicable | Property: layers.markerSettings.opacity<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective opacity={ 0.5 }> </MarkerDirective>

```

```

</MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Legend Text | Not Applicable | Property: layers.markerSettings.legendText<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective legendText='China'>
</MarkerDirective> </MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Offset | Not Applicable | Property: layers.markerSettings.offset<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective offset={ 10 }> </MarkerDirective>
</MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Animation Duration | Not Applicable | Property:
layers.markerSettings.animationDuration<br/><br/> <MapsComponent id='maps'><br/><Inject
services=[[Marker]] /> <LayersDirective> <LayerDirective> <MarkersDirective>
<MarkerDirective animationDuration={ 500 }> </MarkerDirective> </MarkersDirective>
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Animation Delay | Not Applicable | Property:
layers.markerSettings.animationDelay<br/><br/> <MapsComponent id='maps'><br/><Inject
services=[[Marker]] /> <LayersDirective> <LayerDirective> <MarkersDirective>
<MarkerDirective animationDelay={ 100 }> </MarkerDirective> </MarkersDirective>
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker DashArray | Not Applicable | Property: layers.markerSettings.dashArray<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective dashArray={ 1,2 }> </MarkerDirective>
</MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Selection | Not Applicable | Property: layers.markerSettings.selectionSettings<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective selectionSettings={ enable : true, fill :
'#D2691E', opacity : 0.5,enableMultiSelect=false }> </MarkerDirective> </MarkersDirective>
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

| Marker Highlight | Not Applicable | Property: layers.markerSettings.highlightSettings<br/><br/>
<MapsComponent id='maps'><br/><Inject services=[[Marker]] /> <LayersDirective>
<LayerDirective> <MarkersDirective> <MarkerDirective highlightSettings={ enable : true, fill :
'#D2691E', opacity : 0.5 }> </MarkerDirective> </MarkersDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|

```

| Marker Tooltip | Not Applicable | **Property:** *layers.markerSettings.tooltipSettings*

 <MapsComponent id='maps'>
<Inject services=[[Marker]] /> <LayersDirective>
 <LayerDirective> <MarkersDirective> <MarkerDirective tooltipSettings={ visible : true, fill :
 '#363F4C', valuePath : "State", template = ' <div> //... </div>' }> </MarkerDirective>
 </MarkersDirective> </LayerDirective>
 </LayersDirective>
</MapsComponent>,

document.getElementById('maps');|

Bubble Customization

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

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

| Visible | **Property:** *layers.bubbleSettings.visible*

 var layers = [{ bubbleSettings:{
 showBubble: true } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps');| **Property:**
layers.bubbleSettings.visible

 <MapsComponent id='maps'>
 <Inject
 services=[[Bubble]] /> <LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective
 visible={ true }> </BubbleDirective> </BubblesDirective> </LayerDirective>
 </LayersDirective>
</MapsComponent>,

document.getElementById('maps');|

| ValuePath | **Property:** *layers.bubbleSettings.valuePath*

 var layers = [{ bubbleSettings:{
 valuePath: "population" } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps');| **Property:**
layers.bubbleSettings.valuePath

 <MapsComponent id='maps'>
 <Inject
 services=[[Bubble]] /> <LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective
 valuePath='value'> </BubbleDirective> </BubblesDirective> </LayerDirective>
 </LayersDirective>
</MapsComponent>,

document.getElementById('maps');|

| MinValue | **Property:** *layers.bubbleSettings.minValue*

 var layers = [{ bubbleSettings:{
 minValue: "20" } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps');| **Property:**
layers.bubbleSettings.minRadius

 <MapsComponent id='maps'>
 <Inject
 services=[[Bubble]] /> <LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective
 minRadius={ 10 }> </BubbleDirective> </BubblesDirective> </LayerDirective>
 </LayersDirective>
</MapsComponent>,

document.getElementById('maps');|

| MaxValue | **Property:** *layers.bubbleSettings.maxValue*

var layers = [{ bubbleSettings:{
 maxValue: "20" } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps');| **Property:**
layers.bubbleSettings.maxRadius

 <MapsComponent id='maps'>
 <Inject
 services=[[Bubble]] /> <LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective
 maxRadius={ 20 }> </BubbleDirective> </BubblesDirective> </LayerDirective>
 </LayersDirective>
</MapsComponent>,

document.getElementById('maps');|

| Bubble Type | Not Applicable | **Property:** *layers.bubbleSettings.bubbleType*

 <MapsComponent id='maps'>
 <Inject services=[[Bubble]] /> <LayersDirective>
 <LayerDirective> <BubblesDirective> <BubbleDirective bubbleType='Circle'> </BubbleDirective>

```

</BubblesDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|

| Color | Property: layers.bubbleSettings.color<br/><br/> var layers = [{ bubbleSettings:{ color:
"red" } }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property: layers.bubbleSettings.fill<br/><br/>
<MapsComponent id='maps'><br/> <Inject services=[[Bubble]] /> <LayersDirective>
<LayerDirective> <BubblesDirective> <BubbleDirective fill='red'> </BubbleDirective>
</BubblesDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|

| Opacity | Property: layers.bubbleSettings.bubbleOpacity<br/><br/> var layers = [{ bubbleSettings:{
opacity: "0.5" } }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.bubbleSettings.opacity<br/><br/><MapsComponent id='maps'><br/> <Inject
services=[[Bubble]] /> <LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective
opacity={ 0.5 }> </BubbleDirective> </BubblesDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|

| Color Value Path | Property: layers.bubbleSettings.colorValuePath<br/><br/> var layers = [{
bubbleSettings:{ colorValuePath: "Candidate" } }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.bubbleSettings.colorValuePath<br/><br/> <MapsComponent id='maps'><br/> <Inject
services=[[Bubble]] /> <LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective
colorValuePath='color'> </BubbleDirective> </BubblesDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|

| Enable Tooltip | Property: layers.bubbleSettings.showTooltip<br/><br/> var layers = [{
bubbleSettings:{ showTooltip: true } }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.bubbleSettings.tooltipSettings.visible<br/><br/> <MapsComponent id='maps'><br/> <Inject
services=[[Bubble]] /> <LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective
tooltipSettings={ visible:true }> </BubbleDirective> </BubblesDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|

| Tooltip Template | Property: layers.bubbleSettings.tooltipTemplate<br/><br/><div
id="template"><br/>// .. <br/></div><br/>var layers = [{ bubbleSettings:{ tooltipTemplate:
"template" } }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.bubbleSettings.tooltipSettings.template<br/><br/> <MapsComponent id='maps'><br/> <Inject
services=[[Bubble,MapsTooltip]] /> <LayersDirective> <LayerDirective> <BubblesDirective>
<BubbleDirective tooltipSettings={ template:'<div> //... </div>' }> </BubbleDirective>
</BubblesDirective> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|

| Bubble Selection | Not Applicable | Property: layers.bubbleSettings.selectionSettings<br/><br/>
<MapsComponent id='maps'><br/> <Inject services=[[Bubble]] /> <LayersDirective>

```



```

<LayerDirective> <BubblesDirective> <BubbleDirective selectionSettings={ enable : true, fill :
'#D2691E', opacity : 0.5,enableMultiSelect=false }> </BubbleDirective> </BubblesDirective>
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|
| Bubble Highlight | Not Applicable | Property: layers.bubbleSettings.highlightSettings<br><br>
<MapsComponent id='maps'><br/> <Inject services=[[Bubble,MapsTooltip]] />
<LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective highlightSettings={
enable : true, fill : '#D2691E', opacity : 0.5 }> </BubbleDirective> </BubblesDirective>
</LayerDirective>
</LayersDirective><br/></MapsComponent>,<br><br/>document.getElementById('maps');|
| Range Color Mapping | Property: layers.bubbleSettings.colorMappings.rangeColorMapping<br><br>
var layers = [{ bubbleSettings:{
colorMappings:{rangeColorMapping:[{from:50000,to:10000,gradientColors:["red","green"]}]}
}];<br><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br>document.getElementById('maps'); | Property:
layers.bubbleSettings.colorMapping<br><br> <MapsComponent id='maps'><br> <Inject
services=[[Bubble]] /> <LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective
colorMapping:[{ from:0.00001, to:100, color:'rgb(153,174,214)' }] > </BubbleDirective>
</BubblesDirective> </LayerDirective>
</LayersDirective><br></MapsComponent>,<br><br>document.getElementById('maps');|
| Equal Color Mapping | Property: layers.bubbleSettings.colorMappings.equalColorMapping<br><br>
var layers = [{ bubbleSettings:{
colorMappings:{equalColorMapping:[{value:"trump",color:"red"}]} }];<br><EJ.Map id="maps"
layers = {layers} > </EJ.Map><br>document.getElementById('maps'); | Property:
layers.bubbleSettings.colorMapping<br><br> <MapsComponent id='maps'><br> <Inject
services=[[Bubble]] /> <LayersDirective> <LayerDirective> <BubblesDirective> <BubbleDirective
colorMapping:[{ value:'Trump', color:'#D84444' }]> </BubbleDirective> </BubblesDirective>
</LayerDirective>
</LayersDirective><br></MapsComponent>,<br><br>document.getElementById('maps');|

```

DataLabel Customization

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

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

```

| Visible | Property: layers.labelSettings.showLabels<br><br> var layers = [{ labelSettings:{
showLabels: true } }];<br><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br>document.getElementById('maps'); | Property:
layers.dataLabelSettings.visible<br><br> <MapsComponent id='maps'><br> <Inject
services=[[DataLabel]] /> <LayersDirective> <LayerDirective dataLabelSettings={ visible: true }>
</LayerDirective>
</LayersDirective><br></MapsComponent>,<br><br>document.getElementById('maps');|
| Label Path | Property: layers.labelSettings.labelPath<br><br> var layers = [{ labelSettings:{
labelPath: "name" } }];<br><EJ.Map id="maps" layers = {layers} >

```



```

</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.dataLabelSettings.labelPath<br/><br/> <MapsComponent id='maps'><br/> <Inject
services=[[DataLabel]] /> <LayersDirective> <LayerDirective dataLabelSettings={ labelPath:
'name' }> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|
| Enable Smart Label | Property: layers.labelSettings.enableSmartLabel<br/><br/> var layers = [{
labelSettings:{ enableSmartLabel: true} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Not Applicable |
| Smart Label Size | Property: layers.labelSettings.smartLabelSize<br/><br/> var layers = [{
labelSettings:{ smartLabelSize: 10} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Not Applicable |
| Label Length | Property: layers.labelSettings.labelLength<br/><br/> var layers = [{ labelSettings:{
labelLength: 10} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Not Applicable |
| Opacity | Not Applicable | Property: layers.dataLabelSettings.opacity<br/><br/> <MapsComponent
id='maps'><br/> <Inject services=[[DataLabel]] /> <LayersDirective> <LayerDirective
dataLabelSettings={ opacity: { 0.5 } }> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|
| Smart Label Mode | Not Applicable | Property: layers.dataLabelSettings.smartLabelMode<br/><br/>
<MapsComponent id='maps'><br/> <Inject services=[[DataLabel]] /> <LayersDirective>
<LayerDirective dataLabelSettings={ smartLabelMode: 'Trim' }> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|
| InterSectAction | Not Applicable | Property: layers.dataLabelSettings.intersectionAction<br/><br/>
<MapsComponent id='maps'><br/> <Inject services=[[DataLabel]] /> <LayersDirective>
<LayerDirective dataLabelSettings={ intersectionAction: 'None' }> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|
| Template | Not Applicable | Property: layers.dataLabelSettings.template<br/><br/> <div
id="template"><br/>//...<br/></div><br/><MapsComponent id='maps'><br/> <Inject
services=[[DataLabel]] /> <LayersDirective> <LayerDirective dataLabelSettings={ template:
'<div> //... </div>' }> </LayerDirective>
</LayersDirective><br/></MapsComponent>,<br/><br/>document.getElementById('maps');|

```

Legend Customization

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

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

```

| Visible | Property: layers.legendSettings.showLegend<br/><br/> var layers = [{ legendSettings:{
showLegend: true} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property: legendSettings.visible<br/><br/>
<MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = { visible:true}>
<Inject services=[[Legend]] />
</MapsComponent>,<br/><br/>document.getElementById('maps');|

```

| Toggle Visibility | **Property:** *layers.legendSettings.toggleVisibility*

 var layers = [{
 legendSettings:{ toggleVisibility: true } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | **Property:**
legendSettings.toggleVisibility

 <MapsComponent id='maps' ref={m => this.mapInstance
 = m} legendSettings = { toggleVisibility:true}> <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps');|

| Legend Location X | **Property:** *layers.legendSettings.positionX*

var layers = [{
 legendSettings:{ location: {x:250} } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | **Property:** *legendSettings.location*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = { location:{ x:
 250}}> <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps');|

| Legend Location Y | **Property:** *layers.legendSettings.positionY*

 var layers = [{
 legendSettings:{ location: {y:350} } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | **Property:** *legendSettings.location*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = { location:{ y:
 350}}> <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps');|

| Legend Type | **Property:** *layers.legendSettings.type*

 var layers = [{ legendSettings:{
 type:"Layers" } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | **Property:** *legendSettings.type*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = { type:'Layers'}>
 <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps'); |

| Label Orientation | **Property:** *layers.legendSettings.labelOrientation*

 var layers = [{
 legendSettings:{ labelOrientation:"Vertical" } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | Not Applicable |

| Legend Title | **Property:** *layers.legendSettings.title*

 var layers = [{ legendSettings:{ title:
 "Union territories of India" } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | **Property:** *legendSettings.title*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = { title:'Union
 territories of India'}> <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps'); |

| Legend Mode | **Property:** *layers.legendSettings.mode*

 var layers = [{ legendSettings:{
 mode: "Default" } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | **Property:** *legendSettings.mode*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = {
 mode:'Default'}> <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps'); |

| Legend Position | **Property:** *layers.legendSettings.position*

 var layers = [{ legendSettings:{
 position: "TopLeft" } }];
<EJ.Map id="maps" layers = {layers} >

```

</EJ.Map><br/>document.getElementById('maps'); | Property: legendSettings.position<br/><br/>
<MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = {
position:'Top'}> <Inject services=[[Legend]] />
</MapsComponent>,<br/><br/>document.getElementById('maps'); |

| Legend DockOnMap | Property: layers.legendSettings.dockOnMap<br/><br/> var layers = [{
legendSettings:{ dockOnMap: true} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Not Applicable |

| Legend Alignment | Property: layers.legendSettings.dockPosition<br/><br/> var layers = [{
legendSettings:{ dockPosition: "Right"} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property:
legendSettings.alignment<br/><br/> <MapsComponent id='maps' ref={m => this.mapInstance =
m} legendSettings = { alignment:'Center'}> <Inject services=[[Legend]] />
</MapsComponent>,<br/><br/>document.getElementById('maps'); |

| Legend Left Label | Property: layers.legendSettings.leftLabel<br/><br/> var layers = [{
legendSettings:{ leftLabel: "1000M"} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Not Applicable |

| Legend Right Label | Property: layers.legendSettings.rightLabel<br/><br/> var layers = [{
legendSettings:{ rightLabel: "1000M"} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Not Applicable |

| Legend Shape | Property: layers.legendSettings.icon<br/><br/> var layers = [{ legendSettings:{ icon:
"Circle"} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property: legendSettings.shape<br/><br/>
<MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = {
shape:'Circle'}> <Inject services=[[Legend]] />
</MapsComponent>,<br/><br/>document.getElementById('maps'); |

| Legend Shape Height | Property: layers.legendSettings.iconHeight<br/><br/> var layers = [{
legendSettings:{ iconHeight: "20"} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property:
legendSettings.shapeHeight<br/><br/> <MapsComponent id='maps' ref={m => this.mapInstance =
m} legendSettings = { shapeHeight:10}> <Inject services=[[Legend]] />
</MapsComponent>,<br/><br/>document.getElementById('maps'); |

| Legend Shape Width | Property: layers.legendSettings.iconWidth<br/><br/> var layers = [{
legendSettings:{ iconWidth: "20"} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property:
legendSettings.shapeWidth<br/><br/> <MapsComponent id='maps' ref={m => this.mapInstance =
m} legendSettings = { shapeWidth:20}> <Inject services=[[Legend]] />
</MapsComponent>,<br/><br/>document.getElementById('maps'); |

| Height | Property: layers.legendSettings.height<br/><br/> var layers = [{ legendSettings:{ height:
"50"} }];<br/><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br/>document.getElementById('maps'); | Property: legendSettings.width<br/><br/>

```

```

<MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = { height:150}>
<Inject services=[[Legend]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Width | Property: layers.legendSettings.width<br><br> var layers = [{ legendSettings:{ width:
"50"} }];<br><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br>document.getElementById('maps'); | Property: legendSettings.width<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = { width:'150'}>
<Inject services=[[Legend]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Show Labels | Property: layers.legendSettings.showLabels<br><br> var layers = [{ legendSettings:{
showLabels: true} }];<br><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br>document.getElementById('maps'); | Not Applicable |

| Background | Not Applicable | Property: legendSettings.background<br><br> <MapsComponent
id='maps' ref={m => this.mapInstance = m} legendSettings = { background:'Transparent'}>
<Inject services=[[Legend]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Label Position | Not Applicable | Property: legendSettings.labelPosition<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = {
labelPosition:'After'}> <Inject services=[[Legend]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Label Display Mode | Not Applicable | Property:
legendSettings.labelDisplayMode<br><br><MapsComponent id='maps' ref={m =>
this.mapInstance = m} legendSettings = { labelDisplayMode:'Trim'}> <Inject services=[[Legend]]
/> </MapsComponent>,<br><br>document.getElementById('maps'); |

| Legend Orientation | Not Applicable | Property: legendSettings.orientation<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = {
legendOrientation:'Horizontal'}> <Inject services=[[Legend]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Legend Item Fill | Not Applicable | Property: legendSettings.fill<br><br> <MapsComponent
id='maps' ref={m => this.mapInstance = m} legendSettings = { fill:'red'}> <Inject
services=[[Legend]] /> </MapsComponent>,<br><br>document.getElementById('maps'); |

| Legend Shape Padding | Not Applicable | Property: legendSettings.shapePadding<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = {
shapePadding:20}> <Inject services=[[Legend]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Legend Shape Border Color | Not Applicable | Property: legendSettings.shapeBorder.color<br><br>
<MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = { shapeBorder: {
color:'green'} }> <Inject services=[[Legend]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |

```

| Legend Shape Border Width | Not Applicable | **Property:** *legendSettings.shapeBorder.width*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = { shapeBorder: {
 width:2}}> <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps'); |

| Inverter Pointer | Not Applicable | **Property:** *legendSettings.invertedPointer*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} legendSettings = {
 invertedPointer : true }> <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps'); |

| Item Text Style | Not Applicable | **Property:** *legendSettings.textStyle*

 <MapsComponent
 id='maps' ref={m => this.mapInstance = m} legendSettings = { textStyle:{fontWeight:'400'
 size:'14px'}}> <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps'); |

| Title Style | Not Applicable | **Property:** *legendSettings.titleStyle*

 <MapsComponent
 id='maps' ref={m => this.mapInstance = m} legendSettings = { titleStyle:{fontWeight:'400'
 size:'14px'}}> <Inject services=[[Legend]] />
 </MapsComponent>,

document.getElementById('maps'); |

Highlight And Selection Customization

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

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

| Highlight Fill | **Property:** *layers.shapeSettings.highlightColor*

 var layers = [{
 shapeSettings:{ highlightColor: "green" } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | **Property:** *fill*

 <MapsComponent
 id='maps'> <LayersDirective> <LayerDirective highlightSettings={ fill:'green'}> <LayerDirective>
 </LayersDirective> <Inject services=[[Highlight]] />
 </MapsComponent>,

document.getElementById('maps'); |

| Enable Highlight | **Property:** *layers.enableMouseHover*

 var layers = [{ shapeSettings:{
 enableMouseHover: true } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | **Property:** *enable*

 <MapsComponent id='maps'> <LayersDirective> <LayerDirective highlightSettings={
 enable:'true'}> <LayerDirective> </LayersDirective> <Inject services=[[Highlight]] />
 </MapsComponent>,

document.getElementById('maps'); |

| Highlight Border Color | **Property:** *layers.shapeSettings.highlightStroke*

 var layers = [{
 shapeSettings:{ highlightStroke: "red" } }];
<EJ.Map id="maps" layers = {layers} >
 </EJ.Map>
document.getElementById('maps'); | **Property:**
layers.highlightSettings.border.color

 <MapsComponent id='maps'> <LayersDirective>
 <LayerDirective highlightSettings={ border:{color:'red'} }> <LayerDirective> </LayersDirective>
 <Inject services=[[Highlight]] />
 </MapsComponent>,

document.getElementById('maps'); |

| Highlight Border Width | **Property:** *layers.shapeSettings.highlightBorderWidth*

 var layers =
 [{ shapeSettings:{ highlightBorderWidth: "2" } }];
<EJ.Map id="maps" layers = {layers} >

```

</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.highlightSettings.border.width<br/><br/> <MapsComponent id='maps'> <LayersDirective>
<LayerDirective highlightSettings={ border:{width:2} }> <LayerDirective> </LayersDirective>
<Inject services={[Highlight]} />
</MapsComponent>,<br/><br/>document.getElementById('maps');|

| Highlight Opacity | Not Applicable | Property: layers.layers.highlightSettings.opacity<br/><br/>
<MapsComponent id='maps'> <LayersDirective> <LayerDirective highlightSettings={ opacity:0.5
}> <LayerDirective> </LayersDirective> <Inject services={[Highlight]} />
</MapsComponent>,<br/><br/>document.getElementById('maps');|

| Selection Fill | Property: layers.shapeSettings.selectionColor<br/><br/> var layers = [{
shapeSettings:{ selectionColor: "#D2691E"} }];<br/><EJ.Map id="maps" layers = {layers}>
</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.selectionSettings.fill<br/><br/> <MapsComponent id='maps'> <LayersDirective>
<LayerDirective selectionSettings={ fill:'#D2691E' }> <LayerDirective> </LayersDirective> <Inject
services={[Selection]} /> </MapsComponent>,<br/><br/>document.getElementById('maps');|

| Selection Enable | Property: layers.enableSelection<br/><br/> var layers = [{ shapeSettings:{
enableSelection: true} }];<br/><EJ.Map id="maps" layers = {layers}>
</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.selectionSettings.enable<br/><br/> <MapsComponent id='maps'> <LayersDirective>
<LayerDirective selectionSettings={ enable:'true' }> <LayerDirective> </LayersDirective> <Inject
services={[Selection]} /> </MapsComponent>,<br/><br/>document.getElementById('maps');|

| Selection Border Width | Property: layers.selectionSettings.selectionStrokeWidth<br/><br/> var
layers = [{ shapeSettings:{ selectionStrokeWidth: "2"} }];<br/><EJ.Map id="maps" layers =
{layers}> </EJ.Map><br/>document.getElementById('maps'); | Property:
layers.selectionSettings.border.width<br/><br/> <MapsComponent id='maps'> <LayersDirective>
<LayerDirective selectionSettings={ border:{width:2} }> <LayerDirective> </LayersDirective>
<Inject services={[Selection]} />
</MapsComponent>,<br/><br/>document.getElementById('maps');|

| Selection Border Color | Property: layers.selectionSettings.selectionStroke<br/><br/> var layers = [{
shapeSettings:{ selectionStroke: "red"} }];<br/><EJ.Map id="maps" layers = {layers}>
</EJ.Map><br/>document.getElementById('maps'); | Property:
layers.selectionSettings.border.color<br/><br/> <MapsComponent id='maps'> <LayersDirective>
<LayerDirective selectionSettings={ border:{color:'red'} }> <LayerDirective> </LayersDirective>
<Inject services={[Selection]} />
</MapsComponent>,<br/><br/>document.getElementById('maps');|

| Selection Opacity | Not Applicable | Property: layers.selectionSettings.opacity<br/><br/>
<MapsComponent id='maps'> <LayersDirective> <LayerDirective selectionSettings={ opacity:
0.5 }> <LayerDirective> </LayersDirective> <Inject services={[Highlight]} />
</MapsComponent>,<br/><br/>document.getElementById('maps');|

```

Navigation Line Customization

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

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

| Visible | Not Applicable | **Property:** *layers.navigationLineSettings.visible*


```
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLinesDirective>
<NavigationLineDirective visible= {true}> </NavigationLineDirective>
</NavigationLinesDirective> <LayerDirective> </LayersDirective> <Inject
services=[[NavigationLine]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |
```

| Width | Not Applicable | **Property:** *layers.navigationLineSettings.width*


```
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLineDirective width= {2}>
</NavigationLineDirective> </NavigationLinesDirective> <LayerDirective> </LayersDirective>
<Inject services=[[NavigationLine]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |
```

| Longitude | Not Applicable | **Property:** *layers.navigationLineSettings.longitude*


```
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLineDirective
longitude=[[-74.0060, -51.9253]]> </NavigationLineDirective> </NavigationLinesDirective>
<LayerDirective> </LayersDirective> <Inject services=[[NavigationLine]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |
```

| Latitude | Not Applicable | **Property:** *layers.navigationLineSettings.latitude*


```
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLineDirective
latitude=[[37.6276571, -14.2350]]> </NavigationLineDirective> </NavigationLinesDirective>
<LayerDirective> </LayersDirective> <Inject services=[[NavigationLine]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |
```

| DashArray | Not Applicable | **Property:** *layers.navigationLineSettings.dashArray*


```
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLineDirective
dashArray={1,2}> </NavigationLineDirective> </NavigationLinesDirective> <LayerDirective>
</LayersDirective> <Inject services=[[NavigationLine]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |
```

| Color | Not Applicable | **Property:** *layers.navigationLineSettings.color*


```
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLineDirective color= "black">
</NavigationLineDirective> </NavigationLinesDirective> <LayerDirective> </LayersDirective>
<Inject services=[[NavigationLine]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |
```

| Angle | Not Applicable | **Property:** *layers.navigationLineSettings.angle*


```
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLineDirective angle={-180}>
</NavigationLineDirective> </NavigationLinesDirective> <LayerDirective> </LayersDirective>
<Inject services=[[NavigationLine]] />
</MapsComponent>,<br><br>document.getElementById('maps'); |
```

| Arrow Position | Not Applicable | **Property:** *layers.navigationLineSettings.arrow.position*


```
var arrow = new MapsArrow {<br>Position="Start"<br>}<br><MapsComponent id='maps'>
<LayersDirective> <LayerDirective> <NavigationLineDirective arrow:{position:'Start'}>
</NavigationLineDirective> </NavigationLinesDirective> <LayerDirective> </LayersDirective>
```



```

<Inject services={[NavigationLine]} />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Show Arrow | Not Applicable | Property: layers.navigationLineSettings.arrow.showArrow<br><br>
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLineDirective
arrow:{showArrow:true}> </NavigationLineDirective> </NavigationLinesDirective>
<LayerDirective> </LayersDirective> <Inject services={[NavigationLine]} />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Arrow size | Not Applicable | Property: layers.navigationLineSettings.arrow.size<br><br>
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLineDirective
arrow:{size:'10px'}> </NavigationLineDirective> </NavigationLinesDirective> <LayerDirective>
</LayersDirective> <Inject services={[NavigationLine]} />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Arrow Color | Not Applicable | Property: layers.navigationLineSettings.arrow.color<br><br>
<MapsComponent id='maps'> <LayersDirective> <LayerDirective> <NavigationLineDirective
arrow:{color:'green'}> </NavigationLineDirective> </NavigationLinesDirective> <LayerDirective>
</LayersDirective> <Inject services={[NavigationLine]} />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Arrow Offset | Not Applicable | Property:
layers.navigationLineSettings.arrow.offset<br><br><MapsComponent id='maps'>
<LayersDirective> <LayerDirective> <NavigationLineDirective arrow:{offset:10}>
</NavigationLineDirective> </NavigationLinesDirective> </LayerDirective> </LayersDirective>
<Inject services={[NavigationLine]} />
</MapsComponent>,<br><br>document.getElementById('maps'); |

```

Tooltip Customization

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

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

```

| Tooltip Enable | Property: layers.showTooltip<br><br> var layers = [{ shapeSettings:{
showTooltip:true } }];<br><EJ.Map id="maps" layers = {layers} >
</EJ.Map><br>document.getElementById('maps'); | Property:
layers.tooltipSettings.visible<br><br><MapsComponent id='maps'> <LayersDirective
tooltipSettings={ visible:true}> <LayerDirective> </LayerDirective> </LayersDirective> <Inject
services={[MapsTooltip]} /> </MapsComponent>,<br><br>document.getElementById('maps'); |

| Tooltip Template | Property: layers.tooltipTemplate<br><br> <div id=template> <br> \\.
<br></div><br>var layers = [{ shapeSettings:{ tooltipTemplate:"template" } }];<br><EJ.Map
id="maps" layers = {layers} > </EJ.Map><br>document.getElementById('maps'); | Property:
layers.tooltipSettings.template<br><br> <MapsComponent id='maps'> <LayersDirective>
<LayerDirective tooltipSettings={ template:'<div> //... </div>'}> </LayerDirective>
</LayersDirective> <Inject services={[MapsTooltip]} />
</MapsComponent>,<br><br>document.getElementById('maps'); |

| Value Path | Not Applicable | Property: layers.tooltipSettings.valuePath<br><br>
<MapsComponent id='maps'> <LayersDirective tooltipSettings={ valuePath:'population'}>

```



```
<LayerDirective> </LayerDirective> </LayersDirective> <Inject services=[[MapsTooltip]] />
</MapsComponent>,<br><br>document.getElementById('maps');|
```

| Format | Not Applicable | **Property:** *layers.tooltipSettings.format*

<MapsComponent id='maps'> <LayersDirective tooltipSettings={ format:'\${State} \${Population}'}>
<LayerDirective> </LayerDirective> </LayersDirective> <Inject services=[[MapsTooltip]] />
</MapsComponent>,

document.getElementById('maps');|

| Border Color | Not Applicable | **Property:** *layers.tooltipSettings.border.color*

<MapsComponent id='maps'> <LayersDirective tooltipSettings={ border:{color:'red'}}>
<LayerDirective> </LayerDirective> </LayersDirective> <Inject services=[[MapsTooltip]] />
</MapsComponent>,

document.getElementById('maps');|

| Border Width | Not Applicable | **Property:** *layers.tooltipSettings.border.width*

<MapsComponent id='maps'> <LayersDirective tooltipSettings={ color:{width:2}}>
<LayerDirective> </LayerDirective> </LayersDirective> <Inject services=[[MapsTooltip]] />
</MapsComponent>,

document.getElementById('maps');|

Annotation Cutomization

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

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

| Content | Not Applicable | **Property:** *legendSettings.annotations.content*

 <div id="template"> //... </div>
<MapsComponent id='maps' ref={m => this.mapInstance = m} annotation =
[[content:'#template']]></MapsComponent>,

document.getElementById('maps'); |

| Location X | Not Applicable | **Property:** *legendSettings.annotations.x*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} annotation =
[[y:'250px']]></MapsComponent>,

document.getElementById('maps'); |

| Location Y | Not Applicable | **Property:** *legendSettings.annotations.y*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} annotation =
[[x:'250px']]></MapsComponent>,

document.getElementById('maps'); |

| Vertical Alignment | Not Applicable | **Property:**
legendSettings.annotations.verticalAlignment

 <MapsComponent id='maps' ref={m =>
this.mapInstance = m} annotation =
[[verticalAlignnebt:'Center']]></MapsComponent>,

document.getElementById('maps')
; |

| Horizontal Alignment | Not Applicable | **Property:**
legendSettings.annotations.horizontalAlignment

 <MapsComponent id='maps' ref={m =>
this.mapInstance = m} annotation =
[[horizontalAlignment:'Center']]></MapsComponent>,

document.getElementById('ma
ps'); |

| Zindex | Not Applicable | **Property:** *legendSettings.annotations.zIndex*

 <MapsComponent id='maps' ref={m => this.mapInstance = m} annotation =
[[zIndex:2]]></MapsComponent>,

document.getElementById('maps'); |

Maps Other Properties Customization

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

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

| Projection Type | Not Applicable | **Property:** *projectionType*

 <MapsComponent id='maps' projectionType='Mercator'></MapsComponent>;

 document.getElementById('maps'); |

| Background | **Property:** *background*

 <EJ.Map id="maps" background="red"></EJ.Map>
document.getElementById('maps'); | **Property:** *background*

 <MapsComponent id='maps' background='red'></MapsComponent>;

 document.getElementById('maps'); |

| Enable Group Separator | **Property:** *enableGroupSeparator*

 <EJ.Map id="maps" enableGroupSeparator={true} > | **Property:** *useGroupingSeparator*

 <MapsComponent id='maps' useGroupingSeparator={true}></MapsComponent>;

 document.getElementById('maps'); |

| Base Layer Index | **Property:** *baseMapIndex*

 <EJ.Map id="maps" baseMapIndex={0} > | **Property:** *baseLayerIndex*

 <MapsComponent id='maps' baseMapIndex={0}></MapsComponent>;

 document.getElementById('maps'); |

| locale | **Property:** *locale*

 <EJ.Map id="maps" locale="" > | Not Applicable |

| Responsive | **Property:** *isResponsive*

 <EJ.Map id="maps" isResponsive={true} > | Not Applicable |

| Enable Pan | **Property:** *enablePan*

 <EJ.Map id="maps" enablePan={true} > | Not Applicable |

| Enable Navigation | **Property:** *navigationControl.enableNavigation*

 var navigationControl = {enableNavigation:true};
 <EJ.Map id="maps" navigationControl={navigationControl} > | Not Applicable |

| Navigation Orientation | **Property:** *navigationControl.orientation*

 var navigationControl = {orientation:'Horizontal'};
 <EJ.Map id="maps" navigationControl={navigationControl} > | Not Applicable |

| Navigation Dock Position | **Property:** *navigationControl.dockPosition*

 var navigationControl = {dockPosition:'TopLeft'};
 <EJ.Map id="maps" navigationControl={navigationControl} > | Not Applicable |

| Navigation Absolute Position | **Property:** *navigationControl.absolutePosition*

 var navigationControl = {absolutePosition:{x:10 , y:10}};
 <EJ.Map id="maps" navigationControl={navigationControl} > | Not Applicable |

| Dragging Selection | **Property:** *draggingOnSelection*

 <EJ.Map id="maps" draggingOnSelection={true} > | Not Applicable |

| Resize | **Property:** *enableResize*

 <EJ.Map id="maps" enableResize={true} > | Not Applicable |

| Enable Animation | **Property:** *enableAnimation*

 <EJ.Map id="maps"
enableAnimation={true} > | Not Applicable |

| Enable Layer Animation | **Property:** *enableLayerChangeAnimation*

 <EJ.Map id="maps"
enableLayerAnimation={true} > | Not Applicable |

| Center Position | **Property:** *centerPosition*

 <EJ.Map id="maps"
centerPosition=[38.5000, -98] > | **Property:** *centerPosition*

 <MapsComponent
id='maps' centerPosition=[38.5000, -
98]></MapsComponent>,

 document.getElementById('maps'); |

Events

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

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

| Shape Selected | **Property:** *shapeSelected*

 <EJ.Map id="maps"
shapeSelected={shapeSelected}></EJ.Map>,
document.getElementById('maps')
functi
on shapeSelected(args) {} | **Property:** *shapeSelected*

 <MapsComponent id='maps'
shapeSelected={this.shapeSelected.bind(this)}></MapsComponent>,
document.getElemen
tById('maps');
 public shapeSelected(args: IShapeSelectedEventArgs): void {} || Marker
Selected | **Property:** *markerSelected*

 <EJ.Map id="maps"
markerSelected={markerSelected}></EJ.Map>,
document.getElementById('maps')
fun
ction markerSelected(args) {} | **Property:** *markerClick*

 <MapsComponent id='maps'
markerClick={this.markerClick.bind(this)}></MapsComponent>,
document.getElementById
('maps');
 public markerClick(args: IMarkerClickEventArgs): void {} || Marker Move |
Property: *markerEnter*

 <EJ.Map id="maps"
markerEnter={markerEnter}></EJ.Map>,
document.getElementById('maps')
function
markerEnter(args) {} | **Property:** *markerMouseMove*

 <MapsComponent id='maps'
markerMouseMove={this.markerMouseMove.bind(this)}></MapsComponent>,
document.
getElementById('maps');
 public markerMouseMove(args: IMarkerMouseMoveEventArgs):
void {} || Marker Leave | **Property:** *markerLeave*

 <EJ.Map id="maps"
markerLeave={markerLeave}></EJ.Map>,
document.getElementById('maps')
function
markerLeave(args) {} | Not Applicable |

| Legend Item Rendering | **Property:** *legendItemRendering*

 <EJ.Map id="maps"
legendItemRendering={legendItemRendering}></EJ.Map>,
document.getElementById('ma
ps')
function legendItemRendering(args) {} | Not Applicable |

| Display Text Rendering | **Property:** *displayTextRendering*

 <EJ.Map id="maps"
displayTextRendering={displayTextRendering}></EJ.Map>,
document.getElementById('map
s')
function displayTextRendering(args) {} | **Property:** *dataLabelRendering*

 <ejs-
maps id="maps" dataLabelRendering="dataLabelRendering"></ejs-maps>
 function
dataLabelRendering(args){} |

| Legend Item Click | **Property:** *legendItemClick*

 <EJ.Map id="maps"
legendItemClick={legendItemClick}></EJ.Map>,
document.getElementById('maps')
fun
ction legendItemClick(args) {} | Not Applicable |

| Bubble Rendering | **Property:** *bubbleRendering*

 <EJ.Map id="maps"
 bubbleRendering={bubbleRendering}></EJ.Map>,
document.getElementById('maps')
f
 unction bubbleRendering(args) {} | **Property:** *bubbleRendering*

 <MapsComponent
 id='maps'
 bubbleRendering={this.bubbleRendering.bind(this)}></MapsComponent>,
document.getEl
 ementById('maps');
 public bubbleRendering(args: IBubbleRenderingEventArgs): void {}

 function bubbleRendering(args){} |

| Shape Rendering | **Property:** *shapeRendering*

 <EJ.Map id="maps"
 shapeRendering={shapeRendering}></EJ.Map>,
document.getElementById('maps')
fun
 ction shapeRendering(args) {} | **Property:** *shapeRendering*

 <MapsComponent id='maps'
 shapeRendering={this.shapeRendering.bind(this)}></MapsComponent>,
document.getEle
 mentById('maps');
 public shapeRendering(args: IShapeRenderingEventArgs): void {} |

| Zoomed In | **Property:** *zoomedIn*

 <EJ.Map id="maps"
 zoomedIn={zoomedIn}></EJ.Map>,
document.getElementById('maps')
function
 zoomedIn(args) {} | Not Applicable |

| Render Completed | **Property:** *onRenderComplete*

 <EJ.Map id="maps"
 onRenderComplete={onRenderComplete}></EJ.Map>,
document.getElementById('maps')<
 br/>function onRenderComplete(args) {} | **Property:** *loaded*

 <MapsComponent
 id='maps'
 loaded={this.loaded.bind(this)}></MapsComponent>,
document.getElementById('maps');<
 br/> public loaded(args: ILoadedEventArgs): void {} |

| Panned | **Property:** *panned*

 <EJ.Map id="maps"
 panned={panned}></EJ.Map>,
document.getElementById('maps')
function
 panned(args) {} | Not Applicable |

| zoomed Out | **Property:** *zoomedOut*

 <EJ.Map id="maps"
 zoomedOut={zoomedOut}></EJ.Map>,
document.getElementById('maps')
function
 zoomedOut(args) {} | Not Applicable |

| Mouse Over | **Property:** *mouseover*

 <EJ.Map id="maps"
 mouseover={mouseover}></EJ.Map>,
document.getElementById('maps')
function
 mouseover(args) {} | Not Applicable |

| Mouse Leave | **Property:** *mouseleave*

 <EJ.Map id="maps"
 mouseleave={mouseleave}></EJ.Map>,
document.getElementById('maps')
function
 mouseleave(args) {} | Not Applicable |

| Click | **Property:** *click*

 <EJ.Map id="maps"
 click={click}></EJ.Map>,
document.getElementById('maps')
function click(args) {} |
Property: *click*

 <MapsComponent id='maps'
 click={this.click.bind(this)}></MapsComponent>,
document.getElementById('maps');

 public click(args: IClickEventArgs): void {} |

| Double Click | **Property:** *doubleClick*

 <EJ.Map id="maps"
 doubleClick={doubleClick}></EJ.Map>,
document.getElementById('maps')
function

doubleClick(args) {} | **Property:** *doubleClick*
 doubleClick={this.doubleClick.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public doubleClick(args: IDoubleClickEventArgs): void {} |

| Right Click | **Property:** *rightClick*
 rightClick={this.rightClick.bind(this)}></MapsComponent>,
document.getElementById('maps');
function rightClick(args) {} | **Property:** *rightClick*
 rightClick={this.rightClick.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public rightClick(args: IRightClickEventArgs): void {} |

| Initial Load | **Property:** *onLoad*
 onLoad={this.onLoad.bind(this)}></MapsComponent>,
document.getElementById('maps');
function onLoad(args) {} | **Property:** *load*
 load={this.load.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public load(args: ILoadEventArgs): void {} |

| Before Print | Not Applicable | **Property:** *beforePrint*
 beforePrint={this.beforePrint.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public beforePrint(args: IBeforePrintEventArgs): void {} |

| Resize | Not Applicable | **Property:** *resize*
 resize={this.resize.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public resize(args: IResizeEventArgs): void {} |

| Tooltip Render | Not Applicable | **Property:** *tooltipRender*
 tooltipRender={this.tooltipRender.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public tooltipRender(args: ITooltipRenderEventArgs): void {} |

| Item Selection | Not Applicable | **Property:** *itemSelection*
 itemSelection={this.itemSelection.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public itemSelection(args: IItemSelectionEventArgs): void {} |

| Item Highlight | Not Applicable | **Property:** *itemHighlight*
 itemHighlight={this.itemHighlight.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public itemHighlight(args: IItemHighlightEventArgs): void {} |

| Shape Highlight | Not Applicable | **Property:** *shapeHighlight*
 shapeHighlight={this.shapeHighlight.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public shapeHighlight(args: IShapeHighlightEventArgs): void {} |

| Layer Rendering | Not Applicable | **Property:** *layerRendering*
 layerRendering={this.layerRendering.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public layerRendering(args: ILayerRenderingEventArgs): void {} |

| Marker Rendering | Not Applicable | **Property:** *markerRendering*
 markerRendering={this.markerRendering.bind(this)}></MapsComponent>,
document.getElementById('maps');
 public markerRendering(args: IMarkerRenderingEventArgs): void {} |

```
| Bubble Mouse Move | Not Applicable | Property: bubbleMouseMove<br/><br/> <MapsComponent
id='maps'
bubbleMouseMove={this.bubbleMouseMove.bind(this)}></MapsComponent>,<br/>document.
getElementById('maps');<br/> public bubbleMouseMove(args: IBubbleMouseMoveEventArgs):
void {} |
```

```
| Bubble Mouse Move | Not Applicable | Property: annotationRendering<br/><br/>
<MapsComponent id='maps'
annotationRendering={this.annotationRendering.bind(this)}></MapsComponent>,<br/>docume
nt.getElementById('maps');<br/> public annotationRendering(args:
IAnnotationRenderingEventArgs): void {} |
```

```
| Animation Complete | Not Applicable | Property: animationComplete<br/><br/> <MapsComponent
id='maps'
animationComplete={this.animationComplete.bind(this)}></MapsComponent>,<br/>document.
getElementById('maps');<br/> public animationComplete(args: IAnimationCompleteEventArgs):
void {} |
```

```
{% endraw %}
```

How To

Annotation in React Maps component

Annotations are used to mark the specific area of interest in the Maps with texts, shapes, or images. Any number of annotations can be added to the Maps component.

Initialize the Maps component with annotation option, text content or ID of an HTML element or an HTML string can be specified to render a new element that needs to be displayed in the Maps by using the [content](#) property. To specify the content position with [x](#) and [y](#) properties as mentioned in the following example. In annotation, import the image in the specified Map area by using require function as mentioned in the below example.

```
,
const logo = require('./compass.png');
<img src={logo} height="75px" width="75px"/>
,
```

[app.tsx]

INDEX.JSX

```
{% raw %}
import { africa_continent } from 'africa-continent.ts';
import * as React from 'react';
import { MapsComponent, LayersDirective, LayerDirective, Inject, Annotations
} from '@syncfusion/ej2-react-maps';
import * as ReactDOM from 'react-dom';
const SAMPLE_CSS = `
    .control-fluid {
        padding: 0px !important;
    }
    #annotation {
```

```

    color: #DDDDDD;
    font-size: 12px;
    font-family: Roboto;
    background: #3E464C;
    margin: 20px;
    padding: 10px;
    -webkit-border-radius: 2px;
    -moz-border-radius: 2px;
    border-radius: 2px;
    width: 300px;
    -moz-box-shadow: 0px 2px 5px #666;
    -webkit-box-shadow: 0px 2px 5px #666;
    box-shadow: 0px 2px 5px #666;
  };
  export function App() {
    return (
      <div className="control-pane">
        <style>{SAMPLE_CSS}</style>
        <div className="control-section row">
          <MapsComponent
            annotations={[
              {
                content: '#maps-annotation',
                x: '0%',
                y: '70%'
              }
            ]}
          >
            <Inject services={[Annotations]} />
            <LayersDirective>
              <LayerDirective shapeData={africa_continent} />
            </LayersDirective>
          </MapsComponent>
        </div>
        <div id="maps-annotation" style={{ display: 'none' }}>
          <div id="annotation">
            <div
              style={{ marginLeft: '10px', fontSize: '13px', fontWeight: 500
is
              >
                <h5 style={{ marginLeft: '40px' }}>Facts about Africa</h5>
              </div>
              <hr />
              <div>
                <ul style={{ listStyleType: 'disc' }}>
                  <li>
                    Africa is the second largest and second most populated
                    continent in the world.
                  </li>
                  <li style={{ padding: '5px' }}>
                    Africa has 54 sovereign states and 10 non-sovereign
                    territories.
                  </li>
                  <li style={{ padding: '5px' }}>
                    Algeria is the largest country in Africa, where as Mayotte
                    the smallest.

```

```

        </li>
      </ul>
    </div>
  </div>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { africa_continent } from 'africa-continent.ts';
import * as React from 'react';
import { MapsComponent, LayersDirective, LayerDirective, Inject, Annotations }
  from '@syncfusion/ej2-react-maps';
import * as ReactDOM from 'react-dom';
const SAMPLE_CSS = `
  .control-fluid {
    padding: 0px !important;
  }
  #annotation {
    color: #DDDDDD;
    font-size: 12px;
    font-family: Roboto;
    background: #3E464C;
    margin: 20px;
    padding: 10px;
    -webkit-border-radius: 2px;
    -moz-border-radius: 2px;
    border-radius: 2px;
    width: 300px;
    -moz-box-shadow: 0px 2px 5px #666;
    -webkit-box-shadow: 0px 2px 5px #666;
    box-shadow: 0px 2px 5px #666;
  }`;
export function App() {
  return (
    <div className="control-pane">
      <style>{SAMPLE_CSS}</style>
      <div className="control-section row">
        <MapsComponent
          annotations={ [
            {
              content: '#maps-annotation',
              x: '0%',
              y: '70%'
            }
          ] }
        >
        <Inject services={[Annotations]} />
        <LayersDirective>
          <LayerDirective shapeData={africa_continent} />

```



```

        </LayersDirective>
      </MapsComponent>
    </div>
    <div id="maps-annotation" style={{ display: 'none' }}>
      <div id="annotation">
        <div
          style={{ marginLeft: '10px', fontSize: '13px', fontWeight: 500
}}
        >
          <h5 style={{ marginLeft: '40px' }}>Facts about Africa</h5>
        </div>
        <hr />
        <div>
          <ul style={{ listStyleType: 'disc' }}>
            <li>
              Africa is the second largest and second most populated
              continent in the world.
            </li>
            <li style={{ paddingTop: '5px' }}>
              Africa has 54 sovereign states and 10 non-sovereign
              territories.
            </li>
            <li style={{ paddingTop: '5px' }}>
              Algeria is the largest country in Africa, where as Mayotte
is
              the smallest.
            </li>
          </ul>
        </div>
      </div>
    </div>
  </div>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Custom path in React Maps component

Maps component can be customized as the desired layout using the custom path map feature. Here, the Maps component has been showcased with normal geometry type shapes to represent the bus seat selection layout.

```
{% raw %}
```

```
`ts
```

```
import { seatData } from 'seat.ts';
```

```
import * as React from 'react';
```

```
import {
```

```
  MapsComponent, LayersDirective, LayerDirective,
```

```
  Inject, Selection
```

```

} from '@syncfusion/ej2-react-maps';
import * as ReactDOM from 'react-dom';
export function App() {
  return (
    <div className='control-section row'>
      <div className='col-md-8'>
        <div style={{ width: 200, margin: 'auto', paddingBottom: 20 }}>
          <div id="sampletitle">Bus seat selection</div>
        </div>
        <div style={{ border: '3px solid darkgray', width: 200, display: 'block', margin: 'auto' }}>
          <MapsComponent height='400'>
            <Inject services={{[Selection]}} />
            <LayersDirective>
              <LayerDirective shapeData={seat} geometryType='Normal'>
            </LayerDirective>
            </LayersDirective>
          </MapsComponent>
        </div></div></div>;
    )
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  `
  {% endraw %}

```

Drilldown in React Maps component

By clicking a continent, all the countries available in that continent can be viewed using the drill-down feature. For example, the countries in the **Africa** continent have been showcased here. To showcase all the countries in **Africa** continent by clicking the [shapeSelected](#) event as mentioned in the following example.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { africa_continent } from 'africa-continent.ts';
import { defaultData } from 'data.ts';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { MapsComponent, LayersDirective, LayerDirective, Inject, Highlight,
Marker, MarkerDirective, MarkersDirective } from '@syncfusion/ej2-react-
maps';

```

```

export function App() {
  let mapInstance;
  function shapeSelected(args) {
    let shape = args.shapeData.continent;
    if (mapInstance.baseLayerIndex === 0) {
      if (shape === 'Africa') {
        mapInstance.baseLayerIndex = 1;
        mapInstance.refresh();
      }
    }
  }
  return (
    <MapsComponent height="400" ref={m => (mapInstance = m)}
      shapeSelected={shapeSelected}>
      <Inject services={[Highlight, Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map} layerType="Geometry"
shapeDataPath="continent" shapePropertyPath="continent"
dataSource={defaultData}
        shapeSettings={{
          colorValuePath: 'drillColor'
        }}>
        <MarkersDirective>
          <MarkerDirective visible={true}
            template='<div style="font-size:
12px;color:white;text-shadow: 0px 1px 1px black;font-weight:
500;width:50px">Africa</div>'
            dataSource=[
              { latitude: 10.97274101999902, longitude:
16.390625 }
            ]
            animationDuration={0}/>
          </MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
      <LayerDirective layerType="Geometry" shapeData={africa_continent}
        shapeSettings={{
          fill: '#80306A'
        }}
        highlightSettings={{
          enable: true,
          fill: '#80306A'
        }}
      </LayerDirective>
    </LayersDirective>
  </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import { africa_continent } from 'africa-continent.ts';

```

```

import { defaultData } from 'data.ts';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { MapsComponent, LayersDirective, LayerDirective, Inject, Highlight,
Marker, MarkerDirective, MarkersDirective, IShapeSelectedEventArgs } from
'@syncfusion/ej2-react-maps';
export function App() {
  let mapInstance : MapsComponent;
  function shapeSelected(args: IShapeSelectedEventArgs) {
    let shape = args.shapeData.continent;
    if (mapInstance.baseLayerIndex === 0) {
      if (shape === 'Africa') {
        mapInstance.baseLayerIndex = 1;
        mapInstance.refresh();
      }
    }
  }
  return (
    <MapsComponent height="400" ref={m => (mapInstance = m)}
      shapeSelected={shapeSelected}>
      <Inject services={[Highlight, Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map} layerType="Geometry"
shapeDataPath="continent" shapePropertyPath="continent"
dataSource={defaultData}
shapeSettings={{
  colorValuePath: 'drillColor'
}}>
        <MarkersDirective>
          <MarkerDirective visible={true}
            template='<div style="font-size:
12px;color:white;text-shadow: 0px 1px 1px black;font-weight:
500;width:50px">Africa</div>'
            dataSource=[
              { latitude: 10.97274101999902, longitude:
16.390625 }
            ]
            animationDuration={0}/>
          </MarkerDirective>
        </MarkersDirective>
      </LayerDirective>
      <LayerDirective layerType="Geometry" shapeData={africa_continent}
        shapeSettings={{
          fill: '#80306A'
        }}
        highlightSettings={{
          enable: true,
          fill: '#80306A'
        }}
      />
    </LayersDirective>
  </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Marker type in React Maps component

Add different types of markers

Different marker objects can be added to the Maps component using the marker settings. To update different marker settings in Maps, please follow the given steps:

Step 1:

Initialize the Maps component with marker settings. Here, a marker has been added with specified latitude and longitude of California by using the [dataSource](#) property. To customize the shape of the marker using the [shape](#) property and change the border color and width of the marker using the [border](#) property as mentioned in the following example.

INDEX.JSX

```
{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
  MarkerDirective, Inject, Marker
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}>
          <MarkersDirective>
            <MarkerDirective visible={true} shape='Circle' fill='white'
width={3} animationDuration={0}
              dataSource=[
                { latitude: 37.0000, longitude: -120.0000, city:
'California' }
              ]
              border={{ width: 2, color: '#333' }}>
            </MarkerDirective>
          </MarkersDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
```

```

    MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
    MarkerDirective, Inject, Marker
  } from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}>
          <MarkersDirective>
            <MarkerDirective visible={true} shape='Circle' fill='white'
width={3} animationDuration={0}
              dataSource=[
                { latitude: 37.0000, longitude: -120.0000, city:
'California' }
              ]
            </MarkerDirective>
          </MarkersDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Step 2:

Customize the above option for n number of markers as mentioned in the following example.

INDEX.JSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
  MarkerDirective, Inject, Marker
} from '@syncfusion/ej2-react-maps';
//tslint disable
export function App() {
  return (
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}>
          <MarkersDirective>
            <MarkerDirective visible={true} shape='Circle' fill='white'
width={3} animationDuration={0}
              dataSource=[
                { latitude: 37.0000, longitude: -120.0000, city:
'California' }
              ]
            </MarkerDirective>
          </MarkersDirective>
        </LayerDirective>
      </LayersDirective>
    </MapsComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

```

        { width: 2, color: '#333' }
      }>
    </MarkerDirective>
    <MarkerDirective visible={true} shape='Rectangle'
fill='yellow' width={15} height={4} animationDuration={0}
      dataSource=[
        { latitude: 40.7127, longitude: -74.0059, city: 'New York'
}
      ]
      border={
        { width: 2, color: '#333' }
      }>
    </MarkerDirective>
    <MarkerDirective visible={true} shape='Diamond' fill='white'
width={10} height={10} animationDuration={0}
      dataSource=[
        { latitude: 42, longitude: -93, city: 'Iowa' }
      ]
      border={
        { width: 2, color: 'blue' }
      }>
    </MarkerDirective>
    <MarkerDirective visible={true} shape='Balloon' fill='red'
width={10} height={15} animationDuration={0}
      dataSource=[
        { latitude: 36.499589049395055, longitude: -
103.042108197135548, city: 'New Mexico' }
      ]
      border={
        { width: 2, color: '#333' }
      }>
    </MarkerDirective>
    <MarkerDirective visible={true} shape='Triangle' fill='blue'
width={10} animationDuration={0}
      dataSource=[
        { latitude: 36.360624205142919, longitude: -
94.595916790727287, city: 'Oklahoma' }
      ]
      border={
        { width: 2, color: '#333' }
      }>
    </MarkerDirective>
  </MarkersDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { usa_map } from 'usa.ts';

```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  MapsComponent, LayersDirective, LayerDirective, MarkersDirective,
  MarkerDirective, Inject, Marker
} from '@syncfusion/ej2-react-maps';
//tslint disable
export function App() {
  return(
    <MapsComponent >
      <Inject services={[Marker]} />
      <LayersDirective>
        <LayerDirective shapeData={usa_map}>
          <MarkersDirective>
            <MarkerDirective visible={true} shape='Circle' fill='white'
width={3} animationDuration={0}
              dataSource=[
                { latitude: 37.0000, longitude: -120.0000, city:
'California' }
              ]
              border={
                { width: 2, color: '#333' }
              }
            </MarkerDirective>
            <MarkerDirective visible={true} shape='Rectangle'
fill='yellow' width={15} height={4} animationDuration={0}
              dataSource=[
                { latitude: 40.7127, longitude: -74.0059, city: 'New York'
}
              ]
              border={
                { width: 2, color: '#333' }
              }
            </MarkerDirective>
            <MarkerDirective visible={true} shape='Diamond' fill='white'
width={10} height={10} animationDuration={0}
              dataSource=[
                { latitude: 42, longitude: -93, city: 'Iowa' }
              ]
              border={
                { width: 2, color: 'blue' }
              }
            </MarkerDirective>
            <MarkerDirective visible={true} shape='Balloon' fill='red'
width={10} height={15} animationDuration={0}
              dataSource=[
                { latitude: 36.499589049395055, longitude: -
103.042108197135548, city: 'New Mexico' }
              ]
              border={
                { width: 2, color: '#333' }
              }
            </MarkerDirective>
            <MarkerDirective visible={true} shape='Triangle' fill='blue'
width={10} animationDuration={0}
              dataSource=[

```



```

        { latitude: 36.360624205142919, longitude: -
94.595916790727287, city: 'Oklahoma' }
      ]}
      border={
        { width: 2, color: '#333' }
      }>
    </MarkerDirective>
  </MarkersDirective>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

Multiple layer in React Maps component

Adding multiple layers in the Map

The multilayer support allows loading multiple shape files in a single container and enables Maps to display more information. The shape layer is the main layer of the Maps. Multiple layers can be added in a shape layer as **SubLayer** using the [type](#) property.

INDEX.JSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import { california } from 'california.ts';
import { texas } from 'texas.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings={ {
            fill: '#E5E5E5',
            border: {
              color: 'black',
              width: 0.1
            }
          } }>
        </LayerDirective>
        <LayerDirective shapeData={texas}
          type='SubLayer'
          shapeSettings= { {
            fill: 'rgba(141, 206, 255, 0.6)',
            border: {
              color: '#1a9cff',
              width: 0.25
            }
          } }>
      </LayersDirective>
    </MapsComponent >
  );
}

```

```

    </LayerDirective>
    <LayerDirective shapeData={california}
      type='SubLayer'
      shapeSettings= { {
        fill: 'rgba(141, 206, 255, 0.6)',
        border: {
          color: '#1a9cff',
          width: 0.25
        }
      } }>
  </LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { usa_map } from 'usa.ts';
import { california } from 'california.ts';
import { texas } from 'texas.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-react-maps';
export function App() {
  return(
    <MapsComponent >
      <LayersDirective>
        <LayerDirective shapeData={usa_map}
          shapeSettings={ {
            fill: '#E5E5E5',
            border: {
              color: 'black',
              width: 0.1
            }
          } }>
      </LayerDirective>
      <LayerDirective shapeData={texas}
        type='SubLayer'
        shapeSettings= { {
          fill: 'rgba(141, 206, 255, 0.6)',
          border: {
            color: '#1a9cff',
            width: 0.25
          }
        } }>
      </LayerDirective>
      <LayerDirective shapeData={california}
        type='SubLayer'
        shapeSettings= { {
          fill: 'rgba(141, 206, 255, 0.6)',

```

```

        border: {
            color: '#1a9cff',
            width: 0.25
        }
    } }>
</LayerDirective>
</LayersDirective>
</MapsComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);

{% endraw %}

```

Zooming in React Maps component

The center position zooming can be achieved by using the [centerPosition](#) and [zoomFactor](#) properties as mentioned in the following example. The center position is used to configure the zoom level of Maps, and the zoom factor is used to specify the center position where the Maps should be displayed.

INDEX.JSX

```

{% raw %}
import { world_map } from 'world-map.ts';
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    MapsComponent, LayersDirective, LayerDirective,
    Inject, Zoom
} from '@syncfusion/ej2-react-maps';
export function App() {
    return (
        <MapsComponent zoomSettings={{
            enable: false,
            zoomFactor: 13
        }} centerPosition={{
            latitude: 25.54244147012483,
            longitude: -89.62646484375
        }} >
            <Inject services={[Zoom]} />
            <LayersDirective>
                <LayerDirective shapeData={world_map}>
            </LayerDirective>
            </LayersDirective>
        </MapsComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('container'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { world_map } from 'world-map.ts';

```

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  MapsComponent, LayersDirective, LayerDirective,
  Inject, Zoom
} from '@syncfusion/ej2-react-maps';
export function App() {
  return (
    <MapsComponent zoomSettings={{
      enable: false,
      zoomFactor: 13
    }} centerPosition={{
      latitude: 25.54244147012483,
      longitude: -89.62646484375
    }} >
      <Inject services={[Zoom]} />
      <LayersDirective>
        <LayerDirective shapeData={world_map}>
          </LayerDirective>
        </LayersDirective>
      </MapsComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('container'));
  root.render(<App />);
  {% endraw %}
```

MaskedTextBox

Getting Started

The following section explains the required steps to build the MaskedTextBox component with its basic usage in step-by-step procedure.

Dependencies

The following list of dependencies are required to use the MaskedTextBox component in your application.

```
`javascript
|-- @syncfusion/ej2-react-inputs
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-react-base
`
```

Installation and configuration

You can use [Create-react-app](#) to setup the applications.

To install `create-react-app` run the following command.

```
`bash
npm install -g create-react-app
```

Start a new project using create-react-app command as follows

```
`bash
create-react-app quickstart --scripts-version=react-scripts-ts
cd quickstart
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry.

You can choose the component that you want to install. For this application, we are going to use MaskedTextBox component.

To install MaskedTextBox component, use the following command

```
`bash
npm install @syncfusion/ej2-react-inputs --save
```

Adding MaskedTextBox to the application

Now, you can start adding MaskedTextBox component to the application. We have added MaskedTextBox component in `src/App.tsx` file using following code.

[Class-component]

```
`ts
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes MaskedTextBox component
ReactDOM.render(<MaskedTextBoxComponent />,document.getElementById('maskedContainer'));
```

[Functional-component]

```
`ts
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes MaskedTextBox component
// sets mask format to the MaskedTextBox
function App() {
  return <MaskedTextBoxComponent />;
```

```
}  
ReactDOM.render(<App />, document.getElementById('maskedContainer'));  
`
```

Set the mask

You can set the mask to the MaskedTextBox to validate the user input by using the [mask](#) property. To know more about the usage of mask and configuration, refer to this [link](#).

The following example demonstrates the usage of mask element 0 that allows any single digit from 0 to 9.

[Class-component]

```
`ts  
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';  
import * as React from "react";  
import * as ReactDOM from "react-dom";  
// initializes MaskedTextBox component  
// sets mask format to the MaskedTextBox  
ReactDOM.render(<MaskedTextBoxComponent mask={'000-000-0000'}  
/>,document.getElementById('maskedContainer'));  
`
```

[Functional-component]

```
`ts  
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';  
import * as React from 'react';  
import * as ReactDOM from 'react-dom';  
// initializes MaskedTextBox component  
// sets mask format to the MaskedTextBox  
function App() {  
  return <MaskedTextBoxComponent mask={'000-000-0000'} />;  
}  
ReactDOM.render(<App />, document.getElementById('maskedContainer'));  
`
```

Adding CSS reference

Import the MaskedTextBox component's required CSS references as follows in `src/App.css`.

```
`css  
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

```
@import "../node_modules/@syncfusion/ej2-react-inputs/styles/material.css";
,
```

Note: If you want to refer the combined component styles, please make use of our [CRG](#) (Custom Resource Generator) in your application.

Run the application

Use the `npm run start` command to run the application in the browser.

```
npm run start
,
```

The following example shows the MaskedTextBox.

[Class-component]

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes MaskedTextBox component
// sets value to the MaskedTextBox
export default class App extends React.Component {
  render() {
    return (
      <div>
        <label className='label'>Enter the mobile number</label>
        <MaskedTextBoxComponent mask={'000-000-0000'} />
      </div>
    );
  }
}
;
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes MaskedTextBox component
// sets value to the MaskedTextBox
export default class App extends React.Component<{}, {}> {
  public render() {
    return (
      <div>
        <label className='label'>Enter the mobile number</label>
        <MaskedTextBoxComponent mask={'000-000-0000'} />
      </div>
    );
  }
}
;
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

[Functional-component]**INDEX.JSX**

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes MaskedTextBox component
// sets value to the MaskedTextBox
function App() {
    return (<div>
        <label className="label">Enter the mobile number</label>
        <MaskedTextBoxComponent mask={'000-000-0000'}/>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes MaskedTextBox component
// sets value to the MaskedTextBox
function App() {
    return (
        <div>
            <label className="label">Enter the mobile number</label>
            <MaskedTextBoxComponent mask={'000-000-0000'} />
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

See Also

- [How to perform custom validation using FormValidator](#)
- [How to customize the UI appearance of the control](#)
- [How to set cursor position while focus on the input textbox](#)
- [How to display numeric keypad when focus on mobile devices](#)

Mask configuration in React Maskedtextbox component

The mask is a combination of standard and custom mask elements, that validates the user input based on its behavior.

When the mask value is empty, the MaskedTextBox behaves as an input element with text type.

Standard mask elements

The following table shows the list of mask elements and its behavior based on [MSDN](#) standard.

The mask can be formed by combining any one or more of these mask elements.

Mask Element	Description
-----	-----

| 0 | Digit required. This element will accept any single digit from **0** to **9**. |

| 9 | Digit or space, optional. |

| # | Digit or space, optional, Plus(+) and minus(-) signs are allowed. |

| L | Letter required. It will accept letters **a-z** and **A-Z**. |

| ? | Letter or space, optional. |

| & | Requires a character. |

| C | Character or space, optional. |

| A | Alphanumeric (**A-Za-z0-9**) required. |

| a | Alphanumeric (**A-Za-z0-9**) or space, optional. |

| < | Shift down. Converts all characters to lower case. |

| > | Shift up. Converts all characters to upper case. |

| | | Disable a previous shift up or shift down. |

| \\\ | Escapes a mask character, turning it into a literal. |

| All other characters | Literals. All non-mask elements (literals) will appear as themselves within MaskedTextBox. |

The following example demonstrates the usage of standard mask elements.

[Class-component]

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// sets mask with the mask element '#' which accepts any single digit from
'0' to '9',
// space, + and - signs
ReactDOM.render(<MaskedTextBoxComponent mask={'#####'} placeholder={'Mask
##### (ex: 012+-)'} floatLabelType='Always'/>,
document.getElementById('mask1'));
// sets mask format with the mask element 'L' which allows only
alphabets('A-Z and a-z')
ReactDOM.render(<MaskedTextBoxComponent mask={'LLLLLL'} placeholder={'Mask
LLLLLL (ex: Sample)'} floatLabelType='Always'/>,
document.getElementById('mask2'));
// sets mask format with the mask element '&' which allows `alphabets`,
`numbers`
// and `special characters`
ReactDOM.render(<MaskedTextBoxComponent mask={'&&&&'} placeholder={'Mask
&&&& (ex: A12@#)'} floatLabelType='Always'/>,
document.getElementById('mask3'));
// sets mask format with the mask element '>' which converts all characters
that follow
// to upper case and '<' which converts all characters that follow to lower
case
```

```
ReactDOM.render(<MaskedTextBoxComponent mask={'>LLL<LLL'} placeholder={'Mask
>LLL<LL (ex: SAMple)'} floatLabelType='Always'/>,
document.getElementById('mask4'));
// sets mask format with the mask element '\\' which turns mask element `A`
into
// a literal and it displays the alphabet `A`
ReactDOM.render(<MaskedTextBoxComponent mask={'\\A999'} placeholder={'Mask
\\A999 (ex: A321)'} floatLabelType='Always'/>,
document.getElementById('mask5'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// sets mask with the mask element '#' which accepts any single digit from
'0' to '9',
// space, + and - signs
ReactDOM.render(<MaskedTextBoxComponent mask={'#####'} placeholder= {'Mask
##### (ex: 012+-)'} floatLabelType='Always'/>
,document.getElementById('mask1'));
// sets mask format with the mask element 'L' which allows only
alphabets('A-Z and a-z')
ReactDOM.render(<MaskedTextBoxComponent mask={'LLLLLL'} placeholder= {'Mask
LLLLLL (ex: Sample)'} floatLabelType='Always'/>
,document.getElementById('mask2'));
// sets mask format with the mask element '&' which allows `alphabets`,
`numbers`
// and `special characters`
ReactDOM.render(<MaskedTextBoxComponent mask={'&&&&'} placeholder= {'Mask
&&&& (ex: A12@#)'} floatLabelType='Always'/>
,document.getElementById('mask3'));
// sets mask format with the mask element `>` which converts all characters
that follow
// to upper case and `<` which converts all characters that follow to lower
case
ReactDOM.render(<MaskedTextBoxComponent mask={'>LLL<LLL'} placeholder=
{'Mask >LLL<LL (ex: SAMple)'} floatLabelType='Always'/>
,document.getElementById('mask4'));
// sets mask format with the mask element '\\' which turns mask element `A`
into
// a literal and it displays the alphabet `A`
ReactDOM.render(<MaskedTextBoxComponent mask={'\\A999'} placeholder= {'Mask
\\A999 (ex: A321)'} floatLabelType='Always'/>
,document.getElementById('mask5'));
```

[Functional-component]

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// sets mask with the mask element '#' which accepts any single digit from
'0' to '9',
// space, + and - signs
```

```

function App1() {
    return (<MaskedTextBoxComponent mask={'#####'} placeholder={'Mask #####
(ex: 012+-)'} floatLabelType="Always"/>);
}
ReactDOM.render(<App1 />, document.getElementById('mask1'));
// sets mask format with the mask element 'L' which allows only
alphabets('A-Z and a-z')
function App2() {
    return (<MaskedTextBoxComponent mask={'LLLLLL'} placeholder={'Mask
LLLLLL (ex: Sample)'} floatLabelType="Always"/>);
}
ReactDOM.render(<App2 />, document.getElementById('mask2'));
// sets mask format with the mask element '&' which allows `alphabets`,
`numbers`
// and `special characters`
function App3() {
    return (<MaskedTextBoxComponent mask={'&&&&&'} placeholder={'Mask &&&&&
(ex: A12@#)'} floatLabelType="Always"/>);
}
ReactDOM.render(<App3 />, document.getElementById('mask3'));
// sets mask format with the mask element `>` which converts all characters
that follow
// to upper case and `<` which converts all characters that follow to lower
case
function App4() {
    return (<MaskedTextBoxComponent mask={'>LLL<LLL'} placeholder={'Mask
>LLL<LL (ex: SAMple)'} floatLabelType="Always"/>);
}
ReactDOM.render(<App4 />, document.getElementById('mask4'));
// sets mask format with the mask element `\\` which turns mask element `A`
into
// a literal and it displays the alphabet `A`
function App5() {
    return (<MaskedTextBoxComponent mask={'\\A999'} placeholder={'Mask
\\A999 (ex: A321)'} floatLabelType="Always"/>);
}
ReactDOM.render(<App5 />, document.getElementById('mask5'));

```

INDEX.TSX

```

import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// sets mask with the mask element '#' which accepts any single digit from
'0' to '9',
// space, + and - signs
function App1() {
    return (
        <MaskedTextBoxComponent
            mask={'#####'}
            placeholder={'Mask ##### (ex: 012+-)'}
            floatLabelType="Always"
        />
    );
}
ReactDOM.render(<App1 />, document.getElementById('mask1'));

```

```

// sets mask format with the mask element 'L' which allows only
// alphabets('A-Z and a-z')
function App2() {
  return (
    <MaskedTextBoxComponent
      mask={'LLLLLL'}
      placeholder={'Mask LLLLLL (ex: Sample)'}
      floatLabelType="Always"
    />
  );
}
ReactDOM.render(<App2 />, document.getElementById('mask2'));
// sets mask format with the mask element '&' which allows `alphabets`,
// `numbers`
// and `special characters`
function App3() {
  return (
    <MaskedTextBoxComponent
      mask={'&&&&&'}
      placeholder={'Mask &&&&& (ex: A12@#)'}
      floatLabelType="Always"
    />
  );
}
ReactDOM.render(<App3 />, document.getElementById('mask3'));
// sets mask format with the mask element `>` which converts all characters
// that follow
// to upper case and `<` which converts all characters that follow to lower
// case
function App4() {
  return (
    <MaskedTextBoxComponent
      mask={'>LLL<LLL'}
      placeholder={'Mask >LLL<LL (ex: SAMple)'}
      floatLabelType="Always"
    />
  );
}
ReactDOM.render(<App4 />, document.getElementById('mask4'));
// sets mask format with the mask element '\\' which turns mask element `A`
// into
// a literal and it displays the alphabet `A`
function App5() {
  return (
    <MaskedTextBoxComponent
      mask={'\\A999'}
      placeholder={'Mask \\A999 (ex: A321)'}
      floatLabelType="Always"
    />
  );
}
ReactDOM.render(<App5 />, document.getElementById('mask5'));

```

Custom mask elements

Other than the above standard mask elements, the mask can be configured with the custom characters or regular expression to define a custom behavior.

Custom characters

You can define any of the non-mask element as the mask element and its behavior through the [customCharacters](#) property.

In the following example, non-mask element **P** accepts the values **P, A, p, a** and **M** accepts the values **M, m** as mentioned in the custom characters collection.

[Class-component]

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  chars = { P: 'P,A,a,p', M: 'M,m' };
  render() {
    return (
      // sets custom characters collection for non-mask elements 'P' and 'M'
      <MaskedTextBoxComponent mask='00:00' >PM'
      customCharacters={this.chars} placeholder='Time (ex: 10:00 PM, 10:00 AM)'
      floatLabelType='Always' />);
  }
}
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  public chars: any = { P: 'P,A,a,p', M: 'M,m' };
  public render() {
    return (
      // sets custom characters collection for non-mask elements 'P' and 'M'
      <MaskedTextBoxComponent mask='00:00' >PM' customCharacters=
      {this.chars} placeholder='Time (ex: 10:00 PM, 10:00 AM)'
      floatLabelType='Always' />
    );
  }
}
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

[Functional-component]

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
```

```
import * as ReactDOM from 'react-dom';
function App() {
  let chars = { P: 'P,A,a,p', M: 'M,m' };
  return (
    // sets custom characters collection for non-mask elements 'P' and 'M'
    <MaskedTextBoxComponent mask="00:00 >PM" customCharacters={chars}
    placeholder="Time (ex: 10:00 PM, 10:00 AM)" floatLabelType="Always"/>);
}
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let chars: any = { P: 'P,A,a,p', M: 'M,m' };
  return (
    // sets custom characters collection for non-mask elements 'P' and 'M'
    <MaskedTextBoxComponent
      mask="00:00 >PM"
      customCharacters={chars}
      placeholder="Time (ex: 10:00 PM, 10:00 AM)"
      floatLabelType="Always"
    />
  );
}
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

Regular expression

Instead of the mask element, you can define your own regular expression to validate the input of a particular input place.

The regular expressions should be wrapped by the square brackets (e.g.,: [Regex]).

In the following example, regular expression has been set for each input places.

[Class-component]**INDEX.JSX**

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
ReactDOM.render(<MaskedTextBoxComponent mask='[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9]' placeholder='IP Address (ex: 212.212.111.222)' floatLabelType='Always'/>,
  document.getElementById('masktextbox'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
```

```
ReactDOM.render(<MaskedTextBoxComponent mask='[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9]' placeholder='IP Address (ex: 212.212.111.222)' floatLabelType='Always' />, document.getElementById('maskedtextbox'));
```

[Functional-component]

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    return (<MaskedTextBoxComponent mask="[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9]" placeholder="IP Address (ex: 212.212.111.222)" floatLabelType="Always"/>);
}
ReactDOM.render(<App />, document.getElementById('maskedtextbox'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    return (
        <MaskedTextBoxComponent
            mask="[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9]"
            placeholder="IP Address (ex: 212.212.111.222)"
            floatLabelType="Always"
        />
    );
}
ReactDOM.render(<App />, document.getElementById('maskedtextbox'));
```

Prompt character

The Prompt character is a prompting symbol in the MaskedTextBox for the mask elements. The symbol is used to show the input positions in the MaskedTextBox. You can customize the prompt character of MaskedTextBox by using the [promptChar](#) property.

The following example demonstrates the MaskedTextBox with customized prompt character as `#`.

[Class-component]

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
ReactDOM.render(<MaskedTextBoxComponent mask='{ "999-999-9999" }' promptChar='{ "#" }'/>, document.getElementById('maskedContainer'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
ReactDOM.render(<MaskedTextBoxComponent mask={'999-999-9999'}
promptChar={'#'} />, document.getElementById('maskedContainer'));
```

[Functional-component]

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    return <MaskedTextBoxComponent mask={'999-999-9999'} promptChar={'#'} />;
}
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    return <MaskedTextBoxComponent mask={'999-999-9999'} promptChar={'#'} />;
}
ReactDOM.render(<App />, document.getElementById('maskedContainer'));
```

Accessibility in React Maskedtextbox component

The Maskedtextbox 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 Maskedtextbox 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 MaskedTextBox is characterized with complete ARIA Accessibility support that helps to access through the on-screen readers and other assistive technology devices. This component is designed with the reference of the guidelines document given in [WAI ARAI Accessibility practices](#).

The MaskedTextBox uses the `textbox` role and following ARIA properties for its element based on its state.

| Property | Functionality |

| --- | --- |

| aria-live | The `aria-live` attribute indicates the priority of updates to a live region. |

| aria-disabled | The `aria-disabled` property indicates the disabled state of the MaskedTextBox. |

| aria-valuenow | The `aria-valuenow` property specifies the current value of the MaskedTextBox. |

| aria-invalid | The `aria-invalid` property indicates that the user input is incorrect or not within the acceptable ranges. |

| aria-placeholder | The `aria-placeholder` is a short hint to help the users with data entry when the MaskedTextBox has no value. |

| aria-labelledby | The `aria-labelledby` property indicates the floating label element of the MaskedTextBox. |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

React functional component in React Maskedtextbox component

This section explains how to render the masked textbox component in the functional component with react hooks methods. Please find the list of basic hook methods in the following table.

Hooks	Description
-----	-----
useState	useState is a Hook that allows you to define the state in the functional components. If you pass the initial state to this function, then it will return a state variable with value and function to update this state value. Here, useState is used to store the value of product key field.
useEffect	useEffect is a Hook that allows you to execute code after rendering and re-rendering the component. Here, focusing the product key field on initial loading.
useRef	useRef is a Hook function that allows to directly create a reference to the DOM element in the functional component. Here, assigned the product key field reference for focusing the field on initial loading.
useReducer	useReducer is a Hook function that accepts a reducer function and an initial state as argument. It returns a state value and another function to dispatch an action. Here, dispatched the update action to update the mobile number and postal code state values.

The following example shows how to render a simple masked textbox with react hooks.

APP.JSX

```
import { useState, useEffect, useRef, useReducer } from 'react';
import * as React from 'react';
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
function App() {
  const [productKey, setProductKey] = useState('124-234-765-234');
  const initialState = { mobileNo: '', postalCode: '' };
  const productValueRef = useRef(null);
  const reducer = (state, action) => {
    switch (action.type) {
      case 'update':
        return { ...state, [action.field]: action.value };
      default:
        return initialState;
    }
  };
  const [state, dispatch] = useReducer(reducer, initialState);
  const changeHandler = (event) => {
```

```

        setProductKey(event.value);
    };
    const update = (field) => (event) => {
        //update action is dispatched and triggered state changes
        dispatch({ type: 'update', field, value: event.value });
    };
    useEffect(() => {
        productValueRef.current.focusIn();
    }, []);
    return (<div>
        <div className="control_wrapper" id="control_wrapper">
            <h3 className="form-title">Purchase</h3>
            <div className="control_wrapper textbox-form">
                <form id="form1" method="post">
                    <div className="form-group">
                        <MaskedTextBoxComponent ref={productValueRef} name="product"
mask={'000-000-000-000'} value={productKey} change={changeHandler}
multiline="true" placeholder="Product key" floatLabelType="Always" data-msg-
containerid="errroForAmount"/>
                        <div id="errroForAmount"/>
                    </div>
                    <div className="form-group">
                        <MaskedTextBoxComponent name="mobile" mask={'00-0000-0000'}
value={state.mobileNo} change={update('mobileNo')} placeholder="Mobile
Number" floatLabelType="Always" data-msg-containerid="errroForMobileNo"/>
                        <div id="errroForMobileNo"/>
                    </div>
                    <div className="form-group">
                        <MaskedTextBoxComponent name="postal" mask={'000-000'}
value={state.postalCode} change={update('postalCode')} placeholder="Postal
code" floatLabelType="Always" data-msg-containerid="errroForPostalCode"/>
                        <div id="errroForPostalCode"/>
                    </div>
                </form>
            </div>
            <br />
            <br />
        </div>
    </div>);
}
export default App;

```

APP.TSX

```

import { useState, useEffect, useRef, useReducer } from 'react';
import * as React from 'react';
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
function App() {
    const [productKey, setProductKey] = useState('124-234-765-234');
    const initialState = { mobileNo: '', postalCode: '' };
    const productValueRef = useRef(null);
    const reducer = (state, action) => {
        switch (action.type) {
            case 'update':
                return { ...state, [action.field]: action.value };
            default:

```

```

        return initialState;
    }
};
const [state, dispatch] = useReducer(reducer, initialState);
const changeHandler = (event) => {
    setProductKey(event.value);
};
const update = (field) => (event) => {
    //update action is dispatched and triggered state changes
    dispatch({ type: 'update', field, value: event.value });
};
useEffect(() => {
    productValueRef.current.focusIn();
}, []);
return (
    <div>
        <div className="control_wrapper" id="control_wrapper">
            <h3 className="form-title">Purchase</h3>
            <div className="control_wrapper textbox-form">
                <form id="form1" method="post">
                    <div className="form-group">
                        <MaskedTextBoxComponent
                            ref={productValueRef}
                            name="product"
                            mask={ '000-000-000-000' }
                            value={productKey}
                            change={changeHandler}
                            multiline="true"
                            placeholder="Product key"
                            floatLabelType="Always"
                            data-msg-containerid="errroForAmount"
                        />
                        <div id="errroForAmount" />
                    </div>
                    <div className="form-group">
                        <MaskedTextBoxComponent
                            name="mobile"
                            mask={ '00-0000-0000' }
                            value={state.mobileNo}
                            change={update('mobileNo')}
                            placeholder="Mobile Number"
                            floatLabelType="Always"
                            data-msg-containerid="errroForMobileNo"
                        />
                        <div id="errroForMobileNo" />
                    </div>
                    <div className="form-group">
                        <MaskedTextBoxComponent
                            name="postal"
                            mask={ '000-000' }
                            value={state.postalCode}
                            change={update('postalCode')}
                            placeholder="Postal code"
                            floatLabelType="Always"
                            data-msg-containerid="errroForPostalCode"
                        />
                        <div id="errroForPostalCode" />
                    </div>
                </form>
            </div>
        </div>
    </div>
);

```

```

        </div>
      </form>
    </div>
    <br />
    <br />
  </div>
</div>
);
}
export default App;

```

Style appearance in React Maskedtextbox component

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 MaskedTextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
```

/ To specify height, font size, and border /

```
.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;
```

```
border-color: red;
```

```
height: 40px;
```

```
border: 2px solid;
```

```
}
```

```
,
```

Customizing the MaskedTextBox element on hovering

Use the following CSS to customize the Input Mask element on hovering

```
`css
```

/ To specify border /

```
.e-input-group input.e-input, .e-input-group input.e-input:hover:not(.e-success):not(.e-warning):not(.e-error):not([disabled]):not(:focus), .e-input-group.e-control-wrapper input.e-input, .e-input-group.e-control-wrapper input.e-input:hover:not(.e-success):not(.e-warning):not(.e-error):not([disabled]):no(:focus){
```

```
border: 3px solid red;
```

```
}
```

```
,
```

How To

Perform custom validation using form validator in React Maskedtextbox component

To perform custom validation on the MaskedTextBox use the FormValidator along with custom validation rules.

In the following example, the MaskedTextBox is validated for invalid mobile number by adding custom validation in the rules collection of the FormValidator.

INDEX.JSX

```
{% raw %}
import { FormValidator } from '@syncfusion/ej2-inputs';
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  maskInstance;
  componentDidMount() {
    // checks the length of mask value and returns corresponding boolean
    value
    const customFn = (args) => {
      const argsLength = args.element.ej2_instances[0].value.length;
      if (argsLength !== 0) {
        return argsLength >= 10;
      }
      else {
        return true;
      }
    };
    // if mask value length is 0, 0 is returned else the length is
    returned
    const custom = (args) => {
      const argsLength = args.element.ej2_instances[0].value.length;
      if (argsLength === 0) {
        return 0;
      }
      else {
        return argsLength;
      }
    };
    const options = {
      customPlacement: (inputElement, errorElement) => {
        // to place the error message in custom position
        // errorElement - error text which will be displayed.
        document.querySelector("#masktextbox").appendChild(errorElement);
      },
      rules: {
        // must specify the name attribute value in rules section
        with required validation
        'mask_value': { numberValue: [customFn, 'Enter valid mobile
        number'] }
      }
    };
    const formObject = new FormValidator('#form-element', options);
    // FormValidator rule is added for empty MaskedTextBox
  }
}
```

```

        formObject.addRules('mask_value', { maxLength: [custom, 'Enter
mobile number'] });
        document.getElementById('submit_btn').addEventListener('click', ()
=> {
            formObject.validate('mask_value');
            // checks for incomplete value and alerts the formt submit
            if (this.maskInstance.value !== "" &&
this.maskInstance.value.indexOf(this.maskInstance.promptChar) === -1) {
                alert("Submitted");
            }
        });
    }
    render() {
        return (<div>
            <br /><MaskedTextBoxComponent id="mask" name="mask_value"
ref={ (mask) => { this.maskInstance = mask; }} placeholder="Mobile Number"
mask="000-000-0000" floatLabelType='Always' />
            <label className='e-error' htmlFor='mask_value' />
            <button id="submit_btn" type='button' style={{ marginTop: '10px'
}}>Submit</button>
        </div>);
    }
}
;
ReactDOM.render(<App />, document.getElementById('masktextbox'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
    public maskInstance: any;
    public componentDidMount(): void {
        // checks the length of mask value and returns corresponding boolean
        value
        const customFn: (args: { [key: string]: string }) => boolean = (args: {
[key: string]: string }) => {
            const argsLength = (args.element as any).ej2_instances[0].value.length;
            if(argsLength !== 0) {
                return argsLength >= 10;
            }
            else {
                return true;
            }
        };
        // if mask value length is 0, 0 is returned else the length is returned
        const custom: (args: { [key: string]: string }) => boolean = (args: {
[key: string]: string }) => {
            const argsLength = (args.element as any).ej2_instances[0].value.length;
            if(argsLength === 0) {
                return 0;
            }
        }
    }
}

```

```

    else {
        return argsLength;
    }
};

const options: FormValidatorModel = {
    customPlacement: (inputElement: HTMLElement, errorElement:
HTMLElement) => {
        // to place the error message in custom position
        // errorElement - error text which will be displayed.
        (document.querySelector("#masktextbox") as
HTMLElement).appendChild(errorElement);
    },
    rules: {
        // must specify the name attribute value in rules section with
        required validation
        'mask_value': { numberValue: [customFn, 'Enter valid mobile
number'] }
    }
};

const formObject: FormValidator = new FormValidator('#form-element',
options);
// FormValidator rule is added for empty MaskedTextBox
formObject.addRules('mask_value', { maxLength: [custom, 'Enter mobile
number'] });
(document.getElementById('submit_btn') as
HTMLElement).addEventListener('click', () => {
    formObject.validate('mask_value');
    // checks for incomplete value and alerts the formt submit
    if (this.maskInstance.value !== "" &&
this.maskInstance.value.indexOf(this.maskInstance.promptChar) === -1) {
        alert("Submitted");
    }
});
}

public render() {
    return (
        <div>
            <br/><MaskedTextBoxComponent id="mask" name="mask_value" ref =
{(mask) => {this.maskInstance = mask as MaskedTextBoxComponent}}
placeholder="Mobile Number" mask= "000-000-0000" floatLabelType='Always' />
            <label className='e-error' htmlFor='mask_value' />
            <button id="submit_btn" type='button' style={{ marginTop: '10px'
}}>Submit</button>
        </div>
    );
}
};

ReactDOM.render(<App />, document.getElementById('masktextbox'));
{% endraw %}

```

Set cursor position while focus on the input textbox in React Maskedtextbox component

By default, on focusing the MaskedTextBox the entire mask gets selected. You can customize by using any one of the following methods:

- Setting cursor position at the start of the MaskedTextBox.

- Setting cursor position at the end of the MaskedTextBox.
- Setting cursor at the specified position in the MaskedTextBox.

The **selectionStart** and **selectionEnd** set to **0** instead of the input element value's length, when we focus on a MaskedTextBox control filled with all mask characters. This is the default behavior of the HTML 5 input element.

Following is an example that demonstrates the above cases to set cursor position in the MaskedTextBox using [focus](#) event.

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  constructor(props) {
    super(props);
    this.onfocus = this.onfocus.bind(this);
    this.onfocus1 = this.onfocus1.bind(this);
    this.onfocus2 = this.onfocus2.bind(this);
  }
  onfocus(args) {
    // sets cursor position at start of MaskedTextBox
    args.selectionEnd = args.selectionStart = 0;
  }
  onfocus1(args) {
    // sets cursor position at end of MaskedTextBox
    args.selectionStart = args.selectionEnd = args.maskedValue.length;
  }
  onfocus2(args) {
    // sets cursor at specified position
    args.selectionStart = 3;
    args.selectionEnd = 3;
  }
  render() {
    return (
      <div>
        <br /><MaskedTextBoxComponent name="mask_value1"
placeholder="Default cursor position" mask="00000-00000" value="93828-32132"
floatLabelType='Always' />
        <MaskedTextBoxComponent name="mask_value2" placeholder="Cursor
positioned at start" mask="00000-00000" value="83929-4342"
floatLabelType='Always' focus={this.onfocus} />
        <MaskedTextBoxComponent name="mask_value3" placeholder="Cursor
positioned at end" mask="00000-00000" value="83929-3213"
floatLabelType='Always' focus={this.onfocus1} />
        <MaskedTextBoxComponent name="mask_value4" placeholder="Cursor at
specified position" mask="+1 000-000-0000" value="234-432-432"
floatLabelType='Always' focus={this.onfocus2} />
      </div>);
  }
}
;
ReactDOM.render(<App />, document.getElementById('masktextbox'));
```

INDEX.TSX

```

import { MaskedTextBoxComponent, MaskFocusEventArgs } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
    constructor(props: any) {
        super(props);
        this.onfocus = this.onfocus.bind(this);
        this.onfocus1 = this.onfocus1.bind(this);
        this.onfocus2 = this.onfocus2.bind(this);
    }
    public onfocus(args: MaskFocusEventArgs) {
        // sets cursor position at start of MaskedTextBox
        args.selectionEnd= args.selectionStart = 0;
    }
    public onfocus1(args: MaskFocusEventArgs) {
        // sets cursor position at end of MaskedTextBox
        args.selectionStart=args.selectionEnd = args.maskedValue.length;
    }
    public onfocus2(args: MaskFocusEventArgs) {
        // sets cursor at specified position
        args.selectionStart = 3;
        args.selectionEnd = 3;
    }
    public render() {
        return (
            <div>
                <br/><MaskedTextBoxComponent name="mask_value1"
placeholder="Default cursor position" mask= "00000-00000" value="93828-32132" floatLabelType='Always' />
                <MaskedTextBoxComponent name="mask_value2" placeholder="Cursor positioned at start" mask= "00000-00000" value="83929-4342" floatLabelType='Always' focus={this.onfocus}/>
                <MaskedTextBoxComponent name="mask_value3" placeholder="Cursor positioned at end" mask= "00000-00000" value="83929-3213" floatLabelType='Always' focus={this.onfocus1}/>
                <MaskedTextBoxComponent name="mask_value4" placeholder="Cursor at specified position" mask= "+1 000-000-0000" value="234-432-432" floatLabelType='Always' focus={this.onfocus2}/>
            </div>
        );
    }
};
ReactDOM.render(<App />, document.getElementById('masktextbox'));

```

Display numeric keypad when focus on mobile devices in React Maskedtextbox component

By default, on focusing the MaskedTextBox, alphanumeric keypad will be displayed on mobile devices. Sometimes only numeric keypad for number values is needed, and this can be achieved by setting "type" property to `tel`.

Refer to the following example to enable numeric keypad in MaskedTextBox.

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  render() {
    return (<div>
      <br /><MaskedTextBoxComponent name="mask_value" mask='999-99999'
value="342-45432" type="tel"/>
    </div>);
  }
};
ReactDOM.render(<App />, document.getElementById('masktextbox'));
```

INDEX.TSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  public render() {
    return (
      <div>
        <br/><MaskedTextBoxComponent name="mask_value" mask='999-99999'
value= "342-45432" type="tel"/>
      </div>
    );
  }
};
ReactDOM.render(<App />, document.getElementById('masktextbox'));
```

Customize the ui appearance of the control in React Maskedtextbox component

The appearance of the MaskedTextBox can be changed by adding custom `cssClass` to the component and enabling styles.

Refer to the following example to change the appearance of the MaskedTextBox.

INDEX.JSX

```
import { MaskedTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  constructor(props) {
    super(props);
    this.onfocus = this.onfocus.bind(this);
  }
  onfocus(args) {
    // sets cursor position at start of MaskedTextBox
    args.selectionEnd = args.selectionStart;
  }
  render() {
    return (<div>
      <br />
    </div>);
  }
};
ReactDOM.render(<App />, document.getElementById('masktextbox'));
```

```

        <MaskedTextBoxComponent id="mask1" name="mask_value"
placeholder="Enter user ID" floatLabelType='Always' mask="00000"
cssClass="e-style" value="42648" focus={this.onfocus}/>
      </div>;
    }
  }
;
ReactDOM.render(<App />, document.getElementById('masktextbox'));

```

INDEX.TSX

```

import { MaskedTextBoxComponent, MaskFocusEventArgs } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  constructor(props: any) {
    super(props);
    this.onfocus = this.onfocus.bind(this);
  }
  public onfocus(args: MaskFocusEventArgs) {
    // sets cursor position at start of MaskedTextBox
    args.selectionEnd= args.selectionStart;
  }
  public render() {
    return (
      <div>
        <br/>
        <MaskedTextBoxComponent id="mask1" name="mask_value"
placeholder="Enter user ID" floatLabelType='Always' mask= "00000"
cssClass="e-style" value= "42648" focus={this.onfocus}/>
      </div>
    );
  }
};
ReactDOM.render(<App />, document.getElementById('masktextbox'));

```

Ej1 api migration in React Maskedtextbox component

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

Common

```
{% raw %}
```

```
<!-- markdownlint-disable MD033 -->
```

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

```
| --- | --- | --- |
```

```
| Adding custom class | Property: cssClass <br /><br /><EJ.MaskEdit id="mask" maskFormat="9999"
cssClass="custom"></EJ.MaskEdit> | Property: cssClass<br /><br /><MaskedTextBoxComponent
id="mask" mask='9999' cssClass = "custom"></MaskedTextBoxComponent> |
```

| Destroy editor | Not Applicable | **Method:** *destroy*
`<MaskedTextBoxComponent id="mask" mask='00-000'></MaskedTextBoxComponent>
var mask = document.getElementById('mask').ej2_instances[0];
mask.destroy(); |`

| Disable the maskedit control | **Method:** *disable*
`<EJ.MaskEdit id="mask" maskFormat="0000" value={1234}></EJ.MaskEdit>
var maskObj = $("#mask").data("ejMaskEdit");
maskObj.disable(); | Can be achieved using
<MaskedTextBoxComponent id="mask" mask='00-000'></MaskedTextBoxComponent>
var mask = document.getElementById('mask').ej2_instances[0];
mask.enabled= false; |`

| Enable the maskedit control | **Method:** *enable*
`<EJ.MaskEdit id="mask" maskFormat="0000" value={1234}></EJ.MaskEdit>
var maskObj = $("#mask").data("ejMaskEdit");
maskObj.enable(); | Can be achieved using
<MaskedTextBoxComponent id="mask" mask='00-000'></MaskedTextBoxComponent>
var mask = document.getElementById('mask').ej2_instances[0];
mask.enabled= true; |`

| Control state | **Property:** *enabled*
`<EJ.MaskEdit id="mask" maskFormat="0-0000" enabled={false}></EJ.MaskEdit> | Property: enabled
<MaskedTextBoxComponent id="mask" mask='00-000' enabled=false></MaskedTextBoxComponent> |`

| Persistence | **Property:** *enablePersistence*
`<EJ.MaskEdit id="mask" maskFormat="0000" enablePersistence={{true}}></EJ.MaskEdit> | Property: enablePersistence
<MaskedTextBoxComponent id="mask" mask='0000' enablePersistence=true></MaskedTextBoxComponent> |`

| Triggers when editor is focused in | **Event:** *focusIn*
`<EJ.MaskEdit id="mask" maskFormat="00-00" focusIn={this.focusIn}></EJ.MaskEdit>
<script>focusIn: function() {} | Event: focus
<MaskedTextBoxComponent id="mask" mask='00-00' focus={this.onFocus.bind(this)}></MaskedTextBoxComponent>
<script>public onFocus() {} |`

| Triggers when editor is focused out | **Event:** *focusOut*
`<EJ.MaskEdit id="mask" maskFormat="0000" focusOut={this.focusOut}></EJ.MaskEdit>
<script>focusOut: function() {} | Event: blur
<MaskedTextBoxComponent id="mask" mask='00-00' blur={this.onBlur.bind(this)}></MaskedTextBoxComponent>
<script>public onBlur() {} |`

| Sets height | **Property:** *height*
`<EJ.MaskEdit id="mask" maskFormat="0000" height="30px"></EJ.MaskEdit> | Can be achieved using,
<MaskedTextBoxComponent id="mask" mask='0000' cssClass = "custom"></MaskedTextBoxComponent>
<css>.e-maskedtextbox.custom{
height: 30px
} |`

| HTML Attributes | **Property:** *htmlAttributes*
`<EJ.MaskEdit id="mask" maskFormat="0000" htmlAttributes= {htmlAttr}></EJ.MaskEdit>
<script>var htmlAttr = {name: "maskedtextbox"}; | Can be achieved using
<MaskedTextBoxComponent id="mask" mask='00-000' name="textbox"></MaskedTextBoxComponent> |`

| Specifies Input mode | **Property:** *inputMode*
`<EJ.MaskEdit id="mask" maskFormat="0000" inputMode="password"></EJ.MaskEdit>` | **Can be achieved using**
`<MaskedTextBoxComponent type="password" name="mask"></MaskedTextBoxComponent>` |

| Triggers on key press | **Event:** *keyPress*
`<EJ.MaskEdit id="mask" maskFormat="0000" keyPress={this.keyPress}></EJ.MaskEdit>`
`<script>keyPress: function() {}` | **Can be achieved using**
`<MaskedTextBoxComponent mask="9999-999-999" value= '9789660245' id="mask" placeholder="Numeric TextBox" created={this.onCreated.bind(this)} floatLabelType="Always"> </MaskedTextBoxComponent>`
`public onCreated(){document.getElementById("mask").addEventListener('keypress',function({});`
`
}` |

| Triggers on key up | **Event:** *keyUp*
`<EJ.MaskEdit id="mask" maskFormat="0000" keyUp={this.keyUp}></EJ.MaskEdit>`
`<script>keyUp: function() {}` | **Can be achieved using**
`<MaskedTextBoxComponent mask="9999-999-999" value= '9789660245' id="mask" placeholder="Numeric TextBox" created={this.onCreated.bind(this)} floatLabelType="Always"> </MaskedTextBoxComponent>`
`public onCreated(){document.getElementById("mask").addEventListener('keyup',function({});`
`
}` |

| Triggers on mouse out in maskedit control | **Event:** *mouseOut*
`<EJ.MaskEdit id="mask" maskFormat="0000-000" mouseOut={this.onMouseOut}></EJ.MaskEdit>`
`<script>onMouseOut: function() {}` | **Can be achieved using**
`<MaskedTextBoxComponent mask="9999-999-999" value= '9789660245' id="mask" placeholder="Numeric TextBox" created={this.onCreated.bind(this)} floatLabelType="Always"> </MaskedTextBoxComponent>`
`public onCreated(){document.getElementById("mask").addEventListener('mouseout',function({};`
`
}` |

| Triggers when mouse over in maskedit control | **Event:** *mouseOver*
`<EJ.MaskEdit id="mask" maskFormat="00-00" mouseOver={this.onMouseOver}></EJ.MaskEdit>`
`<script>onMouseOver: function() {}` | **Can be achieved using**
`<MaskedTextBoxComponent mask="9999-999-999" value= '9789660245' id="mask" placeholder="Numeric TextBox" created={this.onCreated.bind(this)} floatLabelType="Always"> </MaskedTextBoxComponent>`
`public onCreated(){document.getElementById("mask").addEventListener('mouseover',function({};`
`
}` |

| Name of maskedit control | **Property:** *name*
`<EJ.MaskEdit id="mask" maskFormat="0000" name="pin"></EJ.MaskEdit>` | **Can be achieved using**
`<MaskedTextBoxComponent id="mask" mask='00-000' name="textbox"></MaskedTextBoxComponent>` |

| Triggers on keydown | **Event:** *onKeyDown*
`<EJ.MaskEdit id="mask" maskFormat="9999-9999" onKeyDown={this.onKeyDown}></EJ.MaskEdit>`
`<script>onKeyDown: function() {}` | **Can be achieved using**
`<MaskedTextBoxComponent`

```
mask="9999-999-999" value= '9789660245' id="mask" placeholder="Numeric TextBox"
created={this.onCreated.bind(this)} floatLabelType="Always">
</MaskedTextBoxComponent><br/>public
onCreated(){<br/>document.getElementById("mask").addEventListener('keydown',function({});
<br/>}
```

| Read only | **Property:** *readOnly*

<EJ.MaskEdit id="mask" maskFormat="99-999"
readOnly={true}></EJ.MaskEdit> | **Can be achieved by**
<MaskedTextBoxComponent
mask='000-000-0000' enabled={false}></MaskedTextBoxComponent>
CSS
.e-
mask{
background-image: none !important ;
border-bottom-color: rgba(0, 0, 0, 0.42)
!important ;
}

| Right to left | **Property:** *textAlign*

<EJ.MaskEdit id="mask" maskFormat="9999"
textAlign="Right"></EJ.MaskEdit> | **Property:** *enableRtl*

<MaskedTextBoxComponent
id="mask" mask='9999' enableRtl = true></MaskedTextBoxComponent> |

| Sets width | **Property:** *width*

<EJ.MaskEdit id="mask" maskFormat="0000"
width="100px"></EJ.MaskEdit> | **Property:** *width*

<MaskedTextBoxComponent
id="mask" mask='0000' width = "100px"></MaskedTextBoxComponent> |

Mask Configuration

<!-- markdownlint-disable MD033 -->

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

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

| Triggers on value change | **Event** *change*

<EJ.MaskEdit id="mask" maskFormat="00-00"
change={this.onChange}></EJ.MaskEdit>

script
onChange: function() {} | **Event:** *change*

<MaskedTextBoxComponent id="mask" mask='00-00'
change={this.onChange.bind(this)}></MaskedTextBoxComponent>

public
onChange() {} |

| Clears maskedit text/value | **Method:** *clear*

<EJ.MaskEdit id="mask"
maskFormat="0000" value={1234}></EJ.MaskEdit>
var maskObj =
\$("#mask").data("ejMaskEdit");
maskObj.clear(); | **Can be achieved
using**
<MaskedTextBoxComponent id="mask" mask='00-000'
value="1234"></MaskedTextBoxComponent>
var mask =
document.getElementById('mask').ej2_instances[0];
mask.value= ""; |

| Triggers on creation | **Event:** *create*

<EJ.MaskEdit id="mask" maskFormat="00-00"
create={this.onCreate}></EJ.MaskEdit>

script
onCreate: function() {} | **Event:**
created

<MaskedTextBoxComponent id="mask" mask='00-00'
created={this.onCreated.bind(this)}></MaskedTextBoxComponent>

public
onCreated() {} |

| Custom Character | **Property:** *customCharacter*

<EJ.MaskEdit id="mask"
maskFormat="C-0000" customCharacter="#"></EJ.MaskEdit> | **Property:** *customCharacters*

<MaskedTextBoxComponent id="mask" mask='C-0000' customCharacters=

```
{this.chars}></MaskedTextBoxComponent><br /><br />public customCharacters: object = {C:
'#'}; |
```

```
| Triggers when maskedit control is destroyed | Event: destroy<br /><br /><EJ.MaskEdit id="mask"
maskFormat="00-00" destroy={this.onDestroy}></EJ.MaskEdit><br /><br /><script><br
/>onDestroy: function() {} | Event: destroyed<br /><br /><MaskedTextBoxComponent id="mask"
mask='00-00' destroyed={this.onDestroyed.bind(this)}></MaskedTextBoxComponent><br /><br
/>public onDestroyed() {} |
```

```
| Placeholder float type | Not Applicable | Property: floatLabelType<br /><br />
/><MaskedTextBoxComponent id="mask" mask='9,999' placeholder="Enter value"
floatLabelType="Auto"></MaskedTextBoxComponent> |
```

```
| Gets pure value of maskedit control | Method: getStrippedValue<br /><br /><EJ.MaskEdit id="mask"
maskFormat="0000" value={123}></EJ.MaskEdit><br /><br /><script><br />var maskObj =
$("#mask").data("ejMaskEdit");<br />maskObj.getStrippedValue(); | Can be achieved
using<br /><MaskedTextBoxComponent id="mask" mask='00-
000'></MaskedTextBoxComponent><br />var mask =
document.getElementById('mask').ej2_instances[0];<br />alert(mask.value); |
```

```
| Get whole maskedit value | Method: getUnstrippedValue<br /><br /><EJ.MaskEdit id="mask"
maskFormat="00-00" value={1234}></EJ.MaskEdit><br />var maskObj =
$("#mask").data("ejMaskEdit");<br />maskObj.getUnstrippedValue(); | Method: getMaskedValue<br
/><br /><script><br /><MaskedTextBoxComponent id="mask" mask='00-00'
value="1234"></MaskedTextBoxComponent><br />var mask =
document.getElementById('mask').ej2_instances[0];<br />mask.getMaskedValue(); |
```

```
| Hides prompt character on focus out | Property: hidePromptOnLeave<br /><br /><EJ.MaskEdit
id="mask" maskFormat="aaaa" hidePromptOnLeave={true}></EJ.MaskEdit> | Can be achieved
by<br /><MaskedTextBoxComponent mask="9999-999-999" value= '9789660245' id="mask"
placeholder="Numeric TextBox" created={this.onCreate.bind(this)}
focus={this.onfocus.bind(this)} floatLabelType="Always">
</MaskedTextBoxComponent><br /><script><br />public onCreate(){<br />var maskObj =
document.getElementById("mask");<br />maskObj.addEventListener('focusout',function(){<br />
maskObj.ej2_instances[0].promptChar = " "<br />}}<br /><br />public onfocus(){<br />var
maskObj = document.getElementById("mask").ej2_instances[0];<br />maskObj.promptChar=
"_"<br />}<br /> |
```

```
| Mask format | Property: maskFormat<br /><br /><EJ.MaskEdit id="mask"
maskFormat="99,999"></EJ.MaskEdit> | Property: mask<br /><br /><MaskedTextBoxComponent
id="mask" mask='99,999'></MaskedTextBoxComponent> |
```

```
| Prompt character | Not Applicable | Property: promptChar<br /><br /><MaskedTextBoxComponent
id="mask" mask='0000' promptChar="#"></MaskedTextBoxComponent> |
```

```
| Clear Button | Not Applicable | Property: showClearButton<br /><br /><MaskedTextBoxComponent
id="mask" mask='aaaa' showClearButton=true></MaskedTextBoxComponent> |
```


| Prompt character display | **Property:** `showPromptChar`
`<EJ.MaskEdit id="mask" maskFormat="$99-999" showPromptChar={false}>` | **Can be achieved using**
`<MaskedTextBoxComponent id="mask" mask='0000' promptChar=" ">` |

| Show rounded corner | **Property:** `showRoundedCorner`
`<EJ.MaskEdit id="mask" maskFormat="0000" showRoundedCorner={true}>` | **Can be achieved using**
`<MaskedTextBoxComponent mask="9999-999-999" value= '9789660245' id="mask1" placeholder="Numeric TextBox" cssClass = "e-style" floatLabelType="Always">`
`</MaskedTextBoxComponent>` **CSS**
`#mask1 {border: 2px solid grey; padding: 7px; border-radius: 10px; .e-control-wrapper.e-mask.e-float-input.e-style .e-float-line::before, .e-control-wrapper.e-mask.e-float-input.e-style .e-float-line::after {background: none ;}}` |

| Value of maskedit control | **Property:** `value`
`<EJ.MaskEdit id="mask" maskFormat="0000" value={1234}>` | **Property:** `value`
`<MaskedTextBoxComponent id="mask" mask='0000' value="1234">` |

| Displays hint on maskedit control | **Property:** `watermarkText`
`<EJ.MaskEdit id="mask" maskFormat="9999" watermarkText="Enter value">` | **Property:** `placeholder`
`<MaskedTextBoxComponent id="mask" mask='9999' placeholder="Enter value">` |

Validation

`<!-- markdownlint-disable MD033 -->`

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

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

| Displays error until correct value is entered | **Property:** `showError`
`<EJ.MaskEdit id="mask" maskFormat="99-999" showError={true}>` | **MaskedTextBox by default shows error until the correct value is entered**
`<MaskedTextBoxComponent id="mask" mask='0000' value="1234">` |

| Validation message | **Property:** `validationMessage`
`<EJ.MaskEdit id="mask" maskFormat="0000" validationRules= {validationRules} validationMessage= {validationMessage}>`
`<var validationRules = {required: {true}};>`
`<validationMessage = {required: "Required value"};>` | **Validation can be performed using Form Validation**
`<MaskedTextBoxComponent id="mask" name="maskvalue" placeholder="Mobile Number" mask= "000-000-0000" floatLabelType='Always' />`
`let options: FormValidatorModel = {rules: { 'maskvalue': { numberValue: [required, 'Enter valid mobile number'] }, customPlacement: (inputElement: HTMLElement, errorElement: HTMLElement) =>`
`{document.querySelector("#masktextbox").appendChild(errorElement);}}`
`et formObject: FormValidator = new FormValidator('#form-element', options);`
HTML
`<form id="form-element" class="form-horizontal">`
`<div id='masktextbox'>`
`</div>`
`</form>` |

```
| Validation Rules | Property: validationRules<br /><br /><EJ.MaskEdit id="mask" maskFormat="00-00" validationRules= {validationRules}></EJ.MaskEdit><br /><br />var validationRules = {required: {true}}; | Validation can be performed using Form  
Validation<br /><MaskedTextBoxComponent id="mask" name="maskvalue" placeholder="Mobile Number" mask= "000-000-0000" floatLabelType='Always' /><br />let options: FormValidatorModel = {<br />rules: {<br />'maskvalue': { numberValue: [required, true] }<br />},<br />customPlacement: (inputElement: HTMLElement, errorElement: HTMLElement) => {<br />document.querySelector("#masktextbox").appendChild(errorElement);<br />}<br />};<br />let formObject: FormValidator = new FormValidator('#form-element', options);<br />HTML<br /><form id="form-element" class="form-horizontal"><br /><div id='masktextbox'></div><br /></form> |  
{% endraw %}
```

Mention

Getting Started

This section explains how to create a simple [Link to the Video](#) component and configure its available functionalities in React.

To get start quickly with React Mention, you can check on this video:

Dependencies

The following list of dependencies are required to use the **Mention** component in your application.

```
`javascript  
|-- @syncfusion/ej2-react-dropdowns  
|-- @syncfusion/ej2-dropdowns  
|-- @syncfusion/ej2-base  
|-- @syncfusion/ej2-data  
|-- @syncfusion/ej2-lists  
|-- @syncfusion/ej2-popups  
|-- @syncfusion/ej2-buttons  
|-- @syncfusion/ej2-react-base  
,
```

Setup your development environment

You can use [Create-react-app](#) to setup the applications.

To install **create-react-app** run the following command.

```
`bash  
npm install -g create-react-app  
,
```

Start a new project using create-react-app command as follows

```
<div class='tsx'>
`

create-react-app quickstart --scripts-version=react-scripts-ts
cd quickstart
`

</div>

<div class='jsx'>
`

create-react-app quickstart
cd quickstart
`

</div>
```

'react-scripts-ts' is used for creating React app with typescript.

Adding syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. You can choose the component that you want to install.

To install Mention component, use the following command

```
`bash
npm install @syncfusion/ej2-react-dropdowns --save
`
```

The above command installs [Mention dependencies](#) which are required to render the component in the **React** environment.

Adding Style sheet to the Application

To render Mention component, need to import dropdowns and its dependent components styles as given below in **src/App.css**.

```
`css

/ import the Mention dependency styles /
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-react-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-react-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-react-list/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-react-dropdowns/styles/bootstrap5.css";
`
```

Adding Mention component

Now, you can add the **Mention** component in the application. To use the Mention component properly, the **target** property should be configured so that it renders the Mention component in the configured element. Add **Mention** component in **src/App.tsx** file using the following code snippet.

[Class-component]

```
`ts
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
// Defines the target in which the Mention component is rendered.
private mentionTarget: string = '#mentionElement';
public render() {
return (
<div id='mention_default'>

<MentionComponent target={this.mentionTarget} ></MentionComponent>
</div>
);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

```
`ts
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
// Defines the target in which the Mention component is rendered.
mentionTarget = '#mentionElement';
render() {
return (<div id='mention_default'>
```

Comments

```
<MentionComponent target={this.mentionTarget}></MentionComponent>
</div>;
}
}
ReactDOM.render(<App />, document.getElementById('sample'));
、
```

[Functional-component]

```
`ts
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App (){
// Defines the target in which the Mention component is rendered.
let mentionTarget: string = '#mentionElement';
return (
<div id='mention_default'>
```

Comments

```
<MentionComponent target={mentionTarget} ></MentionComponent>
</div>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));
、

`ts
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
// Defines the target in which the Mention component is rendered.
let mentionTarget = '#mentionElement';
```

```
return (<div id='mention_default'>
```

Comments

```
<MentionComponent target={mentionTarget}></MentionComponent>
```

```
</div>;
```

```
}
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

```
,
```

Binding data source

After initialization, populate the data using the [dataSource](#) property. Here, an array of string values is passed to the Mention component.

[Class-component]

```
`ts
```

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
```

```
import * as React from 'react';
```

```
import * as ReactDOM from "react-dom";
```

```
export default class App extends React.Component<{}, {}> {
```

```
// Defines the target in which the Mention component is rendered.
```

```
private mentionTarget: string = '#mentionElement';
```

```
// Defines the array of data.
```

```
private userData: string[] = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
```

```
public render() {
```

```
return (
```

```
<div id='mention_default'>
```

Comments

```
<MentionComponent target={this.mentionTarget} dataSource={this.userData}></MentionComponent>
```

```
</div>
```

```
);
```

```
}
```

```
}
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

```
,
```

```
`ts
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  // Defines the target in which the Mention component is rendered.
  mentionTarget = '#mentionElement';
  // Defines the array of data.
  userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
  render() {
    return (<div id='mention_default'>
```

Comments

```
<MentionComponent target={this.mentionTarget} dataSource={this.userData}></MentionComponent>
</div>);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

[Functional-component]

```
`ts
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App () {
  // Defines the target in which the Mention component is rendered.
  let mentionTarget: string = '#mentionElement';
  // Defines the array of data.
  let userData: string[] = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
  return (
    <div id='mention_default'>
```

Comments

```

<MentionComponent target={mentionTarget} dataSource={userData}></MentionComponent>
</div>
);
}

```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

```
,
```

```
`ts
```

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
```

```
import * as React from 'react';
```

```
import * as ReactDOM from "react-dom";
```

```
function App() {
```

```
// Defines the target in which the Mention component is rendered.
```

```
let mentionTarget = '#mentionElement';
```

```
// Defines the array of data.
```

```
let userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
```

```
return (<div id='mention_default'>
```

```
Comments
```

```
<MentionComponent target={mentionTarget} dataSource={userData}></MentionComponent>
```

```
</div>;
```

```
}
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

```
,
```

[Run the application](#)

Run the application in the browser using the following command:

```
,
```

```
npm start
```

```
,
```

The following example shows a basic Mention component.

[Class-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";

```



```

export default class App extends React.Component {
  // Defines the target in which the Mention component is rendered.
  mentionTarget = '#mentionElement';
  // Defines the array of data.
  userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
  render() {
    return (<div id='mention_default'>
      <table>
        <tr>
          <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
user'></div>
          </td>
        </tr>
      </table>
      <MentionComponent target={this.mentionTarget}
dataSource={this.userData}></MentionComponent>
    </div>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  // Defines the target in which the Mention component is rendered.
  private mentionTarget: string = '#mentionElement';
  // Defines the array of data.
  private userData: string[] = ['Selma Rose', 'Garth', 'Robert', 'William',
'Joseph'];
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag user'
></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.userData}></MentionComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
    // Defines the target in which the Mention component is rendered.
    let mentionTarget = '#mentionElement';
    // Defines the array of data.
    let userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                </td>
            </tr>
        </table>
        <MentionComponent target={mentionTarget}
dataSource={userData}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App(){
    // Defines the target in which the Mention component is rendered.
    let mentionTarget: string = '#mentionElement';
    // Defines the array of data.
    let userData: string[] = ['Selma Rose', 'Garth', 'Robert', 'William',
'Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag user'
></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={mentionTarget}
dataSource={userData}></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Display Mention character

By using the [showMentionChar](#) property, the text content can be displayed along with the mention character. You can customize the mention character by using the [mentionChar](#) property in the Mention component.

By default, the [mentionChar](#) is @ and the [showMentionChar](#) property is disabled.

The following example displays the text content along with the mention character configured as #.

[Class-component]

INDEX.JSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
    // Defines the target in which the Mention component is rendered.
    mentionTarget = '#mentionElement';
    // Defines the array of data.
    userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
    // Defines the character in which the mention component is initialized
    when pressing.
    mentionCharacter = "#";
    render() {
        return (<div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type # and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={this.mentionTarget}
dataSource={this.userData} showMentionChar={true}
mentionChar={this.mentionCharacter}></MentionComponent>
            </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
    // Defines the target in which the Mention component is rendered.
    private mentionTarget: string = '#mentionElement';
    // Defines the array of data.
    private userData: string[] = ['Selma Rose', 'Garth', 'Robert', 'William',
'Joseph'];
```

```

    // Defines the character in which the mention component is initialized when pressing.
    private mentionCharacter: string = "#";
    public render() {
        return (
            <div id='mention_default'>
                <table>
                    <tr>
                        <td>
                            <label id="comment">Comments</label>
                            <div id="mentionElement" placeholder='Type # and tag user'
                        ></div>
                        </td>
                    </tr>
                </table>
                <MentionComponent target={this.mentionTarget}
                dataSource={this.userData} showMentionChar={true}
                mentionChar={this.mentionCharacter}></MentionComponent>
            </div>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
    // Defines the target in which the Mention component is rendered.
    let mentionTarget = '#mentionElement';
    // Defines the array of data.
    let userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
    // Defines the character in which the mention component is initialized when pressing.
    let mentionCharacter = "#";
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type # and tag
user'></div>
                </td>
            </tr>
        </table>
        <MentionComponent target={mentionTarget} dataSource={userData}
        showMentionChar={true} mentionChar={mentionCharacter}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App () {
    // Defines the target in which the Mention component is rendered.
    let mentionTarget: string = '#mentionElement';
    // Defines the array of data.
    let userData: string[] = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];

    // Defines the character in which the mention component is initialized when pressing.
    let mentionCharacter: string = "#";
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type # and tag user'
                    </div>
                </td>
            </tr>
        </table>
        <MentionComponent target={mentionTarget} dataSource={userData}
        showMentionChar={true} mentionChar={mentionCharacter}></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Working with data in React Mention component

The Mention loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of either **array** or **DataManager**.

The Mention also supports different kinds of data services such as OData V4 and Web API, and data formats such as XML, JSON, and JSONP with the help of **DataManager** adaptors.

Fields	Type	Description
text	string	Specifies the display text of each list item.
value	number or string	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	string	Specifies the category under which the list item has to be grouped.
iconCss	string	Specifies the icon class of each list item.

When binding complex data to the Mention, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in three ways as described in the following.

Array of simple data

The Mention has provided support to load an array of primitive data such as strings and numbers. Here, both the value and text fields act the same.

[Class-component]

INDEX.JSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
  render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
user'></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.userData}></MentionComponent>
      </div>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  private mentionTarget: string = '#mentionElement';
  private userData: string[] = ['Selma Rose', 'Garth', 'Robert', 'William',
'Joseph'];
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
user'></div>
            </td>
          </tr>
        </table>
      </div>);
  }
}
```

```

        <MentionComponent target={this.mentionTarget}
        dataSource={this.userData} ></MentionComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
  let mentionTarget = '#mentionElement';
  let userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
  return (<div id='mention_default'>
    <table>
      <tr>
        <td>
          <label id="comment">Comments</label>
          <div id="mentionElement" placeholder='Type @ and tag
user'></div>
        </td>
      </tr>
    </table>
    <MentionComponent target={mentionTarget}
    dataSource={userData}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App () {
  let mentionTarget: string = '#mentionElement';
  let userData: string[] = ['Selma Rose', 'Garth', 'Robert', 'William',
'Joseph'];
  return (
    <div id='mention_default'>
      <table>
        <tr>
          <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
user'></div>
          </td>
        </tr>
      </table>
      <MentionComponent target={mentionTarget} dataSource={userData}
></MentionComponent>
    </div>);
}

```

```

    </div>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Array of JSON data

The Mention can generate its list of items through an array of JSON data. Therefore the appropriate columns should be mapped to the [fields](#) property.

In the following example, **ID** column and **Game** column from complex data have been mapped to the **value** field and **text** field, respectively.

[Class-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  sportsData = [
    { ID: 'game1', Game: 'Badminton' },
    { ID: 'game2', Game: 'Football' },
    { ID: 'game3', Game: 'Tennis' },
    { ID: 'game4', Game: 'Hockey' },
    { ID: 'game5', Game: 'Basketball' }
  ];
  fields = { text: 'Game', value: 'ID' };
  render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
sport'></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.sportsData} fields={this.fields}></MentionComponent>
      </div>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  private mentionTarget: string = '#mentionElement';
  private sportsData: { [key: string]: Object }[] = [
    { ID: 'game1', Game: 'Badminton' },

```



```

    { ID: 'game2', Game: 'Football' },
    { ID: 'game3', Game: 'Tennis' },
    { ID: 'game4', Game: 'Hockey' },
    { ID: 'game5', Game: 'Basketball' }
  ];
  private fields:Object = { text: 'Game', value: 'ID' }
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
sport'></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.sportsData} fields={this.fields} ></MentionComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
  let mentionTarget = '#mentionElement';
  let sportsData = [
    { ID: 'game1', Game: 'Badminton' },
    { ID: 'game2', Game: 'Football' },
    { ID: 'game3', Game: 'Tennis' },
    { ID: 'game4', Game: 'Hockey' },
    { ID: 'game5', Game: 'Basketball' }
  ];
  let fields = { text: 'Game', value: 'ID' };
  return (<div id='mention_default'>
    <table>
      <tr>
        <td>
          <label id="comment">Comments</label>
          <div id="mentionElement" placeholder='Type @ and tag
sport'></div>
        </td>
      </tr>
    </table>
    <MentionComponent target={mentionTarget} dataSource={sportsData}
fields={fields}></MentionComponent>
  </div>);
}

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App () {
  let mentionTarget: string = '#mentionElement';
  let sportsData: { [key: string]: Object }[] = [
    { ID: 'game1', Game: 'Badminton' },
    { ID: 'game2', Game: 'Football' },
    { ID: 'game3', Game: 'Tennis' },
    { ID: 'game4', Game: 'Hockey' },
    { ID: 'game5', Game: 'Basketball' }
  ];
  let fields:Object = { text: 'Game', value: 'ID' }
  return (
    <div id='mention_default'>
      <table>
        <tr>
          <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
sport'></div>
          </td>
        </tr>
      </table>
      <MentionComponent target={mentionTarget} dataSource={sportsData}
fields={fields} ></MentionComponent>
    </div>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Array of Complex data

The Mention can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Code.ID** column and **Country.Name** column from complex data have been mapped to the **value** field and **text** field, respectively.

[Class-component]

INDEX.JSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  countriesData = [
    { Country: { Name: 'Australia' }, Code: { ID: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { ID: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { ID: 'CA' } },
  ];
}
```

```

    { Country: { Name: 'Cameroon' }, Code: { ID: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { ID: 'DK' } },
    { Country: { Name: 'France' }, Code: { ID: 'FR' } },
  ];
  fields = { text: 'Country.Name', value: 'Code.ID' };
  render() {
    return (<div id='mention_default'>
      <table>
        <tr>
          <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
country'></div>
          </td>
        </tr>
      </table>
      <MentionComponent target={this.mentionTarget}
dataSource={this.countriesData} fields={this.fields}></MentionComponent>
    </div>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  private mentionTarget: string = '#mentionElement';
  private countriesData: { [key: string]: Object }[] = [
    { Country: { Name: 'Australia' }, Code: { ID: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { ID: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { ID: 'CA' } },
    { Country: { Name: 'Cameroon' }, Code: { ID: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { ID: 'DK' } },
    { Country: { Name: 'France' }, Code: { ID: 'FR' } },
  ];
  private fields: Object = { text: 'Country.Name', value: 'Code.ID' };
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
country'></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.countriesData} fields={this.fields} ></MentionComponent>
      </div>
    );
  }
}

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
    let mentionTarget = '#mentionElement';
    let countriesData = [
        { Country: { Name: 'Australia' }, Code: { ID: 'AU' } },
        { Country: { Name: 'Bermuda' }, Code: { ID: 'BM' } },
        { Country: { Name: 'Canada' }, Code: { ID: 'CA' } },
        { Country: { Name: 'Cameroon' }, Code: { ID: 'CM' } },
        { Country: { Name: 'Denmark' }, Code: { ID: 'DK' } },
        { Country: { Name: 'France' }, Code: { ID: 'FR' } },
    ];
    let fields = { text: 'Country.Name', value: 'Code.ID' };
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
country'></div>
                </td>
            </tr>
        </table>
        <MentionComponent target={mentionTarget} dataSource={countriesData}
fields={fields}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App () {
    let mentionTarget: string = '#mentionElement';
    let countriesData: { [key: string]: Object }[] = [
        { Country: { Name: 'Australia' }, Code: { ID: 'AU' } },
        { Country: { Name: 'Bermuda' }, Code: { ID: 'BM' } },
        { Country: { Name: 'Canada' }, Code: { ID: 'CA' } },
        { Country: { Name: 'Cameroon' }, Code: { ID: 'CM' } },
        { Country: { Name: 'Denmark' }, Code: { ID: 'DK' } },
        { Country: { Name: 'France' }, Code: { ID: 'FR' } },
    ];
    let fields: Object = { text: 'Country.Name', value: 'Code.ID' };
    return (
        <div id='mention_default'>
            <table>
```

```

        <tr>
            <td>
                <label id="comment">Comments</label>
                <div id="mentionElement" placeholder='Type @ and tag
country'></div>
            </td>
        </tr>
    </table>
    <MentionComponent target={mentionTarget} dataSource={countriesData}
fields={fields} ></MentionComponent>
</div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Binding remote data

The Mention supports retrieval of data from remote data services with the help of **DataManager** component. The [query](#) property is used to fetch the data from the database and bind it to the Mention component.

OData v4 adaptor - Binding OData v4 service

The ODataV4 is an improved version of OData protocols, and the **DataManager** can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [odata documentation](#). To bind OData v4 service, use the **ODataV4Adaptor**.

The following sample displays the first 6 contacts from **Customers** table of the **Northwind** Data Service.

[Class-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component {
    mentionTarget = '#mentionElement';
    dataSource = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
    fields = { text: 'ContactName', value: 'CustomerID' };
    render() {
        return (<div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
        </div>);
    }
}

```

```

        </table>
        <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} fields={this.fields} query={this.query}
popupWidth={'250px'}></MentionComponent>
    </div>;
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component<{}, {}> {
    private mentionTarget: string = '#mentionElement';
    private dataSource: DataManager = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    private query: Query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
    private fields: Object = { text: 'ContactName', value: 'CustomerID' };
    public render() {
        return (
            <div id='mention_default'>
                <table>
                    <tr>
                        <td>
                            <label id="comment">Comments</label>
                            <div id="mentionElement" placeholder='Type @ and tag
user' ></div>
                        </td>
                    </tr>
                </table>
                <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} fields={this.fields} query={this.query}
popupWidth={'250px'}></MentionComponent>
            </div>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App() {
    let mentionTarget = '#mentionElement';

```

```

let dataSource = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
let fields = { text: 'ContactName', value: 'CustomerID' };
return (<div id='mention_default'>
    <table>
        <tr>
            <td>
                <label id="comment">Comments</label>
                <div id="mentionElement" placeholder='Type @ and tag
user'></div>
            </td>
        </tr>
    </table>
    <MentionComponent dataSource={dataSource} target={mentionTarget}
fields={fields} query={query} popupWidth={'250px'}></MentionComponent>
</div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App () {
let mentionTarget: string = '#mentionElement';
let dataSource: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let query:Query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
let fields:Object = { text: 'ContactName', value: 'CustomerID' };
return (
    <div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
user' ></div>
                </td>
            </tr>
        </table>
        <MentionComponent dataSource={dataSource} target={mentionTarget}
fields={fields} query={query} popupWidth={'250px'}></MentionComponent>
    </div>
    );
}

```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

Web API adaptor

You can use **WebApiAdaptor** to bind mention with Web API created using OData endpoint.

[Class-component]

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query, DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  dataSource = new DataManager({
    url:
'https://services.syncfusion.com/react/production/api/Employees',
    adaptor: new WebApiAdaptor(),
    crossDomain: true
  });
  query = new Query().select(['FirstName', 'EmployeeID']).take(7);
  fields = { text: 'FirstName', value: 'EmployeeID' };
  render() {
    return (<div id='mention_default'>
      <table>
        <tr>
          <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
user'></div>
          </td>
        </tr>
      </table>
      <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} fields={this.fields} query={this.query}
popupWidth={'250px'}></MentionComponent>
    </div>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query, DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
export default class App extends React.Component<{}, {}> {
  private mentionTarget: string = '#mentionElement';
  private dataSource: DataManager = new DataManager({
    url:
'https://services.syncfusion.com/react/production/api/Employees',
    adaptor: new WebApiAdaptor(),
    crossDomain: true
  });
};
```



```

private query:Query = new Query().select(['FirstName',
'EmployeeID']).take(7);
private fields:Object = { text: 'FirstName', value: 'EmployeeID' };

public render() {
return (
<div id='mention_default'>
<table>
<tr>
<td>
<label id="comment">Comments</label>
<div id="mentionElement" placeholder='Type @ and tag user'
></div>
</td>
</tr>
</table>
<MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} fields={this.fields} query={this.query}
popupWidth={'250px'}></MentionComponent>
</div>
);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query, DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
function App() {
let mentionTarget = '#mentionElement';
let dataSource = new DataManager({
url:
'https://services.syncfusion.com/react/production/api/Employees',
adaptor: new WebApiAdaptor(),
crossDomain: true
});
let query = new Query().select(['FirstName', 'EmployeeID']).take(7);
let fields = { text: 'FirstName', value: 'EmployeeID' };
return (<div id='mention_default'>
<table>
<tr>
<td>
<label id="comment">Comments</label>
<div id="mentionElement" placeholder='Type @ and tag
user'></div>
</td>
</tr>
</table>
<MentionComponent dataSource={dataSource} target={mentionTarget}
fields={fields} query={query} popupWidth={'250px'}></MentionComponent>
</div>);

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query, DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
function App() {
    let mentionTarget: string = '#mentionElement';
    let dataSource: DataManager = new DataManager({
        url:
        'https://services.syncfusion.com/react/production/api/Employees',
        adaptor: new WebApiAdaptor(),
        crossDomain: true
    });
    let query: Query = new Query().select(['FirstName', 'EmployeeID']).take(7);
    let fields: Object = { text: 'FirstName', value: 'EmployeeID' };
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag user'
                    ></div>
                    </td>
                </tr>
            </table>
            <MentionComponent dataSource={dataSource} target={mentionTarget}
            fields={fields} query={query} popupWidth={'250px'}></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

See Also

- [Customization](#)
- [How to perform filtering](#)

Filtering data in React Mention component

The Mention component has built-in support to filter data items. The filter operation starts as soon as you start typing characters in the mention element.

Limit the minimum filter character

You can control the minimum length of user input to initiate the search action using [minLength](#) property. This can be useful if you have a very large list of data. The default value is 0, where suggestion the list opened as soon as the user inputs the mention character.

The remote request does not fetch the search data until the search key contains three characters as shown in the following example.

[Class-component]**INDEX.JSX**

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
export default class App extends React.Component {
    mentionTarget = '#mentionElement';
    searchData = new DataManager({
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    dataFields = { text: 'ContactName', value: 'CustomerID' };
    query = new Query().select(['ContactName', 'CustomerID']).take(7);
    minLength = 3;
    render() {
        return (<div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent dataSource={this.searchData}
target={this.mentionTarget} fields={this.dataFields} query={this.query}
minLength={this.minLength}></MentionComponent>
        </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
export default class App extends React.Component<{}, {}> {
    private mentionTarget: string = '#mentionElement';
    private searchData: DataManager = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    private dataFields: Object = { text: 'ContactName', value: 'CustomerID' };
    private query: Query = new Query().select(['ContactName',
'CustomerID']).take(7);
    private minLength = 3;
    public render() {
        return(

```

```

        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent dataSource={this.searchData}
target={this.mentionTarget} fields={this.dataFields} query={this.query}
minLength={this.minLength}></MentionComponent>
        </div>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
function App() {
    let mentionTarget = '#mentionElement';
    let searchData = new DataManager({
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    let dataFields = { text: 'ContactName', value: 'CustomerID' };
    let query = new Query().select(['ContactName', 'CustomerID']).take(7);
    let minLength = 3;
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                </td>
            </tr>
        </table>
        <MentionComponent dataSource={searchData} target={mentionTarget}
fields={dataFields} query={query} minLength={minLength}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
function App() {
    let mentionTarget: string = '#mentionElement';
    let searchData: DataManager = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    let dataFields: Object = { text: 'ContactName', value: 'CustomerID' };
    let query: Query = new Query().select(['ContactName',
'CustomerID']).take(7);
    let minLength = 3;
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent dataSource={searchData} target={mentionTarget}
fields={dataFields} query={query} minLength={minLength}></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Change the filter type

While filtering, you can change the filter type to **Contains**, **StartsWith**, or **EndsWith** in the [filterType](#) property. The default filter operator is **Contains**.

- **StartsWith** - Filter the items that begin with the specified text value.
- **Contains** - Filter the items that contain the specified text value.
- **EndsWith** - Filter the items that end with the specified text value.

[Class-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
export default class App extends React.Component {
    mentionTarget = '#mentionElement';
    searchData = new DataManager({

```

```

        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    query = new Query().select(['ContactName', 'CustomerID']).take(7);
    fields = { text: 'ContactName', value: 'CustomerID' };
    filterType = 'EndsWith';
    render() {
        return (<div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={this.mentionTarget}
dataSource={this.searchData} query={this.query} fields={this.fields}
filterType={this.filterType}></MentionComponent>
        </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
export default class App extends React.Component<{}, {}> {
    private mentionTarget: string = '#mentionElement';
    private searchData: DataManager = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    private query: Query = new Query().select(['ContactName',
'CustomerID']).take(7);
    private fields: Object = { text: 'ContactName', value: 'CustomerID' };
    private filterType: FilterType = 'EndsWith';

    public render() {
        return (
            <div id='mention_default'>
                <table>
                    <tr>
                        <td>
                            <label id="comment">Comments</label>
                            <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                        </td>
                    </tr>
                </table>
            </div>
        );
    }
}

```

```

        </table>
        <MentionComponent target={this.mentionTarget}
        dataSource={this.searchData} query={this.query} fields={this.fields}
        filterType={this.filterType} ></MentionComponent>
    </div>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
function App() {
    let mentionTarget = '#mentionElement';
    let searchData = new DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    let query = new Query().select(['ContactName', 'CustomerID']).take(7);
    let fields = { text: 'ContactName', value: 'CustomerID' };
    let filterType = 'EndsWith';
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                </td>
            </tr>
        </table>
        <MentionComponent target={mentionTarget} dataSource={searchData}
        query={query} fields={fields} filterType={filterType}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
function App () {
    let mentionTarget: string = '#mentionElement';
    let searchData: DataManager = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,

```

```

        crossDomain: true
    });
    let query: Query = new Query().select(['ContactName',
'CustomerID']).take(7);
    let fields: Object = { text: 'ContactName', value: 'CustomerID' };
    let filterType: FilterType = 'EndsWith';
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={mentionTarget} dataSource={searchData}
query={query} fields={fields} filterType={filterType} ></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Allow spacing between search

While filtering the data in the data source, you can allow the space in the middle of the mention by using the [allowSpaces](#) property. If the data source does not match with the mentioned element data, the popup will be hidden on the space key press. The default value of [allowSpaces](#) is `false`.

By default, the [allowSpaces](#) property is disabled, and the space ends the mention component search.

[Class-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
    mentionTarget = '#mentionElement';
    userData = [
        { Name: "Andrew Fuller", ID: "1" },
        { Name: "Anne Dodsworth", ID: "2" },
        { Name: "Janet Leverling", ID: "3" },
        { Name: "Laura Callahan", ID: "4" },
        { Name: "Margaret Peacock", ID: "5" }
    ];
    fields = { text: 'Name' };
    render() {
        return (<div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={mentionTarget} dataSource={searchData}
query={query} fields={fields} filterType={filterType} ></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```



```

        </td>
      </tr>
    </table>
    <MentionComponent target={this.mentionTarget}
dataSource={this.userData} fields={this.fields}
allowSpaces={true}></MentionComponent>
  </div>);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  private mentionTarget: string = '#mentionElement';
  private userData: {[key: string]: Object}[] = [
    { Name : "Andrew Fuller", ID : "1" },
    { Name : "Anne Dodsworth" , ID : "2" },
    { Name : "Janet Leverling" , ID : "3" },
    { Name : "Laura Callahan" , ID : "4" },
    { Name : "Margaret Peacock" , ID : "5" }
  ];
  private fields: Object = {text:'Name'};
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
user'></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.userData} fields={this.fields} allowSpaces={true}
></MentionComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
  let mentionTarget = '#mentionElement';

```

```

let userData = [
  { Name: "Andrew Fuller", ID: "1" },
  { Name: "Anne Dodsworth", ID: "2" },
  { Name: "Janet Leverling", ID: "3" },
  { Name: "Laura Callahan", ID: "4" },
  { Name: "Margaret Peacock", ID: "5" }
];
let fields = { text: 'Name' };
return (<div id='mention_default'>
  <table>
    <tr>
      <td>
        <label id="comment">Comments</label>
        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
      </td>
    </tr>
  </table>
  <MentionComponent target={mentionTarget} dataSource={userData}
fields={fields} allowSpaces={true}></MentionComponent>
</div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App () {
  let mentionTarget: string = '#mentionElement';
  let userData: {[key: string]: Object}[] = [
    { Name : "Andrew Fuller", ID : "1" },
    { Name : "Anne Dodsworth", ID : "2" },
    { Name : "Janet Leverling", ID : "3" },
    { Name : "Laura Callahan", ID : "4" },
    { Name : "Margaret Peacock", ID : "5" }
  ];
  let fields: Object = {text:'Name'};
  return (
    <div id='mention_default'>
      <table>
        <tr>
          <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
user'></div>
          </td>
        </tr>
      </table>
      <MentionComponent target={mentionTarget} dataSource={userData}
fields={fields} allowSpaces={true} ></MentionComponent>
    </div>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Customize the suggestion item count

While filtering, you can customize the number of list items to be displayed in the suggestion list by using the [suggestionCount](#) property.

[Class-component]

INDEX.JSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
    // Defines the target in which the Mention component is rendered.
    mentionTarget = '#mentionElement';
    // Defines the array of data.
    emailData = [
        { Name: "Selma Rose", EmailId: "selma@gmail.com" },
        { Name: "Maria", EmailId: "maria@gmail.com" },
        { Name: "Russo Kay", EmailId: "russo@gmail.com" },
        { Name: "Robert", EmailId: "robert@gmail.com" },
        { Name: "Camden Kate", EmailId: "camden@gmail.com" },
        { Name: "Garth", EmailId: "garth@gmail.com" },
        { Name: "Andrew James", EmailId: "james@gmail.com" },
        { Name: "Olivia", EmailId: "olivia@gmail.com" },
        { Name: "Sophia", EmailId: "sophia@gmail.com" },
        { Name: "Margaret", EmailId: "margaret@gmail.com" },
        { Name: "Ursula Ann", EmailId: "ursula@gmail.com" },
        { Name: "Laura Grace", EmailId: "laura@gmail.com" },
        { Name: "Albert", EmailId: "albert@gmail.com" },
        { Name: "William", EmailId: "william@gmail.com" }
    ];
    fields = { text: 'Name' };
    render() {
        return (<div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={this.mentionTarget}
dataSource={this.emailData} fields={this.fields}
suggestionCount={8}></MentionComponent>
        </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
```

```

import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}>, {}> {
  // Defines the target in which the Mention component is rendered.
  private mentionTarget: string = '#mentionElement';
  // Defines the array of data.
  private emailData: { [key: string]: Object }[] = [
    { Name: "Selma Rose", EmailId: "selma@gmail.com" },
    { Name: "Maria", EmailId: "maria@gmail.com" },
    { Name: "Russo Kay", EmailId: "russo@gmail.com" },
    { Name: "Robert", EmailId: "robert@gmail.com" },
    { Name: "Camden Kate", EmailId: "camden@gmail.com" },
    { Name: "Garth", EmailId: "garth@gmail.com" },
    { Name: "Andrew James", EmailId: "james@gmail.com" },
    { Name: "Olivia", EmailId: "olivia@gmail.com" },
    { Name: "Sophia", EmailId: "sophia@gmail.com" },
    { Name: "Margaret", EmailId: "margaret@gmail.com" },
    { Name: "Ursula Ann", EmailId: "ursula@gmail.com" },
    { Name: "Laura Grace", EmailId: "laura@gmail.com" },
    { Name: "Albert", EmailId: "albert@gmail.com" },
    { Name: "William", EmailId: "william@gmail.com" }
  ];
  private fields: object = { text: 'Name' };
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
user' ></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.emailData} fields={this.fields}
suggestionCount={8}></MentionComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
  // Defines the target in which the Mention component is rendered.
  let mentionTarget = '#mentionElement';
  // Defines the array of data.
  let emailData = [
    { Name: "Selma Rose", EmailId: "selma@gmail.com" },

```

```

    { Name: "Maria", EmailId: "maria@gmail.com" },
    { Name: "Russo Kay", EmailId: "russo@gmail.com" },
    { Name: "Robert", EmailId: "robert@gmail.com" },
    { Name: "Camden Kate", EmailId: "camden@gmail.com" },
    { Name: "Garth", EmailId: "garth@gmail.com" },
    { Name: "Andrew James", EmailId: "james@gmail.com" },
    { Name: "Olivia", EmailId: "olivia@gmail.com" },
    { Name: "Sophia", EmailId: "sophia@gmail.com" },
    { Name: "Margaret", EmailId: "margaret@gmail.com" },
    { Name: "Ursula Ann", EmailId: "ursula@gmail.com" },
    { Name: "Laura Grace", EmailId: "laura@gmail.com" },
    { Name: "Albert", EmailId: "albert@gmail.com" },
    { Name: "William", EmailId: "william@gmail.com" }
  ];
  let fields = { text: 'Name' };
  return (<div id='mention_default'>
    <table>
      <tr>
        <td>
          <label id="comment">Comments</label>
          <div id="mentionElement" placeholder='Type @ and tag
user'></div>
        </td>
      </tr>
    </table>
    <MentionComponent target={mentionTarget} dataSource={emailData}
fields={fields} suggestionCount={8}></MentionComponent>
  </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App () {
  // Defines the target in which the Mention component is rendered.
  let mentionTarget: string = '#mentionElement';
  // Defines the array of data.
  let emailData: { [key: string]: Object }[] = [
    { Name: "Selma Rose", EmailId: "selma@gmail.com" },
    { Name: "Maria", EmailId: "maria@gmail.com" },
    { Name: "Russo Kay", EmailId: "russo@gmail.com" },
    { Name: "Robert", EmailId: "robert@gmail.com" },
    { Name: "Camden Kate", EmailId: "camden@gmail.com" },
    { Name: "Garth", EmailId: "garth@gmail.com" },
    { Name: "Andrew James", EmailId: "james@gmail.com" },
    { Name: "Olivia", EmailId: "olivia@gmail.com" },
    { Name: "Sophia", EmailId: "sophia@gmail.com" },
    { Name: "Margaret", EmailId: "margaret@gmail.com" },
    { Name: "Ursula Ann", EmailId: "ursula@gmail.com" },
    { Name: "Laura Grace", EmailId: "laura@gmail.com" },
    { Name: "Albert", EmailId: "albert@gmail.com" },
    { Name: "William", EmailId: "william@gmail.com" }
  ];
}

```

```

let fields:object = { text: 'Name' };
return (
  <div id='mention_default'>
    <table>
      <tr>
        <td>
          <label id="comment">Comments</label>
          <div id="mentionElement" placeholder='Type @ and tag
user' ></div>
        </td>
      </tr>
    </table>
    <MentionComponent target={mentionTarget} dataSource={emailData}
fields={fields} suggestionCount={8}></MentionComponent>
  </div>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

See Also

- [Templates](#)

Sorting in React Mention component

You can display the suggestion list items in a specific order. It has possible types as **Ascending**, **Descending**, and **None** in the [sortOrder](#) property.

- **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.

In the following sample, the popup list data is rendered in **Descending** order.

[Class-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  sportData = [
    { ID: "game1", Game: "Badminton" },
    { ID: "game2", Game: "Football" },
    { ID: "game3", Game: "Tennis" },
    { ID: "game4", Game: "Hockey" },
    { ID: "game5", Game: "Basketball" },
    { ID: "game6", Game: "Cricket" }
  ];
  fields = { text: 'Game' };
  render() {
    return (<div id='mention_default'>

```

```

        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
sport'></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.sportData} fields={this.fields}
sortOrder={"Descending"}></MentionComponent>
      </div>;
    }
  }
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  private mentionTarget: string = '#mentionElement';
  private sportData: { [key: string]: Object }[] = [
    { ID : "game1" ,Game : "Badminton"},
    { ID : "game2" ,Game : "Football" },
    { ID : "game3" ,Game : "Tennis"},
    { ID : "game4" ,Game : "Hockey"},
    { ID : "game5" ,Game : "Basketball"},
    { ID : "game6" ,Game : "Cricket"}
  ];
  private fields: Object = {text:'Game'};
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag sport '
></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.sportData} fields={this.fields} sortOrder={"Descending"}
></MentionComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
    let mentionTarget = '#mentionElement';
    let sportData = [
        { ID: "game1", Game: "Badminton" },
        { ID: "game2", Game: "Football" },
        { ID: "game3", Game: "Tennis" },
        { ID: "game4", Game: "Hockey" },
        { ID: "game5", Game: "Basketball" },
        { ID: "game6", Game: "Cricket" }
    ];
    let fields = { text: 'Game' };
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
sport'></div>
                </td>
            </tr>
        </table>
        <MentionComponent target={mentionTarget} dataSource={sportData}
fields={fields} sortOrder={"Descending"}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App () {
    let mentionTarget: string = '#mentionElement';
    let sportData: { [key: string]: Object }[] = [
        { ID : "game1" ,Game : "Badminton"},
        { ID : "game2" ,Game : "Football" },
        { ID : "game3" ,Game : "Tennis"},
        { ID : "game4" ,Game : "Hockey"},
        { ID : "game5" ,Game : "Basketball"},
        { ID : "game6" ,Game : "Cricket"}
    ];
    let fields: Object = {text:'Game'};
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag sport '
></div>

```



```

        </td>
      </tr>
    </table>
    <MentionComponent target={mentionTarget} dataSource={sportData}
fields={fields} sortOrder={"Descending"} ></MentionComponent>
  </div>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Template in React Mention component

The Mention has been provided with several options to customize each suggestion list item, display item, and data loading indication.

Item template

The content of each list item in Mention can be customized using [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data using [itemTemplate](#).

[Class-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  dataSource = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(10);
  fields = { text: "FirstName", value: "EmployeeID" };
  itemTemplate(data) {
    return (<span><span>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
  }
  render() {
    return (<div id='mention_default'>
      <table>
        <tr>
          <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
user'></div>
          </td>
        </tr>
      </table>
      <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} fields={this.fields}

```

```

itemTemplate={this.itemTemplate} query={this.query} sortOrder={'Ascending'}
popupWidth={'250px'}></MentionComponent>
</div>);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component<{}, {}> {
    private mentionTarget: string = '#mentionElement';
    private dataSource: DataManager = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    private query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(10);
    private fields: Object = { text: "FirstName", value: "EmployeeID" };
    private itemTemplate(data: any): JSX.Element {
        return (
            <span><span>{data.FirstName}</span><span className
='city'>{data.City}</span></span>
        );
    }
    public render() {
        return (
            <div id='mention_default'>
                <table>
                    <tr>
                        <td>
                            <label id="comment">Comments</label>
                            <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                        </td>
                    </tr>
                </table>
                <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} fields={this.fields}
itemTemplate={this.itemTemplate} query={this.query} sortOrder={'Ascending'}
popupWidth={'250px'}></MentionComponent>
            </div>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';

```

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App() {
    let mentionTarget = '#mentionElement';
    let dataSource = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    let query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(10);
    let fields = { text: "FirstName", value: "EmployeeID" };
    function itemTemplate(data) {
        return (<span><span>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
    }
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                </td>
            </tr>
        </table>
        <MentionComponent dataSource={dataSource} target={mentionTarget}
fields={fields} itemTemplate={itemTemplate} query={query}
sortOrder={'Ascending'} popupWidth={'250px'}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App() {
    let mentionTarget: string = '#mentionElement';
    let dataSource: 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(['FirstName', 'City', 'EmployeeID']).take(10);
    let fields: Object = { text: "FirstName", value: "EmployeeID" };
    function itemTemplate(data: any): JSX.Element {
        return (
            <span><span>{data.FirstName}</span><span className
='city'>{data.City}</span></span>
        );
    }
}

```

```

return (
  <div id='mention_default'>
    <table>
      <tr>
        <td>
          <label id="comment">Comments</label>
          <div id="mentionElement" placeholder='Type @ and tag
user'></div>
        </td>
      </tr>
    </table>
    <MentionComponent dataSource={dataSource} target={mentionTarget}
fields={fields} itemTemplate={itemTemplate} query={query}
sortOrder={ 'Ascending' } popupWidth={ '250px' }></MentionComponent>
  </div>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Display template

You can customize the mentioned value's display appearance using the [displayTemplate](#) property.

In the following sample, the selected value is displayed as a combined text of both FirstName and City in the mention element, which is separated by a hyphen.

[Class-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  dataSource = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(10);
  fields = { text: "FirstName", value: "EmployeeID" };
  itemTemplate(data) {
    return (<span><span>{data.FirstName}</span><span>
className='city'>{data.City}</span></span>);
  }
  displayTemplate(data) {
    return (<React.Fragment>
      <span>{data.FirstName} - {data.City}</span>
    </React.Fragment>);
  }
  render() {
    return (<div id='mention_default'>
      <table>
        <tr>

```

```

        <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
user'></div>
        </td>
    </tr>
</table>
    <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} fields={this.fields}
itemTemplate={this.itemTemplate} displayTemplate={this.displayTemplate}
query={this.query} sortOrder={ 'Ascending' }
popupWidth={ '250px' }></MentionComponent>
    </div>;
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component<{}, {}> {
    private mentionTarget: string = '#mentionElement';
    private dataSource: DataManager = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    private query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(10);
    private fields: Object = { text: "FirstName", value: "EmployeeID" };
    private itemTemplate(data: any): JSX.Element {
        return (
            <span><span>{data.FirstName}</span><span className
='city'>{data.City}</span></span>
        );
    }
    private displayTemplate(data: any): JSX.Element {
        return (
            <React.Fragment>
                <span>{data.FirstName} - {data.City}</span>
            </React.Fragment>
        );
    }

    public render() {
        return (
            <div id='mention_default'>
                <table>
                    <tr>
                        <td>
                            <label id="comment">Comments</label>
                            <div id="mentionElement" placeholder='Type @ and tag
user'></div>

```

```

        </td>
      </tr>
    </table>
    <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} fields={this.fields}
itemTemplate={this.itemTemplate} displayTemplate={this.displayTemplate}
query={this.query} sortOrder={'Ascending'}
popupWidth={'250px'}></MentionComponent>
  </div>
);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App() {
  let mentionTarget = '#mentionElement';
  let dataSource = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  let query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(10);
  let fields = { text: "FirstName", value: "EmployeeID" };
  function itemTemplate(data) {
    return (<span><span>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
  }
  function displayTemplate(data) {
    return (<React.Fragment>
      <span>{data.FirstName} - {data.City}</span>
    </React.Fragment>);
  }
  return (<div id='mention_default'>
    <table>
      <tr>
        <td>
          <label id="comment">Comments</label>
          <div id="mentionElement" placeholder='Type @ and tag
user'></div>
        </td>
      </tr>
    </table>
    <MentionComponent dataSource={dataSource} target={mentionTarget}
fields={fields} itemTemplate={itemTemplate} displayTemplate={displayTemplate}
query={query} sortOrder={'Ascending'}
popupWidth={'250px'}></MentionComponent>
  </div>);
}

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App() {
    let mentionTarget: string = '#mentionElement';
    let dataSource: 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(['FirstName', 'City', 'EmployeeID']).take(10);
    let fields: Object = { text: "FirstName", value: "EmployeeID" };
    function itemTemplate(data: any): JSX.Element {
        return (
            <span><span>{data.FirstName}</span><span className
            ='city'>{data.City}</span></span>
        );
    }
    function displayTemplate(data: any): JSX.Element {
        return (
            <React.Fragment>
                <span>{data.FirstName} - {data.City}</span>
            </React.Fragment>
        );
    }
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
                        user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent dataSource={dataSource} target={mentionTarget}
            fields={fields} itemTemplate={itemTemplate} displayTemplate={displayTemplate}
            query={query} sortOrder={'Ascending'}
            popupWidth={'250px'}></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

No records template

You can show the custom design of the popup list content when no data is found and no matches are found on search with the help of [noRecordsTemplate](#) property.

[Class-component]**INDEX.JSX**

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
export default class App extends React.Component {
    mentionTarget = '#mentionElement';
    dataSource = [];
    noRecordsTemplate = "<span class='norecord'> NO DATA AVAILABLE</span>";
    render() {
        return (<div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget}
noRecordsTemplate={this.noRecordsTemplate}></MentionComponent>
        </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
export default class App extends React.Component<{}, {}> {
    private mentionTarget: string = '#mentionElement';
    private dataSource :[] = [];
    private noRecordsTemplate: string = "<span class='norecord'> NO DATA
AVAILABLE</span>";
    public render() {
        return (
            <div id='mention_default'>
                <table>
                    <tr>
                        <td>
                            <label id="comment">Comments</label>
                            <div id="mentionElement" placeholder='Type @ and tag user'
></div>
                        </td>
                    </tr>
                </table>
                <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} noRecordsTemplate={this.noRecordsTemplate}
></MentionComponent>
            </div>
        );
    }
}

```



```

    }
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    let mentionTarget = '#mentionElement';
    let dataSource = [];
    let noRecordsTemplate = "<span class='norecord'> NO DATA AVAILABLE</span>";
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag user'></div>
                </td>
            </tr>
        </table>
        <MentionComponent dataSource={dataSource} target={mentionTarget} noRecordsTemplate={noRecordsTemplate}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
function App () {
    let mentionTarget: string = '#mentionElement';
    let dataSource :[] = [];
    let noRecordsTemplate: string = "<span class='norecord'> NO DATA AVAILABLE</span>";
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag user'
                    ></div>
                    </td>
                </tr>
            </table>
            <MentionComponent dataSource={dataSource} target={mentionTarget} noRecordsTemplate={noRecordsTemplate} ></MentionComponent>
        </div>
    );
}

```

```
);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Spinner template

Display the customized waiting spinner, when data fetching takes time to load in the suggestion list by using the [spinnerTemplate](#) property.

[Class-component]

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  dataSource = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(10);
  fields = { text: 'FirstName', value: 'EmployeeID' };
  spinnerTemplate() {
    return (<React.Fragment>
      <div className="spinner_loader"></div>
    </React.Fragment>);
  }
  render() {
    return (<div id='mention_default'>
      <table>
        <tr>
          <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag user'></div>
          </td>
        </tr>
      </table>
      <MentionComponent dataSource={this.dataSource}
        target={this.mentionTarget} fields={this.fields} query={this.query}
        sortOrder={'Ascending'} spinnerTemplate={this.spinnerTemplate}
        popupWidth={'250px'}></MentionComponent>
    </div>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
```

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component<{}>, {}> {
  private mentionTarget: string = '#mentionElement';
  private dataSource: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  private query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(10);
  private fields: Object = { text: "FirstName", value: "EmployeeID" };

  private spinnerTemplate(): JSX.Element {
    return (
      <React.Fragment>
        <div className="spinner_loader"></div>
      </React.Fragment>
    );
  }
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
user'></div>
            </td>
          </tr>
        </table>
        <MentionComponent dataSource={this.dataSource}
target={this.mentionTarget} fields={this.fields} query={this.query}
sortOrder={'Ascending'} spinnerTemplate={this.spinnerTemplate}
popupWidth={'250px'}></MentionComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App() {
  let mentionTarget = '#mentionElement';
  let dataSource = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
};

```

```

let query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(10);
let fields = { text: "FirstName", value: "EmployeeID" };
function spinnerTemplate() {
    return (<React.Fragment>
        <div className="spinner_loader"></div>
    </React.Fragment>);
}
return (<div id='mention_default'>
    <table>
        <tr>
            <td>
                <label id="comment">Comments</label>
                <div id="mentionElement" placeholder='Type @ and tag
user'></div>
            </td>
        </tr>
    </table>
    <MentionComponent dataSource={dataSource} target={mentionTarget}
fields={fields} query={query} sortOrder={'Ascending'}
spinnerTemplate={spinnerTemplate} popupWidth={'250px'}></MentionComponent>
</div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from "react-dom";
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App () {
    let mentionTarget: string = '#mentionElement';
    let dataSource: 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(['FirstName',
'City','EmployeeID']).take(10);
    let fields:Object = { text: "FirstName", value: "EmployeeID" };

    function spinnerTemplate(): JSX.Element {
        return (
            <React.Fragment>
                <div className="spinner_loader"></div>
            </React.Fragment>
        );
    }
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>

```

```

        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
        </td>
      </tr>
    </table>
    <MentionComponent dataSource={dataSource} target={mentionTarget}
fields={fields} query={query} sortOrder={'Ascending'}
spinnerTemplate={spinnerTemplate} popupWidth={'250px'}></MentionComponent>
    </div>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Localization in React Mention component

The Localization library allows you to localize static text content of the [noRecordsTemplate](#) property according to the culture currently assigned to the Mention.

| Locale key | en-US (default) |

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

| noRecordsTemplate | No records found |

Loading translations

To load the translation object to your application, use the load function of the **L10n** class.

In the following sample, French culture is set to the mention component and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially.

[Class-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { L10n } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  customerData = new DataManager({
    url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  fields = { text: 'ContactName', value: 'CustomerID' };
  query = new Query().select(['ContactName', 'CustomerID']).take(0);
  componentWillMount() {
    L10n.load({
      'fr-BE': {
        'dropdowns': {
          'noRecordsTemplate': "Aucun enregistrement trouvé"
        }
      }
    });
  }
}

```

```

render() {
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                </td>
            </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.customerData} locale="fr-BE" fields={this.fields}
query={this.query}></MentionComponent>
    </div>);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
export default class App extends React.Component<{}, {}> {
    private mentionTarget: string = '#mentionElement';
    private customerData: DataManager = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    private fields: Object = { text: 'ContactName', value: 'CustomerID' };
    private query: Query = new Query().select(['ContactName',
'CustomerID']).take(0);
    public componentWillMount() {
        L10n.load({
            'fr-BE': {
                'dropdowns': {
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
        });
    }
    public render() {
        return (
            <div id='mention_default'>
                <table>
                    <tr>
                        <td>
                            <label id="comment">Comments</label>
                            <div id="mentionElement" placeholder='Type @ and tag user'
                        ></div>
                        </td>
                    </tr>
                </table>
            </div>

```

```

        </table>
        <MentionComponent target={this.mentionTarget}
        dataSource={this.customerData} locale="fr-BE" fields={this.fields}
        query={this.query} ></MentionComponent>
    </div>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { L10n } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App() {
    let mentionTarget = '#mentionElement';
    let customerData = new DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    let fields = { text: 'ContactName', value: 'CustomerID' };
    let query = new Query().select(['ContactName', 'CustomerID']).take(0);
    function componentWillMount() {
        L10n.load({
            'fr-BE': {
                'dropdowns': {
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
        });
    }
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                </td>
            </tr>
        </table>
        <MentionComponent target={mentionTarget} dataSource={customerData}
        locale="fr-BE" fields={fields} query={query}></MentionComponent>
    </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
function App () {
    let mentionTarget: string = '#mentionElement';
    let customerData: DataManager = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    });
    let fields: Object = { text: 'ContactName', value: 'CustomerID' };
    let query: Query = new Query().select(['ContactName',
'CustomerID']).take(0);
    function componentWillMount() {
        L10n.load({
            'fr-BE': {
                'dropdowns': {
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
        });
    }
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag user'
></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={mentionTarget} dataSource={customerData}
locale="fr-BE" fields={fields} query={query} ></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

See Also

- [Accessibility](#)
- [How to bind the data to the mention](#)

Customization in React Mention component

Show or hide mention character

By default, the `showMentionChar` which does not display the text content with the mentioned character is disabled. If the property `showMentionChar` is enabled, it allows the respective `mentionChar` configured along with the text content opted from the suggested list to display.

[Class-component]**INDEX.JSX**

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
export default class App extends React.Component {
    // Defines the target in which the Mention component is rendered.
    mentionTarget = '#mentionElement';
    // Defines the array of data.
    userData = [
        { Name: "Selma Rose", EmailId: "selma@gmail.com" },
        { Name: "Maria", EmailId: "maria@gmail.com" },
        { Name: "Russo kay", EmailId: "russo@gmail.com" },
        { Name: "Robert", EmailId: "robert@gmail.com" },
        { Name: "Garth", EmailId: "garth@gmail.com" }
    ];
    fields = { text: 'Name' };
    render() {
        return (<div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={this.mentionTarget}
dataSource={this.userData} fields={this.fields}
showMentionChar={true}></MentionComponent>
        </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
export default class App extends React.Component<{}, {}> {
    // Defines the target in which the Mention component is rendered.
    private mentionTarget: string = '#mentionElement';
    // Defines the array of data.
    private userData: { [key: string]: Object }[] = [
        { Name : "Selma Rose", EmailId : "selma@gmail.com"},
        { Name : "Maria", EmailId : "maria@gmail.com" },
        { Name : "Russo kay", EmailId : "russo@gmail.com" },
        { Name : "Robert", EmailId : "robert@gmail.com" },
        { Name : "Garth", EmailId : "garth@gmail.com" }
    ];
    private fields: Object = {text:'Name'};
    public render() {
        return (

```

```

        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={this.mentionTarget}
dataSource={this.userData} fields={this.fields} showMentionChar=
{true}></MentionComponent>
        </div>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {
    // Defines the target in which the Mention component is rendered.
    let mentionTarget = '#mentionElement';
    // Defines the array of data.
    let userData = [
        { Name: "Selma Rose", EmailId: "selma@gmail.com" },
        { Name: "Maria", EmailId: "maria@gmail.com" },
        { Name: "Russo kay", EmailId: "russo@gmail.com" },
        { Name: "Robert", EmailId: "robert@gmail.com" },
        { Name: "Garth", EmailId: "garth@gmail.com" }
    ];
    let fields = { text: 'Name' };
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                </td>
            </tr>
        </table>
        <MentionComponent target={mentionTarget} dataSource={userData}
fields={fields} showMentionChar={true}></MentionComponent>
        </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
function App () {
    // Defines the target in which the Mention component is rendered.
    let mentionTarget: string = '#mentionElement';
    // Defines the array of data.
    let userData: { [key: string]: Object }[] = [
        { Name : "Selma Rose", EmailId : "selma@gmail.com" },
        { Name : "Maria", EmailId : "maria@gmail.com" },
        { Name : "Russo kay", EmailId : "russo@gmail.com" },
        { Name : "Robert", EmailId : "robert@gmail.com" },
        { Name : "Garth", EmailId : "garth@gmail.com" }
    ];
    let fields: Object = {text:'Name'};
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={mentionTarget} dataSource={userData}
fields={fields}showMentionChar= {true}></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Adding the suffix character after selection

The Mention has provided support to specify the custom suffix to append alongside the mentioned selected item while inserting. You can append space or new line character as [suffixText](#).

[Class-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
    mentionTarget = '#mentionElement';
    userData = [
        { Country: "Australia", Code: "AU" },
        { Country: "Bermuda", Code: "BM" },
        { Country: "Canada", Code: "CA" },
        { Country: "Cameroon", Code: "CM" },
        { Country: "Denmark", Code: "DK" }
    ];
    fields = { text: 'Country' };
    render() {
        return (<div id='mention_default'>

```

```

        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
country'></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.userData} fields={this.fields} showMentionChar={true}
suffixText={'&#160;'}></MentionComponent>
      </div>;
    }
  }
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  private mentionTarget: string = '#mentionElement';
  private userData: { [key: string]: Object }[] = [
    { Country : "Australia", Code : "AU" },
    { Country : "Bermuda" , Code : "BM" },
    { Country : "Canada" , Code : "CA" },
    { Country : "Cameroon" , Code : "CM" },
    { Country : "Denmark" , Code : "DK" }
  ];
  private fields: Object = {text:'Country'};
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag country'
></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.userData} fields={this.fields} showMentionChar= {true}
suffixText={'&#160;'}></MentionComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
    let mentionTarget = '#mentionElement';
    let userData = [
        { Country: "Australia", Code: "AU" },
        { Country: "Bermuda", Code: "BM" },
        { Country: "Canada", Code: "CA" },
        { Country: "Cameroon", Code: "CM" },
        { Country: "Denmark", Code: "DK" }
    ];
    let fields = { text: 'Country' };
    return (<div id='mention_default'>
        <table>
            <tr>
                <td>
                    <label id="comment">Comments</label>
                    <div id="mentionElement" placeholder='Type @ and tag
country'></div>
                </td>
            </tr>
        </table>
        <MentionComponent target={mentionTarget} dataSource={userData}
fields={fields} showMentionChar={true}
suffixText={'&#160;'}></MentionComponent>
        </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App () {
    let mentionTarget: string = '#mentionElement';
    let userData: { [key: string]: Object }[] = [
        { Country : "Australia", Code : "AU" },
        { Country : "Bermuda" , Code : "BM" },
        { Country : "Canada" , Code : "CA" },
        { Country : "Cameroon" , Code : "CM" },
        { Country : "Denmark" , Code : "DK" }
    ];
    let fields: Object = {text:'Country'};
    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag country'
></div>
                    </td>

```

```

        </tr>
      </table>
      <MentionComponent target={mentionTarget} dataSource={userData}
fields={fields} showMentionChar= {true}
suffixText={'&#160;'}></MentionComponent>
    </div>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Configure the popup list

You can customize the suggestion list width and height using the [popupHeight](#) and [popupWidth](#) properties.

By default, the popup list width value is set as **auto**. Depending on the mentioned suggestion data list, the width value is automatically adjusted. The popup list height value is set as **300px**.

[Class-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  sportsData = [
    { ID: "game1", Game: "Badminton" },
    { ID: "game2", Game: "Football" },
    { ID: "game3", Game: "Tennis" },
    { ID: "game4", Game: "Hockey" },
    { ID: "game5", Game: "Basketball" },
    { ID: "game6", Game: "Cricket" }
  ];
  fields = { text: 'Game' };
  render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag
sport'></div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
dataSource={this.sportsData} fields={this.fields} popupHeight={'200px'}
popupWidth={'250px'}></MentionComponent>
      </div>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}>, {}> {
  private mentionTarget: string = '#mentionElement';
  private sportsData: {[key: string]: Object}[] = [
    { ID : "game1" ,Game : "Badminton"},
    { ID : "game2" ,Game : "Football" },
    { ID : "game3" ,Game : "Tennis"},
    { ID : "game4" ,Game : "Hockey"},
    { ID : "game5" ,Game : "Basketball"},
    { ID : "game6" ,Game : "Cricket"}
  ];
  private fields: Object = {text:'Game'};
  public render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
              <label id="comment">Comments</label>
              <div id="mentionElement" placeholder='Type @ and tag sport'
            </div>
            </td>
          </tr>
        </table>
        <MentionComponent target={this.mentionTarget}
        dataSource={this.sportsData} fields={this.fields} popupHeight={'200px'}
        popupWidth={'250px'}></MentionComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
  let mentionTarget = '#mentionElement';
  let sportsData = [
    { ID: "game1", Game: "Badminton" },
    { ID: "game2", Game: "Football" },
    { ID: "game3", Game: "Tennis" },
    { ID: "game4", Game: "Hockey" },
    { ID: "game5", Game: "Basketball" },
    { ID: "game6", Game: "Cricket" }
  ];
  let fields = { text: 'Game' };
  return (<div id='mention_default'>
    <table>
      <tr>

```

```

        <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
sport'></div>
        </td>
    </tr>
</table>
    <MentionComponent target={mentionTarget} dataSource={sportsData}
fields={fields} popupHeight={'200px'}
popupWidth={'250px'}></MentionComponent>
</div>;
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
    let mentionTarget: string = '#mentionElement';
    let sportsData: {[key: string]: Object}[] = [
        { ID : "game1" ,Game : "Badminton"},
        { ID : "game2" ,Game : "Football" },
        { ID : "game3" ,Game : "Tennis"},
        { ID : "game4" ,Game : "Hockey"},
        { ID : "game5" ,Game : "Basketball"},
        { ID : "game6" ,Game : "Cricket"}
    ];
    let fields: Object = {text:'Game'};

    return (
        <div id='mention_default'>
            <table>
                <tr>
                    <td>
                        <label id="comment">Comments</label>
                        <div id="mentionElement" placeholder='Type @ and tag sport'
></div>
                    </td>
                </tr>
            </table>
            <MentionComponent target={mentionTarget} dataSource={sportsData}
fields={fields} popupHeight={'200px'}
popupWidth={'250px'}></MentionComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Trigger character

You can customize the trigger character by using the [mentionChar](#) property in the Mention component. The trigger character triggers the suggestion list to display in the target area.

By default, the [mentionChar](#) is @.

Accessibility in React Mention component

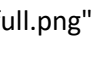
Web accessibility makes web content and web applications more accessible for people with disabilities. Mention component provides built-in compliance with WAI-ARIA specifications. The WAI-ARIA support is achieved using the attributes such as `aria-selected` and `aria-activedescendent`.

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

| Properties | Functionalities |

| --- | --- |

| aria-selected | Indicates the selected option. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

Keyboard interaction

You can use the following key shortcuts to access the Mention without interruptions.

| Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | Selects the first item in the Mention list. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Esc(Escape) | Closes the popup list when it is in an open state. |

| Enter | Selects the focused item, and when it is in an open state the popup list closes. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, inserts the selected popup list item and closes the popup list. |

[Class-component]

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
export default class App extends React.Component {
  mentionTarget = '#mentionElement';
  userData = [
    { Name: "Andrew Fuller", ID: "1" },
    { Name: "Anne Dodsworth", ID: "2" },
    { Name: "Janet Leverling", ID: "3" },
    { Name: "Laura Callahan", ID: "4" },
    { Name: "Margaret Peacock", ID: "5" }
  ];
  fields = { text: 'Name' };
  render() {
    return (
      <div id='mention_default'>
        <table>
          <tr>
            <td>
```

```

        <label id="comment">Comments</label>
        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
    </td>
</tr>
</table>
<MentionComponent target={this.mentionTarget}
dataSource={this.userData} fields={this.fields}></MentionComponent>
</div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
export default class App extends React.Component<{}, {}> {
    private mentionTarget: string = '#mentionElement';
    private userData: { [key: string]: Object }[] = [
        { Name : "Andrew Fuller", ID : "1" },
        { Name : "Anne Dodsworth", ID : "2" },
        { Name : "Janet Leverling", ID : "3" },
        { Name : "Laura Callahan", ID : "4" },
        { Name : "Margaret Peacock", ID : "5" }
    ];
    private fields: Object = { text: 'Name' }
    public render() {
        return (
            <div id='mention_default'>
                <table>
                    <tr>
                        <td>
                            <label id="comment">Comments</label>
                            <div id="mentionElement" placeholder='Type @ and tag
user' ></div>
                        </td>
                    </tr>
                </table>
                <MentionComponent target={this.mentionTarget}
dataSource={this.userData} fields={this.fields} ></MentionComponent>
            </div>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
function App() {

```

```

let mentionTarget = '#mentionElement';
let userData = [
  { Name: "Andrew Fuller", ID: "1" },
  { Name: "Anne Dodsworth", ID: "2" },
  { Name: "Janet Leverling", ID: "3" },
  { Name: "Laura Callahan", ID: "4" },
  { Name: "Margaret Peacock", ID: "5" }
];
let fields = { text: 'Name' };
return (<div id='mention_default'>
  <table>
    <tr>
      <td>
        <label id="comment">Comments</label>
        <div id="mentionElement" placeholder='Type @ and tag
user'></div>
      </td>
    </tr>
  </table>
  <MentionComponent target={mentionTarget} dataSource={userData}
fields={fields}></MentionComponent>
</div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { MentionComponent } from '@syncfusion/ej2-react-dropdowns';
function App () {
  let mentionTarget: string = '#mentionElement';
  let userData: { [key: string]: Object }[] = [
    { Name : "Andrew Fuller", ID : "1" },
    { Name : "Anne Dodsworth", ID : "2" },
    { Name : "Janet Leverling", ID : "3" },
    { Name : "Laura Callahan", ID : "4" },
    { Name : "Margaret Peacock", ID : "5" }
  ];
  let fields: Object = { text: 'Name' }
  return (
    <div id='mention_default'>
      <table>
        <tr>
          <td>
            <label id="comment">Comments</label>
            <div id="mentionElement" placeholder='Type @ and tag
user' ></div>
          </td>
        </tr>
      </table>
      <MentionComponent target={mentionTarget} dataSource={userData}
fields={fields} ></MentionComponent>
    </div>
  );
}

```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Menu

Getting Started

This section explains how to create a simple Menu, and configure its available functionalities in React.

Dependencies

The following list of dependencies are required to use the Menu component in your application.

```
`javascript
|-- @syncfusion/ej2-react-navigations
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
`,`
```

Setup your development environment

You can use [Create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

To install `create-react-app` run the following command.

```
`bash
npm install -g create-react-app
`,`
```

Start a new project using create-react-app command as follows

```
<div class='tsx'>
```

```
`bash
create-react-app quickstart --scripts-version=react-scripts-ts
cd quickstart
`
```

</div>

<div class='jsx'>

```
`bash
create-react-app quickstart
cd quickstart
`
```

</div>

'react-scripts-ts' is used for creating React app with typescript.

[Adding Syncfusion packages](#)

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry.

To install **Menu** component, use the following command

```
`bash
npm install @syncfusion/ej2-react-navigations --save
`
```

The above command installs [Menu dependencies](#) which are required to render the component in the **React** environment.

[Adding Style sheet to the Application](#)

Add Menu component's styles as given below in **App.css**.

```
`css
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
`
```

[Add Menu to the project](#)

Now, you can add **Menu** component in the application. For getting started, add **Menu** component in **src/App.tsx** file. Using the following code snippet.

```
`ts
import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
function App() {
// Menu items definition
```

```
let menuItems: MenuItemModel[] = [
{
  items: [
    { text: 'Open' },
    { text: 'Save' },
    { text: 'Exit' }
  ],
  text: 'File'
},
{
  items: [
    { text: 'Cut' },
    { text: 'Copy' },
    { text: 'Paste' }
  ],
  text: 'Edit'
},
{
  items: [
    { text: 'Toolbar' },
    { text: 'Sidebar' }
  ],
  text: 'View'
},
{
  items: [
    { text: 'Spelling & Grammar' },
    { text: 'Customize' },
    { text: 'Options' }
  ],
  text: 'Tools'
},
{ text: 'Go' },
```

```
{ text: 'Help' }  
];  
return (  
  <MenuComponent items={menuItems}/>  
);  
}  
export default App;  
`ts  
import { MenuComponent } from '@syncfusion/ej2-react-navigations';  
import * as React from 'react';  
function App() {  
  // Menu items definition  
  let menuItems = [  
    {  
      items: [  
        { text: 'Open' },  
        { text: 'Save' },  
        { text: 'Exit' }  
      ],  
      text: 'File'  
    },  
    {  
      items: [  
        { text: 'Cut' },  
        { text: 'Copy' },  
        { text: 'Paste' }  
      ],  
      text: 'Edit'  
    },  
    {  
      items: [  
        { text: 'Toolbar' },
```



```

{ text: 'Sidebar' }
],
text: 'View'
},
{
items: [
{ text: 'Spelling & Grammar' },
{ text: 'Customize' },
{ text: 'Options' }
],
text: 'Tools'
},
{ text: 'Go' },
{ text: 'Help' }
];
return (<MenuComponent items={menuItems}/>);
}
export default App;

```

Run the application

Run the application in the browser using the following command:

```
npm start
```

The following example shows a basic Menu component.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu items definition
  let menuItems = [
    {
      items: [
        { text: 'Open' },
        { text: 'Save' },

```

```

        { text: 'Exit' }
      ],
      text: 'File'
    },
    {
      items: [
        { text: 'Cut' },
        { text: 'Copy' },
        { text: 'Paste' }
      ],
      text: 'Edit'
    },
    {
      items: [
        { text: 'Toolbar' },
        { text: 'Sidebar' }
      ],
      text: 'View'
    },
    {
      items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
      ],
      text: 'Tools'
    },
    { text: 'Go' },
    { text: 'Help' }
  ];
  return (<MenuComponent items={menuItems}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu items definition
  let menuItems: MenuItemModel[] = [
    {
      items: [
        { text: 'Open' },
        { text: 'Save' },
        { text: 'Exit' }
      ],
      text: 'File'
    },
    {
      items: [

```

```

        { text: 'Cut' },
        { text: 'Copy' },
        { text: 'Paste' }
      ],
      text: 'Edit'
    },
    {
      items: [
        { text: 'Toolbar' },
        { text: 'Sidebar' }
      ],
      text: 'View'
    },
    {
      items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
      ],
      text: 'Tools'
    },
    { text: 'Go' },
    { text: 'Help' }
  ];
  return (
    <MenuComponent items={menuItems}/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

This example demonstrates the basic rendering of Menu with items support. For more information about data source support, refer to the [Data Source Binding](#) section.

See Also

- [Create menu using data source](#)
- [Customize menu items using template support](#)
- [Integrating with Toolbar component](#)

Icons and sub menu items in React Menu component

Icons

The menu item contains an icon/image in it to provide a visual representation of an action. To place the icon on a menu item, set the [iconCss](#) property with the required icon CSS. By default, the icon is positioned at the left of the menu item. In the following sample, the icons of File and Edit menu items and Open, Save, Cut, Copy, and Paste sub menu items are added using the [iconCss](#) property.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';

```

```

enableRipple(true);
function App() {
  // Menu items definition
  let menuItems = [
    {
      iconCss: 'em-icons e-file',
      items: [
        { text: 'Open', iconCss: 'em-icons e-open' },
        { text: 'Save', iconCss: 'em-icons e-save' },
        { separator: true },
        { text: 'Exit' }
      ],
      text: 'File'
    },
    {
      iconCss: 'em-icons e-edit',
      items: [
        { text: 'Cut', iconCss: 'em-icons e-cut' },
        { text: 'Copy', iconCss: 'em-icons e-copy' },
        { text: 'Paste', iconCss: 'em-icons e-paste' }
      ],
      text: 'Edit'
    },
    {
      items: [
        { text: 'Toolbar' },
        { text: 'Sidebar' },
        { text: 'Full Screen' }
      ],
      text: 'View'
    },
    {
      items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
      ],
      text: 'Tools'
    },
    { text: 'Go' },
    { text: 'Help' }
  ];
  return (<MenuComponent items={menuItems}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {

```

```
// Menu items definition
let menuItems: MenuItemModel[] = [
  {
    iconCss: 'em-icons e-file',
    items: [
      { text: 'Open', iconCss: 'em-icons e-open' },
      { text: 'Save', iconCss: 'em-icons e-save' },
      { separator: true },
      { text: 'Exit' }
    ],
    text: 'File'
  },
  {
    iconCss: 'em-icons e-edit',
    items: [
      { text: 'Cut', iconCss: 'em-icons e-cut' },
      { text: 'Copy', iconCss: 'em-icons e-copy' },
      { text: 'Paste', iconCss: 'em-icons e-paste' }
    ],
    text: 'Edit'
  },
  {
    items: [
      { text: 'Toolbar' },
      { text: 'Sidebar' },
      { text: 'Full Screen' }
    ],
    text: 'View'
  },
  {
    items: [
      { text: 'Spelling & Grammar' },
      { text: 'Customize' },
      { text: 'Options' }
    ],
    text: 'Tools'
  },
  { text: 'Go' },
  { text: 'Help' }
];

return (
  <MenuComponent items={menuItems}/>
);

export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

Navigation

Navigation in Menu is used to navigate to the other web page when a menu item is clicked. It can be achieved by providing a link to the menu item using the [url](#) property. In the following sample, the Navigation URL is added to sub menu items using the url property.

INDEX.JSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
```

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu items definition
  let menuItems = [
    {
      items: [
        { text: 'Washing Machine', url:
'https://www.google.com/search?q=washing+machine' },
        { text: 'Air Conditioners', url:
'https://www.google.com/search?q=air+conditioners' }
      ],
      text: 'Appliances'
    },
    {
      items: [
        { text: 'Headphones', url:
'https://www.google.com/search?q=headphones' },
        { text: 'Memory Cards', url:
'https://www.google.com/search?q=memory+cards' },
        { text: 'Power Banks', url:
'https://www.google.com/search?q=power+banks' }
      ],
      text: 'Mobile'
    },
    {
      items: [
        { text: 'Televisions', url:
'https://www.google.com/search?q=televisions' },
        { text: 'Home Theatres', url:
'https://www.google.com/search?q=home+theatres' },
        { text: 'Gaming Laptops', url:
'https://www.google.com/search?q=gaming+laptops' }
      ],
      text: 'Entertainment'
    },
    { text: 'Fashion', url: 'https://www.google.com/search?q=fashion' },
    { text: 'Offers', url: 'https://www.google.com/search?q=offers' }
  ];
  function beforeItemRender(args) {
    if (args.item.url) {
      args.element.getElementsByTagName('a')[0].setAttribute('target',
'_blank');
    }
    return (<MenuComponent items={menuItems}
beforeItemRender={beforeItemRender}/>);
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';

```

```

import { MenuComponent, MenuEventArgs, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    // Menu items definition
    let menuItems: MenuItemModel[] = [
        {
            items: [
                { text: 'Washing Machine', url: 'https://www.google.com/search?q=washing+machine' },
                { text: 'Air Conditioners', url: 'https://www.google.com/search?q=air+conditioners' }
            ],
            text: 'Appliances'
        },
        {
            items: [
                { text: 'Headphones', url: 'https://www.google.com/search?q=headphones' },
                { text: 'Memory Cards', url: 'https://www.google.com/search?q=memory+cards' },
                { text: 'Power Banks', url: 'https://www.google.com/search?q=power+banks' }
            ],
            text: 'Mobile'
        },
        {
            items: [
                { text: 'Televisions', url: 'https://www.google.com/search?q=televisions' },
                { text: 'Home Theatres', url: 'https://www.google.com/search?q=home+theatres' },
                { text: 'Gaming Laptops', url: 'https://www.google.com/search?q=gaming+laptops' }
            ],
            text: 'Entertainment'
        },
        { text: 'Fashion', url: 'https://www.google.com/search?q=fashion' },
        { text: 'Offers', url: 'https://www.google.com/search?q=offers' }
    ];
    function beforeItemRender(args: MenuEventArgs): void {
        if (args.item.url) {
            args.element.getElementsByTagName('a')[0].setAttribute('target', '_blank');
        }
        return (
            <MenuComponent items={menuItems}
            beforeItemRender={beforeItemRender}/>
        );
    }
    export default App;
    ReactDOM.render(<App />, document.getElementById('element'));
}

```

Multilevel nesting

The Menu supports multiple level nesting, and it can be achieved by mapping the [items](#) property inside the parent [menuItems](#). In the following sample, three-level nesting of menu has been provided.

INDEX.JSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    // Menu items definition
    let menuItems = [
        {
            items: [
                {
                    items: [
                        { text: 'Trimmers' },
                        { text: 'Shavers' }
                    ],
                    text: 'Personal Care'
                },
                {
                    items: [
                        { text: 'Shirts' },
                        { text: 'Jackets' },
                        { text: 'Track Suits' }
                    ],
                    text: 'Clothing'
                },
                { text: 'Footwear' }
            ],
            text: 'Men Fashion'
        },
        {
            items: [
                {
                    items: [
                        { text: 'Kurtas' },
                        { text: 'Salwars' },
                        { text: 'Sarees' }
                    ],
                    text: 'Clothing'
                },
                {
                    items: [
                        { text: 'Nosepins' },
                        { text: 'Anklets' }
                    ],
                    text: 'Jewellery'
                }
            ],
            text: 'Women Fashion'
        }
    ]
}
```



```

        ],
        text: 'Fashion'
    },
    {
        items: [
            {
                items: [
                    { text: 'Fully Automatic' },
                    { text: 'Semi Automatic' }
                ],
                text: 'Washing Machine'
            },
            {
                items: [
                    { text: 'Inverter ACs' },
                    { text: 'Split ACs' }
                ],
                text: 'Air Conditioners'
            }
        ],
        text: 'Home & Living'
    },
    { text: 'Accessories' },
    { text: 'Sports' },
    { text: 'Gaming' }
];
return (<MenuComponent items={menuItems}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    // Menu items definition
    let menuItems: MenuItemModel[] = [
        {
            items: [
                {
                    items: [
                        { text: 'Trimmers' },
                        { text: 'Shavers' }
                    ],
                    text: 'Personal Care'
                },
                {
                    items: [
                        { text: 'Shirts' },

```

```

        { text: 'Jackets' },
        { text: 'Track Suits' }
      ],
      text: 'Clothing'
    },
    { text: 'Footwear' }
  ],
  text: 'Men Fashion'
},
{
  items: [
    {
      items: [
        { text: 'Kurtas' },
        { text: 'Salwars' },
        { text: 'Sarees' }
      ],
      text: 'Clothing'
    },
    {
      items: [
        { text: 'Nosepins' },
        { text: 'Anklets' }
      ],
      text: 'Jewellery'
    }
  ],
  text: 'Women Fashion'
}
],
text: 'Fashion'
},
{
  items: [
    {
      items: [
        { text: 'Fully Automatic' },
        { text: 'Semi Automatic' }
      ],
      text: 'Washing Machine'
    },
    {
      items: [
        { text: 'Inverter ACs' },
        { text: 'Split ACs' }
      ],
      text: 'Air Conditioners'
    }
  ],
  text: 'Home & Living'
},
{ text: 'Accessories' },
{ text: 'Sports' },
{ text: 'Gaming' }
];
return (
  <MenuComponent items={menuItems} />

```

```

    );
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('element'));

```

You can achieve multi level nesting with data source by mapping `name` of the child items to the `children` sub-property of `fields` property. Also, we can specify `id` property for menu items. For more information, refer to the [data source binding](#) section. To open sub menu items only on item click, `showItemOnClick` should be set as `true`.

The below table represents the MenuItem properties and it's description.

Property Name | Type | Description

|iconCss|string|Defines class/multiple classes separated by a space for the menu Item that is used to include an icon. Menu Item can include font icon and sprite image.

|id|string|Specifies the id for menu item.

|separator|boolean|Specifies separator between the menu items. Separator are either horizontal or vertical lines used to group menu items.

|items|MenuItemModel[]|Specifies the sub menu items that is the array of MenuItem model/

|text|string|Specifies text for menu item.

|url|string|Specifies url for menu item that creates the anchor link to navigate to the url provided.

See Also

- [Customize menu items](#)
- [Group menu items with separator](#)

Data source binding and custom menu items in React Menu component

Data binding

The Menu supports data source bindings such as array of JavaScript objects that can be structured as either `hierarchical` or `self-referential` data.

Hierarchical data

The Menu can be populated with hierarchical data source by assigning it to the `items` property, and the fields with corresponding keys can be mapped to the `fields` property.

JSON data

The Menu can generate its menu items through an array of complex data source by mapping fields from the `fields` property.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  let dataSource = [

```

```
{
  continent: 'Asia',
  countries: [
    {
      country: 'China',
      languages: [
        { language: 'Chinese' },
        { language: 'Cantonese' }
      ]
    },
    {
      country: 'India',
      languages: [
        { language: 'English' },
        { language: 'Hindi' },
        { language: 'Tamil' }
      ]
    },
    {
      country: 'Japan',
      languages: [
        { language: 'Japanese' }
      ]
    }
  ]
},
{
  continent: 'Africa',
  countries: [
    {
      country: 'Nigeria',
      languages: [
        { language: 'English' },
        { language: 'Hausa' }
      ]
    },
    {
      country: 'Egypt',
      languages: [
        { language: 'Arabic' }
      ]
    },
    {
      country: 'South Africa',
      languages: [
        { language: 'Tswana' },
        { language: 'Swati' }
      ]
    }
  ]
},
{
  continent: 'North America',
  countries: [
    {
      country: 'Canada',
      languages: [
```

```
        { language: 'French' }
      ]
    },
    {
      country: 'Mexico',
      languages: [
        { language: 'Spanish' }
      ]
    },
    {
      country: 'USA',
      languages: [
        { language: 'English' }
      ]
    }
  ]
},
{
  continent: 'South America',
  countries: [
    {
      country: 'Brazil',
      languages: [
        { language: 'Portuguese' }
      ]
    },
    {
      country: 'Colombia',
      languages: [
        { language: 'Spanish' }
      ]
    },
    {
      country: 'Argentina',
      languages: [
        { language: 'Spanish' }
      ]
    }
  ]
},
{
  continent: 'Oceania',
  countries: [
    {
      country: 'Australia'
    },
    {
      country: 'New Zealand'
    },
    {
      country: 'Samoa'
    }
  ]
}
];
// Menu fields definition
let menuFields = {
```

```

        children: ['countries', 'languages'],
        text: ['continent', 'country', 'language']
    };
    return (<MenuComponent items={dataSource} fields={menuFields}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { FieldSettingsModel, MenuComponent, } from '@syncfusion/ej2-react-
navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let dataSource: Array<{ [key: string]: any }> = [
        {
            continent: 'Asia',
            countries: [
                {
                    country: 'China',
                    languages: [
                        { language: 'Chinese' },
                        { language: 'Cantonese' }
                    ]
                },
                {
                    country: 'India',
                    languages: [
                        { language: 'English' },
                        { language: 'Hindi' },
                        { language: 'Tamil' }
                    ]
                },
                {
                    country: 'Japan',
                    languages: [
                        { language: 'Japanese' }
                    ]
                }
            ]
        },
        {
            continent: 'Africa',
            countries: [
                {
                    country: 'Nigeria',
                    languages: [
                        { language: 'English' },
                        { language: 'Hausa' }
                    ]
                },
                {
                    country: 'Egypt',

```

```
        languages: [
          { language: 'Arabic' }
        ]
      },
      {
        country: 'South Africa',
        languages: [
          { language: 'Tswana' },
          { language: 'Swati' }
        ]
      }
    ]
  },
  {
    continent: 'North America',
    countries: [
      {
        country: 'Canada',
        languages: [
          { language: 'French' }
        ]
      },
      {
        country: 'Mexico',
        languages: [
          { language: 'Spanish' }
        ]
      },
      {
        country: 'USA',
        languages: [
          { language: 'English' }
        ]
      }
    ]
  },
  {
    continent: 'South America',
    countries: [
      {
        country: 'Brazil',
        languages: [
          { language: 'Portuguese' }
        ]
      },
      {
        country: 'Colombia',
        languages: [
          { language: 'Spanish' }
        ]
      },
      {
        country: 'Argentina',
        languages: [
          { language: 'Spanish' }
        ]
      }
    ]
  }
]
```

```

    ],
    {
      continent: 'Oceania',
      countries: [
        {
          country: 'Australia'
        },
        {
          country: 'New Zealand'
        },
        {
          country: 'Samoa'
        },
      ],
    }
  ];
  // Menu fields definition
  let menuFields: FieldSettingsModel = {
    children: ['countries', 'languages'],
    text: ['continent', 'country', 'language']
  };
  return (
    <MenuComponent items={dataSource} fields={menuFields}/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Data Service

In application level, remote data binding can be achieved using [DataManager](#). To create Menu, assign items property with resultant data from [callback](#) function.

The following example displays five employees' **FirstName** from **Employees** table and **ShipName** details from **Orders** table of the **Northwind** Data Service.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import { useState, useEffect } from "react";
import * as React from 'react';
import * as ReactDOM from 'react-dom';
const SERVICE_URI = 'https://services.odata.org/v4/Northwind/Northwind.svc/';
enableRipple(true);
function App() {
  const [menuItems, setState] = useState([]);
  // Menu fields definition.
  let menuFields = {
    children: ['Orders'],
    text: ['FirstName', 'ShipName']
  };
  useEffect(() => {
    new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor(),
      crossDomain: true })
      .executeQuery(

```



```

        new Query().from('Employees').take(5).hierarchy(
            new Query()
                .foreignKey('EmployeeID')
                .from('Orders').take(13),
            select
        ))
        .then((e) => {
            setState(e.result);
        });
    }, []);
    function select() {
        return [1, 2, 3, 4, 5];
    }
    return (
        <div>
            {
                menuItems.length ?
                <MenuComponent items={menuItems} fields={menuFields}/> : ''
            }
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query, ReturnOption } from
 '@syncfusion/ej2-data';
import { FieldSettingsModel, MenuComponent } from '@syncfusion/ej2-react-
navigations';
import { useState, useEffect } from "react";
import * as React from 'react';
import * as ReactDOM from 'react-dom';
const SERVICE_URI: string =
'https://services.odata.org/v4/Northwind/Northwind.svc/';
enableRipple(true);
function App() {
    const [menuItems, setState] = useState([]);
    // Menu fields definition.
    let menuFields: FieldSettingsModel = {
        children: ['Orders'],
        text: ['FirstName', 'ShipName']
    };
    useEffect(() => {
        new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor(),
crossDomain: true })
            .executeQuery(
                new Query().from('Employees').take(5).hierarchy(
                    new Query()
                        .foreignKey('EmployeeID')
                        .from('Orders').take(13),
                    select
                ))
            .then((e: ReturnOption) => {
                setState(e.result as any);
            });
    });
}

```

```

    });
  }, []);
  function select(): number [] {
    return [1,2,3,4,5];
  }
  return (
    <div>
      {
        menuItems.length ?
        <MenuComponent items={menuItems} fields={menuFields}/> : ''
      }
    </div>
  );
}
ReactDOM.render(<App />, document.getElementById('element'));

```

Self-referential data

Menu can be populated from self-referential data structure that contains array of JSON objects with `parentId` mapping.

In the following example, the `id`, `pId`, and `text` columns from self-referential data have been mapped to the `itemId`, `parentId`, and `text` fields, respectively.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu datasource
  let data = [
    { id: 'parent1', text: 'Events' },
    { id: 'parent2', text: 'Movies' },
    { id: 'parent3', text: 'Directory' },
    { id: 'parent4', pId: null, text: 'Queries' },
    { id: 'parent5', pId: null, text: 'Services' },
    { id: 'parent6', pId: 'parent1', text: 'Conferences' },
    { id: 'parent7', pId: 'parent1', text: 'Music' },
    { id: 'parent8', pId: 'parent1', text: 'Workshops' },
    { id: 'parent9', pId: 'parent2', text: 'Now Showing' },
    { id: 'parent10', pId: 'parent2', text: 'Coming Soon' },
    { id: 'parent10', pId: 'parent3', text: 'Media Gallery' },
    { id: 'parent11', pId: 'parent3', text: 'Newsletters' },
    { id: 'parent12', pId: 'parent4', text: 'Our Policy' },
    { id: 'parent13', pId: 'parent4', text: 'Site Map' },
    { id: 'parent14', pId: 'parent7', text: 'Pop' },
    { id: 'parent15', pId: 'parent7', text: 'Folk' },
    { id: 'parent16', pId: 'parent7', text: 'Classical' }
  ];
  // Menu fields definition
  let menuFields = {
    itemId: 'id',
    parentId: 'pId',
    text: 'text'
  }

```

```

    };
    return (<MenuComponent items={data} fields={menuFields}/>);
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { FieldSettingsModel, MenuComponent, } from '@syncfusion/ej2-react-
navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu datasource
  let data: Array<{key: string}: any > = [
    { id: 'parent1', text: 'Events' },
    { id: 'parent2', text: 'Movies' },
    { id: 'parent3', text: 'Directory' },
    { id: 'parent4', pId: null, text: 'Queries' },
    { id: 'parent5', pId: null, text: 'Services' },
    { id: 'parent6', pId: 'parent1', text: 'Conferences' },
    { id: 'parent7', pId: 'parent1', text: 'Music' },
    { id: 'parent8', pId: 'parent1', text: 'Workshops' },
    { id: 'parent9', pId: 'parent2', text: 'Now Showing' },
    { id: 'parent10', pId: 'parent2', text: 'Coming Soon' },
    { id: 'parent10', pId: 'parent3', text: 'Media Gallery' },
    { id: 'parent11', pId: 'parent3', text: 'Newsletters' },
    { id: 'parent12', pId: 'parent4', text: 'Our Policy' },
    { id: 'parent13', pId: 'parent4', text: 'Site Map' },
    { id: 'parent14', pId: 'parent7', text: 'Pop' },
    { id: 'parent15', pId: 'parent7', text: 'Folk' },
    { id: 'parent16', pId: 'parent7', text: 'Classical' }
  ];
  // Menu fields definition
  let menuFields: FieldSettingsModel = {
    itemId: 'id',
    parentId: 'pId',
    text: 'text'
  };
  return (
    <MenuComponent items={data} fields={menuFields}/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Custom menu items

The Menu can be customized using Essential JS2 **Template engine** to render the elements.

To customize menu items in your application, set your customized template string to the [template](#) property. In the following example, the menu has been rendered with customized menu items.

INDEX.JSX

```

{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(false);
function App() {
    // Menu fields definition
    let menuFields = {
        children: ['options'],
        text: ['category', 'value']
    };
    // Menu data definition
    let data = [
        {
            category: 'Products',
            options: [
                { value: 'JavaScript', url: 'javascript' },
                { value: 'Angular', url: 'angular' },
                { value: 'ASP.NET Core', url: 'core' },
                { value: 'ASP.NET MVC', url: 'mvc' }
            ]
        },
        {
            category: 'Services',
            options: [
                {
                    support: [
                        { id: 1, count: '1200+', value: 'Application Development' },
                        { id: 2, count: '3700+', value: 'Maintenance & Support' },
                        { id: 3, value: 'Quality Assurance' },
                        { id: 4, count: '900+', value: 'Cloud Integration' }
                    ]
                }
            ]
        },
        {
            category: 'About Us',
            options: [
                {
                    about: {
                        value: "We are on a mission to provide world-class best software solutions for web, mobile and desktop platforms. Around 900+ applications are designed and delivered to our customers to make digital & strengthen their businesses."
                    }
                }
            ]
        },
        { category: 'Careers' },
        { category: 'Sign In' }
    ];
    function menuTemplate(data) {
        return (data.category ? <span>{data.category}</span> :
            (data.value && data.url) ?

```

```

        <div className='e-avatar e-avatar-small image' style={{
backgroundImage:
'url(https://ej2.syncfusion.com/react/demos/src/menu/images/' + data.url +
'.png)' }}>{data.value}</div> :
        data.support ?
        <ul>
            {data.support.map((supp) => <li key={supp.id}>
                {supp.value}
                {supp.count ? <span className='e-badge e-
badge-success'>{supp.count}</span> : ""}
            </li>)}
        </ul> :
        <div tabIndex={0} className="e-card">
            <div className="e-card-header">
                <div className="e-card-header-caption">
                    <div className="e-card-header-
title">About Us</div>
                </div>
            </div>
            <div className="e-card-content">
                {data.about.value}
            </div>
            <div className="e-card-actions">
                <button className="e-btn e-outline">
                    Read More
                </button>
            </div>
        </div>);
    }
    return <div className="menu-section">
        <MenuComponent items={data} fields={menuFields}
template={menuTemplate}/>
    </div>;
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { FieldSettingsModel, MenuComponent } from '@syncfusion/ej2-react-
navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(false);
function App() {
    // Menu fields definition
    let menuFields: FieldSettingsModel = {
        children: ['options'],
        text: ['category', 'value']
    };
    // Menu data definition
    let data: Array<{ [key: string]: any }> = [
        {

```

```

        category: 'Products',
        options: [
            { value: 'JavaScript', url: 'javascript' },
            { value: 'Angular', url: 'angular' },
            { value: 'ASP.NET Core', url: 'core' },
            { value: 'ASP.NET MVC', url: 'mvc' }
        ]
    },
    {
        category: 'Services',
        options: [
            {
                support: [
                    { id: 1, count: '1200+', value: 'Application Development' },
                    { id: 2, count: '3700+', value: 'Maintenance & Support' },
                    { id: 3, value: 'Quality Assurance' },
                    { id: 4, count: '900+', value: 'Cloud Integration' }
                ]
            }
        ]
    },
    {
        category: 'About Us',
        options: [
            {
                about: {
                    value: "We are on a mission to provide world-class best software solutions for web, mobile and desktop platforms. Around 900+ applications are desgined and delivered to our customers to make digital & strengthen their businesses."
                }
            }
        ]
    },
    { category: 'Careers' },
    { category: 'Sign In' }
];

function menuTemplate(data: any): JSX.Element {
    return (
        data.category ? <span>{data.category}</span> :
        (data.value && data.url) ?
            <div className='e-avatar e-avatar-small image' style={{
                backgroundImage:
                'url(https://ej2.syncfusion.com/react/demos/src/menu/images/' + data.url +
                '.png)' }}>{data.value}</div> :
            data.support ?
                <ul>
                    {
                        data.support.map((supp: any) => <li
                            key={supp.id}>
                                {supp.value}
                                {
                                    supp.count ? <span className='e-badge
                                    e-badge-success'>{supp.count}</span> : ""
                                }
                            </li>
                    }
                </ul>
            :
    
```

```

        </li>
      }
    </ul> :
    <div tabIndex={0} className="e-card">
      <div className="e-card-header">
        <div className="e-card-header-caption">
          <div className="e-card-header-
title">About Us</div>
        </div>
      </div>
      <div className="e-card-content">
        {data.about.value}
      </div>
      <div className="e-card-actions">
        <button className="e-btn e-outline">
          Read More
        </button>
      </div>
    </div>
  );
}
return (
  <div className="menu-section">
    <MenuComponent items={data} fields={menuFields}
template={menuTemplate}/>
  </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

To prevent sub menu closing, set `args.cancel` to `true` in [beforeClose](#) event.

See Also

- [Render menu with items](#)

Use case scenarios in React Menu component

Scrollable menu

The menu component supports horizontal and vertical scrolling to render large menus and submenus in an adaptive way. This can be achieved by enabling the [enableScrolling](#) property and by restricting the corresponding menu/submenu size.

INDEX.JSX

```

import { closest, enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu items definition
  let menuItems = [

```

```
{
  items: [
    {
      items: [
        { text: 'Electric Cookers' },
        { text: 'Coffee Makers' },
        { text: 'Blenders' },
        { text: 'Microwave Ovens' }
      ],
      text: 'Kitchen'
    },
    {
      items: [
        { text: 'Our Exclusive TVs' },
        { text: 'Smart TVs' },
        { text: 'Big Screen TVs' }
      ],
      text: 'Television'
    },
    {
      text: 'Washing Machine'
    },
    {
      text: 'Refrigerators'
    },
    {
      items: [
        { text: 'Inverter ACs' },
        { text: 'Split ACs' },
        { text: 'Window ACs' }
      ],
      text: 'Air Conditioners'
    },
    {
      text: 'Water Purifiers'
    },
    {
      text: 'Air Purifiers'
    },
    {
      text: 'Chimneys'
    },
    {
      text: 'Inverters'
    },
    {
      text: 'Healthy Living'
    },
    {
      text: 'Vacuum Cleaners'
    },
    {
      text: 'Room Heaters'
    },
    {
      text: 'New Launches'
    }
  ]
}
```



```
    ],
    text: 'Appliances'
  },
  {
    items: [
      {
        items: [
          { text: 'Headphones' },
          { text: 'Memory Cards' },
          { text: 'Power Banks' },
          { text: 'Mobile Cases' },
          { text: 'Screen Protectors' }
        ],
        text: 'Mobile'
      },
      {
        text: 'Laptops'
      },
      {
        items: [
          { text: 'Pendrives' },
          { text: 'External Hard Disks' },
          { text: 'Monitors' },
          { text: 'Keyboards' }
        ],
        text: 'Desktop PC'
      },
      {
        items: [
          { text: 'Lens' },
          { text: 'Tripods' }
        ],
        text: 'Camera'
      }
    ],
    text: 'Accessories'
  },
  {
    items: [
      {
        text: 'Men'
      },
      {
        text: 'Women'
      }
    ],
    text: 'Fashion'
  },
  {
    items: [
      {
        text: 'Furniture'
      },
      {
        text: 'Decor'
      }
    ]
  }
]
```

```

        text: 'Smart Home Automation'
      },
      {
        text: 'Dining & Serving'
      }
    ],
    text: 'Home & Living'
  },
  {
    items: [
      {
        text: 'Televisions'
      },
      {
        text: 'Home Theatres'
      },
      {
        text: 'Gaming Laptops'
      }
    ],
    text: 'Entertainment'
  },
  {
    text: 'Contact Us'
  },
  {
    text: 'Help'
  }
];
function onBeforeOpen(args) {
  // Restricting sub menu wrapper height
  if (args.parentItem.text === 'Appliances') {
    closest(args.element, '.e-menu-wrapper').style.height = '230px';
  }
}
return (<MenuComponent items={menuItems} cssClass='e-scrollable-menu'
enableScrolling={true} beforeOpen={onBeforeOpen}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { closest, enableRipple } from '@syncfusion/ej2-base';
import { BeforeOpenCloseMenuEventArgs, MenuComponent, MenuItemModel } from
 '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu items definition
  let menuItems: MenuItemModel[] = [
    {
      items: [
        {
          items: [

```

```

        { text: 'Electric Cookers' },
        { text: 'Coffee Makers' },
        { text: 'Blenders' },
        { text: 'Microwave Ovens' }
      ],
      text: 'Kitchen'
    },
    {
      items: [
        { text: 'Our Exclusive TVs' },
        { text: 'Smart TVs' },
        { text: 'Big Screen TVs' }
      ],
      text: 'Television'
    },
    {
      text: 'Washing Machine'
    },
    {
      text: 'Refrigerators'
    },
    {
      items: [
        { text: 'Inverter ACs' },
        { text: 'Split ACs' },
        { text: 'Window ACs' },
      ],
      text: 'Air Conditioners'
    },
    {
      text: 'Water Purifiers'
    },
    {
      text: 'Air Purifiers'
    },
    {
      text: 'Chimneys'
    },
    {
      text: 'Inverters'
    },
    {
      text: 'Healthy Living'
    },
    {
      text: 'Vacuum Cleaners'
    },
    {
      text: 'Room Heaters'
    },
    {
      text: 'New Launches'
    }
  ],
  text: 'Appliances'
},
{

```

```
items: [
  {
    items: [
      { text: 'Headphones' },
      { text: 'Memory Cards' },
      { text: 'Power Banks' },
      { text: 'Mobile Cases' },
      { text: 'Screen Protectors' }
    ],
    text: 'Mobile'
  },
  {
    text: 'Laptops'
  },
  {
    items: [
      { text: 'Pendrives' },
      { text: 'External Hard Disks' },
      { text: 'Monitors' },
      { text: 'Keyboards' }
    ],
    text: 'Desktop PC'
  },
  {
    items: [
      { text: 'Lens' },
      { text: 'Tripods' }
    ],
    text: 'Camera'
  }
],
text: 'Accessories'
},
{
  items: [
    {
      text: 'Men'
    },
    {
      text: 'Women'
    }
  ],
  text: 'Fashion'
},
{
  items: [
    {
      text: 'Furniture'
    },
    {
      text: 'Decor'
    },
    {
      text: 'Smart Home Automation'
    },
    {
      text: 'Dining & Serving'
    }
  ]
}
```

```

    },
    text: 'Home & Living'
  },
  {
    items: [
      {
        text: 'Televisions'
      },
      {
        text: 'Home Theatres'
      },
      {
        text: 'Gaming Laptops'
      }
    ],
    text: 'Entertainment'
  },
  {
    text: 'Contact Us'
  },
  {
    text: 'Help'
  }
];
function onBeforeOpen(args: BeforeOpenCloseMenuEventArgs) {
  // Restricting sub menu wrapper height
  if (args.parentItem.text === 'Appliances') {
    (closest(args.element, '.e-menu-wrapper') as
HTMLElement).style.height = '230px';
  }
}
return (
  <MenuComponent items={menuItems} cssClass='e-scrollable-menu'
enableScrolling={true} beforeOpen={onBeforeOpen}/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Menu in toolbar

The following example demonstrates how to integrate Menu with Toolbar component.

INDEX.JSX

```

{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { ItemDirective, ItemsDirective, MenuComponent, ToolbarComponent }
from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  let tbObj;
  // Menu items definition
  let data = [

```

```

    {
      header: 'Events',
      subItems: [
        { text: 'Conferences' },
        { text: 'Music' },
        { text: 'Workshops' }
      ]
    },
    {
      header: 'Movies',
      subItems: [
        { text: 'Now Showing' },
        { text: 'Coming Soon' }
      ]
    },
    {
      header: 'Directory',
      subItems: [
        { text: 'Media Gallery' },
        { text: 'Newsletters' }
      ]
    },
    {
      header: 'Queries',
      subItems: [
        { text: 'Our Policy' },
        { text: 'Site Map' },
        { text: '24x7 Support' }
      ]
    },
    { header: 'Services' }
  ];
  let menuFields = {
    children: ['subItems', 'options'],
    text: ['header', 'text', 'value']
  };
  let animation = { effect: 'None' };
  function menuTemplate() {
    return (<MenuComponent id="menu" items={data} fields={menuFields}
animationSettings={animation}/>);
  }
  function onCreated() {
    tbObj.refreshOverflow();
  }
  return (<div className="toolbar-menu-control">
    <ToolbarComponent id="toolbar" created={onCreated} ref={ (scope)
=> { tbObj = scope; }}>
      <ItemsDirective>
        <ItemDirective template={menuTemplate}/>
        <ItemDirective prefixIcon='em-icons e-shopping-cart'
align='Right'/>
      </ItemsDirective>
    </ToolbarComponent>
  </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

```
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { FieldSettingsModel, ItemDirective, ItemsDirective, MenuComponent,
ToolbarComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  let tbObj: ToolbarComponent;
  // Menu items definition
  let data: Array<{key: string}: any > = [
    {
      header: 'Events',
      subItems: [
        { text: 'Conferences' },
        { text: 'Music' },
        { text: 'Workshops' }
      ]
    },
    {
      header: 'Movies',
      subItems: [
        { text: 'Now Showing' },
        { text: 'Coming Soon' }
      ]
    },
    {
      header: 'Directory',
      subItems: [
        { text: 'Media Gallery' },
        { text: 'Newsletters' }
      ]
    },
    {
      header: 'Queries',
      subItems: [
        { text: 'Our Policy' },
        { text: 'Site Map' },
        { text: '24x7 Support' }
      ]
    },
    { header: 'Services' }
  ];
  let menuFields: FieldSettingsModel = {
    children: ['subItems', 'options'],
    text: ['header', 'text', 'value']
  };
  let animation: any = { effect: 'None' };
  function menuTemplate(): JSX.Element {
    return (
      <MenuComponent id="menu" items={data} fields={menuFields}
animationSettings={animation} />

```

```

    );
  }
  function onCreate(): void {
    tbObj.refreshOverflow();
  }
  return (
    <div className="toolbar-menu-control">
      <ToolbarComponent id="toolbar" created={onCreated} ref={ (scope)
=> { tbObj = scope as ToolbarComponent; }} >
        <ItemsDirective>
          <ItemDirective template={menuTemplate} />
          <ItemDirective prefixIcon='em-icons e-shopping-cart'
align='Right' />
        </ItemsDirective>
      </ToolbarComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Hamburger menu

The following example demonstrates the use case of menu with Accordion component integrated in SideBar.

INDEX.JSX

```

{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { AccordionComponent, AccordionItemDirective, AccordionItemsDirective,
SidebarComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  let sidebarObj;
  let accordionObj;
  let nestedAccordionObj;
  let width = '200px';
  let type = 'Over';
  function clicked(e) {
    const target = (e.originalEvent).target;
    if (!e.item && !(target.closest('.e-acrdn-
item')).getElementsByClassName('e-tgl-collapse-icon').length) {
      sidebarObj.hide();
    }
  }
  function appliancesAccordion() {
    return ReactDOM.render(
      <AccordionComponent ref={scope => nestedAccordionObj = scope}
expanding={nestedExpand} clicked={clicked}>
        <AccordionItemsDirective>
          <AccordionItemDirective header='Kitchen' content='<div
id="Appliances_Kitchen_Items"></div>' />

```



```

        <AccordionItemDirective header='Washing Machine'
content='<div id="Appliances_Washing_Items"></div>' />
        <AccordionItemDirective header='Air Conditioners'
content='<div id="Appliances_Conditioners_Items"></div>' />
    </AccordionItemsDirective>
</AccordionComponent>,
    document.getElementById("Appliances_Items"));
}
function kitchenAccordion() {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Electric Cookers' />
                <AccordionItemDirective header='Coffee Makers' />
                <AccordionItemDirective header='Blenders' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Appliances_Kitchen_Items"));
}
function acAccordion() {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Inverter ACs' />
                <AccordionItemDirective header='Split ACs' />
                <AccordionItemDirective header='Window ACs' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Appliances_Conditioners_Items"));
}
function washingAccordian() {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Fully Automatic' />
                <AccordionItemDirective header='Semi Automatic' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Appliances_Washing_Items"));
}
function accessoriesAccordion() {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Mobile' />
                <AccordionItemDirective header='Computer' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Accessories_Items"));
}
function homeAccordion() {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Furniture' />
                <AccordionItemDirective header='Decor' />
            </AccordionItemsDirective>

```

```

        </AccordionComponent>,
        document.getElementById("Home_Living_Items"));
    }
    function fashionAccordion() {
        return ReactDOM.render(
            <AccordionComponent clicked={clicked}>
                <AccordionItemsDirective>
                    <AccordionItemDirective header='Men' />
                    <AccordionItemDirective header='Women' />
                </AccordionItemsDirective>
            </AccordionComponent>,
            document.getElementById("Fashion_Items"));
    }
    function entertainmentAccordion() {
        return ReactDOM.render(
            <AccordionComponent clicked={clicked}>
                <AccordionItemsDirective>
                    <AccordionItemDirective header='Televisions' />
                    <AccordionItemDirective header='Home Theatres' />
                    <AccordionItemDirective header='Gaming Laptops' />
                </AccordionItemsDirective>
            </AccordionComponent>,
            document.getElementById("Entertainment_Items"));
    }
    // Expanding Event function for main Accordion component.
    function expand(e) {
        if (e.isExpanded && [].indexOf.call(accordionObj.items, e.item) ===
0) {
            if ((e.element).querySelectorAll('.e-accordion').length > 0) {
                return;
            }
            appliancesAccordion();
        } else if (e.isExpanded && [].indexOf.call(accordionObj.items,
e.item) === 1) {
            if ((e.element).querySelectorAll('.e-accordion').length > 0) {
                return;
            }
            accessoriesAccordion();
        } else if (e.isExpanded && [].indexOf.call(accordionObj.items,
e.item) === 2) {
            if ((e.element).querySelectorAll('.e-accordion').length > 0) {
                return;
            }
            fashionAccordion();
        } else if (e.isExpanded && [].indexOf.call(accordionObj.items,
e.item) === 3) {
            if ((e.element).querySelectorAll('.e-accordion').length > 0) {
                return;
            }
            homeAccordion();
        } else if (e.isExpanded && [].indexOf.call(accordionObj.items,
e.item) === 4) {
            if ((e.element).querySelectorAll('.e-accordion').length > 0) {
                return;
            }
            entertainmentAccordion();
        }
    }
}

```

```

    }
    // Expanding Event function for nested Accordion component.
    function nestedExpand(e) {
        if (e.isExpanded && [].indexOf.call(nestedAccordionObj.items, e.item)
=== 0) {
            if ((e.element).querySelectorAll('.e-accordion').length > 0) {
                return;
            }
            kitchenAccordion();
        } else if (e.isExpanded && [].indexOf.call(nestedAccordionObj.items,
e.item) === 1) {
            if ((e.element).querySelectorAll('.e-accordion').length > 0) {
                return;
            }
            washingAccordian();
        } else if (e.isExpanded && [].indexOf.call(nestedAccordionObj.items,
e.item) === 2) {
            if ((e.element).querySelectorAll('.e-accordion').length > 0) {
                return;
            }
            acAccordion();
        }
    }
    function hamburgerClick() {
        sidebarObj.show();
        accordionObj.refresh();
    }
    function close() {
        sidebarObj.hide();
    }
    return (
        <div>
            <div className="header">
                <span id="hamburger" className="e-icons menu default"
onClick={hamburgerClick}/>
                <div className="content">Header content</div>
            </div>
            <SidebarComponent id='default-sidebar' width={width} type={type}
ref={scope => sidebarObj = scope}>
                <div className="title-header">
                    <div style={{display:'inline-block'}}>Menu</div>
                    <span id="close" className="e-icons" onClick={close}/>
                </div>
                <div className="content-area">
                    <AccordionComponent ref={scope => accordionObj = scope}
expanding={expand} clicked={clicked}>
                        <AccordionItemsDirective>
                            <AccordionItemDirective header='Appliances'
content='<div id="Appliances_Items"></div>' />
                            <AccordionItemDirective header='Accessories'
content='<div id="Accessories_Items"></div>' />
                            <AccordionItemDirective header='Fashion'
content='<div id="Fashion_Items"></div>' />
                            <AccordionItemDirective header='Home & Living'
content='<div id="Home_Living_Items"></div>' />
                            <AccordionItemDirective header='Entertainment'
content='<div id="Entertainment_Items"></div>' />

```

```

        </AccordionItemsDirective>
      </AccordionComponent>
    </div>
  </SidebarComponent>
  <div>
    <div className="main-content">Main content</div>
  </div>
</div>
);
}
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { AccordionClickArgs, AccordionComponent, AccordionItemDirective,
AccordionItemsDirective, ExpandEventArgs, SidebarComponent, SidebarType }
from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  let sidebarObj: SidebarComponent;
  let accordionObj: AccordionComponent;
  let nestedAccordionObj: AccordionComponent;
  let width: string = '200px';
  let type: SidebarType = 'Over';
  function clicked(e: AccordionClickArgs): void {
    const target = (e.originalEvent as Event).target as HTMLElement;
    if (!e.item && !(target.closest('.e-acrdn-item') as
HTMLElement).getElementsByClassName('e-tgl-collapse-icon').length) {
      sidebarObj.hide();
    }
  }
  function appliancesAccordion(): void {
    return ReactDOM.render(
      <AccordionComponent ref={scope => nestedAccordionObj = scope as
AccordionComponent} expanding={nestedExpand} clicked={clicked}>
        <AccordionItemsDirective>
          <AccordionItemDirective header='Kitchen' content='<div
id="Appliances_Kitchen_Items"></div>' />
          <AccordionItemDirective header='Washing Machine'
content='<div id="Appliances_Washing_Items"></div>' />
          <AccordionItemDirective header='Air Conditioners'
content='<div id="Appliances_Conditioners_Items"></div>' />
        </AccordionItemsDirective>
      </AccordionComponent>,
      document.getElementById("Appliances_Items"));
  }
  function kitchenAccordion(): void {
    return ReactDOM.render(
      <AccordionComponent clicked={clicked}>
        <AccordionItemsDirective>
          <AccordionItemDirective header='Electric Cookers' />

```

```

        <AccordionItemDirective header='Coffee Makers' />
        <AccordionItemDirective header='Blenders' />
    </AccordionItemsDirective>
</AccordionComponent>,
document.getElementById("Appliances_Kitchen_Items"));
}
function acAccordion(): void {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Inverter ACs' />
                <AccordionItemDirective header='Split ACs' />
                <AccordionItemDirective header='Window ACs' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Appliances_Conditioners_Items"));
}
function washingAccordion(): void {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Fully Automatic' />
                <AccordionItemDirective header='Semi Automatic' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Appliances_Washing_Items"));
}
function accessoriesAccordion(): void {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Mobile' />
                <AccordionItemDirective header='Computer' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Accessories_Items"));
}
function homeAccordion(): void {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Furniture' />
                <AccordionItemDirective header='Decor' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Home_Living_Items"));
}
function fashionAccordion(): void {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Men' />
                <AccordionItemDirective header='Women' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Fashion_Items"));
}

```

```

function entertainmentAccordion(): void {
    return ReactDOM.render(
        <AccordionComponent clicked={clicked}>
            <AccordionItemsDirective>
                <AccordionItemDirective header='Televisions' />
                <AccordionItemDirective header='Home Theatres' />
                <AccordionItemDirective header='Gaming Laptops' />
            </AccordionItemsDirective>
        </AccordionComponent>,
        document.getElementById("Entertainment_Items"));
}
// Expanding Event function for main Accordion component.
function expand(e: ExpandEventArgs): void {
    if (e.isExpanded && [].indexOf.call(accordionObj.items, e.item) ===
0) {
        if ((e.element as Element).querySelectorAll('.e-
accordion').length > 0) {
            return;
        }
        appliancesAccordion();
    } else if (e.isExpanded && [].indexOf.call(accordionObj.items,
e.item) === 1) {
        if ((e.element as Element).querySelectorAll('.e-
accordion').length > 0) {
            return;
        }
        accessoriesAccordion();
    } else if (e.isExpanded && [].indexOf.call(accordionObj.items,
e.item) === 2) {
        if ((e.element as Element).querySelectorAll('.e-
accordion').length > 0) {
            return;
        }
        fashionAccordion();
    } else if (e.isExpanded && [].indexOf.call(accordionObj.items,
e.item) === 3) {
        if ((e.element as Element).querySelectorAll('.e-
accordion').length > 0) {
            return;
        }
        homeAccordion();
    } else if (e.isExpanded && [].indexOf.call(accordionObj.items,
e.item) === 4) {
        if ((e.element as Element).querySelectorAll('.e-
accordion').length > 0) {
            return;
        }
        entertainmentAccordion();
    }
}
// Expanding Event function for nested Accordion component.
function nestedExpand(e: ExpandEventArgs): void {
    if (e.isExpanded && [].indexOf.call(nestedAccordionObj.items, e.item)
=== 0) {
        if ((e.element as Element).querySelectorAll('.e-
accordion').length > 0) {
            return;

```

```

        }
        kitchenAccordion();
    } else if (e.isExpanded && [].indexOf.call(nestedAccordionObj.items,
e.item) === 1) {
        if ((e.element as Element).querySelectorAll('.e-
accordion').length > 0) {
            return;
        }
        washingAccordian();
    } else if (e.isExpanded && [].indexOf.call(nestedAccordionObj.items,
e.item) === 2) {
        if ((e.element as Element).querySelectorAll('.e-
accordion').length > 0) {
            return;
        }
        acAccordion();
    }
}
function hamburgerClick(): void {
    sidebarObj.show();
    accordionObj.refresh();
}
function close(): void {
    sidebarObj.hide();
}
return (
    <div>
        <div className="header">
            <span id="hamburger" className="e-icons menu default"
onClick={hamburgerClick}/>
            <div className="content">Header content</div>
        </div>
        <SidebarComponent id='default-sidebar' width={width} type={type}
ref={scope => sidebarObj = scope as SidebarComponent}>
            <div className="title-header">
                <div style={{display: 'inline-block'}}>Menu</div>
                <span id="close" className="e-icons" onClick={close}/>
            </div>
            <div className="content-area">
                <AccordionComponent ref={scope => accordionObj = scope as
AccordionComponent} expanding={expand} clicked={clicked}>
                    <AccordionItemsDirective>
                        <AccordionItemDirective header='Appliances'
content='<div id="Appliances_Items"></div>' />
                        <AccordionItemDirective header='Accessories'
content='<div id="Accessories_Items"></div>' />
                        <AccordionItemDirective header='Fashion'
content='<div id="Fashion_Items"></div>' />
                        <AccordionItemDirective header='Home & Living'
content='<div id="Home_Living_Items"></div>' />
                        <AccordionItemDirective header='Entertainment'
content='<div id="Entertainment_Items"></div>' />
                    </AccordionItemsDirective>
                </AccordionComponent>
            </div>
        </SidebarComponent>
    </div>

```

```

        <div className="main-content">Main content</div>
      </div>
    </div>
  );
}
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Mobile view

The following example demonstrates the use case of Menu in Mobile mode by using ListView component with hamburger.

INDEX.JSX

```

import { Animation, enableRipple } from '@syncfusion/ej2-base';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  let listObj;
  let listViewDisplay = { display: 'none' };
  let closeSpanDisplay = { display: 'none' };
  let dataSource = [
    {
      child: [
        {
          child: [
            { id: 'list1_1_1', text: 'Electric Cookers' },
            { id: 'list1_1_2', text: 'Coffee Makers' },
            { id: 'list1_1_3', text: 'Blenders' },
          ],
          id: 'list1_1',
          text: 'Kitchen'
        },
        {
          child: [
            { id: 'list1_2_1', text: 'Fully Automatic' },
            { id: 'list1_2_2', text: 'Semi Automatic' }
          ],
          id: 'list1_2',
          text: 'Washing Machine'
        },
        {
          child: [
            { id: 'list1_3_1', text: 'Inverter ACs' },
            { id: 'list1_3_2', text: 'Split ACs' },
            { id: 'list1_3_3', text: 'Window ACs' },
          ],
          id: 'list1_3',
          text: 'Air Conditioners'
        }
      ],
      id: 'list1',
      text: 'Appliances'
    },
  ],

```



```

    {
      child: [
        {
          child: [
            { id: 'list2_1_1', text: 'Headphones' },
            { id: 'list2_1_2', text: 'Memory Cards' },
            { id: 'list2_1_3', text: 'Power Banks' }
          ],
          id: 'list2_1',
          text: 'Mobile'
        },
        {
          child: [
            { id: 'list2_2_1', text: 'Pendrives' },
            { id: 'list2_2_2', text: 'External Hard Disks' },
            { id: 'list2_2_3', text: 'Monitors' }
          ],
          id: 'list2_2',
          text: 'Computer'
        }
      ],
      id: 'list2',
      text: 'Accessories'
    },
    {
      child: [
        { id: 'list3_1', text: 'Men' },
        { id: 'list3_2', text: 'Women' }
      ],
      id: 'list3',
      text: 'Fashion'
    },
    {
      child: [
        { id: 'list4_1', text: 'Furniture' },
        { id: 'list4_2', text: 'Decor' }
      ],
      id: 'list4',
      text: 'Home & Living'
    },
    {
      child: [
        { id: 'list5_1', text: 'Televisions' },
        { id: 'list5_2', text: 'Home Theatres' },
        { id: 'list5_3', text: 'Gaming Laptops' }
      ],
      id: 'list5',
      text: 'Entertainment'
    }
  ];
  function onSelect(e) {
    if (e.data && !e.data.child) {
      listViewDisplay = { display: 'none' };
      closeSpanDisplay = { display: 'none' };
      listObj.refresh();
    }
  }
}

```

```

function onClick() {
  listViewDisplay = { display: 'none' };
  closeSpanDisplay = { display: 'none' };
}
function hamburgerClick() {
  const animation = new Animation({ duration: 500 });
  animation.animate(listObj.element, {
    begin: () => {
      listViewDisplay = { display: 'block' };
      closeSpanDisplay = { display: 'block' };
    },
    name: 'SlideDown'
  });
}
return (<div className='layoutWrapper'>
  <div className='speaker'>
    <div className='camera' />
  </div>
  <div className='layout'>
    <div id='container'>
      <div id='header'>
        <span id='hamburger' onClick={hamburgerClick}
className='e-icons menu default' />
        <div className='content'>Header</div>
      </div>
      <ListViewComponent id='listview' ref={scope => listObj =
scope} dataSource={dataSource} showHeader={true} headerTitle='Menu'
select={onSelect} />
      <span id='close' onClick={onClick} className='e-icons'
style={closeSpanDisplay} />
    </div>
  </div>
  <div className='outerButton' />
</div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { Animation, enableRipple } from '@syncfusion/ej2-base';
import { ListViewComponent, SelectEventArgs } from '@syncfusion/ej2-react-
lists';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  let listObj: ListViewComponent;
  let listViewDisplay: any = { display: 'none' };
  let closeSpanDisplay: any = { display: 'none' };
  let dataSource: Array<{ [key: string]: any }> = [
    {
      child: [
        {
          child: [
            { id: 'list1_1_1', text: 'Electric Cookers' },

```

```

        { id: 'list1_1_2', text: 'Coffee Makers' },
        { id: 'list1_1_3', text: 'Blenders' },
      ],
      id: 'list1_1',
      text: 'Kitchen'
    },
    {
      child: [
        { id: 'list1_2_1', text: 'Fully Automatic' },
        { id: 'list1_2_2', text: 'Semi Automatic' }
      ],
      id: 'list1_2',
      text: 'Washing Machine'
    },
    {
      child: [
        { id: 'list1_3_1', text: 'Inverter ACs' },
        { id: 'list1_3_2', text: 'Split ACs' },
        { id: 'list1_3_3', text: 'Window ACs' },
      ],
      id: 'list1_3',
      text: 'Air Conditioners'
    }
  ],
  id: 'list1',
  text: 'Appliances'
},
{
  child: [
    {
      child: [
        { id: 'list2_1_1', text: 'Headphones' },
        { id: 'list2_1_2', text: 'Memory Cards' },
        { id: 'list2_1_3', text: 'Power Banks' }
      ],
      id: 'list2_1',
      text: 'Mobile'
    },
    {
      child: [
        { id: 'list2_2_1', text: 'Pendrives' },
        { id: 'list2_2_2', text: 'External Hard Disks' },
        { id: 'list2_2_3', text: 'Monitors' }
      ],
      id: 'list2_2',
      text: 'Computer'
    }
  ],
  id: 'list2',
  text: 'Accessories'
},
{
  child: [
    { id: 'list3_1', text: 'Men' },
    { id: 'list3_2', text: 'Women' }
  ],
  id: 'list3',
  text: 'Fashion'
},

```

```

    {
      child: [
        { id: 'list4_1', text: 'Furniture' },
        { id: 'list4_2', text: 'Decor' }
      ],
      id: 'list4',
      text: 'Home & Living'
    },
    {
      child: [
        { id: 'list5_1', text: 'Televisions' },
        { id: 'list5_2', text: 'Home Theatres' },
        { id: 'list5_3', text: 'Gaming Laptops' }
      ],
      id: 'list5',
      text: 'Entertainment'
    }
  ];
  function onSelect(e: SelectEventArgs): void {
    if (e.data && !(e.data as { child: object }).child) {
      listViewDisplay = { display: 'none' };
      closeSpanDisplay = { display: 'none' };
      listObj.refresh();
    }
  }
  function onClick(): void {
    listViewDisplay = { display: 'none' };
    closeSpanDisplay = { display: 'none' };
  }
  function hamburgerClick(): void {
    const animation = new Animation({ duration: 500 });
    animation.animate(listObj.element, {
      begin: () => {
        listViewDisplay = { display: 'block' };
        closeSpanDisplay = { display: 'block' };
      },
      name: 'SlideDown'
    });
  }
  return (
    <div className='layoutWrapper'>
      <div className='speaker'>
        <div className='camera' />
      </div>
      <div className='layout'>
        <div id='container'>
          <div id='header'>
            <span id='hamburger' onClick={hamburgerClick}
className='e-icons menu default' />
            <div className='content'>Header</div>
            </div>
            <ListViewComponent id='listview' ref={scope => listObj =
scope as ListViewComponent} dataSource={dataSource} showHeader={true}
headerTitle='Menu' select={onSelect} />
            <span id='close' onClick={onClick} className='e-icons'
style={closeSpanDisplay} />
          </div>
        </div>
      </div>
    </div>
  );

```

```

        </div>
        <div className='outerButton' />
      </div>
    );
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('element'));

```

Accessibility in React Menu component

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

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates Menu component's root menu as **menubar**, popup as **menu**, and the popup items as **menuitem**. |

| **aria-haspopup** | Indicates the availability and type of interactive popup element. |

| **aria-expanded** | Indicates whether the subtree can be expanded or collapsed, and indicates whether its current state can be expanded or collapsed. |

| **aria-orientation** | Indicates whether the orientation is horizontal or vertical. The default orientation is horizontal. |

| **aria-label** | Indicates the menu item text. |

| **aria-disabled** | Indicates the state of menu item whether it is disabled. |

Keyboard interaction

The Menu 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 Menu component.

| Press | To do this |

| --- | --- |

| **Esc** | Closes the sub menu that contains focus and returns focus to the parent element. |

| **Enter** | Opens the sub menu if focused menu item has sub menu, and places focus on its first item or activates the item and closes the sub menu. |

| **Up** | Navigates up or to the previous menu item. |

| **Down** | Navigates down or to the next menu item. |

| **Left** | Closes the current sub menu and navigates to the parent menu. |

| **Right** | Navigates and open the next sub menu. |

| **Home** | Focuses the first item. |

| **End** | Focuses the last item. |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Style and appearance in React Menu component

To modify the Menu appearance, you need to override the default CSS of Menu component. Please find the list of CSS classes and its corresponding section in Menu component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

.e-menu-wrapper	To customize the menu wrapper
.e-menu-wrapper.e-menu-popup	To customize the menu popup wrapper
.e-menu-wrapper ul .e-menu-item	To customize the menu items
.e-menu-wrapper.e-menu-popup .e-menu-item	To customize the menu popup items
.e-menu-wrapper ul .e-menu-item .e-caret	To customize the menu items caret icon

How To

Render with separator in React Menu component

The separators are both horizontal and vertical lines used to separate the menu items. You cannot select the separators, but you can enable separators to group the menu items using the [separator](#) property.

The **Open** and **Save** sub menu items are grouped using the **separator** property in the following sample.

INDEX.JSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    // Menu items definition
    let menuItems = [
        {
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { separator: true },
                { text: 'Exit' }
            ],
            text: 'File'
        },
        {
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }
            ],
            text: 'Edit'
        }
    ];
```

```

    },
    {
      items: [
        { text: 'Toolbar' },
        { text: 'Sidebar' },
        { text: 'Full Screen' }
      ],
      text: 'View'
    },
    {
      items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
      ],
      text: 'Tools'
    },
    { text: 'Go' },
    { text: 'Help' }
  ];
  return (<MenuComponent items={menuItems}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu items definition
  let menuItems: MenuItemModel[] = [
    {
      items: [
        { text: 'Open' },
        { text: 'Save' },
        { separator: true },
        { text: 'Exit' }
      ],
      text: 'File'
    },
    {
      items: [
        { text: 'Cut' },
        { text: 'Copy' },
        { text: 'Paste' }
      ],
      text: 'Edit'
    },
    {
      items: [
        { text: 'Toolbar' },

```



```

        { text: 'Sidebar' },
        { text: 'Full Screen' }
    ],
    text: 'View'
},
{
    items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
    ],
    text: 'Tools'
},
{ text: 'Go' },
{ text: 'Help' }
];
return (
    <MenuComponent items={menuItems}/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

You can also enable the separator to group **horizontal** menu items.

Change orientation in React Menu component

Orientation in menu items can be changed horizontally or vertically using the [orientation](#) property. By default, it is horizontally aligned.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuItems = [
        {
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { text: 'Exit' }
            ],
            text: 'File'
        },
        {
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }
            ],
            text: 'Edit'
        },
        {
            items: [

```

```

        { text: 'Toolbar' },
        { text: 'Sidebar' },
        { text: 'Full Screen' }
    ],
    text: 'View'
},
{
    items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
    ],
    text: 'Tools'
},
{ text: 'Help' }
];
return (<MenuComponent items={menuItems} orientation="Vertical"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuItems: MenuItemModel[] = [
        {
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { text: 'Exit' }
            ],
            text: 'File'
        },
        {
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }
            ],
            text: 'Edit'
        },
        {
            items: [
                { text: 'Toolbar' },
                { text: 'Sidebar' },
                { text: 'Full Screen' }
            ],
            text: 'View'
        },
        {

```

```

        items: [
            { text: 'Spelling & Grammar' },
            { text: 'Customize' },
            { text: 'Options' }
        ],
        text: 'Tools'
    },
    { text: 'Help' }
];
return (
    <MenuComponent items={menuItems} orientation="Vertical"/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Customize menu using events in React Menu component

The Menu provides a set of [events](#) to enable users to customize it.

The available events are [beforeOpen](#), [beforeClose](#), [onClose](#), [onOpen](#), and [select](#).

INDEX.JSX

```

{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import { useState } from "react";
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuItems = [
        {
            items: [
                { text: 'Conferences' },
                { text: 'Music' },
                { text: 'Workshops' }
            ],
            text: 'Events'
        },
        {
            items: [
                { text: 'Now Showing' },
                { text: 'Coming Soon' }
            ],
            text: 'Movies'
        },
        {
            items: [
                { text: 'Media Gallery' },
                { text: 'Newsletters' }
            ],
            text: 'Directory'
        },
        {

```

```

        items: [
            { text: 'Our Policy' },
            { text: 'Site Map' }
        ],
        text: 'Queries'
    },
    { text: 'Services' }
];
const [state, setState] = useState({
    eventTrace: ''
});
function beforeOpen(args) {
    updateEventLog(args);
}
function beforeClose(args) {
    updateEventLog(args);
}
function onClose(args) {
    updateEventLog(args);
}
function onOpen(args) {
    updateEventLog(args);
}
function select(args) {
    updateEventLog(args);
}
function updateEventLog(args) {
    setState({ eventTrace: state.eventTrace + args.name + ' Event
triggered. <br />' });
}
function btnClick() {
    setState({ eventTrace: '' });
}
return (<div className='control-section'>
    <div className='menu-section'>
        <MenuComponent id='menu' items={menuItems}
beforeOpen={beforeOpen} beforeClose={beforeClose} onClose={onClose}
onOpen={onOpen} select={select}/>
    </div>
    <div className='property-section'>
        <table id='propertyTable' title='Event trace'>
            <tbody>
                <th>Event trace:-</th>
                <tr>
                    <td dangerouslySetInnerHTML={{ __html:
state.eventTrace }}/>
                </tr>
            </tbody>
        </table>
    </div>
    <ButtonComponent id='clear' cssClass='e-small' content='Clear'
onClick={btnClick}/>
</div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { BeforeOpenCloseMenuEventArgs, MenuComponent, MenuEventArgs,
MenuItemModel, OpenCloseMenuEventArgs } from '@syncfusion/ej2-react-
navigations';
import * as React from 'react';
import { useState } from "react";
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuItems: MenuItemModel[] = [
        {
            items: [
                { text: 'Conferences' },
                { text: 'Music' },
                { text: 'Workshops' }
            ],
            text: 'Events'
        },
        {
            items: [
                { text: 'Now Showing' },
                { text: 'Coming Soon' }
            ],
            text: 'Movies'
        },
        {
            items: [
                { text: 'Media Gallery' },
                { text: 'Newsletters' }
            ],
            text: 'Directory'
        },
        {
            items: [
                { text: 'Our Policy' },
                { text: 'Site Map' }
            ],
            text: 'Queries'
        },
        { text: 'Services' }
    ];
    const [state, setState] = useState({
        eventTrace: ''
    });
    function beforeOpen(args: BeforeOpenCloseMenuEventArgs): void {
        updateEventLog(args);
    }
    function beforeClose(args: BeforeOpenCloseMenuEventArgs): void {
        updateEventLog(args);
    }
    function onClose(args: OpenCloseMenuEventArgs): void {

```

```

        updateEventLog(args);
    }
    function onOpen(args: OpenCloseMenuEventArgs): void {
        updateEventLog(args);
    }
    function select(args: MenuEventArgs): void {
        updateEventLog(args);
    }
    function updateEventLog(args: any): void {
        setState({ eventTrace: state.eventTrace + args.name + ' Event
triggered. <br />' });
    }
    function btnClick(): void {
        setState({ eventTrace: '' });
    }
    return (
        <div className='control-section'>
            <div className='menu-section'>
                <MenuComponent id='menu' items={menuItems}
beforeOpen={beforeOpen} beforeClose={beforeClose} onClose={onClose}
onOpen={onOpen} select={select}/>
            </div>
            <div className='property-section'>
                <table id='propertyTable' title='Event trace'>
                    <tbody>
                        <th>Event trace:-</th>
                        <tr>
                            <td dangerouslySetInnerHTML={{__html:
state.eventTrace}}/>
                        </tr>
                    </tbody>
                </table>
            </div>
            <ButtonComponent id='clear' cssClass='e-small' content='Clear'
onClick={btnClick}/>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
[% endraw %]

```

Customize menu items in React Menu component

Add or remove menu items

Menu items can be added or removed using the [insertAfter](#), [insertBefore](#) and [removeItems](#) methods.

In the following example, the **Europe** menu items are added before the **Oceania** item, the **Africa** menu items are added after the **Asia**, and the **South America** and **Mexico** items are removed from menu.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);

```

```
function App() {
  let menuObj;
  // Menu data source
  let data = [
    {
      continent: 'Asia',
      countries: [
        { country: 'China' },
        { country: 'India' },
        { country: 'Japan' }
      ]
    },
    {
      continent: 'North America',
      countries: [
        { country: 'Canada' },
        { country: 'Mexico' },
        { country: 'USA' }
      ]
    },
    {
      continent: 'South America',
      countries: [
        { country: 'Brazil' },
        { country: 'Colombia' },
        { country: 'Argentina' }
      ]
    },
    {
      continent: 'Oceania',
      countries: [
        { country: 'Australia' },
        { country: 'New Zealand' },
        { country: 'Samoa' }
      ]
    },
    { continent: 'Antarctica' }
  ];
  // Menu fields definition
  let menuFields = {
    children: ['countries'],
    text: ['continent', 'country'],
  };
  function created() {
    let insertItem = [
      {
        continent: 'Europe',
        countries: [
          { country: 'Finland' },
          { country: 'Austria' }
        ]
      }
    ];
    // Add items before to 'Oceania'
    menuObj.insertBefore(insertItem, 'Oceania', false);
    insertItem = [
      {
```

```

        continent: 'Africa',
        countries: [
            { country: 'Nigeria' }
        ]
    }
];
// Add items after to 'Asia'
menuObj.insertAfter(insertItem, 'Asia', false);
// Remove items
menuObj.removeItem(['South America', 'Mexico'], false);
}
return (<MenuComponent ref={scope => menuObj = scope} items={data}
fields={menuFields} created={created}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { FieldSettingsModel, MenuComponent } from '@syncfusion/ej2-react-
navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuObj: MenuComponent;
    // Menu data source
    let data: Array<{ [key: string]: any }> = [
        {
            continent: 'Asia',
            countries: [
                { country: 'China' },
                { country: 'India' },
                { country: 'Japan' }
            ]
        },
        {
            continent: 'North America',
            countries: [
                { country: 'Canada' },
                { country: 'Mexico' },
                { country: 'USA' }
            ]
        },
        {
            continent: 'South America',
            countries: [
                { country: 'Brazil' },
                { country: 'Colombia' },
                { country: 'Argentina' }
            ]
        },
        {
            continent: 'Oceania',
            countries: [

```



```

        { country: 'Australia' },
        { country: 'New Zealand' },
        { country: 'Samoa' },
    ]
},
{ continent: 'Antarctica' }
];
// Menu fields definition
let menuFields: FieldSettingsModel = {
    children: ['countries'],
    text: ['continent', 'country'],
};
function created(): void {
    let insertItem: Array<{ [key: string]: any }> = [
        {
            continent: 'Europe',
            countries: [
                { country: 'Finland' },
                { country: 'Austria' }
            ]
        }
    ];
    // Add items before to 'Oceania'
    menuObj.insertBefore(insertItem, 'Oceania', false);
    insertItem = [
        {
            continent: 'Africa',
            countries: [
                { country: 'Nigeria' }
            ]
        }
    ];
    // Add items after to 'Asia'
    menuObj.insertAfter(insertItem, 'Asia', false);
    // Remove items
    menuObj.removeItem(['South America', 'Mexico'], false);
}
return (
    <MenuComponent ref={scope => menuObj = scope} items={data}
    fields={menuFields} created={created}/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

To process items with ID values, set `isUnique` to `true`.

Enable or disable menu items

You can enable and disable the menu items using the [enableItems](#) method in Menu. To enable menu items, set the `enable` property in argument to `true` and vice-versa.

In the following example, the **Directory** header item, **Conferences**, and **Music** sub menu items are disabled.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuObj;
    // Menu items definition
    let menuItems = [
        {
            items: [
                { text: 'Conferences' },
                { text: 'Music' },
                { text: 'Workshops' }
            ],
            text: 'Events'
        },
        {
            items: [
                { text: 'Now Showing' },
                { text: 'Coming Soon' }
            ],
            text: 'Movies'
        },
        {
            items: [
                { text: 'Media Gallery' },
                { text: 'Newsletters' }
            ],
            text: 'Directory'
        },
        {
            items: [
                { text: 'Our Policy' },
                { text: 'Site Map' }
            ],
            text: 'Queries'
        },
        { text: 'Services' }
    ];
    let disableItems = ['Conferences', 'Music', 'Directory'];
    function beforeOpen(args) {
        // Handling sub menu items
        for (const item of args.items) {
            if (disableItems.indexOf(item.text) > -1) {
                menuObj.enableItems([item.text], false, false);
            }
        }
    }
    function created() {
        // Disable items
        menuObj.enableItems(disableItems, false, false);
    }
    function btnClick() {
        // Enable items
        menuObj.enableItems(disableItems, true, false);
    }
}

```

```

        disableItems = [];
    }
    return (<div className="control-section">
        <ButtonComponent id='enable' content='Enable all items'
onClick={btnClick}/>
        <div className="menu-section">
            <MenuComponent ref={scope => menuObj = scope}
items={menuItems} beforeOpen={beforeOpen} created={created}/>
        </div>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { BeforeOpenCloseMenuEventArgs, MenuComponent, MenuItemModel } from
 '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuObj: MenuComponent;
    // Menu items definition
    let menuItems: MenuItemModel[] = [
        {
            items: [
                { text: 'Conferences' },
                { text: 'Music' },
                { text: 'Workshops' }
            ],
            text: 'Events'
        },
        {
            items: [
                { text: 'Now Showing' },
                { text: 'Coming Soon' }
            ],
            text: 'Movies'
        },
        {
            items: [
                { text: 'Media Gallery' },
                { text: 'Newsletters' }
            ],
            text: 'Directory'
        },
        {
            items: [
                { text: 'Our Policy' },
                { text: 'Site Map' }
            ],
            text: 'Queries'
        }
    ],

```

```

        { text: 'Services' }
    ];
    let disableItems: string[] = ['Conferences', 'Music', 'Directory'];
    function beforeOpen(args: BeforeOpenCloseMenuEventArgs): void {
        // Handling sub menu items
        for (const item of args.items) {
            if (disableItems.indexOf(item.text) > -1) {
                menuObj.enableItems([item.text], false, false);
            }
        }
    }
    function created(): void {
        // Disable items
        menuObj.enableItems(disableItems, false, false);
    }
    function btnClick(): void {
        // Enable items
        menuObj.enableItems(disableItems, true, false);
        disableItems = [];
    }
    return (
        <div className="control-section">
            <ButtonComponent id='enable' content='Enable all items'
onClick={btnClick}/>
            <div className="menu-section">
                <MenuComponent ref={scope => menuObj = scope}
items={menuItems} beforeOpen={beforeOpen} created={created}/>
            </div>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

To disable sub menu items, use the [beforeOpen](#) event.

Hide or show menu items

You can show or hide the menu items using the [showItems](#) and [hideItems](#) methods.

In the following example, the **Movies** header item, **Workshops**, and **Music** sub menu items are hidden in menu.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuObj;
    let menuItems = [
        {
            items: [
                { text: 'Conferences' },

```

```

        { text: 'Music' },
        { text: 'Workshops' }
      ],
      text: 'Events'
    },
    {
      items: [
        { text: 'Now Showing' },
        { text: 'Coming Soon' }
      ],
      text: 'Movies'
    },
    {
      items: [
        { text: 'Media Gallery' },
        { text: 'Newsletters' }
      ],
      text: 'Directory'
    },
    {
      items: [
        { text: 'Our Policy' },
        { text: 'Site Map' }
      ],
      text: 'Queries'
    },
    { text: 'Services' }
  ];
  let hiddenItems = ['Workshops', 'Music', 'Movies'];
  function beforeOpen(args) {
    // Handling sub menu items
    for (const item of args.items) {
      if (hiddenItems.indexOf(item.text) > -1) {
        menuObj.hideItems([item.text], false);
      }
    }
  }
  function created() {
    // Disable items
    menuObj.hideItems(hiddenItems, false);
  }
  function btnClick() {
    // show items
    menuObj.showItems(hiddenItems, false);
    hiddenItems = [];
  }
  return (
    <div className="control-section">
      <ButtonComponent content='Show all items' onClick={btnClick}/>
      <div className="menu-section">
        <MenuComponent ref={scope => menuObj = scope}
items={menuItems} beforeOpen={beforeOpen} created={created}/>
      </div>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { BeforeOpenCloseMenuEventArgs, MenuComponent, MenuItemModel } from
 '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuObj: MenuComponent;
    let menuItems: MenuItemModel[] = [
        {
            items: [
                { text: 'Conferences' },
                { text: 'Music' },
                { text: 'Workshops' }
            ],
            text: 'Events'
        },
        {
            items: [
                { text: 'Now Showing' },
                { text: 'Coming Soon' }
            ],
            text: 'Movies'
        },
        {
            items: [
                { text: 'Media Gallery' },
                { text: 'Newsletters' }
            ],
            text: 'Directory'
        },
        {
            items: [
                { text: 'Our Policy' },
                { text: 'Site Map' }
            ],
            text: 'Queries'
        },
        { text: 'Services' }
    ];
    let hiddenItems: string[] = ['Workshops', 'Music', 'Movies'];
    function beforeOpen(args: BeforeOpenCloseMenuEventArgs): void {
        // Handling sub menu items
        for (const item of args.items) {
            if (hiddenItems.indexOf(item.text) > -1) {
                menuObj.hideItems([item.text], false);
            }
        }
    }
    function created(): void {
        // Disable items
        menuObj.hideItems(hiddenItems, false);
    }
}

```

```

    }
    function btnClick(): void {
        // show items
        menuObj.showItems(hiddenItems, false);
        hiddenItems = [];
    }
    return (
        <div className="control-section">
            <ButtonComponent content='Show all items' onClick={btnClick}/>
            <div className="menu-section">
                <MenuComponent ref={scope => menuObj = scope}
items={menuItems} beforeOpen={beforeOpen} created={created}/>
            </div>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Using the [beforeOpen](#) event, you can hide the sub menu items as in the above example since the menu supports to hide items only for headers initially.

Create mnemonic ui in menuitem in React Menu component

A particular character in a text can be underlined in the [beforeItemRender](#) event by adding `<u>` tag in between the text and assign the innerHTML to the `li` element.

In the following example, the first character in **File**, **Open**, and **Save** menu items are underlined.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuObj;
    let menuItems = [
        {
            iconCss: 'em-icons e-file',
            items: [
                { text: 'Open', iconCss: 'em-icons e-open' },
                { text: 'Save', iconCss: 'em-icons e-save' },
                { separator: true },
                { text: 'Exit' }
            ],
            text: 'File'
        },
        {
            iconCss: 'em-icons e-edit',
            items: [
                { text: 'Cut', iconCss: 'em-icons e-cut' },
                { text: 'Copy', iconCss: 'em-icons e-copy' },
                { text: 'Paste', iconCss: 'em-icons e-paste' }
            ],
            text: 'Edit'
        }
    ];
}

```

```

    },
    { text: 'Format' },
    { text: 'View' },
    { text: 'Bookmarks' },
    { text: 'Tools' },
    { separator: true },
    { text: 'Help' }
  ];
  function beforeItemRender(args) {
    if (['File', 'Open', 'Save'].indexOf(args.item.text) > -1) {
      // To underline a First character.
      const underlinedText = '<u>' + args.item.text.slice(0, 1) +
        '</u>' + args.item.text.slice(1);
      args.element.innerHTML =
        args.element.innerHTML.replace(args.item.text, underlinedText);
    }
  }
  return (<MenuComponent items={menuItems}
    beforeItemRender={beforeItemRender}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, MenuEventArgs, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  let menuObj: MenuComponent;
  let menuItems: MenuItemModel[] = [
    {
      iconCss: 'em-icons e-file',
      items: [
        { text: 'Open', iconCss: 'em-icons e-open' },
        { text: 'Save', iconCss: 'em-icons e-save' },
        { separator: true },
        { text: 'Exit' }
      ],
      text: 'File'
    },
    {
      iconCss: 'em-icons e-edit',
      items: [
        { text: 'Cut', iconCss: 'em-icons e-cut' },
        { text: 'Copy', iconCss: 'em-icons e-copy' },
        { text: 'Paste', iconCss: 'em-icons e-paste' }
      ],
      text: 'Edit'
    },
    { text: 'Format' },
    { text: 'View' },
    { text: 'Bookmarks' },

```



```

        { text: 'Tools' },
        { separator: true },
        { text: 'Help' }
    ];
    function beforeItemRender(args: MenuEventArgs): void {
        if ([ 'File', 'Open', 'Save' ].indexOf(args.item.text) > -1) {
            // To underline a First character.
            const underlinedText = '<u>' + args.item.text.slice(0, 1) +
            '</u>' + args.item.text.slice(1);
            args.element.innerHTML =
            args.element.innerHTML.replace(args.item.text, underlinedText);
        }
        return (
            <MenuComponent items={menuItems}
            beforeItemRender={beforeItemRender}/>
        );
    }
    export default App;
    ReactDOM.render(<App />, document.getElementById('element'));

```

Change sub menu position in React Menu component

The submenu position can be changed by using the [beforeOpen](#) event. Assign the top and left position where you want to open the submenu to the [beforeOpen](#) event arguments `args.top` and `args.left` respectively.

In the below sample, the sub menu opens above the parent menu item.

INDEX.JSX

```

import { closest, enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    // Menu items definition
    let menuItems = [
        {
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { text: 'Exit' }
            ],
            text: 'File'
        },
        {
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }
            ],
            text: 'Edit'
        },
        {
            items: [

```

```

        { text: 'Toolbar' },
        { text: 'Sidebar' }
    ],
    text: 'View'
},
{
    items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
    ],
    text: 'Tools'
},
{ text: 'Go' },
{ text: 'Help' }
];
function onBeforeOpen(args) {
    // Getting parent menu item element offset
    const relativeOffset = closest(args.event.target, '.e-menu-
item').getBoundingClientRect();
    // Getting sub menu wrapper element using closest method
    const subMenuEle = closest(args.element, '.e-menu-wrapper');
    subMenuEle.style.display = 'block';
    args.top = (relativeOffset.top -
subMenuEle.getBoundingClientRect().height) + pageYOffset;
    args.left = relativeOffset.left + pageXOffset;
    subMenuEle.style.display = '';
}
return (<MenuComponent items={menuItems} beforeOpen={onBeforeOpen}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { closest, enableRipple } from '@syncfusion/ej2-base';
import { BeforeOpenCloseMenuEventArgs, MenuComponent, MenuItemModel } from
 '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    // Menu items definition
    let menuItems: MenuItemModel[] = [
        {
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { text: 'Exit' }
            ],
            text: 'File'
        },
        {
            items: [
                { text: 'Cut' },
                { text: 'Copy' },

```

```

        { text: 'Paste' }
      ],
      text: 'Edit'
    },
    {
      items: [
        { text: 'Toolbar' },
        { text: 'Sidebar' }
      ],
      text: 'View'
    },
    {
      items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
      ],
      text: 'Tools'
    },
    { text: 'Go' },
    { text: 'Help' }
  ];
  function onBeforeOpen(args: BeforeOpenCloseMenuEventArgs) {
    // Getting parent menu item element offset
    const relativeOffset = closest(args.event.target as Element, '.e-menu-item').getBoundingClientRect();
    // Getting sub menu wrapper element using closest method
    const subMenuEle = closest(args.element, '.e-menu-wrapper') as
    HTMLInputElement;
    subMenuEle.style.display = 'block';
    args.top = (relativeOffset.top -
    subMenuEle.getBoundingClientRect().height) + pageYOffset;
    args.left = relativeOffset.left + pageXOffset;
    subMenuEle.style.display = '';
  }
  return (
    <MenuComponent items={menuItems} beforeOpen={onBeforeOpen}/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

> For custom positioning, set both **top** and **left** position in the [beforeOpen](#) event.

Rounded corner in React Menu component

The rounded corner can be achieved by using the [cssClass](#) property. Add a custom class to the menu component and customize it using the **border-radius** CSS property. For more information, refer to the [style.css](#) file mapped under the source tab.

INDEX.JSX

```

import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // Menu items definition

```

```

let menuItems = [
  {
    items: [
      { text: 'Open' },
      { text: 'Save' },
      { text: 'Exit' }
    ],
    text: 'File'
  },
  {
    items: [
      { text: 'Cut' },
      { text: 'Copy' },
      { text: 'Paste' }
    ],
    text: 'Edit'
  },
  {
    items: [
      { text: 'Toolbar' },
      { text: 'Sidebar' }
    ],
    text: 'View'
  },
  {
    items: [
      { text: 'Spelling & Grammar' },
      { text: 'Customize' },
      { text: 'Options' }
    ],
    text: 'Tools'
  },
  { text: 'Go' },
  { text: 'Help' }
];
return (<MenuComponent items={menuItems} cssClass='e-rounded-menu'/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-
navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // Menu items definition
  let menuItems: MenuItemModel[] = [
    {
      items: [
        { text: 'Open' },
        { text: 'Save' },
        { text: 'Exit' }
      ],
      text: 'File'
    }
  ];
}

```

```

    },
    {
      items: [
        { text: 'Cut' },
        { text: 'Copy' },
        { text: 'Paste' }
      ],
      text: 'Edit'
    },
    {
      items: [
        { text: 'Toolbar' },
        { text: 'Sidebar' }
      ],
      text: 'View'
    },
    {
      items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
      ],
      text: 'Tools'
    },
    { text: 'Go' },
    { text: 'Help' }
  ];
  return (
    <MenuComponent items={menuItems} cssClass='e-rounded-menu' />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Change animation settings in React Menu component

To change the animation of the Menu, [animationSettings](#) property is used. The supported effects for Menu are,

| Effect | Functionality |

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

| None | Specifies the sub menu transform with no animation effect. |

| SlideDown | Specifies the sub menu transform with slide down effect. |

| ZoomIn | Specifies the sub menu transform with zoom in effect. |

| FadeIn | Specifies the sub menu transform with fade in effect. |

The following sample illustrates how to open Menu with **FadeIn** [effect](#) with the [duration](#) of **800ms**. Also we can set [easing](#) for menu items.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';

```

```

import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu items definition
  let menuItems = [
    {
      items: [
        { text: 'Open' },
        { text: 'Save' },
        { text: 'Exit' }
      ],
      text: 'File'
    },
    {
      items: [
        { text: 'Cut' },
        { text: 'Copy' },
        { text: 'Paste' }
      ],
      text: 'Edit'
    },
    {
      items: [
        { text: 'Toolbar' },
        { text: 'Sidebar' }
      ],
      text: 'View'
    },
    {
      items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
      ],
      text: 'Tools'
    },
    { text: 'Go' },
    { text: 'Help' }
  ];
  let animationSettings = {
    duration: 800,
    effect: 'FadeIn'
  };
  return (<MenuComponent items={menuItems}
animationSettings={animationSettings}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-
navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';

```

```

enableRipple(true);
function App() {
  // Menu items definition
  let menuItems: MenuItemModel[] = [
    {
      items: [
        { text: 'Open' },
        { text: 'Save' },
        { text: 'Exit' }
      ],
      text: 'File'
    },
    {
      items: [
        { text: 'Cut' },
        { text: 'Copy' },
        { text: 'Paste' }
      ],
      text: 'Edit'
    },
    {
      items: [
        { text: 'Toolbar' },
        { text: 'Sidebar' }
      ],
      text: 'View'
    },
    {
      items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
      ],
      text: 'Tools'
    },
    { text: 'Go' },
    { text: 'Help' }
  ];
  let animationSettings: any = {
    duration: 800,
    effect: 'FadeIn'
  }
  return (
    <MenuComponent items={menuItems}
    animationSettings={animationSettings}/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Right to left in React Menu component

Menu 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 Menu component.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    // Menu items definition
    let menuItems = [
        {
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { text: 'Exit' }
            ],
            text: 'File'
        },
        {
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }
            ],
            text: 'Edit'
        },
        {
            items: [
                { text: 'Toolbar' },
                { text: 'Sidebar' }
            ],
            text: 'View'
        },
        {
            items: [
                { text: 'Spelling & Grammar' },
                { text: 'Customize' },
                { text: 'Options' }
            ],
            text: 'Tools'
        },
        { text: 'Go' },
        { text: 'Help' }
    ];
    return (<MenuComponent items={menuItems} enableRTL={true}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {

```



```
// Menu items definition
let menuItems: MenuItemModel[] = [
  {
    items: [
      { text: 'Open' },
      { text: 'Save' },
      { text: 'Exit' }
    ],
    text: 'File'
  },
  {
    items: [
      { text: 'Cut' },
      { text: 'Copy' },
      { text: 'Paste' }
    ],
    text: 'Edit'
  },
  {
    items: [
      { text: 'Toolbar' },
      { text: 'Sidebar' }
    ],
    text: 'View'
  },
  {
    items: [
      { text: 'Spelling & Grammar' },
      { text: 'Customize' },
      { text: 'Options' }
    ],
    text: 'Tools'
  },
  { text: 'Go' },
  { text: 'Help' }
];
return (
  <MenuComponent items={menuItems} enableRtl={true}/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));
```

Set title in React Menu component

In this sample, the title for menu item can be achieved by using 'beforeItemRender' client-side event in Menu component.

INDEX.JSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  let menuItems = [
```

```

        {
            id: 'settingIcon',
            iconCss: 'em-icons e-file',
            items: [
                { text: 'Open',
                  items: [
                      { text: 'Sub Option1' },
                      { text: 'Sub Option2' },
                  ]
                },
                { text: 'Save' },
                { separator: true },
                { text: 'Exit' }
            ]
        }
    ];
    function beforeItemRender(args) {
        if (args.item.id == 'settingIcon') {
            args.element.setAttribute('title', 'Settings');
        }
    }
    return (<MenuComponent items={menuItems}
beforeItemRender={beforeItemRender}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { MenuComponent, MenuEventArgs, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    let menuItems: MenuItemModel[] = [
        {
            id: 'settingIcon',
            iconCss: 'em-icons e-file',
            items: [
                { text: 'Open',
                  items: [
                      { text: 'Sub Option1' },
                      { text: 'Sub Option2' },
                  ]
                },
                { text: 'Save' },
                { separator: true },
                { text: 'Exit' }
            ]
        }
    ];
    function beforeItemRender(args: MenuEventArgs): void {
        if (args.item.id == 'settingIcon') {

```

```

        args.element.setAttribute('title', 'Settings');
    }
}
return (
    <MenuComponent items={menuItems}
    beforeItemRender={beforeItemRender}/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React ContextMenu</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
cards/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
notifications/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
layouts/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"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>
    <div id='element'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Menu item click in React Menu component

You can open menu items and sub menu on menu item click by setting [showItemOnClick](#) property of the Menu. To open sub menu items only on item click, should be set as `true`.

INDEX.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';

```

```

import { MenuComponent } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
    // Menu items definition
    let menuItems = [
        {
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { separator: true },
                { text: 'Exit' }
            ],
            text: 'File'
        },
        {
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }
            ],
            text: 'Edit'
        },
        {
            items: [
                {
                    text: 'Toolbars',
                    items: [
                        { text: 'Menu Bar' },
                        { text: 'Bookmarks Toolbar' },
                        { text: 'Customize' }
                    ]
                },
                {
                    text: 'Zoom',
                    items: [
                        { text: 'Zoom In' },
                        { text: 'Zoom Out' },
                        { text: 'Reset' }
                    ]
                },
                { text: 'Full Screen' }
            ],
            text: 'View'
        },
        {
            items: [
                { text: 'Spelling & Grammar' },
                { text: 'Customize' },
                { text: 'Options' }
            ],
            text: 'Tools'
        },
        { text: 'Go' },
        { text: 'Help' }
    ];
};

```

```

    return (<MenuComponent items={menuItems} showItemOnClick={true}/>);
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { MenuComponent, MenuItemModel } from '@syncfusion/ej2-react-navigations';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
function App() {
  // Menu items definition
  let menuItems: MenuItemModel[] = [
    {
      items: [
        { text: 'Open' },
        { text: 'Save' },
        { separator: true },
        { text: 'Exit' }
      ],
      text: 'File'
    },
    {
      items: [
        { text: 'Cut' },
        { text: 'Copy' },
        { text: 'Paste' }
      ],
      text: 'Edit'
    },
    {
      items: [
        {
          text: 'Toolbars',
          items: [
            { text: 'Menu Bar' },
            { text: 'Bookmarks Toolbar' },
            { text: 'Customize' }
          ]
        },
        {
          text: 'Zoom',
          items: [
            { text: 'Zoom In' },
            { text: 'Zoom Out' },
            { text: 'Reset' }
          ]
        },
        { text: 'Full Screen' }
      ],
      text: 'View'
    },
    {

```

```

        items: [
          { text: 'Spelling & Grammar' },
          { text: 'Customize' },
          { text: 'Options' }
        ],
        text: 'Tools'
      },
      { text: 'Go' },
      { text: 'Help' }
    ];
    return (
      <MenuComponent items={menuItems} showItemOnClick={true}/>
    );
  }
}

export default App;
ReactDOM.render(<App />, document.getElementById('element'));

```

Ej1 api migration in React Menu component

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

Properties

```
{% raw %}
```

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

 Animation type on hover or click on the menu items 	Property: <i>animationType</i> var animationType = ej.AnimationType.Default; <EJ.Menu id="menu" animationType={animationType}></EJ.Menu>	 Not applicable
---	--	---------------------------

Context menu target Property: <code>contextMenuTarget</code> <EJ.Menu id="menu" contextMenuTarget="#target"></EJ.Menu> Not applicable

 Container element for submenu's collision 	Property: <i>container</i> <i><EJ.Menu id="menu" contextMenuTarget="#target" container="#target"></i>	 Not applicable
--	---	---------------------------

```
| Menu items | Not applicable | Property: items <br/><br/> <MenuComponent id="menu"
items={this.menuItems}></MenuComponent> <br/> constructor(props: {}) { <br/> &#160;
this.menuItems = [ <br/> &#160; <br/> &#160; &#160; text: "File", id: "fileid" <br/> &#160; &#160; }, <br/>
&#160; <br/> &#160; &#160; text: "Edit", <br/> &#160; &#160; &#160; id: "editid", <br/> &#160; &#160;
items: [ <br/> &#160; &#160; &#160; &#160; { text: "Cut", iconCss: "e-icons e-cut" }, <br/> &#160; &#160;
&#160; &#160; { text: "Copy", iconCss: "e-icons e-copy" }, <br/> &#160; &#160; &#160; &#160; { text: "Paste", iconCss:
"e-icons e-paste" } <br/> &#160; &#160; &#160; ] <br/> &#160; &#160; }, <br/> &#160; &#160; { <br/> &#160; &#160; &#160; text:
"view", id: "viewid", url: "#" <br/> &#160; &#160; }, <br/> &#160; &#160; <br/> &#160; &#160; &#160; &#160; text: "Help", id:
"helpid" <br/> &#160; &#160; } <br/> ]; <br/> } |
```

```
| Adding custom class | Property: cssClass   
 <EJ.Menu id="menu" cssClass="custom-  
class"></EJ.Menu> | Property: cssClass   
 <MenuComponent id="menu"  
cssClass="custom-class" items={this.menuItems}></MenuComponent> |
```

 Enables or disables the animation on hover or click on the menu items Property: <i>enableAnimation</i>

 <EJ.Menu id="menu" enableAnimation={true}></EJ.Menu> Not applicable
--

| **Animation settings** | **Not applicable** | **Property:** *animationSettings*

 <MenuComponent id="menu" animationSettings={this.animation} items={this.menuItems}></MenuComponent>
 constructor(props: {}) {
 this.animationSettings = { effect: 'ZoomIn' };
 } |

| **Root menu items to be aligned center in horizontal menu** | **Property:** *enableCenterAlign*

 <EJ.Menu id="menu" enableCenterAlign={true}></EJ.Menu> | **Not applicable** |

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

 <EJ.Menu id="menu" enabled={false}></EJ.Menu> | **Property:** *disabled*

 <MenuComponent id="menu" disabled={true} items={this.menuItems}></MenuComponent> |

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

 <EJ.Menu id="menu" enableRTL={true}></EJ.Menu> | **Property:** *enableRtl*

 <MenuComponent id="menu" disabled={true} items={this.menuItems} enableRtl={true}></MenuComponent> |

| **Enables/Disables the separator** | **Property:** *enableSeparator*

 <EJ.Menu id="menu" enableSeparator={false}></EJ.Menu> | **Not applicable** |

| **Menu Type** | **Property:** *menuType*

 <EJ.Menu id="menu" menuType="contextmenu"></EJ.Menu> | **Not applicable** |

| **Exclude target for context menu** | **Property:** *excludeTarget*

 <EJ.Menu id="menu" menu-type="contextmenu" contextMenuTarget="#target" excludeTarget=".inner"></EJ.Menu> | **Not applicable** |

| **Fields** | **Property:** *fields*

 var data = [
 { id: "fileid", text: "File", parentId: null },
 { id: "editid", text: "Edit", parentId: null },
 { id: "viewid", text: "View", parentId: null },
 { id: "helpid", text: "Help", parentId: null }
];
 <EJ.Menu id="menu" fields-dataSource={data} fields-id="id" fields-text="text" fields-parentid="parentId"></EJ.Menu> | **Property:** *fields*

 <MenuComponent id="menu" disabled={true} items={this.menuItems} fields={this.menuFields}></MenuComponent>
 constructor(props: {}) {
 this.menuFields = { text: ["text"], children: ["children"] };
 } |

| **Height** | **Property:** *height*

 <EJ.Menu id="menu" height="50px"></EJ.Menu> | **Not applicable** |

| **Width** | **Property:** *width*

 <EJ.Menu id="menu" width="800px"></EJ.Menu> | **Not applicable** |

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

 var attributes = { "aria-label": "menu" }
 <EJ.Menu id="menu" htmlAttribute={attributes}></EJ.Menu> | **Not applicable** |

| **Responsive** | **Property:** *isResponsive*

 <EJ.Menu id="menu" isResponsive={true}></EJ.Menu> | **Not applicable** |

| **Show item on click** | **Property:** *openOnClick*

 <EJ.Menu id="menu" openOnClick={true}></EJ.Menu> | **Property:** *showItemOnClick*

 <MenuComponent id="menu" showItemOnClick={true} items={this.menuItems}></MenuComponent> |

| **Orientation** | **Property:** *orientation*

 <EJ.Menu id="menu" orientation="vertical"></EJ.Menu> | **Property:** *orientation*

 <MenuComponent id="menu" orientation="Vertical" items={this.menuItems}></MenuComponent> |

| **Show root level arrows** | **Property:** *showRootLevelArrows*

 <EJ.Menu id="menu" showRootLevelArrows={false}></EJ.Menu> | **Not applicable** |

| **Show sub level arrows** | **Property:** *showSubLevelArrows*

 <EJ.Menu id="menu" showSubLevelArrows={false}></EJ.Menu> | **Not applicable** |

| **Sub menu direction** | **Property:** *subMenuDirection*

 <EJ.Menu id="menu" subMenuDirection="left"></EJ.Menu> | **Not applicable** |

| **Title** | **Property:** *titleText*

 <EJ.Menu id="menu" titleText="Title of the Menu"></EJ.Menu> | **Not applicable** |

| **Template** | **Not applicable** | **Property:** *template*

 <MenuComponent id="menu" items={this.data} fields={this.menuFields} template="#menuTemplate"></MenuComponent> |

| **Pop up menu height** | **Property:** *overflowHeight*

 <EJ.Menu id="menu" overflowHeight="200px"></EJ.Menu> | **Not applicable** |

Methods

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

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

| **Disable Method** | **Method:** *disable*

 <EJ.Menu id="menu"></EJ.Menu>
 var menu = \$('#menu').ejMenu('instance');
 menu.disable(); | **Not applicable** |

| **Disable menu items** | **Method:** *disableItem*

 <EJ.Menu id="menu"></EJ.Menu>
 var menu = \$('#menu').ejMenu('instance');
 menu.disableItem("File"); | **Method:** *enableItems*

 <MenuComponent id="menu" ref={(scope) => {this.menu = scope}} items={this.menuItems}></MenuComponent>
 constructor(props: {}) {
 this.menu.enableItems("File", false);
 } |

| **Disable menu items by ID** | **Method:** *disableItemByID*

 <EJ.Menu id="menu"></EJ.Menu>
 var menu = \$('#menu').ejMenu('instance');
 menu.disableItemByID("fileid"); | **Method:** *enableItems*

 <MenuComponent id="menu" ref={(scope) => {this.menu = scope}} items={this.menuItems}></MenuComponent>
 constructor(props: {}) {
 this.menu.enableItems("fileid", false, true);
 } |

| **Enable Method** | **Method:** *enable*

 <EJ.Menu id="menu"></EJ.Menu>
 var menu = \$('#menu').ejMenu('instance');
 menu.enable(); | **Not applicable** |

| **Enable menu items** | **Method:** *enableItem*

 <EJ.Menu id="menu"></EJ.Menu>
 var menu = \$('#menu').ejMenu('instance');
 menu.enableItem("File"); | **Method:** *enableItems*

 <MenuComponent id="menu" ref={(scope) => {this.menu = scope}} items={this.menuItems}></MenuComponent>
 constructor(props: {}) {
 this.menu.enableItems("File", true);
 } |

| **Enable menu items by ID** | **Method:** *enableItemByID*

 <EJ.Menu id="menu"></EJ.Menu>
 var menu = \$('#menu').ejMenu('instance');
 menu.enableItemByID("fileid"); | **Method:** *enableItems*

 <MenuComponent id="menu" ref={(scope) => {this.menu = scope}} items={this.menuItems}></MenuComponent>
 constructor(props: {}) {
 this.menu.enableItems("fileid", true, true);
 } |

| **Hide Method** | **Method:** `hide`
 `<EJ.Menu id="menu"></EJ.Menu>`
 `var menu = $('#menu').ejMenu('instance');`
 `menu.hide();` | **Not applicable** |

| **Hide menu items** | **Method:** `hideItems`
 `<EJ.Menu id="menu"></EJ.Menu>`
 `var menu = $('#menu').ejMenu('instance');`
 `menu.hideItems(["#helpid", "#editid"]);` | **Method:** `hideItems`
 `<MenuComponent id="menu" ref={scope => {this.menu = scope}}`
 `items={this.menuItems}></MenuComponent>`
 `constructor(props: {}) {`
 ` `
 `this.menu.hideItems(["editid", "helpid"], true);`
 `}` |

| **Insert menu items** | **Method:** `insert`
 `<EJ.Menu id="menu"></EJ.Menu>`
 `var menu = $('#menu').ejMenu('instance');`
 `menu.insert([{ id: "viewid", text: "View" }, {"#editid"}];` | **Not applicable** |

| **Insert menu items after the specified menu item** | **Method:** `insertAfter`
 `<EJ.Menu id="menu"></EJ.Menu>`
 `var menu = $('#menu').ejMenu('instance');`
 `menu.insertAfter([{ id: "viewid", text: "View" }, {"#editid"}];` | **Method:** `insertAfter`
 `<MenuComponent id="menu" ref={scope => {this.menu = scope}}`
 `items={this.menuItems}></MenuComponent>`
 `constructor(props: {}) {`
 ` `
 `this.menu.insertAfter([{ id: "view_id", text: "View" }, {"Edit"}];`
 `}` |

| **Insert menu items before the specified menu item** | **Method:** `insertBefore`
 `<EJ.Menu id="menu"></EJ.Menu>`
 `var menu = $('#menu').ejMenu('instance');`
 `menu.insertBefore([{ id: "viewid", text: "View" }, {"#helpid"}];` | **Method:** `insertBefore`
 `<MenuComponent id="menu" ref={scope => {this.menu = scope}}`
 `items={this.menuItems}></MenuComponent>`
 `constructor(props: {}) {`
 ` `
 `this.menu.insertBefore([{ id: "view_id", text: "View" }, {"Help"}];`
 `}` |

| **Remove menu items** | **Method:** `remove`
 `<EJ.Menu id="menu"></EJ.Menu>`
 `var menu = $('#menu').ejMenu('instance');`
 `menu.remove(["#Edit"]);` | **Method:** `removeItems`
 `<MenuComponent id="menu" ref={scope => {this.menu = scope}}`
 `items={this.menuItems}></MenuComponent>`
 `constructor(props: {}) {`
 ` `
 `this.menu.removeItems(["Edit"]);`
 `}` |

| **To show menu** | **Method:** `show`
 `<EJ.Menu id="menu"></EJ.Menu>`
 `var menu = $('#menu').ejMenu('instance');`
 `menu.show();` | **Not applicable** |

| **Destroy method** | **Method:** `destroy`
 `<EJ.Menu id="menu"></EJ.Menu>`
 `var menu = $('#menu').ejMenu('instance');`
 `menu.destroy();` | **Method:** `destroy`
 `<MenuComponent id="menu" ref={scope => {this.menu = scope}}`
 `items={this.menuItems}></MenuComponent>`
 `constructor(props: {}) {`
 ` `
 `this.menu.destroy();`
 `}` |

Events

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

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

| **Triggers before opening the menu** | **Events:** `beforeOpen`
 `<EJ.Menu id="menu" beforeOpen={beforeOpen}></EJ.Menu>`
 `function beforeOpen(args) {`
 ` `
 ` `
 `code block /`
 `}` | **Events:** `beforeOpen`
 `<MenuComponent id="menu"`

**items={this.menuitems} beforeOpen={this.beforeOpen.bind(this)}></MenuComponent>

beforeOpen(args) {
 / code block /
 } |**

**| Triggers before closing the menu | Not applicable | Events: beforeClose

<MenuComponent id="menu" items={this.menuitems}
beforeClose={this.beforeClose.bind(this)}></MenuComponent>
 beforeClose(args) {

 / code block */
 } |**

**| Triggers before rendering each menu item | Not applicable | Events: beforeItemRender

<MenuComponent id="menu" items={this.menuitems}
beforeItemRender={this.beforeItemRender.bind(this)}></MenuComponent>

beforeItemRender(args) {
 / code block */
 } |**

**| Triggers while selecting the menu item | Events: click

 <EJ.Menu id="menu"
click={click}></EJ.Menu>
 function click(args) {
 / code block /
 } |
Events: select

 <MenuComponent id="menu" items={this.menuitems}
select={this.select.bind(this)}></MenuComponent>
 select(args) {
 / code
block /
 } |**

**| Triggers after closing the menu | Events: close

 <EJ.Menu id="menu"
close={close}></EJ.Menu>
 function close(args) {
 / code block /
 } |
Events: onClose

 <MenuComponent id="menu" items={this.menuitems}
onClose={this.onClose.bind(this)}></MenuComponent>
 onClose(args) {
 /
code block /
 } |**

**| Triggers after opening the menu | Events: open

 <EJ.Menu id="menu"
open={open}></EJ.Menu>
 function open(args) {
 / code block /
 } |
Events: onOpen

 <MenuComponent id="menu" items={this.menuitems}
onOpen={this.onOpen.bind(this)}></MenuComponent>
 onOpen(args) {
 /
code block /
 } |**

**| Triggers once the component rendering is completed | Events: create

 <EJ.Menu
id="menu" create={create}></EJ.Menu>
 function create(args) {
 / code
block /
 } | Events: created

 <MenuComponent id="menu" items={this.menuitems}
created={this.created.bind(this)}></MenuComponent>
 created() {
 / code
block /
 } |**

**| Triggers once the component is destroyed | Events: destroy

 <EJ.Menu id="menu"
destroy={destroy}></EJ.Menu>
 function destroy(args) {
 / code block /

 } | Not applicable* |**

**| Triggers when key down on menu items | Events: keydown

 <EJ.Menu id="menu"
keydown={keydown}></EJ.Menu>
 function keydown(args) {
 / code block
/
 } | Not applicable* |**

**| Triggers when mouse out from menu items | Events: mouseout

 <EJ.Menu id="menu"
mouseout={mouseout}></EJ.Menu>
 function mouseout(args) {
 / code
block /
 } | Not applicable* |**

| **Triggers when mouse over the Menu items** | **Events:** *mouseover*

 <EJ.Menu id="menu" mouseover={mouseover}></EJ.Menu>
 function mouseover(args) {
 / **code block** /
 } | **Not applicable*** |

| **Triggers when overflow popup menu opens** | **Events:** *overflowOpen*

 <EJ.Menu id="menu" overflowOpen={overflowOpen}></EJ.Menu>
 function overflowOpen(args) {
 / **code block** /
 } | **Not applicable*** |

| **Triggers when overflow popup menu closes** | **Events:** *overflowClose*

 <EJ.Menu id="menu" overflowClose={overflowClose}></EJ.Menu>
 function overflowClose(args) {
 / **code block** /
 } | **Not applicable*** |

{% endraw %}

Message

Getting Started

This article provides a step-by-step introduction to get started with the Syncfusion React Message component.

Prerequisites

[System requirements for Syncfusion React UI components](#)

Dependencies

The following list of dependencies are required to use the **Message** component in the application.

```
`javascript
```

```
|-- @syncfusion/ej2-react-notifications
```

```
|-- @syncfusion/ej2-base
```

```
|-- @syncfusion/ej2-buttons
```

```
|-- @syncfusion/ej2-popups
```

```
|-- @syncfusion/ej2-notifications
```

```
|-- @syncfusion/ej2-react-base
```

```
`
```

Create the React application

To set-up, a React application, choose any of the following ways. The best and easiest way is to use the [create-react-app](#). It sets up the development environment in JavaScript and improves the application for production. Refer to the [installation instructions](#) of the **create-react-app**.

```
`bash
```

```
npx create-react-app my-app
```

```
cd my-app
```

```
npm start
```

```
`
```

or

```
`bash
yarn create react-app my-app
cd my-app
yarn start
`
```

To set-up a React application in the **TypeScript** environment, run the following command.

```
`bash
npx create-react-app my-app --template typescript
cd my-app
npm start
`
```

Besides using the [npx](#) package runner tool, also create an application from the **npm init**. To begin with the **npm init**, upgrade the **npm** version to **npm 6+**.

```
`bash
npm init react-app my-app
cd my-app
npm start
`
```

Add Syncfusion React packages

Once you have created the React application, install the required Syncfusion React component package in the application. All Syncfusion React (Essential JS 2) packages are published on the [npmjs.com](https://www.npmjs.com) public registry.

To install the Message component package, use the following command.

```
`bash
npm install @syncfusion/ej2-react-notifications --save
`
```

or

```
`bash
yarn add @syncfusion/ej2-react-notifications
`
```

Also, check out the [installation section](#) to know the different ways of installing the packages.

Import the Syncfusion CSS styles

After installing the Syncfusion component packages in the application, import the required themes based on the components used.

The Syncfusion React component comes with built-in [themes](#), which are available in installed packages. It is quite simple to adapt the Syncfusion React components based on the application style by referring to any of the built-in themes. Import the **Material** theme for the Message component.

Import the CSS styles for the Message component and its dependencies in the **src/App.css** file.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-react-notifications/styles/material.css';
`
```

Check out the [Themes topic](#) to know more about built-in themes and different ways to refer to themes in React applications.

Add Message component to the application

Start adding the required components to the application. Add the Message component in the **src/App.js** or **src/App.tsx** file using the following code.

- Before adding the Message component to the markup, import the Message component in the **src/App.js** or **src/App.tsx** file.

```
`bash
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
`
```

- Then, add the **Message** component in the application using the following code sample.

```
`ts
import './App.css';
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
  return (<MessageComponent content="Please read the comments carefully"></MessageComponent>);
}
export default App;
`
```

Run the application

All are set. Now, run the application using the following command.

```
`bash
npm start
`
```

or

```
`bash
```

```
yarn start
```

```
,
```

The output will appear as follows.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (<MessageComponent content="Please read the comments
carefully"></MessageComponent>);
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (<MessageComponent content="Please read the comments
carefully"></MessageComponent>);
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Message</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>
```

```

<div id='sample'>
  <div id='loader'>Loading....</div>
</div>
<style>
  /* Sample level styles */
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
  .e-message {
    margin: 100px;
  }
</style>
</body>
</html>

```

Severities in React Message component

The severity denotes the importance and context of the message to the user. The message contains different severity types. Use the [severity](#) property to display the messages with different severity levels.

The available severity types are **Normal**, **Success**, **Info**, **Warning** and **Error**. The default severity type for messages is **Normal**.

The following example demonstrates the severity of the messages.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
  return (
    <div className="msg-default-section">
      <div className="content-section">
        <MessageComponent id="msg_default" content="Editing is
restricted"></MessageComponent>
        <MessageComponent id="msg_info" content="Please read the comments
carefully" severity="Info"></MessageComponent>
        <MessageComponent id="msg_success" content="Your message has been
sent successfully" severity="Success"></MessageComponent>
        <MessageComponent id="msg_warning" content="There was a problem
with your network connection" severity="Warning"></MessageComponent>
        <MessageComponent id="msg_error" content="A problem occurred
while submitting your data" severity="Error"></MessageComponent>
      </div>
    </div>);
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
  return (
    <div className="msg-default-section">
      <div className="content-section">
        <MessageComponent id="msg_default" content="Editing is
restricted"></MessageComponent>
        <MessageComponent id="msg_info" content="Please read the comments
carefully" severity="Info"></MessageComponent>
        <MessageComponent id="msg_success" content="Your message has been
sent successfully" severity="Success"></MessageComponent>
        <MessageComponent id="msg_warning" content="There was a problem
with your network connection" severity="Warning"></MessageComponent>
        <MessageComponent id="msg_error" content="A problem occurred
while submitting your data" severity="Error"></MessageComponent>
      </div>
    </div>
  );
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Message</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
  <style>
    /* Sample level styles */
    #loader {
      color: #008cff;
      height: 40px;
    }
  </style>

```



```

        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .msg-default-section .content-section {
        margin: 0 auto;
        max-width: 450px;
        padding-top: 10px;
    }
    .msg-default-section .e-message {
        margin: 10px 0;
    }
</style>
</body>
</html>

```

Variants in React Message component

The Message has predefined appearance variants for different visual representations. The variants of the message can be changed based on the [variant](#) property.

The available variants are **Text**, **Outlined** and **Filled**. The default variant type for messages is **Text**.

- **Text** - The severity is differentiated using a text color and a light background color.
- **Outlined** - The severity is differentiated using a text color and a border without a background.
- **Filled** - The severity is differentiated using a text color and a dark background color.

The following example demonstrates the default message with different variant types.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (
        <div className="msg-variant-section">
            <div className="content-section">
                <h4>Filled</h4>
                <MessageComponent id="msg_default_filled"
variant="Filled">Editing is restricted</MessageComponent>
                <MessageComponent id="msg_info_filled" severity="Info"
variant="Filled">Please read the comments carefully</MessageComponent>
                <MessageComponent id="msg_success_filled" severity="Success"
variant="Filled">Your message has been sent successfully</MessageComponent>
                <MessageComponent id="msg_warning_filled" severity="Warning"
variant="Filled">There was a problem with your network
connection</MessageComponent>
                <MessageComponent id="msg_error_filled" severity="Error"
variant="Filled">A problem occurred while submitting your
data</MessageComponent>
            </div>
            <div className="content-section">
                <h4>Outlined</h4>
                <MessageComponent id="msg_default_outlined"
variant="Outlined">Editing is restricted</MessageComponent>
            </div>
        </div>
    );
}

```

```

        <MessageComponent id="msg_info_outlined" severity="Info"
variant="Outlined">Please read the comments carefully</MessageComponent>
        <MessageComponent id="msg_success_outlined" severity="Success"
variant="Outlined">Your message has been sent successfully</MessageComponent>
        <MessageComponent id="msg_warning_outlined" severity="Warning"
variant="Outlined">There was a problem with your network
connection</MessageComponent>
        <MessageComponent id="msg_error_outlined" severity="Error"
variant="Outlined">A problem occurred while submitting your
data</MessageComponent>
    </div>
    <div className="content-section">
        <h4>Text</h4>
        <MessageComponent id="msg_default">Editing is
restricted</MessageComponent>
        <MessageComponent id="msg_info" severity="Info">Please read the
comments carefully</MessageComponent>
        <MessageComponent id="msg_success" severity="Success">Your
message has been sent successfully</MessageComponent>
        <MessageComponent id="msg_warning" severity="Warning">There was a
problem with your network connection</MessageComponent>
        <MessageComponent id="msg_error" severity="Error">A problem
occurred while submitting your data</MessageComponent>
    </div>
</div>);
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (
        <div className="msg-variant-section">
            <div className="content-section">
                <h4>Filled</h4>
                <MessageComponent id="msg_default_filled"
variant="Filled">Editing is restricted</MessageComponent>
                <MessageComponent id="msg_info_filled" severity="Info"
variant="Filled">Please read the comments carefully</MessageComponent>
                <MessageComponent id="msg_success_filled" severity="Success"
variant="Filled">Your message has been sent successfully</MessageComponent>
                <MessageComponent id="msg_warning_filled" severity="Warning"
variant="Filled">There was a problem with your network
connection</MessageComponent>
                <MessageComponent id="msg_error_filled" severity="Error"
variant="Filled">A problem occurred while submitting your
data</MessageComponent>
            </div>
            <div className="content-section">
                <h4>Outlined</h4>

```

```

        <MessageComponent id="msg_default_outlined"
variant="Outlined">Editing is restricted</MessageComponent>
        <MessageComponent id="msg_info_outlined" severity="Info"
variant="Outlined">Please read the comments carefully</MessageComponent>
        <MessageComponent id="msg_success_outlined" severity="Success"
variant="Outlined">Your message has been sent successfully</MessageComponent>
        <MessageComponent id="msg_warning_outlined" severity="Warning"
variant="Outlined">There was a problem with your network
connection</MessageComponent>
        <MessageComponent id="msg_error_outlined" severity="Error"
variant="Outlined">A problem occurred while submitting your
data</MessageComponent>
    </div>
    <div className="content-section">
        <h4>Text</h4>
        <MessageComponent id="msg_default">Editing is
restricted</MessageComponent>
        <MessageComponent id="msg_info" severity="Info">Please read the
comments carefully</MessageComponent>
        <MessageComponent id="msg_success" severity="Success">Your
message has been sent successfully</MessageComponent>
        <MessageComponent id="msg_warning" severity="Warning">There was a
problem with your network connection</MessageComponent>
        <MessageComponent id="msg_error" severity="Error">A problem
occurred while submitting your data</MessageComponent>
    </div>
</div>
);
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Message</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>

```

```

<div id='sample'>
  <div id='loader'>Loading....</div>
</div>
<style>
  /* Sample level styles */
  #loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
  .msg-variant-section .content-section {
    margin: 0 auto;
    max-width: 520px;
    padding: 10px;
  }
  .msg-variant-section .e-message {
    margin: 10px 0;
  }
  .msg-variant-section {
    display: flex;
  }
</style>
</body>
</html>

```

Icons in React Message component

This section explains the message with no icons, how to show or hide the close icon and add the custom severity icon to the message.

No Icon

By default, severity icons can be displayed according to the severity types to make it more understandable to the user by visual information rather than text. To hide the severity icons, set the [showIcon](#) property to `false`.

The following example demonstrates the different severity messages without the severity icons.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
  return (
    <div className="msg-default-section">
      <div className="content-section">
        <MessageComponent id="msg_default" content="Editing is
restricted" showIcon={false}></MessageComponent>
        <MessageComponent id="msg_info" content="Please read the comments
carefully" severity="Info" showIcon={false}></MessageComponent>
        <MessageComponent id="msg_success" content="Your message has been
sent successfully" severity="Success" showIcon={false}></MessageComponent>
        <MessageComponent id="msg_warning" content="There was a problem
with your network connection" severity="Warning"
showIcon={false}></MessageComponent>
      </div>
    </div>
  );
}

```

```

        <MessageComponent id="msg_error" content="A problem occurred
while submitting your data" severity="Error"
showIcon={false}></MessageComponent>
    </div>
</div>);
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (
        <div className="msg-default-section">
            <div className="content-section">
                <MessageComponent id="msg_default" content="Editing is
restricted" showIcon={false}></MessageComponent>
                <MessageComponent id="msg_info" content="Please read the comments
carefully" severity="Info" showIcon={false}></MessageComponent>
                <MessageComponent id="msg_success" content="Your message has been
sent successfully" severity="Success" showIcon={false}></MessageComponent>
                <MessageComponent id="msg_warning" content="There was a problem
with your network connection" severity="Warning"
showIcon={false}></MessageComponent>
                <MessageComponent id="msg_error" content="A problem occurred
while submitting your data" severity="Error"
showIcon={false}></MessageComponent>
            </div>
        </div>
    );
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Message</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />

```

```

<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
  <style>
    /* Sample level styles */
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
    .msg-default-section .content-section {
      margin: 0 auto;
      max-width: 450px;
      padding-top: 10px;
    }
    .msg-default-section .e-message {
      margin: 10px 0;
    }
  </style>
</body>
</html>

```

Custom Icon

By default, the severity icons can be displayed according to the severity type to make it more understandable to the user by visual information rather than text. If the user wants to customize these icons, it can be achieved through the `cssClass` property.

The following example demonstrates how the default message is rendered with a custom severity icon.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
  return (<div className="msg-custom-section">
    <div className="content-section">
      <MessageComponent id="msg_icon" cssClass="custom">Essential JS 2
is a modern JavaScript UI Controls library that has
      been built from the ground up to be lightweight, responsive,
      modular and touch friendly. It is written in TypeScript and has no
      external dependencies. It also includes complete support for
      Angular, React, Vue, ASP.NET MVC and ASP.NET Core
      frameworks.</MessageComponent>
    </div>
  </div>);
}

```

```
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (
        <div className="msg-custom-section">
            <div className="content-section">
                <MessageComponent id="msg_icon" cssClass="custom">Essential JS 2
is a modern JavaScript UI Controls library that has
                been built from the ground up to be lightweight, responsive,
modular and touch friendly. It is written in TypeScript and has no
                external dependencies. It also includes complete support for
Angular, React, Vue, ASP.NET MVC and ASP.NET Core
frameworks.</MessageComponent>
            </div>
        </div>
    );
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Message</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
    <style>
        /* Sample level styles */
```

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

.msg-custom-section .content-section {
    margin: 0 auto;
    max-width: 400px;
    padding-top: 10px;
}

.msg-custom-section .e-message {
    margin: 10px 0;
}

@font-face {
    font-family: 'Message_icons';
    src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMj8BS4YAAAEoAAAAVmNtYX DopOj pAAABiAAAADZnbHlmEsNX
oQAAACgAAAjMAgVhZCgNOH4AAADQAAAAANmhoZWEIPwQDAAAArAAAACRobXR4C4AAAAAAAAAYAAAAIb
G9jYQEmAAAAAAHAAAAABmlheHABEAEOAAABCAAAACBuYwllldofAwAABBQAAAjtcG9zdNGGZXAAAA
aEAAAALwABAAAEAAAAAFwEAAAAAAD4gABAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAecv2Nl8PPPU
ACwQAAAAAAN9TeeEAAAAA3lN54QAAAAAD4gO/AAAACAACAAAAAAAAAAAAEAAAACAQIABAAAAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAA
AAAAAAAAAAAAAAAAAAUUGZFZABA6JTolAQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAAAAACAA
AAwAAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA6JT//wAA6JT//wAAAAEABAAAAAEAAAAAAAAABJgA
AAAQAAAAAA+IDvWANAG4AoAEBAABLwMrATUzPwQXDwMVHxIdAQ8SFR8GMz8RPQEvESMPAicHIw8H
ERUfDD8JES8JIw8BNw8DFR8SHQEPEhUfBj8SPQEvESMPAgHCjwYHCAelpQcIBwaPqQMCawECawUGC
QkIBwCHBQYEBQMDAgIBAQICAwMFBAYFBwCHCAkJBgUDAgEDBQcICQkHBGcGDQwLCwoJCAcHBGUEAw
ICAgIDBAUGBwCICQoLCwWNBGcHBwkJB6LJvwkIBwYFBAIBAwQFBGcECMTJBGcHCAgJBQUEBAMCAQE
BAQIDBAQFBQkJBwCH/QMCawECawUGFRQSEREODgwLCggGBgMDAwMGBggKCwwODhEREhQVBgUDAgED
BQcICQkHBGcGGBgWFBMSDw8NCwkIBgUDAwUGCAkLDQ8PEhMUFHgYBgcHBwkJBwElfwQEAggBAGMFF
wQDBAgICAgHBWYHCAkJCQkKCgSLCgWLDAsMDAsMCwwKCwsKCgkJCQkIBwYHBwgICAgHBwUDAQIDBA
oMDAwNDQ4PDg8QEBAQEBEREBAQEBApDg8ODQ0MDAwKBAMCAQMferMBAGQFBGcECP7/CAgHBgYDagE
BsgUDAgEBawMEBAUFBQYGANyGBgYFBQQEawIBAgROBAMICAgIBwCGERMTFRUWFxcYGRkaGhsbGxsb
GxoaGrkYFxcWFRUTExEGBwCICAgIBwCFawEBAQMEFRUXGBkaGxsDHR4eHx8gICAgHx8eHh0dGxsaG
RgXFRUEawIBawUAAAAAEgDeAAEAAAAAAAAAAAAQAAAAEAAAAAAAAEADQABAAEAAAAAAAAIABwAOAAEAAA
AAAAAMADQAVAAEAAAAAAAAAQADQaiAAEAAAAAAAAUACwAvAAEAAAAAAAAAYADQA6AAEAAAAAAAAoALABHAAE
AAAAAAAAAEgBzAAMAAQQJAAMAAAgCFAAMAAQQJAEEAGgCHAAMAAQQJAAIADgChaAMAAQQJAAMAGgCv
AAMAAQQJAAGAgDJAAMAAQQJAUAUAFgDJAAMAAQQJAAYAGgD5AAMAAQQJAaOAWAETAAMAAQQJAASAJ
AFrIE1lc3NhZ2VfaWNVbnNSZWdlbGZyTWVzc2FnZV9pY29uc01lc3NhZ2VfaWNVbnNWZXJzaW9uID
EuME1lc3NhZ2VfaWNVbnNGb250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWR
pb3d3dy5zeW5jZnVzaW9uLmNvbQAgAE0AZQBzAHMAYQBnAGUAXwBpAGMabwBuAHMAUgBlAGcAdQBz
AGEAcgBNAGUAcwBzAGEAZwBlAF8AaQBjAG8AbgBzAE0AZQBzAHMAYQBnAGUAXwBpAGMabwBuAHMAV
gBlAHIAcwBpAG8AbgAgADEALGAwAE0AZQBzAHMAYQBnAGUAXwBpAGMabwBuAHMARgBvAG4AdAAgAG
cAZQBwAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB
0AHIAbwAgAFMAdABlAGQAaQBVaHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMabwBtAAAA
AAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMABWF1ZGlvAAAA)
format('true-type');

    font-weight: normal;
    font-style: normal;
}

.custom.e-message .e-msg-icon::before {
    font-family: 'Message_icons';
    content: '\e894';
}
```



```

    </style>
  </body>
</html>

```

Close Icon

The message can be rendered with or without the close icon. The close icon is used to hide the message, either by manually clicking the close icon or through keyboard interaction.

By default, the close icon is not rendered in the message. To show the close icon, set the [showCloseIcon](#) property to **true**.

In the following example, the messages are rendered with the close icon.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { useState } from 'react';
function App() {
    const [defaultVisible, setDefaultVisible] = useState(true);
    const [defaultCssClass, setDefaultCssClass] = useState('e-outline e-
primary msg-hidden');
    const [infoVisible, setInfoVisible] = useState(true);
    const [infoCssClass, setinfoCssClass] = useState('e-outline e-primary e-
info msg-hidden');
    const [successVisible, setSuccessVisible] = useState(true);
    const [successCssClass, setSuccessCssClass] = useState('e-outline e-
primary e-success msg-hidden');
    const [warningVisible, setWarningVisible] = useState(true);
    const [warningCssClass, setWarningCssClass] = useState('e-outline e-
primary e-warning msg-hidden');
    const [errorVisible, setErrorVisible] = useState(true);
    const [errorCssClass, setErrorCssClass] = useState('e-outline e-primary
e-error msg-hidden');
    const [showIcon, setShowIcon] = useState(true);
    const [showCloseIcon, setShowCloseIcon] = useState(true);
    const defaultClick = () => {
        setDefaultVisible(true);
        setDefaultCssClass('e-outline e-primary msg-hidden');
    };
    const defaultClosed = () => {
        setDefaultVisible(false);
        setDefaultCssClass('e-outline e-primary');
    };
    const infoClick = () => {
        setInfoVisible(true);
        setinfoCssClass('e-outline e-primary e-info msg-hidden');
    };
    const infoClosed = () => {
        setInfoVisible(false);
        setinfoCssClass('e-outline e-primary e-info');
    };
    const successClick = () => {

```

```

        setSuccessVisible(true);
        setSuccessCssClass('e-outline e-primary e-success msg-hidden');
    };
    const successClosed = () => {
        setSuccessVisible(false);
        setSuccessCssClass('e-outline e-primary e-success');
    };
    const warningClick = () => {
        setWarningVisible(true);
        setWarningCssClass('e-outline e-primary e-warning msg-hidden');
    };
    const warningClosed = () => {
        setWarningVisible(false);
        setWarningCssClass('e-outline e-primary e-warning');
    };
    const errorClick = () => {
        setErrorVisible(true);
        setErrorCssClass('e-outline e-primary e-error msg-hidden');
    };
    const errorClosed = () => {
        setErrorVisible(false);
        setErrorCssClass('e-outline e-primary e-warning');
    };
    return (<div className="msg-icon-section">
        <div className="content-section">
            <ButtonComponent id="btn1" content="Show Default Message"
cssClass={defaultCssClass} onClick={defaultClick.bind(this)} />
            <MessageComponent id="msg_default_icon" visible={defaultVisible}
showCloseIcon={showCloseIcon} closed={defaultClosed.bind(this)}
showIcon={showIcon}>Editing is restricted</MessageComponent>
            <ButtonComponent id="btn2" content="Show Info Message"
cssClass={infoCssClass} onClick={infoClick.bind(this)} />
            <MessageComponent id="msg_info_icon" severity="Info"
showCloseIcon={showCloseIcon} visible={infoVisible}
closed={infoClosed.bind(this)} showIcon={showIcon}>Please read the comments
carefully</MessageComponent>
            <ButtonComponent id="btn3" content="Show Success Message"
cssClass={successCssClass} onClick={successClick.bind(this)} />
            <MessageComponent id="msg_success_icon" severity="Success"
showCloseIcon={showCloseIcon} closed={successClosed.bind(this)}
visible={successVisible} showIcon={showIcon}>Your message has been sent
successfully</MessageComponent>
            <ButtonComponent id="btn4" content="Show Warning Message"
cssClass={warningCssClass} onClick={warningClick.bind(this)} />
            <MessageComponent id="msg_warning_icon" severity="Warning"
showCloseIcon={showCloseIcon} closed={warningClosed.bind(this)}
visible={warningVisible} showIcon={showIcon}>There was a problem with your
network connection</MessageComponent>
            <ButtonComponent id="btn5" content="Show Error Message"
cssClass={errorCssClass} onClick={errorClick.bind(this)} />
            <MessageComponent id="msg_error_icon" severity="Error"
showCloseIcon={showCloseIcon} closed={errorClosed.bind(this)}
visible={errorVisible} showIcon={showIcon}>A problem occurred while
submitting your data</MessageComponent>
        </div>
    </div>);
}

```

```
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { useState } from 'react';
function App() {
    const [defaultVisible, setDefaultVisible] = useState<boolean>(true);
    const [defaultCssClass, setDefaultCssClass] = useState<string>('e-outline
e-primary msg-hidden');
    const [infoVisible, setInfoVisible] = useState<boolean>(true);
    const [infoCssClass, setinfoCssClass] = useState<string>('e-outline e-
primary e-info msg-hidden');
    const [successVisible, setSuccessVisible] = useState<boolean>(true);
    const [successCssClass, setSuccessCssClass] = useState<string>('e-outline
e-primary e-success msg-hidden');
    const [warningVisible, setWarningVisible] = useState<boolean>(true);
    const [warningCssClass, setWarningCssClass] = useState<string>('e-outline
e-primary e-warning msg-hidden');
    const [errorVisible, setErrorVisible] = useState<boolean>(true);
    const [errorCssClass, setErrorCssClass] = useState<string>('e-outline e-
primary e-error msg-hidden');
    const [showIcon, setShowIcon] = useState<boolean>(true);
    const [showCloseIcon, setShowCloseIcon] = useState<boolean>(true);
    const defaultClick = () => {
        setDefaultVisible(true);
        setDefaultCssClass('e-outline e-primary msg-hidden');
    };
    const defaultClosed = () => {
        setDefaultVisible(false);
        setDefaultCssClass('e-outline e-primary');
    };
    const infoClick = () => {
        setInfoVisible(true);
        setinfoCssClass('e-outline e-primary e-info msg-hidden');
    };
    const infoClosed = () => {
        setInfoVisible(false);
        setinfoCssClass('e-outline e-primary e-info');
    };
    const successClick = () => {
        setSuccessVisible(true);
        setSuccessCssClass('e-outline e-primary e-success msg-hidden');
    };
    const successClosed = () => {
        setSuccessVisible(false);
        setSuccessCssClass('e-outline e-primary e-success');
    };
    const warningClick = () => {
```

```

        setWarningVisible(true);
        setWarningCssClass('e-outline e-primary e-warning msg-hidden');
    };
    const warningClosed = () => {
        setWarningVisible(false);
        setWarningCssClass('e-outline e-primary e-warning');
    };
    const errorClick = () => {
        setErrorVisible(true);
        setErrorCssClass('e-outline e-primary e-error msg-hidden');
    };
    const errorClosed = () => {
        setErrorVisible(false);
        setErrorCssClass('e-outline e-primary e-warning');
    };
    return (
        <div className="msg-icon-section">
            <div className="content-section">
                <ButtonComponent id="btn1" content="Show Default Message"
                cssClass={defaultCssClass} onClick={defaultClick.bind(this)} />
                <MessageComponent id="msg_default_icon" visible={defaultVisible}
                showCloseIcon={showCloseIcon} closed={defaultClosed.bind(this)}
                showIcon={showIcon}>Editing is restricted</MessageComponent>
                <ButtonComponent id="btn2" content="Show Info Message"
                cssClass={infoCssClass} onClick={infoClick.bind(this)} />
                <MessageComponent id="msg_info_icon" severity="Info"
                showCloseIcon={showCloseIcon} visible={infoVisible}
                closed={infoClosed.bind(this)} showIcon={showIcon}>Please read the comments
                carefully</MessageComponent>
                <ButtonComponent id="btn3" content="Show Success Message"
                cssClass={successCssClass} onClick={successClick.bind(this)} />
                <MessageComponent id="msg_success_icon" severity="Success"
                showCloseIcon={showCloseIcon} closed={successClosed.bind(this)}
                visible={successVisible} showIcon={showIcon}>Your message has been sent
                successfully</MessageComponent>
                <ButtonComponent id="btn4" content="Show Warning Message"
                cssClass={warningCssClass} onClick={warningClick.bind(this)} />
                <MessageComponent id="msg_warning_icon" severity="Warning"
                showCloseIcon={showCloseIcon} closed={warningClosed.bind(this)}
                visible={warningVisible} showIcon={showIcon}>There was a problem with your
                network connection</MessageComponent>
                <ButtonComponent id="btn5" content="Show Error Message"
                cssClass={errorCssClass} onClick={errorClick.bind(this)} />
                <MessageComponent id="msg_error_icon" severity="Error"
                showCloseIcon={showCloseIcon} closed={errorClosed.bind(this)}
                visible={errorVisible} showIcon={showIcon}>A problem occurred while
                submitting your data</MessageComponent>
            </div>
        </div>
    );
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Message</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
  <style>
    /* Sample level styles */
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
    .msg-icon-section .content-section {
      margin: 0 auto;
      max-width: 450px;
      padding-top: 10px;
    }
    .msg-icon-section .e-message {
      margin: 10px 0;
    }
    .msg-icon-section .e-btn {
      display: block;
      margin: 10px 0;
    }
    .msg-icon-section .e-btn.msg-hidden {
      display: none;
    }
  </style>
</body>
</html>
```

Customization in React Message component

The Message component allows the user to customize the content display positions and appearance. This section explains the details about changing the content alignments and border styles for messages.

Content Alignment

Normally, the message content is aligned to the **left**. The Message component allows the user to align the message content in the **center** or **right** through the built-in classes `e-content-center` and `e-content-right`.

The following example demonstrates the message with different content alignments.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (<div className="msg-custom-section">
        <div className="content-section">
            <h4>Content Alignment</h4>
            <MessageComponent id="msg_content_left" content="Your license has
been activated successfully" severity="Success"></MessageComponent>
            <MessageComponent id="msg_content_center" content="The license
will expire today" cssClass="e-content-center"
severity="Warning"></MessageComponent>
            <MessageComponent id="msg_content_right" content="The license key
is invalid" cssClass="e-content-right" severity="Error"></MessageComponent>
        </div>
    </div>);
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (
        <div className="msg-custom-section">
            <div className="content-section">
                <h4>Content Alignment</h4>
                <MessageComponent id="msg_content_left" content="Your license has
been activated successfully" severity="Success"></MessageComponent>
                <MessageComponent id="msg_content_center" content="The license
will expire today" cssClass="e-content-center"
severity="Warning"></MessageComponent>
                <MessageComponent id="msg_content_right" content="The license key
is invalid" cssClass="e-content-right" severity="Error"></MessageComponent>
            </div>
        </div>
    );
}
export default App;
```

```
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Message</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
  <style>
    /* Sample level styles */
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
    .msg-custom-section .content-section {
      margin: 0 auto;
      max-width: 400px;
      padding-top: 10px;
    }
    .msg-custom-section .e-message {
      margin: 10px 0;
    }
  </style>
</body>
</html>
```

Rounded and Square

To customize the Message component's appearance, add the custom class to the message through the [cssClass](#) property. This custom class will be added to the root element. Based on this custom class, the user can override the message styles at the application level.

The following example shows the rounded and squared appearance of the message, which can be achieved by adding the `cssClass` property.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (<div className="msg-custom-section">
        <div className="content-section">
            <h4>Rounded</h4>
            <MessageComponent content="The license will expire today"
cssClass="rounded" severity="Warning"></MessageComponent>
            <h4>Square</h4>
            <MessageComponent content="The license key is invalid"
cssClass="square" severity="Error"></MessageComponent>
        </div>
    </div>);
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
    return (
        <div className="msg-custom-section">
            <div className="content-section">
                <h4>Rounded</h4>
                <MessageComponent content="The license will expire today"
cssClass="rounded" severity="Warning"></MessageComponent>
                <h4>Square</h4>
                <MessageComponent content="The license key is invalid"
cssClass="square" severity="Error"></MessageComponent>
            </div>
        </div>
    );
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
```

INDEX.HTML

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



```

    <meta name="author" content="Syncfusion" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
    <style>
        /* Sample level styles */
        #loader {
            color: #008cff;
            height: 40px;
            left: 45%;
            position: absolute;
            top: 45%;
            width: 30%;
        }
        .msg-custom-section .content-section {
            margin: 0 auto;
            max-width: 400px;
            padding-top: 10px;
        }
        .msg-custom-section .e-message {
            margin: 10px 0;
        }
        .msg-custom-section .e-message.rounded {
            border-radius: 5px;
        }
        .msg-custom-section .e-message.square {
            border-radius: 1px;
        }
    </style>
</body>
</html>

```

CSS Message

The Essential JS 2 Message has predefined CSS classes that can be defined in the HTML elements, which renders the message without any script reference. This can display a simple message with content and make the code lighter.

The following DOM structure is required to display the simple message with the content.

```
`bash
```

```
<div class="e-message">
```

```
<div class="e-msg-content">..content..</div>
</div>
`
```

The following DOM structure is required to display the simple message with the content and severity icon.

```
`bash
<div class="e-message">
  <span class="e-msg-icon"></span>
  <div class="e-msg-content">..content..</div>
</div>
`
```

The following is the available list of predefined CSS classes to make the appearance of a message.

Class	Description
-----	-----
e-message	Represents the message wrapper.
e-msg-icon	Represents the severity type icon.
e-msg-content	Represents the message content.
e-msg-close-icon	Represents the close icon.
e-info	Represents the information message.
e-success	Represents the success message.
e-warning	Represents the warning message.
e-error	Represents the error message.
e-content-center	Aligns the message content to the center.
e-content-right	Aligns the message content to the right.

The following example shows the message which renders without any script reference.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
function App() {
  return (
    <div class="msg-default-section">
      <div class="content-section">
        <div id="msg-default" class="e-message" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">Editing is restricted</div>
        </div>
        <div id="msg-info" class="e-message e-info" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">Please read the comments
carefully</div>
      </div>
    </div>
  );
}
```

```

        </div>
        <div id="msg-success" class="e-message e-success" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">Your message has been sent
successfully</div>
        </div>
        <div id="msg-warning" class="e-message e-warning" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">There was a problem with your
network connection</div>
        </div>
        <div id="msg-error" class="e-message e-error" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">A problem occurred while
submitting your data</div>
        </div>
      </div>
    </div>);
  }
  export default App;
  const root = ReactDOM.createRoot(document.getElementById('sample'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
function App() {
  return (
    <div class="msg-default-section">
      <div class="content-section">
        <div id="msg-default" class="e-message" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">Editing is restricted</div>
        </div>
        <div id="msg-info" class="e-message e-info" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">Please read the comments
carefully</div>
        </div>
        <div id="msg-success" class="e-message e-success" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">Your message has been sent
successfully</div>
        </div>
        <div id="msg-warning" class="e-message e-warning" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">There was a problem with your
network connection</div>
        </div>
        <div id="msg-error" class="e-message e-error" role="alert">
          <span class="e-msg-icon"></span>
          <div class="e-msg-content">A problem occurred while
submitting your data</div>
        </div>
      </div>
    </div>
  );
}

```

```

        </div>
      </div>
    );
  }
  export default App;
  const root = ReactDOM.createRoot(document.getElementById('sample'));
  root.render(<App />);

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Message</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
  <style>
    /* Sample level styles */
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
    .msg-default-section .content-section {
      margin: 0 auto;
      max-width: 450px;
      padding-top: 10px;
    }
    .msg-default-section .e-message {
      margin: 10px 0;
    }
  </style>
</body>
</html>

```

Template in React Message component

The message supports templates that allows the user to customize the content with a custom structure. The content can be a string, paragraph, or any other HTML element. The template can be rendered through the [content](#) property or added directly to the HTML element.

In the following sample, the Message component content is customized with HTML elements and React Button components, which are directly added to the HTML element.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { useState } from 'react';
function App() {
    const [visible, setVisible] = useState(true);
    const [cssClass, setCssClass] = useState('e-outline e-primary e-success msg-hidden');
    const showClick = () => {
        setVisible(true);
        setCssClass('e-outline e-primary e-success msg-hidden');
    };
    const dismissClick = () => {
        setVisible(false);
    };
    const closed = () => {
        setCssClass('e-outline e-primary e-success');
    };
    const contentTemplate = () => {
        return (
            <div>
                <h1>Merged pull request</h1>
                <p>Pull request #41 merged after a successful build</p>
                <ButtonComponent id="commitBtn" cssClass="e-link"
content="View commit" />
                <ButtonComponent id="closeBtn" cssClass="e-link"
content="Dismiss" onClick={dismissClick.bind(this)} />
            </div>
        );
    };
    return (<div className="msg-template-section">
        <div className="content-section">
            <ButtonComponent id="showBtn" content="Show pull request"
cssClass={cssClass} onClick={showClick.bind(this)} />
            <MessageComponent id="msg_template" visible={visible}
content={contentTemplate.bind(this)} severity="Success"
closed={closed.bind(this)} />
        </div>
    </div>);
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { MessageComponent } from '@syncfusion/ej2-react-notifications';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { useState } from 'react';
function App() {
    const [visible, setVisible] = useState<boolean>(true);
    const [cssClass, setCssClass] = useState<string>(
        'e-outline e-primary e-success msg-hidden'
    );
    const showClick = () => {
        setVisible(true);
        setCssClass('e-outline e-primary e-success msg-hidden');
    };
    const dismissClick = () => {
        setVisible(false);
    };
    const closed = () => {
        setCssClass('e-outline e-primary e-success');
    };
    const contentTemplate = () => {
        return (
            <div>
                <h1>Merged pull request</h1>
                <p>Pull request #41 merged after a successful build</p>
                <ButtonComponent id="commitBtn" cssClass="e-link"
content="View commit" />
                <ButtonComponent id="closeBtn" cssClass="e-link"
content="Dismiss" onClick={dismissClick.bind(this)} />
            </div>
        );
    };
    return (
        <div className="msg-template-section">
            <div className="content-section">
                <ButtonComponent id="showBtn" content="Show pull request"
cssClass={cssClass} onClick={showClick.bind(this)} />
                <MessageComponent id="msg_template" visible={visible}
content={contentTemplate.bind(this)} severity="Success"
closed={closed.bind(this)} />
            </div>
        </div>
    );
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
```

```

<html lang="en">
<head>
  <title>Syncfusion React Message</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
notifications/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
buttons/styles/material.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
  <style>
    /* Sample level styles */
    #loader {
      color: #008cff;
      height: 40px;
      left: 45%;
      position: absolute;
      top: 45%;
      width: 30%;
    }
    .msg-template-section .content-section {
      margin: 0 auto;
      max-width: 450px;
      padding-top: 20px;
    }
    .msg-template-section .e-btn.msg-hidden {
      display: none;
    }
    .msg-template-section .e-message h1 {
      margin: 0;
      font-size: 16px;
      font-weight: 600;
      line-height: 1.25;
    }
    .msg-template-section .e-message .e-msg-icon {
      padding: 0 4px;
      margin-top: 3px;
    }
    .msg-template-section .e-message p {
      margin: 8px 0 4px;
    }
    .msg-template-section .e-message .e-btn {

```

```
padding: 0;
}
</style>
</body>
</html>
```

Accessibility in React Message component

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

| Attributes | Purpose |

| --- | --- |

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

| **aria-label** | Provides an accessible name for the close icon. |

Keyboard interaction

The Message 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 Message component.

| Press | To do this |

| --- | --- |

| Tab / Shift + Tab | To focus the close icon in the message. |

| Enter / Space | Closes the focused close icon's message. |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

MultiSelect

Getting Started

This section explains how to create a simple [Link to the Video](#) component and configure its available functionalities in React.

To get start quickly with React MultiSelect component, you can check on this video:

Dependencies

The following list of dependencies are required to use the **MultiSelect** component in your application.

```
`javascript
```

```
|-- @syncfusion/ej2-react-dropdowns
```

```
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
\
```

Installation and configuration

You can use [Create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

```
`bash
npm install -g create-react-app
\
```

Start a new project using create-react-app command as follows

```
create-react-app quickstart --scripts-version=react-scripts-ts
cd quickstart
\
```

Adding syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. You can choose the component that you want to install.

To install MultiSelect component, use the following command

```
`bash
npm install @syncfusion/ej2-react-dropdowns --save
\
```

Adding MultiSelect component

Now, you can start adding MultiSelect component in the application. For getting started, add the MultiSelect component in `src/App.tsx` file using following code.

Add the below code in the `src/App.tsx` to initialize the MultiSelect.

```
[Class-component]
```

```
`ts
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
```

```
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  public render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id='mtselement' />
    );
  }
}

ReactDOM.render(<App />, document.getElementById('sample'));
`
```

[Functional-component]

```
`ts
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App(){
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id='mtselement' />
  );
}

ReactDOM.render(<App />, document.getElementById('sample'));
`
```

Adding CSS reference

Import the MultiSelect component required CSS references as follows in `src/App.css`.

```
`css
/ import the MultiSelect dependency styles /
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-dropdowns/styles/material.css";
```

,

Binding data source

After initialization, populate the data using [dataSource](#) property. Here, an array of string values is passed to the MultiSelect component.

[Class-component]

```
`ts
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the array of data
  private sportsData: string[] = ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf', 'Gymnastics', 'Hockey',
  'Rugby', 'Snooker', 'Tennis'];
  public render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement" dataSource={this.sportsData} />
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

,

[Functional-component]

```
`ts
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App(){
  // define the array of data
  let sportsData: string[] = ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf', 'Gymnastics', 'Hockey',
  'Rugby', 'Snooker', 'Tennis'];
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" dataSource={sportsData} />
  );
}
```

```
);
}

ReactDOM.render(<App />, document.getElementById('sample'));
`
```

Run the application

After completing the configuration required to render a basic MultiSelect, run the following command to display the output in your default browser.

```
`
npm start
`
```

[Class-component]

INDEX.JSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of data
    sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" dataSource={this.sportsData}
placeholder="Find a game"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data
    private sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" dataSource={this.sportsData}
placeholder="Find a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    let sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={sportsData}
placeholder="Find a game"/>);
    }
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={sportsData}
placeholder="Find a game" />
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Configure the Popup List

By default, the width of the popup list automatically adjusts according to the MultiSelect input element's width and the height of the popup list has '300px'.

You can also customize the suggestion list height and width using [popupHeight](#) and [popupWidth](#) properties respectively.

In the following sample, popup list's width and height are configured.

[Class-component]**INDEX.JSX**

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of data
    sportsData = ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf',
'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis'];
    render() {
        return (
            // specifies the tag for render the MultiSelect component
```

```

        <MultiSelectComponent id="mtselement" dataSource={this.sportsData}
        popupHeight="250px" popupWidth="250px" placeholder="Find a game"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data
    private sportsData: string[] = ['Badminton', 'Basketball', 'Cricket',
    'Football', 'Golf', 'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis'];
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement"
            dataSource={this.sportsData} popupHeight="250px" popupWidth="250px"
            placeholder="Find a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]**INDEX.JSX**

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    let sportsData = ['Badminton', 'Basketball', 'Cricket', 'Football',
    'Golf', 'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis'];
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={sportsData}
        popupHeight="250px" popupWidth="250px" placeholder="Find a game"/>);
    }
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    let sportsData: string[] = ['Badminton', 'Basketball', 'Cricket',
    'Football', 'Golf', 'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis'];
    return (
        // specifies the tag for render the MultiSelect component

```

```

    <MultiSelectComponent id="mtselement" dataSource={sportsData}
    popupHeight="250px" popupWidth="250px" placeholder="Find a game" />
    );
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

See Also

- [How to bind the data](#)

Data binding in React Multi select component

The MultiSelect loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports

the data type of `array` or `DataManager`.

The MultiSelect also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON,

and JSONP with the help of `DataManager` adaptors.

Fields	Type	Description
text	string	Specifies the display text of each list item.
value	number or string	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	string	Specifies the category under which the list item has to be grouped.
iconCss	string	Specifies the icon class of each list item.

When binding complex data to the MultiSelect, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of string

The MultiSelect has support to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of string
  sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
  render() {
    return (

```



```

        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={this.sportsData}
placeholder="Select a game"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
    // define the array of string
    private sportsData: string[] = ['Badminton', 'Cricket', 'Football',
'Golf', 'Tennis'];
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement"
dataSource={this.sportsData} placeholder="Select a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]**INDEX.JSX**

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of string
    const sportsData = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={sportsData}
placeholder="Select a game"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of string
    const sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
    return (
        // specifies the tag for render the MultiSelect component

```

```

    <MultiSelectComponent id="mtselement" dataSource={sportsData}
    placeholder="Select a game" />
    );
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

2. Array of object

The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **id** column and **sports** column from complex data have been mapped to the **value** field and **text** field, respectively.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  sportsData = [
    { id: 'game1', sports: 'Badminton' },
    { id: 'game2', sports: 'Football' },
    { id: 'game3', sports: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'sports', value: 'id' };
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement" dataSource={this.sportsData}
      fields={this.fields} placeholder="Select a game"/>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the JSON of data
  private sportsData: { [key: string]: Object }[] = [
    { id: 'game1', sports: 'Badminton' },
    { id: 'game2', sports: 'Football' },
    { id: 'game3', sports: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  private fields: object = { text: 'sports', value: 'id' };
  public render() {
    return (
      // specifies the tag for render the MultiSelect component

```

```

        <MultiSelectComponent id="mtselement"
        dataSource={this.sportsData} fields={this.fields} placeholder="Select a game"
        />
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData = [
        { id: 'game1', sports: 'Badminton' },
        { id: 'game2', sports: 'Football' },
        { id: 'game3', sports: 'Tennis' }
    ];
    // maps the appropriate column to fields property
    const fields = { text: 'sports', value: 'id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={sportsData}
        fields={fields} placeholder="Select a game"/>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData: { [key: string]: Object }[] = [
        { id: 'game1', sports: 'Badminton' },
        { id: 'game2', sports: 'Football' },
        { id: 'game3', sports: 'Tennis' }
    ];
    // maps the appropriate column to fields property
    const fields: object = { text: 'sports', value: 'id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={sportsData}
        fields={fields} placeholder="Select a game" />
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

3. Array of complex object

The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Code.Id** column and **Country.Name** column from complex data have been mapped to the **value** field and **text** field, respectively.

[Class-component]

INDEX.JSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the JSON of data
    countriesData = [
        { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
        { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
        { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
        { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
        { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
        { Country: { Name: 'France' }, Code: { Id: 'FR' } }
    ];
    // maps the appropriate column to fields property
    fields = { text: 'Country.Name', value: 'Code.Id' };
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" dataSource={this.countriesData}
            fields={this.fields} placeholder="Select a country"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
{% raw %}
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the JSON of data
    private countriesData: { [key: string]: Object }[] = [
        { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
        { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
        { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
        { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
        { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
        { Country: { Name: 'France' }, Code: { Id: 'FR' } }
    ];
    // maps the appropriate column to fields property
    private fields: object = { text: 'Country.Name', value: 'Code.Id' };
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
```

```

        <MultiSelectComponent id="mtselement"
        dataSource={this.countriesData} fields={this.fields} placeholder="Select a
        country" />
      );
    }
  }
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

[Functional-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // define the JSON of data
  const countriesData = [
    { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
    { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
    { Country: { Name: 'France' }, Code: { Id: 'FR' } }
  ];
  // maps the appropriate column to fields property
  const fields = { text: 'Country.Name', value: 'Code.Id' };
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" dataSource={countriesData}
    fields={fields} placeholder="Select a country"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

{% raw %}
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // define the JSON of data
  const countriesData: { [key: string]: Object }[] = [
    { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
    { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
    { Country: { Name: 'France' }, Code: { Id: 'FR' } }
  ];
  // maps the appropriate column to fields property
  const fields: object = { text: 'Country.Name', value: 'Code.Id' };
  return (
    // specifies the tag for render the MultiSelect component

```

```

    <MultiSelectComponent id="mtselement" dataSource={countriesData}
    fields={fields} placeholder="Select a country" />
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}

```

Binding remote data

The MultiSelect supports retrieval of data from remote data services with the help of **DataManager** component. The [Query](#) property is used to fetch data from the database and bind it to the MultiSelect.

The following sample displays the first 6 contacts from “Customers” table of the **Northwind** Data Service.

[Class-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // bind the DataManager instance to dataSource property
  customerData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
  });
  // bind the Query instance to query property
  query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
  // maps the appropriate column to fields property
  fields = { text: 'ContactName', value: 'CustomerID' };
  // sort the resulted items
  sortOrder = 'Ascending';
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement" query={this.query}
dataSource={this.customerData} fields={this.fields}
sortOrder={this.sortOrder} placeholder="Select a customer"/>);
    }
  }
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';

```

```

export default class App extends React.Component<{}, {}> {
  // bind the DataManager instance to dataSource property
  private customerData: DataManager = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
  });
  // bind the Query instance to query property
  private query: Query = new
Query().from('Customers').select(['ContactName', 'CustomerID']).take(6);
  // maps the appropriate column to fields property
  private fields: object = { text: 'ContactName', value: 'CustomerID' };
  // sort the resulted items
  private sortOrder: SortOrder = 'Ascending';
  public render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement" query={this.query}
dataSource={this.customerData}
      fields={this.fields} sortOrder={this.sortOrder}
placeholder="Select a customer" />
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // bind the DataManager instance to dataSource property
  const customerData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
  });
  // bind the Query instance to query property
  const query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
  // maps the appropriate column to fields property
  const fields = { text: 'ContactName', value: 'CustomerID' };
  // sort the resulted items
  const sortOrder = 'Ascending';
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" query={query}
dataSource={customerData} fields={fields} sortOrder={sortOrder}
placeholder="Select a customer"/>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const customerData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    });
    // bind the Query instance to query property
    const query: Query = new Query().from('Customers').select(['ContactName', 'CustomerID']).take(6);
    // maps the appropriate column to fields property
    const fields: object = { text: 'ContactName', value: 'CustomerID' };
    // sort the resulted items
    const sortOrder: SortOrder = 'Ascending';
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" query={query}
        dataSource={customerData}
        fields={fields} sortOrder={sortOrder} placeholder="Select a
        customer" />
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

See Also

- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)

Value binding in ##Platform_Name## Multi select control

Value binding in the MultiSelect control allows you to associate data values with each list item. This facilitates managing and retrieving selected values efficiently. The MultiSelect component provides flexibility in binding both primitive data types and complex objects.

Primitive Data Types

The MultiSelect Dropdown control provides flexible binding capabilities for primitive data types like strings and numbers. You can effortlessly bind local primitive data arrays, fetch and bind data from remote sources, and even custom data binding to suit specific requirements. Bind the value of primitive data to the [value](#) property of the MultiSelect.

Primitive data types include:

- String

- Number
- Boolean
- Null

The following sample shows the example for preselect values for primitive data type

[Class-component]

INDEX.JSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of string
    constructor(props) {
        super(props);
        this.records = ["Item 1", "Item 2", "Item 3", "Item 4", "Item 5",
            "Item 6", "Item 7", "Item 8", "Item 9", "Item 10", "Item 11", "Item 12",
            "Item 13", "Item 14", "Item 15"]
    }
    fields = { text: 'text', value: 'id' };
    value = ["Item 1", "Item 2"];
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.records}
            value={this.value} placeholder="e.g. Item 1" allowFiltering={false}
            popupHeight="200px" >
                </MultiSelectComponent>);
        )
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = { text: 'text', value: 'id' };
    // define the array of string
    private records: { [key: string]: Object }[] = [];
    private value: number[] | string[] = ["Item 1", "Item 2"];
    // define the array of string
    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    public render() {
        return (
```

```

    // specifies the tag for render the DropDownList component
    <MultiSelectComponent id="datas" dataSource={this.records}
value={this.value} placeholder="e.g. Item 1" allowFiltering={false}
popupHeight="200px" >
    </MultiSelectComponent>
    );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Object Data Types

In the MultiSelect Dropdown control, object binding allows you to bind to a dataset of objects. When [allowObjectBinding](#) is enabled, the value of the control will be an object of the same type as the selected item in the [value](#) property. This feature seamlessly binds arrays of objects, whether sourced locally, retrieved from remote endpoints, or customized to suit specific application needs.

The following sample shows the example for preselect values for object data type

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of string
    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    fields = { text: 'text', value: 'id' };
    value = [{ id: 'id1', text: 'Item 1' }, { id: 'id2', text: 'Item 2' }];
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.records}
value={this.value} placeholder="e.g. Item 1" allowObjectBinding={true}
allowFiltering={false} fields={this.fields} popupHeight="200px" >
                </MultiSelectComponent>
            )
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property

```

```

private fields: object = { text: 'text', value: 'id' };
// define the array of string
private records: { [key: string]: Object }[] = [];
private value: object[] = [{ id: 'id1', text: 'Item 1' }, { id: 'id2',
text: 'Item 2' }];
// define the array of string
constructor(props) {
  super(props);
  this.records = Array.from({ length: 150 }, (_, i) => ({
    id: 'id' + (i + 1),
    text: `Item ${i + 1}`,
  }));
}
public render() {
  return (
    // specifies the tag for render the DropDownList component
    <MultiSelectComponent id="datas" dataSource={this.records}
value={this.value} placeholder="e.g. Item 1" allowObjectBinding={true}
allowFiltering={false} fields={this.fields} popupHeight="200px" >
    </MultiSelectComponent>
  );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Templates in React Multi select component

The MultiSelect has been provided with several options to customize each list item, group title, selected value, header, and footer elements.

Item template

The content of each list item within the MultiSelect can be customized with the help of [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data's.

[Class-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // bind the DataManager instance to dataSource property
  employeeData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
  });
  // bind the Query instance to query property
  query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6);
  // maps the appropriate column to fields property
  fields = { text: 'FirstName', value: 'EmployeeID' };

```

```

    // sort the resulted items
    sortOrder = 'Ascending';
    // set the value to itemTemplate property
    itemTemplate(data) {
        return (<span><span className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
    }
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" query={this.query}
itemTemplate={this.itemTemplate} = this.itemTemplate.bind(this)}
dataSource={this.employeeData} sortOrder={this.sortOrder}
fields={this.fields} placeholder="Select an employee"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    private query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    private fields: object = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    // set the value to itemTemplate property
    public itemTemplate(data: any): JSX.Element {
        return (
            <span><span className='name'>{data.FirstName}</span><span className
='city'>{data.City}</span></span>
        );
    }
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" query={this.query}
itemTemplate={this.itemTemplate} = this.itemTemplate.bind(this)}
dataSource={this.employeeData} sortOrder={this.sortOrder}
fields={this.fields} placeholder="Select an employee" />
        );
    }
}

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    const query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    const fields = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    const sortOrder = 'Ascending';
    // set the value to itemTemplate property
    function itemTemplate(data) {
        return (<span><span className='name'>{data.FirstName}</span><span
        className='city'>{data.City}</span></span>);
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" query={query}
        itemTemplate={itemTemplate} itemTemplate = itemTemplate.bind(this)
        dataSource={employeeData} sortOrder={sortOrder} fields={fields}
        placeholder="Select an employee"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App(){
    // bind the DataManager instance to dataSource property
    const employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
```

```

const query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6);
// maps the appropriate column to fields property
const fields: object = { text: 'FirstName', value: 'EmployeeID' };
// sort the resulted items
const sortOrder: SortOrder = 'Ascending';
// set the value to itemTemplate property
function itemTemplate(data: any): JSX.Element {
    return (
        <span><span className='name'>{data.FirstName}</span><span className
='city'>{data.City}</span></span>
    );
}
return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" query={query}
itemTemplate={itemTemplate = itemTemplate.bind(this)}
dataSource={employeeData} sortOrder={sortOrder} fields={fields}
placeholder="Select an employee" />
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Value template

The currently selected value that is displayed by default on the MultiSelect input element can be customized using the [valueTemplate](#) property.

In the following sample, the selected value is displayed as a combined text of both **FirstName** and **City** in the MultiSelect input, which is separated by a hyphen.

[Class-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    fields = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    sortOrder = 'Ascending';
    // set the value to itemTemplate property
    itemTemplate(data) {

```

```

        return (<span><span className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
    }
    // set the value to valueTemplate property
    valueTemplate(data) {
        return (<span>${data.FirstName} - ${data.City}</span>);
    }
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement"
valueTemplate={this.valueTemplate} query={this.query}
itemTemplate={this.itemTemplate} sortOrder={this.sortOrder}
dataSource={this.employeeData} fields={this.fields} placeholder="Select an
employee"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    private query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    private fields: object = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    // set the value to itemTemplate property
    public itemTemplate(data: any): JSX.Element {
        return (
            <span><span className='name'>{data.FirstName}</span><span className
='city'>{data.City}</span></span>
        );
    }
    // set the value to valueTemplate property
    public valueTemplate(data: any): JSX.Element {
        return (
            <span>${data.FirstName} - ${data.City}</span>
        );
    }
    public render() {
        return (

```

```

        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement"
valueTemplate={this.valueTemplate} query={this.query}
itemTemplate={this.itemTemplate} sortOrder={this.sortOrder}
dataSource={this.employeeData} fields={this.fields} placeholder="Select an
employee" />
    );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    const query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    const fields = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    const sortOrder = 'Ascending';
    // set the value to itemTemplate property
    function itemTemplate(data) {
        return (<span><span className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
    }
    // set the value to valueTemplate property
    function valueTemplate(data) {
        return (<span>${data.FirstName} - ${data.City}</span>);
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" valueTemplate={valueTemplate}
query={query} itemTemplate={itemTemplate} sortOrder={sortOrder}
dataSource={employeeData} fields={fields} placeholder="Select an
employee"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';

```



```

import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    const query: Query = new Query().from('Employees').select(['FirstName',
    'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    const fields: object = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    const sortOrder: SortOrder = 'Ascending';
    // set the value to itemTemplate property
    function itemTemplate(data: any): JSX.Element {
        return (
            <span><span className='name'>{data.FirstName}</span><span className
            ='city'>{data.City}</span></span>
        );
    }
    // set the value to valueTemplate property
    function valueTemplate(data: any): JSX.Element {
        return (
            <span>${data.FirstName} - ${data.City}</span>
        );
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" valueTemplate={valueTemplate}
        query={query} itemTemplate={itemTemplate} sortOrder={sortOrder}
        dataSource={employeeData} fields={fields} placeholder="Select an employee" />
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Group template

The group header title under which appropriate sub-items are categorized can also be customize with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

[Class-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Predicate, Query } from
 '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';

```

```

import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // form predicate to fetch the grouped data
    groupPredicate = new Predicate('City', 'equal', 'london').or('City',
    'equal', 'seattle');
    // bind the Query instance to query property
    query = new Query().from('Employees').select(['FirstName', 'City',
    'EmployeeID']).take(6).where(this.groupPredicate);
    // maps the appropriate column to fields property
    fields = { text: 'FirstName', value: 'EmployeeID', groupBy: 'City' };
    // sort the resulted items
    sortOrder = 'Ascending';
    // set the value to groupTemplate
    groupTemplate(data) {
        return (<strong>{data.City}</strong>);
    }
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" query={this.query}
            groupTemplate={this.groupTemplate = this.groupTemplate.bind(this)}
            dataSource={this.employeeData} sortOrder={this.sortOrder}
            fields={this.fields} placeholder="Select an employee"/>);
        }
    }
    ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Predicate, Query } from
    '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // form predicate to fetch the grouped data
    private groupPredicate = new Predicate('City', 'equal',
    'london').or('City', 'equal', 'seattle');
    // bind the Query instance to query property
    private query: Query = new Query().from('Employees').select(['FirstName',
    'City', 'EmployeeID']).take(6).where(this.groupPredicate);
    // maps the appropriate column to fields property

```

```

    private fields: object = { text: 'FirstName', value: 'EmployeeID',
groupBy: 'City' };
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    // set the value to groupTemplate
    public groupTemplate(data: any): JSX.Element {
        return (
            <strong>{data.City}</strong>
        );
    }
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" query={this.query}
groupTemplate={this.groupTemplate = this.groupTemplate.bind(this)}
dataSource={this.employeeData} sortOrder={this.sortOrder}
fields={this.fields} placeholder="Select an employee" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Predicate, Query } from
'@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // form predicate to fetch the grouped data
    const groupPredicate = new Predicate('City', 'equal',
'london').or('City', 'equal', 'seattle');
    // bind the Query instance to query property
    const query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6).where(groupPredicate);
    // maps the appropriate column to fields property
    const fields = { text: 'FirstName', value: 'EmployeeID', groupBy: 'City'
};
    // sort the resulted items
    const sortOrder = 'Ascending';
    // set the value to groupTemplate
    function groupTemplate(data) {
        return (<strong>{data.City}</strong>);
    }
    return (
        // specifies the tag for render the MultiSelect component

```

```

    <MultiSelectComponent id="mtselement" query={query}
    groupTemplate={groupTemplate = groupTemplate.bind(this)}
    dataSource={employeeData} sortOrder={sortOrder} fields={fields}
    placeholder="Select an employee"/>);
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Predicate, Query } from
 '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';

function App() {
  // bind the DataManager instance to dataSource property
  const employeeData: DataManager = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
  });
  // form predicate to fetch the grouped data
  const groupPredicate = new Predicate('City', 'equal',
    'london').or('City', 'equal', 'seattle');
  // bind the Query instance to query property
  const query: Query = new Query().from('Employees').select(['FirstName',
    'City', 'EmployeeID']).take(6).where(groupPredicate);
  // maps the appropriate column to fields property
  const fields: object = { text: 'FirstName', value: 'EmployeeID', groupBy:
    'City' };
  // sort the resulted items
  const sortOrder: SortOrder = 'Ascending';
  // set the value to groupTemplate
  function groupTemplate(data: any): JSX.Element {
    return (
      <strong>{data.City}</strong>
    );
  }
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" query={query}
    groupTemplate={groupTemplate = groupTemplate.bind(this)}
    dataSource={employeeData} sortOrder={sortOrder} fields={fields}
    placeholder="Select an employee" />
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Header template

The header element is shown statically at the top of the popup list items within the MultiSelect, and any custom element can be placed as a header element using the

[headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

[Class-component]

INDEX.JSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    fields = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    sortOrder = 'Ascending';
    // set the value to header template
    headerTemplate(data) {
        return (<span className='head'><span
className='name'>Name</span><span className='city'>City</span></span>);
    }
    // set the value to item template
    itemTemplate(data) {
        return (<span className='item'><span
className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
    }
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" query={this.query}
headerTemplate={this.headerTemplate} dataSource={this.employeeData}
sortOrder={this.sortOrder} itemTemplate={this.itemTemplate}
fields={this.fields} placeholder="Select an employee"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
```

```

// bind the DataManager instance to dataSource property
private employeeData: DataManager = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
});
// bind the Query instance to query property
private query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6);
// maps the appropriate column to fields property
private fields: object = { text: 'FirstName', value: 'EmployeeID' };
// sort the resulted items
private sortOrder: SortOrder = 'Ascending';
// set the value to header template
public headerTemplate(data: any): JSX.Element {
    return (
        <span className='head'><span className='name'>Name</span><span
className='city'>City</span></span>
    );
}
// set the value to item template
public itemTemplate(data: any): JSX.Element {
    return (
        <span className='item' ><span
className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>
    );
}
public render() {
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" query={this.query}
headerTemplate={this.headerTemplate} dataSource={this.employeeData}
sortOrder={this.sortOrder} itemTemplate={this.itemTemplate}
fields={this.fields} placeholder="Select an employee" />
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
}

```

```

// bind the Query instance to query property
const query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6);
// maps the appropriate column to fields property
const fields = { text: 'FirstName', value: 'EmployeeID' };
// sort the resulted items
const sortOrder = 'Ascending';
// set the value to header template
function headerTemplate(data) {
    return (<span className='head'><span
className='name'>Name</span><span className='city'>City</span></span>);
}
// set the value to item template
function itemTemplate(data) {
    return (<span className='item'><span
className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
}
return (
// specifies the tag for render the MultiSelect component
<MultiSelectComponent id="mtselement" query={query}
headerTemplate={headerTemplate} dataSource={employeeData}
sortOrder={sortOrder} itemTemplate={itemTemplate} fields={fields}
placeholder="Select an employee"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    const query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    const fields: object = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    const sortOrder: SortOrder = 'Ascending';
    // set the value to header template
    function headerTemplate(data: any): JSX.Element {
        return (
            <span className='head'><span className='name'>Name</span><span
className='city'>City</span></span>
        );
    }
}

```

```

// set the value to item template
function itemTemplate(data: any): JSX.Element {
    return (
        <span className='item' ><span
className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>
    );
}
return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" query={query}
headerTemplate={headerTemplate} dataSource={employeeData}
sortOrder={sortOrder} itemTemplate={itemTemplate} fields={fields}
placeholder="Select an employee" />
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Footer template

The MultiSelect has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the MultiSelect.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of data
    sportsData = ["BasketBall", "Cricket", "Football", "Golf"];
    // set the value to footer template
    footerTemplate() {
        return (<span className='foot' />);
    }
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement"
footerTemplate={this.footerTemplate} dataSource={this.sportsData}
placeholder="Select a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data

```



```

    private sportsData: string[] = ["BasketBall", "Cricket", "Football",
"Golf"];
    // set the value to footer template
    public footerTemplate(): JSX.Element {
        return (
            <span className='foot' />
        );
    }
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement"
footerTemplate={this.footerTemplate} dataSource={this.sportsData}
placeholder="Select a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const sportsData = ["BasketBall", "Cricket", "Football", "Golf"];
    // set the value to footer template
    function footerTemplate() {
        return (<span className='foot' />);
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" footerTemplate={footerTemplate}
dataSource={sportsData} placeholder="Select a game"/>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App(){
    // define the array of data
    const sportsData: string[] = ["BasketBall", "Cricket", "Football",
"Golf"];
    // set the value to footer template
    function footerTemplate(): JSX.Element {
        return (
            <span className='foot' />
        );
    }
    return (

```

```
// specifies the tag for render the MultiSelect component
<MultiSelectComponent id="mtselement" footerTemplate={footerTemplate}
dataSource={sportsData} placeholder="Select a game" />
);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

No records template

The MultiSelect is provided with support to custom design the popup list content when no data is found and no matches found on search with the help of

[noRecordsTemplate](#) property.

In the following sample, popup list content displays the notification of no data available.

[Class-component]

INDEX.JSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of data
  data = [];
  // set the value to noRecords template
  noRecordsTemplate(data) {
    return (<span className='norecord'> NO DATA AVAILABLE</span>);
  }
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement"
noRecordsTemplate={this.noRecordsTemplate} =
this.noRecordsTemplate.bind(this) dataSource={this.data} placeholder="Select
an item"/>);
    }
  }
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the array of data
  private data: string[] = [];
  // set the value to noRecords template
  public noRecordsTemplate(data: any): JSX.Element {
    return (
      <span className='norecord'> NO DATA AVAILABLE</span>
    );
  }
  public render() {
    return (
```

```

        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement"
noRecordsTemplate={this.noRecordsTemplate =
this.noRecordsTemplate.bind(this)} dataSource={this.data} placeholder="Select
an item" />
    );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const data = [];
    // set the value to noRecords template
    function noRecordsTemplate(data) {
        return (<span className='norecord'> NO DATA AVAILABLE</span>);
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement"
noRecordsTemplate={noRecordsTemplate = noRecordsTemplate.bind(this)}
dataSource={data} placeholder="Select an item"/>);
    }
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const data: string[] = [];
    // set the value to noRecords template
    function noRecordsTemplate(data: any): any {
        return (
            <span className='norecord'> NO DATA AVAILABLE</span>
        );
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement"
noRecordsTemplate={noRecordsTemplate} dataSource={data} placeholder="Select
an item" />
    );
    }
ReactDOM.render(<App />, document.getElementById('sample'));

```

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the

[actionFailureTemplate](#) property.

In the following sample, when the data fetch request fails, the MultiSelect displays the notification.

[Class-component]

INDEX.JSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // bind the DataManager instance to dataSource property
  customerData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'http://services.odata.org/V4/Northwind/Northwind.svc/'
  });
  // bind the Query instance to query property
  query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
  // maps the appropriate column to fields property
  fields = { text: 'ContactName', value: 'CustomerID' };
  // set the value to action failure template
  failureTemplate(data) {
    return (<span className='action-failure'> Data fetch get
fails</span>);
  }
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement" query={this.query}
actionFailureTemplate={this.failureTemplate} =
this.failureTemplate.bind(this) dataSource={this.customerData}
fields={this.fields} placeholder="Select a customer"/>);
    )
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // bind the DataManager instance to dataSource property
  private customerData: DataManager = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
```

```

        url: 'http://services.odata.org/V4/Northwind/Northwind.svcs/'
    });
    // bind the Query instance to query property
    private query: Query = new
Query().from('Customers').select(['ContactName', 'CustomerID']).take(6);
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    // set the value to action failure template
    public failureTemplate(data: any): JSX.Element {
        return (
            <span className='action-failure'> Data fetch get fails</span>
        );
    }
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" query={this.query}
actionFailureTemplate={this.failureTemplate =
this.failureTemplate.bind(this)} dataSource={this.customerData}
fields={this.fields} placeholder="Select a customer" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const customerData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'http://services.odata.org/V4/Northwind/Northwind.svcs/'
    });
    // bind the Query instance to query property
    const query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
    // maps the appropriate column to fields property
    const fields = { text: 'ContactName', value: 'CustomerID' };
    // set the value to action failure template
    function failureTemplate(data) {
        return (<span className='action-failure'> Data fetch get
fails</span>);
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" query={query}
actionFailureTemplate={failureTemplate = failureTemplate.bind(this)}
dataSource={customerData} fields={fields} placeholder="Select a customer"/>
    );
}

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const customerData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'http://services.odata.org/V4/Northwind/Northwind.svcs/'
    });
    // bind the Query instance to query property
    const query: Query = new Query().from('Customers').select(['ContactName',
    'CustomerID']).take(6);
    // maps the appropriate column to fields property
    const fields: object = { text: 'ContactName', value: 'CustomerID' };
    // set the value to action failure template
    function failureTemplate(data: any): JSX.Element {
        return (
            <span className='action-failure'> Data fetch get fails</span>
        );
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" query={query}
        actionFailureTemplate={failureTemplate = failureTemplate.bind(this)}
        dataSource={customerData} fields={fields} placeholder="Select a customer" />
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

See Also

- [How to bind the data](#)
- [How to group the data using header](#)
- [How to customize the options in MultiSelect](#)

Grouping in React Multi select component

The MultiSelect supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using [groupBy](#) field.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the data with category
    vegetableData = [
        { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
        { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
        { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3' },
    ],
    { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
    { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
    { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
    { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
    { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
    { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
    { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
    { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
    ];
    // map the groupBy field with category column
    fields = { groupBy: 'category', text: 'vegetable', value: 'id' };
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" popupHeight='200px'
fields={this.fields} dataSource={this.vegetableData} placeholder="Select a
vegetable"/>);
        }
    }
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the data with category
    private vegetableData: { [key: string]: Object }[] = [
        { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
        { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
        { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3' },
    ],
    { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
    { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
    { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
    { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
    { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
    { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
    { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
    { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
    ];
    // map the groupBy field with category column
    private fields: object = { groupBy: 'category', text: 'vegetable', value:
'id' };
    public render() {

```

```

        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" popupHeight='200px'
            fields={this.fields} dataSource={this.vegetableData} placeholder="Select a
            vegetable" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the data with category
    let vegetableData = [
        { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
        { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
        { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3' },
    ],
    { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
    { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
    { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
    { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
    { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
    { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
    { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
    { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
    ];
    // map the groupBy field with category column
    const fields = { groupBy: 'category', text: 'vegetable', value: 'id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" popupHeight='200px' fields={fields}
        dataSource={vegetableData} placeholder="Select a vegetable"/>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the data with category
    let vegetableData: { [key: string]: Object }[] = [
        { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
        { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
        { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3' },
    ],
    { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
    { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },

```



```

    { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
    { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
    { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
    { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
    { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
    { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
  ];
  // map the groupBy field with category column
  const fields: object = { groupBy: 'category', text: 'vegetable', value:
'id' };
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" popupHeight='200px'
fields={fields} dataSource={vegetableData} placeholder="Select a vegetable"
/>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Customization

The grouping header is also provided with customization option. This allows custom designing using the `groupTemplate` property for both inline and fixed headers.

Grouping with CheckBox

Previously, there is no checkbox for group headers. Now, this feature allow to render checkbox in group header to select the group items in single selection. You can enable this feature by setting `enableGroupCheckBox` property value as **true** and `mode` property as **CheckBox**.

Inject the `CheckBoxSelection` module in the MultiSelect to use the checkbox.

[Class-component]

INDEX.JSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the data with category
  vegetableData = [
    { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
    { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
    { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3' },
  ],
  { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
  { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
  { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
  { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
  { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
  { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
  { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
  { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
];
  // map the groupBy field with category column
  fields = { groupBy: 'category', text: 'vegetable', value: 'id' };

```

```

render() {
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" popupHeight='200px'
    fields={this.fields} dataSource={this.vegetableData} placeholder="Select a
    vegetable" mode="CheckBox" enableGroupCheckBox="true" allowFiltering="true"
    showSelectAll="true" filterBarPlaceholder="Search Vegetables">
      <Inject services={[CheckBoxSelection]} />
    </MultiSelectComponent>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the data with category
  private vegetableData: { [key: string]: Object }[] = [
    { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
    { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
    { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3' },
    { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
    { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
    { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
    { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
    { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
    { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
    { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
    { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
  ];
  // map the groupBy field with category column
  private fields: object = { groupBy: 'category', text: 'vegetable', value:
'id' };
  public render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement" popupHeight='200px'
      fields={this.fields} dataSource={this.vegetableData} placeholder="Select a
      vegetable" mode="CheckBox" enableGroupCheckBox="true" allowFiltering="true"
      showSelectAll="true" filterBarPlaceholder="Search Vegetables">
        <Inject services={[CheckBoxSelection]} />
      </MultiSelectComponent>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the data with category
    const vegetableData = [
        { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
        { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
        { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3' },
    ],
        { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
        { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
        { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
        { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
        { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
        { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
        { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
        { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
    ];
    // map the groupBy field with category column
    const fields = { groupBy: 'category', text: 'vegetable', value: 'id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" popupHeight='200px' fields={fields}
        dataSource={vegetableData} placeholder="Select a vegetable" mode="CheckBox"
        enableGroupCheckBox="true" allowFiltering="true" showSelectAll="true"
        filterBarPlaceholder="Search Vegetables">
            <Inject services={[CheckBoxSelection]}/>
        </MultiSelectComponent>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the data with category
    const vegetableData: { [key: string]: Object }[] = [
        { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
        { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
        { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3' },
    ],
        { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
        { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
        { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
        { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
        { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
        { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
        { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
        { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
    ];
    // map the groupBy field with category column

```

```

const fields: object = { groupBy: 'category', text: 'vegetable', value:
'id' };
return (
  // specifies the tag for render the MultiSelect component
  <MultiSelectComponent id="mtselement" popupHeight='200px'
fields={fields} dataSource={vegetableData} placeholder="Select a vegetable"
mode="CheckBox" enableGroupCheckBox="true" allowFiltering="true"
showSelectAll="true" filterBarPlaceholder="Search Vegetables">
    <Inject services={[CheckBoxSelection]} />
  </MultiSelectComponent>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

See Also

- [Group Template support to MultiSelect.](#)

Filtering in React Multi select component

The MultiSelect has built-in support to filter data items when `allowFiltering` is enabled. The filter operation starts as soon as you start typing characters in the MultiSelect input.

To display filtered items in the popup, filter the required data and return it to the MultiSelect via [updateData](#) method by using the [filtering](#) event.

The following sample illustrates how to query the data source and pass the data to the MultiSelect through the `updateData` method in `filtering` event.

[Class-component]

INDEX.JSX

```

// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the filtering data
  searchData = [
    { index: "s1", country: "Alaska" }, { index: "s2", country:
"California" },
    { index: "s3", country: "Florida" }, { index: "s4", country:
"Georgia" }
  ];
  // maps the appropriate column to fields property
  fields = { text: "country", value: "index" };
  constructor(props) {
    super(props);
    this.onFiltering = this.onFiltering.bind(this);
  }
  // filtering event handler to filter a country
  onFiltering(args) {
    let query = new Query();

```

```

        // frame the query based on search string with filter type.
        query = (args.text !== "") ? query.where("country", "startswith",
args.text, true) : query;
        // pass the filter data source, filter query to updateData method.
        args.updateData(this.searchData, query);
    }
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" popupHeight="250px"
fields={this.fields} filtering={this.onFiltering} allowFiltering={true}
dataSource={this.searchData} placeholder="Select a country"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { FilteringEventArgs, MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the filtering data
    private searchData: { [key: string]: Object }[] = [
        { index: "s1", country: "Alaska" }, { index: "s2", country:
"California" },
        { index: "s3", country: "Florida" }, { index: "s4", country:
"Georgia" }
    ];
    // maps the appropriate column to fields property
    private fields: object = { text: "country", value: "index" };
    constructor(props: any) {
        super(props);
        this.onFiltering = this.onFiltering.bind(this);
    }
    // filtering event handler to filter a country
    public onFiltering(args: FilteringEventArgs) {
        let query = new Query();
        // frame the query based on search string with filter type.
        query = (args.text !== "") ? query.where("country", "startswith",
args.text, true) : query;
        // pass the filter data source, filter query to updateData method.
        args.updateData(this.searchData, query);
    }
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" popupHeight="250px"
fields={this.fields} filtering={this.onFiltering} allowFiltering={true}
dataSource={this.searchData} placeholder="Select a country" />
        );
    }
}

```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the filtering data
    const searchData = [
        { index: "s1", country: "Alaska" }, { index: "s2", country:
"California" },
        { index: "s3", country: "Florida" }, { index: "s4", country:
"Georgia" }
    ];
    // maps the appropriate column to fields property
    const fields = { text: "country", value: "index" };
    // filtering event handler to filter a country
    function onFiltering(args) {
        let query = new Query();
        // frame the query based on search string with filter type.
        query = (args.text !== "") ? query.where("country", "startswith",
args.text, true) : query;
        // pass the filter data source, filter query to updateData method.
        args.updateData(this.searchData, query);
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" popupHeight="250px" fields={fields}
filtering={onFiltering} allowFiltering={true} dataSource={searchData}
placeholder="Select a country"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { FilteringEventArgs, MultiSelectComponent } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App(){
    // define the filtering data
    const searchData: { [key: string]: Object }[] = [
        { index: "s1", country: "Alaska" }, { index: "s2", country:
"California" },
        { index: "s3", country: "Florida" }, { index: "s4", country:
"Georgia" }
    ];
    // maps the appropriate column to fields property
    const fields: object = { text: "country", value: "index" };
```

```

// filtering event handler to filter a country
function onFiltering(args: FilteringEventArgs) {
    let query = new Query();
    // frame the query based on search string with filter type.
    query = (args.text !== "") ? query.where("country", "startswith",
args.text, true) : query;
    // pass the filter data source, filter query to updateData method.
    args.updateData(this.searchData, query);
}
return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" popupHeight="250px"
fields={fields} filtering={onFiltering} allowFiltering={true}
dataSource={searchData} placeholder="Select a country" />
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the MultiSelect. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

[Class-component]

INDEX.JSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    searchData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    fields = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    query = new Query().select(['ContactName', 'CustomerID']).take(7);
    // sort the resulted items
    sortOrder = 'Ascending';
    constructor(props) {
        super(props);
        this.onFiltering = this.onFiltering.bind(this);
    }
    // filtering event handler to filter a customer
    onFiltering(e) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);

```

```

    }
    else {
        // restrict the remote request until search key contains 3
        characters.
        if (e.text.length < 3) {
            return;
        }
        let query = new Query().select(['ContactName', 'CustomerID']);
        query = (e.text !== '') ? query.where('ContactName',
        'startswith', e.text, true) : query;
        e.updateData(this.searchData, query);
    }
}
render() {
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" allowFiltering={true}
        popupHeight="250px" filtering={this.onFiltering} sortOrder={this.sortOrder}
        query={this.query} dataSource={this.searchData} fields={this.fields}
        placeholder="Select a customer"/>);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { FilteringEventArgs, MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private searchData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    private query: Query = new Query().select(['ContactName',
    'CustomerID']).take(7);
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    constructor(props: any) {
        super(props);
        this.onFiltering = this.onFiltering.bind(this);
    }
    // filtering event handler to filter a customer
    public onFiltering(e: FilteringEventArgs) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        }
    }
}

```



```

    } else {
        // restrict the remote request until search key contains 3
        characters.
        if (e.text.length < 3) { return; }
        let query: Query = new Query().select(['ContactName',
        'CustomerID']);
        query = (e.text !== '') ? query.where('ContactName',
        'startswith', e.text, true) : query;
        e.updateData(this.searchData, query);
    }
}
public render() {
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" allowFiltering={true}
        popupHeight="250px" filtering={this.onFiltering} sortOrder={this.sortOrder}
        query={this.query} dataSource={this.searchData} fields={this.fields}
        placeholder="Select a customer" />
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const searchData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    const fields = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    const query = new Query().select(['ContactName', 'CustomerID']).take(7);
    // sort the resulted items
    const sortOrder = 'Ascending';
    // filtering event handler to filter a customer
    function onFiltering(e) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        }
        else {
            // restrict the remote request until search key contains 3
            characters.
            if (e.text.length < 3) {
                return;
            }

```

```

    }
    let query = new Query().select(['ContactName', 'CustomerID']);
    query = (e.text !== '') ? query.where('ContactName',
'startswith', e.text, true) : query;
    e.updateData(this.searchData, query);
  }
}
return (
  // specifies the tag for render the MultiSelect component
  <MultiSelectComponent id="mtselement" allowFiltering={true}
popupHeight="250px" filtering={onFiltering} sortOrder={sortOrder}
query={query} dataSource={searchData} fields={fields} placeholder="Select a
customer"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { FilteringEventArgs, MultiSelectComponent } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // bind the DataManager instance to dataSource property
  const searchData: DataManager = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
  });
  // maps the appropriate column to fields property
  const fields: object = { text: 'ContactName', value: 'CustomerID' };
  // bind the Query instance to query property
  const query: Query = new Query().select(['ContactName',
'CustomerID']).take(7);
  // sort the resulted items
  const sortOrder: SortOrder = 'Ascending';
  // filtering event handler to filter a customer
  function onFiltering(e: FilteringEventArgs) {
    // load overall data when search key empty.
    if (e.text === '') {
      e.updateData(this.searchData);
    } else {
      // restrict the remote request until search key contains 3
characters.
      if (e.text.length < 3) { return; }
      let query: Query = new Query().select(['ContactName',
'CustomerID']);
      query = (e.text !== '') ? query.where('ContactName',
'startswith', e.text, true) : query;
      e.updateData(this.searchData, query);
    }
  }
  return (

```

```
// specifies the tag for render the MultiSelect component
<MultiSelectComponent id="mtselement" allowFiltering={true}
popupHeight="250px" filtering={onFiltering} sortOrder={sortOrder}
query={query} dataSource={searchData} fields={fields} placeholder="Select a
customer" />
);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Change the filter type

While filtering, you can change the filter type to `contains`, `startsWith`, or `endsWith` for string type within the filter event handler.

In the following examples, data filtering is done with `endsWith` type.

[Class-component]

INDEX.JSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    searchData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    fields = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    query = new Query().select(['ContactName', 'CustomerID']).take(7);
    // sort the resulted items
    sortOrder = 'Ascending';
    constructor(props) {
        super(props);
        this.onFiltering = this.onFiltering.bind(this);
    }
    // filtering event handler to filter a customer
    onFiltering(e) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        }
        else {
            let query = new Query().select(["ContactName", "CustomerID"]);
            // change the type of filtering
            query = (e.text !== '') ? query.where('ContactName', 'endsWith',
e.text, true) : query;
            e.updateData(this.searchData, query);
        }
    }
}
```

```

render() {
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" allowFiltering={true}
        popupHeight="250px" filtering={this.onFiltering} query={this.query}
        sortOrder={this.sortOrder} dataSource={this.searchData} fields={this.fields}
        placeholder="Select a customer"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { FilteringEventArgs, MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private searchData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    private query: Query = new Query().select(['ContactName',
'CustomerID']).take(7);
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    constructor(props: any) {
        super(props);
        this.onFiltering = this.onFiltering.bind(this);
    }
    // filtering event handler to filter a customer
    public onFiltering(e: FilteringEventArgs) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        } else {
            let query: Query = new Query().select(["ContactName",
'CustomerID']);
            // change the type of filtering
            query = (e.text !== '') ? query.where('ContactName', 'endsWith',
e.text, true) : query;
            e.updateData(this.searchData, query);
        }
    }
    public render() {
        return (
            // specifies the tag for render the MultiSelect component

```

```

        <MultiSelectComponent id="mtselement" allowFiltering={true}
        popupHeight="250px" filtering={this.onFiltering} query={this.query}
        sortOrder={this.sortOrder} dataSource={this.searchData} fields={this.fields}
        placeholder="Select a customer" />
      );
    }
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // bind the DataManager instance to dataSource property
  const searchData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
  });
  // maps the appropriate column to fields property
  const fields = { text: 'ContactName', value: 'CustomerID' };
  // bind the Query instance to query property
  const query = new Query().select(['ContactName', 'CustomerID']).take(7);
  // sort the resulted items
  const sortOrder = 'Ascending';
  // filtering event handler to filter a customer
  function onFiltering(e) {
    // load overall data when search key empty.
    if (e.text === '') {
      e.updateData(this.searchData);
    }
    else {
      let query = new Query().select(["ContactName", "CustomerID"]);
      // change the type of filtering
      query = (e.text !== '') ? query.where('ContactName', 'endsWith',
e.text, true) : query;
      e.updateData(this.searchData, query);
    }
  }
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" allowFiltering={true}
    popupHeight="250px" filtering={onFiltering} query={query}
    sortOrder={sortOrder} dataSource={searchData} fields={fields}
    placeholder="Select a customer"/>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { FilteringEventArgs, MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const searchData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    const fields: object = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    const query: Query = new Query().select(['ContactName',
'CustomerID']).take(7);
    // sort the resulted items
    const sortOrder: SortOrder = 'Ascending';
    // filtering event handler to filter a customer
    function onFiltering(e: FilteringEventArgs) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        } else {
            let query: Query = new Query().select(["ContactName",
'CustomerID']);
            // change the type of filtering
            query = (e.text !== '') ? query.where('ContactName', 'endsWith',
e.text, true) : query;
            e.updateData(this.searchData, query);
        }
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" allowFiltering={true}
popupHeight="250px" filtering={onFiltering} query={query}
sortOrder={sortOrder} dataSource={searchData} fields={fields}
placeholder="Select a customer" />
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the **where** clause.

The following example shows how to perform case-sensitive filter.

[Class-component]

INDEX.JSX

```

import { Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the JSON of data
    sportsData = [
        { id: 'game1', sports: 'Badminton' },
        { id: 'game2', sports: 'Tennis' },
        { id: 'game3', sports: 'Football' }
    ];
    // maps the appropriate column to fields property
    fields = { text: 'sports', value: 'id' };
    // sort the resulted items
    sortOrder = 'Ascending';
    // filtering event handler to filter a customer
    onFiltering = (e) => {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.sportsData);
        }
        else {
            let query = new Query().select(["sports", "id"]);
            // enable the case sensitive filtering by passing false to 4th
            // parameter.
            query = (e.text !== '') ? query.where('sports', 'startsWith',
            e.text, false) : query;
            e.updateData(this.sportsData, query);
        }
    };
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" dataSource={this.sportsData}
            fields={this.fields} placeholder="Select a game" allowFiltering={true}
            filtering={this.onFiltering = this.onFiltering.bind(this)}
            sortOrder={this.sortOrder}/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { FilteringEventArgs, MultiSelectComponent } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the JSON of data
    private sportsData: { [key: string]: Object }[] = [
        { id: 'game1', sports: 'Badminton' },
        { id: 'game2', sports: 'Tennis' },
    ];

```

```

        { id: 'game3', sports: 'Football' }
    ];
    // maps the appropriate column to fields property
    private fields: object = { text: 'sports', value: 'id' };
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    // filtering event handler to filter a customer
    public onFiltering = (e: FilteringEventArgs) => {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.sportsData);
        } else {
            let query: Query = new Query().select(["sports", "id"]);
            // enable the case sensitive filtering by passing false to 4th
parameter.
            query = (e.text !== '') ? query.where('sports', 'startsWith',
e.text, false) : query;
            e.updateData(this.sportsData, query);
        }
    }
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement"
dataSource={this.sportsData} fields={this.fields} placeholder="Select a game"
allowFiltering={true} filtering={this.onFiltering} =
this.onFiltering.bind(this)} sortOrder={this.sortOrder} />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData = [
        { id: 'game1', sports: 'Badminton' },
        { id: 'game2', sports: 'Tennis' },
        { id: 'game3', sports: 'Football' }
    ];
    // maps the appropriate column to fields property
    const fields = { text: 'sports', value: 'id' };
    // sort the resulted items
    const sortOrder = 'Ascending';
    // filtering event handler to filter a customer
    function onFiltering(e) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.sportsData);
        }
    }
}

```



```

    }
    else {
        let query = new Query().select(["sports", "id"]);
        // enable the case sensitive filtering by passing false to 4th
parameter.
        query = (e.text !== '') ? query.where('sports', 'startsWith',
e.text, false) : query;
        e.updateData(this.sportsData, query);
    }
}
return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" dataSource={sportsData}
fields={fields} placeholder="Select a game" allowFiltering={true}
filtering={onFiltering = onFiltering.bind(this)} sortOrder={sortOrder}/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { FilteringEventArgs, MultiSelectComponent } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData: { [key: string]: Object }[] = [
        { id: 'game1', sports: 'Badminton' },
        { id: 'game2', sports: 'Tennis' },
        { id: 'game3', sports: 'Football' }
    ];
    // maps the appropriate column to fields property
    const fields: object = { text: 'sports', value: 'id' };
    // sort the resulted items
    const sortOrder: SortOrder = 'Ascending';
    // filtering event handler to filter a customer
    function onFiltering(e: FilteringEventArgs) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.sportsData);
        } else {
            let query: Query = new Query().select(["sports", "id"]);
            // enable the case sensitive filtering by passing false to 4th
parameter.
            query = (e.text !== '') ? query.where('sports', 'startsWith',
e.text, false) : query;
            e.updateData(this.sportsData, query);
        }
    }
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={sportsData}
fields={fields} placeholder="Select a game" allowFiltering={true}
filtering={onFiltering = onFiltering.bind(this)} sortOrder={sortOrder} />
    );
}

```

```

    );
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

Diacritics Filtering

MultiSelect supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

In the following sample, data with diacritics are bound as dataSource for MultiSelect.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  diacriticsData = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aeromodelaje',
    'Águilas',
    'Ajedrez',
    'Ala Delta',
    'Álbumes de Música',
    'Alusivos',
    'Análisis de Escritura a Mano'
  ];
  render() {
    return (
      <MultiSelectComponent id="diacritics" ignoreAccent={true}
        allowFiltering={true} dataSource={this.diacriticsData} placeholder="e.g:
        aero"/>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  private diacriticsData: string[] = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aeromodelaje',
    'Águilas',
    'Ajedrez',
    'Ala Delta',
    'Álbumes de Música',
    'Alusivos',
    'Análisis de Escritura a Mano'];
  public render() {

```

```

    return (
      <MultiSelectComponent id="diacritics" ignoreAccent={true}
allowFiltering={true} dataSource={this.diacriticsData} placeholder="e.g:
aero" />
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  const diacriticsData = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aeromodelaje',
    'Águilas',
    'Ajedrez',
    'Ala Delta',
    'Álbumes de Música',
    'Alusivos',
    'Análisis de Escritura a Mano'
  ];
  return (<MultiSelectComponent id="diacritics" ignoreAccent={true}
allowFiltering={true} dataSource={diacriticsData} placeholder="e.g: aero"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App(){
  const diacriticsData: string[] = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aeromodelaje',
    'Águilas',
    'Ajedrez',
    'Ala Delta',
    'Álbumes de Música',
    'Alusivos',
    'Análisis de Escritura a Mano'];
  return (
    <MultiSelectComponent id="diacritics" ignoreAccent={true}
allowFiltering={true} dataSource={diacriticsData} placeholder="e.g: aero" />
  );
}

```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

See Also

- [How to bind the data](#)
- [How to group the data using header](#)
- [How to add custom value to the MultiSelect](#)

Custom value in React Multi select component

The MultiSelect allows user to add a new non-present option to the component value when [allowCustomValue](#) is enabled.

while selecting the new custom value `customValueSelection` event will be triggered.

The following sample demonstrates configuration of custom value support with the MultiSelect component.

[Class-component]

INDEX.JSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  sportsData = [
    { id: 'game1', sports: 'Badminton' },
    { id: 'game2', sports: 'Football' },
    { id: 'game3', sports: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'sports', value: 'id' };
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement" dataSource={this.sportsData}
        fields={this.fields} placeholder="Select a game" allowCustomValue={true}/>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the JSON of data
  private sportsData: { [key: string]: Object }[] = [
    { id: 'game1', sports: 'Badminton' },
    { id: 'game2', sports: 'Football' },
    { id: 'game3', sports: 'Tennis' }
  ];
};
```

```

// maps the appropriate column to fields property
private fields: object = { text: 'sports', value: 'id' };
public render() {
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement"
        dataSource={this.sportsData} fields={this.fields} placeholder="Select a game"
        allowCustomValue={true} />
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData = [
        { id: 'game1', sports: 'Badminton' },
        { id: 'game2', sports: 'Football' },
        { id: 'game3', sports: 'Tennis' }
    ];
    // maps the appropriate column to fields property
    const fields = { text: 'sports', value: 'id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={sportsData}
        fields={fields} placeholder="Select a game" allowCustomValue={true}/>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData: { [key: string]: Object }[] = [
        { id: 'game1', sports: 'Badminton' },
        { id: 'game2', sports: 'Football' },
        { id: 'game3', sports: 'Tennis' }
    ];
    // maps the appropriate column to fields property
    const fields: object = { text: 'sports', value: 'id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="mtselement" dataSource={sportsData}
        fields={fields} placeholder="Select a game" allowCustomValue={true} />
    );
}

```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

Virtualization in MultiSelect Dropdown

MultiSelect Dropdown virtualization is a technique used to efficiently render extensive lists of items while minimizing the impact on performance. This method is particularly advantageous when dealing with large datasets because it ensures that only a fixed number of DOM (Document Object Model) elements are created. When scrolling through the list, existing DOM elements are reused to display relevant data instead of generating new elements for each item. This recycling process is managed internally.

During virtual scrolling, the data retrieved from the data source depends on the popup height and the calculation of the list item height. Enabling the [enableVirtualization](#) option in a MultiSelect Dropdown activates this virtualization technique.

When fetching data from the data source, the [actionBegin](#) event is triggered before data retrieval begins. Then, the [actionComplete](#) event is triggered once the data is successfully fetched.

Furthermore, Incremental Search is supported with virtualization in the MultiSelect component. When a key is typed, the focus is moved to the respective element in the open popup state. In the closed popup state, the popup opens, and focus is moved to the respective element in the popup list based on the typed key. The Incremental Search functionality is well-suited for scenarios involving remote data binding.

Binding local data

The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property. When using virtual scrolling, the list updates based on the scroll offset value, triggering a request to fetch more data from the server.

In the following example, `id` column and `text` column from complex data have been mapped to the `value` field and `text` field, respectively.

[Class-component]

INDEX.JSX

```
import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of string
  constructor(props) {
    super(props);
    this.records = Array.from({ length: 150 }, (_, i) => ({
      id: 'id' + (i + 1),
      text: `Item ${i + 1}`,
    }));
  }
  fields = { text: 'text', value: 'id' };
  render() {
    return (
      // specifies the tag for render the DropDownList component
```

```

        <MultiSelectComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={false}
fields={this.fields} popupHeight="200px" >
            <Inject services={[VirtualScroll]} />
        </MultiSelectComponent>;
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = { text: 'text', value: 'id' };
    // define the array of string
    private records: { [key: string]: Object }[] = [];

    // define the array of string
    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={false}
fields={this.fields} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </MultiSelectComponent>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Binding remote data

The MultiSelect supports retrieval of data from remote data services with the help of **DataManager** component. When using remote data, it initially fetches all the data from the server, triggering the **actionBegin** and **actionComplete** events, and then stores the data locally. During virtual scrolling, additional data is retrieved from the locally stored data, triggering the **actionBegin** and **actionComplete** events at that time as well.

The following sample displays the OrderId from the **Orders** Data Service.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, WebApiAdaptor, Query } from '@syncfusion/ej2-data';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    customerData = new DataManager({
        url: 'https://services.syncfusion.com/react/production/api/Orders',
        adaptor: new WebApiAdaptor,
        crossDomain: true
    });
    customerField = { text: 'OrderID', value: 'OrderID' };
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.customerData}
placeholder="OrderID" enableVirtualization={true} allowFiltering={true}
fields={this.customerField} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </MultiSelectComponent>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import { DataManager, WebApiAdaptor, Query } from '@syncfusion/ej2-data';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private customerField: object = { text: 'OrderID', value: 'OrderID' };

    private customerData: DataManager = new DataManager({
        url: 'https://services.syncfusion.com/react/production/api/Orders',
        adaptor: new WebApiAdaptor,
        crossDomain: true
    });
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.customerData}
placeholder="OrderID" enableVirtualization={true} allowFiltering={true}
fields={this.customerField} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </MultiSelectComponent>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```


Customizing items count in virtualization

When the `enableVirtualization` property is enabled, the `take` property provided by the user within the Query parameter at the initial state or during the `actionBegin` event will be considered. Internally, it calculates the items that fit onto the current page (i.e., probably twice the amount of the popup's height). If the user-provided take value is less than the minimum number of items that fit into the popup, the user-provided take value will not be considered.

The following sample shows the example for Customizing items count in virtualization.

[Class-component]

INDEX.JSX

```
import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import { Query } from '@syncfusion/ej2-data';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of string
  constructor(props) {
    super(props);
    this.records = Array.from({ length: 150 }, (_, i) => ({
      id: 'id' + (i + 1),
      text: `Item ${i + 1}`,
    }));
  }
  fields = { text: 'text', value: 'id' };
  // bind the Query instance to query property
  query = new Query().take(20);
  Begin(args) {
    args.Query = new Query().take(25);
  }
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <MultiSelectComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} query={this.query}
allowFiltering={false} actionBegin={this.Begin} fields={this.fields}
popupHeight="200px" >
        <Inject services={[VirtualScroll]} />
      </MultiSelectComponent>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import { Query } from '@syncfusion/ej2-data';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // maps the appropriate column to fields property
  private fields: object = { text: 'text', value: 'id' };
}
```

```

// define the array of string
private records: { [key: string]: Object }[] = [];
private query: Query = new Query().take(20);
public Begin(e: any): void {
    e.query = new Query().take(25);
}
// define the array of string
constructor(props) {
    super(props);
    this.records = Array.from({ length: 150 }, (_, i) => ({
        id: 'id' + (i + 1),
        text: `Item ${i + 1}`,
    }));
}
public render() {
    return (
        // specifies the tag for render the DropDownList component
        <MultiSelectComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" query={this.query} actionBegin={this.Begin}
enableVirtualization={true} allowFiltering={false} fields={this.fields}
popupHeight="200px" >
        <Inject services={[VirtualScroll]}/>
    </MultiSelectComponent>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Grouping with virtualization

The MultiSelect component supports grouping with Virtualization. It allows you to organize elements into groups based on different categories. Each item in the list can be classified using the [groupBy](#) field in the data table. After grouping, virtualization works similarly to local data binding, providing a seamless user experience. When the data source is bound to remote data, an initial request is made to retrieve all data for the purpose of grouping. Subsequently, the grouped data works in the same way as local data binding on virtualization.

The following sample shows the example for Grouping with Virtualization.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of string
    records = [];
    constructor(props) {
        super(props);
        for (let i = 1; i <= 150; i++) {
            let item = {};
            item.id = 'id' + i;
            item.text = `Item ${i}`;

```

```

        // Generate a random number between 1 and 4 to determine the
group
        const randomGroup = Math.floor(Math.random() * 4) + 1;
        switch (randomGroup) {
            case 1:
                item.group = 'Group A';
                break;
            case 2:
                item.group = 'Group B';
                break;
            case 3:
                item.group = 'Group C';
                break;
            case 4:
                item.group = 'Group D';
                break;
            default:
                break;
        }
        this.records.push(item);
    }
}
fields = { groupBy: 'group', text: 'text', value: 'id' };
render() {
    return (
        // specifies the tag for render the DropDownList component
        <MultiSelectComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={true}
fields={this.fields} popupHeight="200px" >
            <Inject services={[VirtualScroll]} />
        </MultiSelectComponent>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = { groupBy: 'group', text: 'text', value: 'id' };
    // define the array of string
    private records: { [key: string]: Object }[] = [];

    // define the array of string
    constructor(props) {
        super(props);
        for (let i = 1; i <= 150; i++) {
            const item: { id: string, text: string, group: string } = {
                id: 'id' + i,
                text: `Item ${i}`,
                group: ''
            };
        }
    }
}

```

```

// Generate a random number between 1 and 4 to determine the group
const randomGroup = Math.floor(Math.random() * 4) + 1;
switch (randomGroup) {
  case 1:
    item.group = 'Group A';
    break;
  case 2:
    item.group = 'Group B';
    break;
  case 3:
    item.group = 'Group C';
    break;
  case 4:
    item.group = 'Group D';
    break;
  default:
    break;
}
this.records.push(item);
}
}
public render() {
  return (
    // specifies the tag for render the DropDownList component
    <MultiSelectComponent id="datas" dataSource={this.records}
    placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={true}
    fields={this.fields} popupHeight="200px" >
      <Inject services={[VirtualScroll]} />
    </MultiSelectComponent>
  );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Filtering with virtualization

The MultiSelect component supports Filtering with Virtualization. The MultiSelect includes a built-in feature that enables data filtering when the [allowFiltering](#) option is enabled. In the context of Virtual Scrolling, the filtering process operates in response to the typed characters. Specifically, the MultiSelect sends a request to the server, utilizing the full data source, to achieve filtering. Before initiating the request, an action event is triggered. Upon successful retrieval of data from the server, an action complete event is triggered. The initial data is loaded when the popup is opened. Whether the filter list has a selection or not, the popup closes.

The following sample shows the example for Filtering with Virtualization.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of string

```

```

    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    fields = { text: 'text', value: 'id' };
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={true}
fields={this.fields} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </MultiSelectComponent>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = { text: 'text', value: 'id' };
    // define the array of string
    private records: { [key: string]: Object }[] = [];

    // define the array of string
    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={true}
fields={this.fields} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </MultiSelectComponent>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Checkbox with virtualization

The MultiSelect component supports CheckBox selection with Virtualization. The MultiSelect comes with integrated functionality that allows for the selection of multiple values using checkboxes when the [mode](#) property is configured to **CheckBox**. In the context of Virtual Scrolling, the checkbox render with each list element. based on the checkbox selection and unselection, component value property updated with respective values.

The following sample shows the example for checkbox with Virtualization.

[Class-component]

INDEX.JSX

```
import { MultiSelectComponent, Inject, VirtualScroll, CheckBoxSelection }
from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of string
  constructor(props) {
    super(props);
    this.records = Array.from({ length: 150 }, (_, i) => ({
      id: 'id' + (i + 1),
      text: `Item ${i + 1}`,
    }));
  }
  fields = { text: 'text', value: 'id' };
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <MultiSelectComponent id="datas" dataSource={this.records}
      mode="CheckBox" placeholder="e.g. Item 1" enableVirtualization={true}
      allowFiltering={false} fields={this.fields} popupHeight="200px" >
        <Inject services={[VirtualScroll, CheckBoxSelection]} />
      </MultiSelectComponent>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MultiSelectComponent, Inject, VirtualScroll, CheckBoxSelection }
from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // maps the appropriate column to fields property
  private fields: object = { text: 'text', value: 'id' };
  // define the array of string
  private records: { [key: string]: Object }[] = [];

  // define the array of string
  constructor(props) {
    super(props);
    this.records = Array.from({ length: 150 }, (_, i) => ({
      id: 'id' + (i + 1),
```

```

        text: `Item ${i + 1}`,
    }));
}
public render() {
    return (
        // specifies the tag for render the DropDownList component
        <MultiSelectComponent id="datas" dataSource={this.records}
mode="CheckBox" placeholder="e.g. Item 1" enableVirtualization={true}
allowFiltering={false} fields={this.fields} popupHeight="200px" >
        <Inject services={[VirtualScroll, CheckBoxSelection]}/>
        </MultiSelectComponent>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Custom value with virtualization

The MultiSelect component supports custom value with Virtualization. When the [allowCustomValue](#) property is enabled, the MultiSelect enables users to include a new option not currently available in the component value. Upon selecting this newly added custom value, the MultiSelect triggers the [customValueSelection](#) event and also custom value will be added to the end of the complete list.

The following sample shows the example for custom value with Virtualization.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of string
    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    fields = { text: 'text', value: 'id' };
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={false}
allowCustomValue={true} fields={this.fields} popupHeight="200px" >
            <Inject services={[VirtualScroll]} />
            </MultiSelectComponent>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
  // maps the appropriate column to fields property
  private fields: object = { text: 'text', value: 'id' };
  // define the array of string
  private records: { [key: string]: Object }[] = [];

  // define the array of string
  constructor(props) {
    super(props);
    this.records = Array.from({ length: 150 }, (_, i) => ({
      id: 'id' + (i + 1),
      text: `Item ${i + 1}`,
    }));
  }
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <MultiSelectComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowCustomValue={true}
allowFiltering={false} fields={this.fields} popupHeight="200px" >
        <Inject services={[VirtualScroll]} />
      </MultiSelectComponent>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Preselect values with virtualization

The MultiSelect component extends its support for preselected values with Virtualization. When binding values from local or remote data to the MultiSelect component, the corresponding data value is fetched from the server and promptly updated within the component. Moreover, when binding a custom value to the component, the value is updated within the component, and the bound custom value is seamlessly appended to the end of the complete list.

The following sample shows the example for Preselect value with Virtualization.

[Class-component]

INDEX.JSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of string
  constructor(props) {
    super(props);
    this.records = Array.from({ length: 150 }, (_, i) => ({
      id: 'id' + (i + 1),
      text: `Item ${i + 1}`,
    }));
  }
}

```



```

    }
    fields = { text: 'text', value: 'id' };
    value = ['id1', 'id2', 'id3'];
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.records}
            value={this.value} placeholder="e.g. Item 1" enableVirtualization={true}
            allowFiltering={false} fields={this.fields} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </MultiSelectComponent>);
        )
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent, Inject, VirtualScroll } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = { text: 'text', value: 'id' };
    // define the array of string
    private records: { [key: string]: Object }[] = [];
    private value: string[] = ["id1", "id2", "id3"];
    // define the array of string
    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <MultiSelectComponent id="datas" dataSource={this.records}
            value={this.value} placeholder="e.g. Item 1" enableVirtualization={true}
            allowFiltering={false} fields={this.fields} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </MultiSelectComponent>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Checkbox in React Multi select component

The MultiSelect has built-in support to select multiple values through checkbox, when [mode](#) property set as `CheckBox`.

To use checkbox, inject the `CheckBoxSelection` module in the MultiSelect.

[Class-component]

INDEX.JSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
 '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' },
    { Id: 'game4', Game: 'Golf' },
    { Id: 'game5', Game: 'Cricket' },
    { Id: 'game6', Game: 'Handball' },
    { Id: 'game7', Game: 'Karate' },
    { Id: 'game8', Game: 'Fencing' },
    { Id: 'game9', Game: 'Boxing' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'Game', value: 'Id' };
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="checkbox" dataSource={this.sportsData}
fields={this.fields} placeholder="Select game" mode="CheckBox">
        <Inject services={ [CheckBoxSelection]} />
      </MultiSelectComponent>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
 '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the JSON of data
  private sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' },
    { Id: 'game4', Game: 'Golf' },
    { Id: 'game5', Game: 'Cricket' },
    { Id: 'game6', Game: 'Handball' },
    { Id: 'game7', Game: 'Karate' },
    { Id: 'game8', Game: 'Fencing' },
    { Id: 'game9', Game: 'Boxing' }
  ];
  // maps the appropriate column to fields property
  private fields: object = { text: 'Game', value: 'Id' };
  public render() {
    return (
      // specifies the tag for render the MultiSelect component

```

```

        <MultiSelectComponent id="checkbox" dataSource={this.sportsData}
            fields={this.fields} placeholder="Select game"
mode="CheckBox">
            <Inject services={[CheckBoxSelection]} />
        </MultiSelectComponent>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
    ];
    // maps the appropriate column to fields property
    const fields = { text: 'Game', value: 'Id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="checkbox" dataSource={sportsData}
fields={fields} placeholder="Select game" mode="CheckBox">
            <Inject services={[CheckBoxSelection]} />
        </MultiSelectComponent>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData: { [key: string]: Object }[] = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
    ];

```

```

    { Id: 'game6', Game: 'Handball' },
    { Id: 'game7', Game: 'Karate' },
    { Id: 'game8', Game: 'Fencing' },
    { Id: 'game9', Game: 'Boxing' }
  ];
  // maps the appropriate column to fields property
  const fields: object = { text: 'Game', value: 'Id' };
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="checkbox" dataSource={sportsData}
      fields={fields} placeholder="Select game" mode="CheckBox">
      <Inject services={[CheckBoxSelection]} />
    </MultiSelectComponent>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Select All

The MultiSelect component has in-built support to select the all list items using **Select All** options in the header.

When the [showSelectAll](#) property is set to true, by default Select All text will show. You can customize the name attribute of the Select All option by using [selectAllText](#).

For the unSelect All option, by default unSelect All text will show. You can customize the name attribute of the unSelect All option by using [unselectAllText](#).

[Class-component]

INDEX.JSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' },
    { Id: 'game4', Game: 'Golf' },
    { Id: 'game5', Game: 'Cricket' },
    { Id: 'game6', Game: 'Handball' },
    { Id: 'game7', Game: 'Karate' },
    { Id: 'game8', Game: 'Fencing' },
    { Id: 'game9', Game: 'Boxing' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'Game', value: 'Id' };
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="checkbox" dataSource={this.sportsData}
        fields={this.fields} placeholder="Select game" mode="CheckBox"

```

```

selectAllText="Select All" unSelectAllText="unSelect All"
showSelectAll={true}>
    <Inject services={[CheckBoxSelection]} />
  </MultiSelectComponent>);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the JSON of data
  private sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' },
    { Id: 'game4', Game: 'Golf' },
    { Id: 'game5', Game: 'Cricket' },
    { Id: 'game6', Game: 'Handball' },
    { Id: 'game7', Game: 'Karate' },
    { Id: 'game8', Game: 'Fencing' },
    { Id: 'game9', Game: 'Boxing' }
  ];
  // maps the appropriate column to fields property
  private fields: object = { text: 'Game', value: 'Id' };
  public render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="checkbox" dataSource={this.sportsData}
        fields={this.fields} placeholder="Select game"
        mode="CheckBox" selectAllText="Select All" unSelectAllText="unSelect All"
        showSelectAll={true}>
        <Inject services={[CheckBoxSelection]} />
      </MultiSelectComponent>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]**INDEX.JSX**

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // define the JSON of data
  const sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },

```

```

        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
    ];
    // maps the appropriate column to fields property
    const fields = { text: 'Game', value: 'Id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="checkbox" dataSource={sportsData}
        fields={fields} placeholder="Select game" mode="CheckBox"
        selectAllText="Select All" unSelectAllText="unSelect All"
        showSelectAll={true}>
            <Inject services={[CheckBoxSelection]} />
        </MultiSelectComponent>);
    }
    ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData: { [key: string]: Object }[] = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
    ];
    // maps the appropriate column to fields property
    const fields: object = { text: 'Game', value: 'Id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="checkbox" dataSource={sportsData}
        fields={fields} placeholder="Select game" mode="CheckBox"
        selectAllText="Select All" unSelectAllText="unSelect All"
        showSelectAll={true}>
            <Inject services={[CheckBoxSelection]} />
        </MultiSelectComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Selection Limit

Defines the limit of the selected items using [maximumSelectionLength](#).

[Class-component]**INDEX.JSX**

```
import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' },
    { Id: 'game4', Game: 'Golf' },
    { Id: 'game5', Game: 'Cricket' },
    { Id: 'game6', Game: 'Handball' },
    { Id: 'game7', Game: 'Karate' },
    { Id: 'game8', Game: 'Fencing' },
    { Id: 'game9', Game: 'Boxing' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'Game', value: 'Id' };
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="checkbox" dataSource={this.sportsData}
fields={this.fields} placeholder="Select game" mode="CheckBox"
maximumSelectionLength={3}>
        <Inject services={[CheckBoxSelection]} />
      </MultiSelectComponent>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the JSON of data
  private sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' },
    { Id: 'game4', Game: 'Golf' },
    { Id: 'game5', Game: 'Cricket' },
    { Id: 'game6', Game: 'Handball' },
    { Id: 'game7', Game: 'Karate' },
    { Id: 'game8', Game: 'Fencing' },
    { Id: 'game9', Game: 'Boxing' }
  ];
}
```

```

// maps the appropriate column to fields property
private fields: object = { text: 'Game', value: 'Id' };
public render() {
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="checkbox" dataSource={this.sportsData}
            fields={this.fields} placeholder="Select game"
            mode="CheckBox" maximumSelectionLength={3}>
            <Inject services={[CheckBoxSelection]} />
        </MultiSelectComponent>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
    ];
    // maps the appropriate column to fields property
    const fields = { text: 'Game', value: 'Id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="checkbox" dataSource={sportsData}
            fields={fields} placeholder="Select game" mode="CheckBox"
            maximumSelectionLength={3}>
            <Inject services={[CheckBoxSelection]} />
        </MultiSelectComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data

```



```

const sportsData: { [key: string]: Object }[] = [
  { Id: 'game1', Game: 'Badminton' },
  { Id: 'game2', Game: 'Football' },
  { Id: 'game3', Game: 'Tennis' },
  { Id: 'game4', Game: 'Golf' },
  { Id: 'game5', Game: 'Cricket' },
  { Id: 'game6', Game: 'Handball' },
  { Id: 'game7', Game: 'Karate' },
  { Id: 'game8', Game: 'Fencing' },
  { Id: 'game9', Game: 'Boxing' }
];
// maps the appropriate column to fields property
const fields: object = { text: 'Game', value: 'Id' };
return (
  // specifies the tag for render the MultiSelect component
  <MultiSelectComponent id="checkbox" dataSource={sportsData}
    fields={fields} placeholder="Select game" mode="CheckBox"
    maximumSelectionLength={3}>
    <Inject services={[CheckBoxSelection]} />
  </MultiSelectComponent>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Selection Reordering

Using [enableSelectionOrder](#) to Reorder the selected items in popup visibility state.

[Class-component]

INDEX.JSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
 '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' },
    { Id: 'game4', Game: 'Golf' },
    { Id: 'game5', Game: 'Cricket' },
    { Id: 'game6', Game: 'Handball' },
    { Id: 'game7', Game: 'Karate' },
    { Id: 'game8', Game: 'Fencing' },
    { Id: 'game9', Game: 'Boxing' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'Game', value: 'Id' };
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="checkbox" dataSource={this.sportsData}
        fields={this.fields} placeholder="Select game" mode="CheckBox"
        enableSelectionOrder={false}>

```

```

        <Inject services={[CheckBoxSelection]}/>
      </MultiSelectComponent>;
    }
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the JSON of data
  private sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' },
    { Id: 'game4', Game: 'Golf' },
    { Id: 'game5', Game: 'Cricket' },
    { Id: 'game6', Game: 'Handball' },
    { Id: 'game7', Game: 'Karate' },
    { Id: 'game8', Game: 'Fencing' },
    { Id: 'game9', Game: 'Boxing' }
  ];
  // maps the appropriate column to fields property
  private fields: object = { text: 'Game', value: 'Id' };
  public render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="checkbox" dataSource={this.sportsData}
        fields={this.fields} placeholder="Select game"
        mode="CheckBox" enableSelectionOrder={false}>
        <Inject services={[CheckBoxSelection]}/>
      </MultiSelectComponent>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // define the JSON of data
  const sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' },
    { Id: 'game4', Game: 'Golf' },
    { Id: 'game5', Game: 'Cricket' },

```

```

        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
    ];
    // maps the appropriate column to fields property
    const fields = { text: 'Game', value: 'Id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="checkbox" dataSource={sportsData}
        fields={fields} placeholder="Select game" mode="CheckBox"
        enableSelectionOrder={false}>
            <Inject services={[CheckBoxSelection]} />
        </MultiSelectComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { CheckBoxSelection, Inject, MultiSelectComponent } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData: { [key: string]: Object }[] = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
    ];
    // maps the appropriate column to fields property
    const fields: object = { text: 'Game', value: 'Id' };
    return (
        // specifies the tag for render the MultiSelect component
        <MultiSelectComponent id="checkbox" dataSource={sportsData}
        fields={fields} placeholder="Select game" mode="CheckBox"
        enableSelectionOrder={false}>
            <Inject services={[CheckBoxSelection]} />
        </MultiSelectComponent>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

See Also

- [How to bind the data](#)
- [How to filter the bound data](#)
- [How to add custom value to the MultiSelect](#)

- [How to render grouping with checkbox.](#)

Chip customization in React Multi select component

The MultiSelect allows the user to customize the selected chip element through the [tagging](#) event. In that event, you can set the custom classes to chip element via that event argument of `setClass` method.

The following sample demonstrates chip-customization with the MultiSelect component.

[Class-component]

INDEX.JSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the JSON of data
    colorsData = [
        { Color: 'Chocolate', Code: '#75523C' },
        { Color: 'CadetBlue', Code: '#3B8289' },
        { Color: 'DarkOrange', Code: '#FF843D' },
        { Color: 'DarkRed', Code: '#CA3832' },
        { Color: 'Fuchsia', Code: '#D44FA3' },
        { Color: 'HotPink', Code: '#F23F82' },
        { Color: 'Indigo', Code: '#2F5D81' },
        { Color: 'LimeGreen', Code: '#4CD242' },
        { Color: 'OrangeRed', Code: '#FE2A00' },
        { Color: 'Tomato', Code: '#FF745C' }
    ];
    // maps the appropriate column to fields property
    fields = { text: 'Color', value: 'Code' };
    // set the value to MultiSelect
    colorValues = ['#75523C', '#4CD242', '#FF745C'];
    // bind the tagging event
    onTagging = (e) => {
        // set the current selected item text as class to chip element.
        e.setClass(e.itemData[this.fields.text].toLowerCase());
    };
    render() {
        return (
            <MultiSelectComponent id="chip-customization"
            value={this.colorValues} dataSource={this.colorsData} fields={this.fields}
            mode="Box" placeholder="Favorite Colors" tagging={this.onTagging} =
            this.onTagging.bind(this) />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MultiSelectComponent, TaggingEventArgs } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the JSON of data
    private colorsData: { [key: string]: Object }[] = [
```

```

    { Color: 'Chocolate', Code: '#75523C' },
    { Color: 'CadetBlue', Code: '#3B8289' },
    { Color: 'DarkOrange', Code: '#FF843D' },
    { Color: 'DarkRed', Code: '#CA3832' },
    { Color: 'Fuchsia', Code: '#D44FA3' },
    { Color: 'HotPink', Code: '#F23F82' },
    { Color: 'Indigo', Code: '#2F5D81' },
    { Color: 'LimeGreen', Code: '#4CD242' },
    { Color: 'OrangeRed', Code: '#FE2A00' },
    { Color: 'Tomato', Code: '#FF745C' }
  ];
  // maps the appropriate column to fields property
  private fields: { [key: string]: string } = { text: 'Color', value: 'Code'
};
  // set the value to MultiSelect
  private colorValues: string[] = ['#75523C', '#4CD242', '#FF745C'];
  // bind the tagging event
  public onTagging = (e: TaggingEventArgs) => {
    // set the current selected item text as class to chip element.
    e.setClass((e.itemData as any)[this.fields.text].toLowerCase());
  }
  public render() {
    return (
      <MultiSelectComponent id="chip-customization"
value={this.colorValues} dataSource={this.colorsData} fields={this.fields}
mode="Box" placeholder="Favorite Colors" tagging={this.onTagging} =
this.onTagging.bind(this) />
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // define the JSON of data
  const colorsData = [
    { Color: 'Chocolate', Code: '#75523C' },
    { Color: 'CadetBlue', Code: '#3B8289' },
    { Color: 'DarkOrange', Code: '#FF843D' },
    { Color: 'DarkRed', Code: '#CA3832' },
    { Color: 'Fuchsia', Code: '#D44FA3' },
    { Color: 'HotPink', Code: '#F23F82' },
    { Color: 'Indigo', Code: '#2F5D81' },
    { Color: 'LimeGreen', Code: '#4CD242' },
    { Color: 'OrangeRed', Code: '#FE2A00' },
    { Color: 'Tomato', Code: '#FF745C' }
  ];
  // maps the appropriate column to fields property
  const fields = { text: 'Color', value: 'Code' };
  // set the value to MultiSelect

```

```

const colorValues = ['#75523C', '#4CD242', '#FF745C'];
// bind the tagging event
function onTagging() { }
(e) => {
    // set the current selected item text as class to chip element.
    e.setClass(e.itemData[this.fields.text].toLowerCase());
};
return (<MultiSelectComponent id="chip-customization" value={colorValues}
dataSource={colorsData} fields={fields} mode="Box" placeholder="Favorite
Colors" tagging={onTagging = onTagging.bind(this)} />);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { MultiSelectComponent, TaggingEventArgs } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const colorsData: { [key: string]: Object }[] = [
        { Color: 'Chocolate', Code: '#75523C' },
        { Color: 'CadetBlue', Code: '#3B8289' },
        { Color: 'DarkOrange', Code: '#FF843D' },
        { Color: 'DarkRed', Code: '#CA3832' },
        { Color: 'Fuchsia', Code: '#D44FA3' },
        { Color: 'HotPink', Code: '#F23F82' },
        { Color: 'Indigo', Code: '#2F5D81' },
        { Color: 'LimeGreen', Code: '#4CD242' },
        { Color: 'OrangeRed', Code: '#FE2A00' },
        { Color: 'Tomato', Code: '#FF745C' }
    ];
    // maps the appropriate column to fields property
    const fields: { [key: string]: string } = { text: 'Color', value: 'Code' };
    // set the value to MultiSelect
    const colorValues: string[] = ['#75523C', '#4CD242', '#FF745C'];
    // bind the tagging event
    const onTagging = (e: TaggingEventArgs) => {
        // set the current selected item text as class to chip element.
        e.setClass((e.itemData as any)[fields.text].toLowerCase());
    }
    return (
        <MultiSelectComponent id="chip-customization" value={colorValues}
dataSource={colorsData} fields={fields} mode="Box" placeholder="Favorite
Colors" tagging={onTagging} />
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Localization in React Multi select component

The Localization library allows you to localize static text content of the [noRecordsTemplate](#)

and [actionFailureTemplate](#) properties according to the culture currently assigned to the MultiSelect.

| Locale key | en-US (default) |

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

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the MultiSelect and no data is loaded. Hence, the **noRecordsTemplate** property displays its text in French culture initially, and if the sample is run offline, the **actionFailureTemplate** property displays its text appropriately.

[Class-component]

INDEX.JSX

```
// import L10n class for load function
import { L10n } from '@syncfusion/ej2-base';
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind remotedata to showcase actionFailureTemplate in offline.
    customerData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    fields = { text: 'ContactName', value: 'CustomerID' };
    // take 0 item to showcase noRecordsTemplate property.
    query = new Query().select(['ContactName', 'CustomerID']).take(0);
    // set locale culture to MultiSelect
    componentWillMount() {
        L10n.load({
            'fr-BE': {
                'multi-select': {
                    'actionFailureTemplate': "Modèle d'échec d'action",
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
        });
    }
    render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" fields={this.fields}
            locale="fr-BE" query={this.query} dataSource={this.customerData}
            placeholder="Sélectionnez un client"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```

// import L10n class for load function
import { L10n } from '@syncfusion/ej2-base';
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind remotedata to showcase actionFailureTemplate in offline.
    private customerData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    // take 0 item to showcase noRecordsTemplate property.
    private query: Query = new Query().select(['ContactName',
'CustomerID']).take(0);
    // set locale culture to MultiSelect
    public componentWillMount() {
        L10n.load({
            'fr-BE': {
                'multi-select': {
                    'actionFailureTemplate': "Modèle d'échec d'action",
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
        });
    }
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" fields={this.fields}
            locale="fr-BE" query={this.query} dataSource={this.customerData}
            placeholder="Sélectionnez un client" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]**INDEX.JSX**

```

// import L10n class for load function
import { L10n } from '@syncfusion/ej2-base';
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';

```



```
function App() {
  // bind remotedata to showcase actionFailureTemplate in offline.
  const customerData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
  });
  // maps the appropriate column to fields property
  const fields = { text: 'ContactName', value: 'CustomerID' };
  // take 0 item to showcase noRecordsTemplate property.
  const query = new Query().select(['ContactName', 'CustomerID']).take(0);
  // set locale culture to MultiSelect
  React.useEffect(() => {
    L10n.load({
      'fr-BE': {
        'multi-select': {
          'actionFailureTemplate': "Modèle d'échec d'action",
          'noRecordsTemplate': "Aucun enregistrement trouvé"
        }
      }
    });
  }, []);
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" fields={fields} locale="fr-BE"
    query={query} dataSource={customerData} placeholder="Sélectionnez un
client"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import L10n class for load function
import { L10n } from '@syncfusion/ej2-base';
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // bind remotedata to showcase actionFailureTemplate in offline.
  const customerData: DataManager = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
  });
  // maps the appropriate column to fields property
  const fields: object = { text: 'ContactName', value: 'CustomerID' };
  // take 0 item to showcase noRecordsTemplate property.
  const query: Query = new Query().select(['ContactName',
'CustomerID']).take(0);
  // set locale culture to MultiSelect
  React.useEffect(() => {
    L10n.load({
```

```

        'fr-BE': {
            'multi-select': {
                'actionFailureTemplate': "Modèle d'échec d'action",
                'noRecordsTemplate': "Aucun enregistrement trouvé"
            }
        }
    });
}, []);
return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" fields={fields} locale="fr-BE"
    query={query} dataSource={customerData} placeholder="Sélectionnez un client"
    />
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

See Also

- [Accessibility](#)
- [How to bind the data to the combobox](#)

Style in React Multi select component

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 background color of wrapper element

Use the following CSS to customize the background color of wrapper element.

```

`css
.e-multiselect.e-input-group .e-multi-select-wrapper {
background-color: red;
}
`

```

Customizing the appearance of the delimiter wrapper element

Use the following CSS to customize the appearance of delimiter wrapper element.

```

`css
.e-multiselect .e-delim-values {
-webkit-text-fill-color: blue;
font-size: 16px;
font-family: cursive;
}
`

```

Customizing the appearance of chips

Use the following CSS to customize the appearance of selected chips.

```
`css
.e-multiselect .e-multi-select-wrapper .e-chips .e-chipcontent {
font-family: cursive;
font-size: 20px;
-webkit-text-fill-color: blue;
}
.e-multi-select-wrapper .e-chips {
background-color: aqua;
height: 26px;
}
`
```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```
`css
.e-multiselect.e-input-group .e-input-group-icon, .e-multiselect.e-input-group.e-control-wrapper .e-
input-group-icon:hover {
color: red;
font-size: 14px;
}
`
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
`css
.e-multiselect.e-input-group.e-control-wrapper.e-input-focus::before, .e-multiselect.e-input-group.e-
control-wrapper.e-input-focus::after {
background: #c000ff;
}
`
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
`css
.e-multiselect.e-disabled .e-multi-select-wrapper .e-delim-values {
-webkit-text-fill-color: red;
}
```

```
}
`
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
`css
.e-multiselect input.e-dropdownbase::placeholder {
color: red;
}
`
```

Customizing the placeholder to add mandatory indicator(*)

Use the following CSS to add the mandatory indicator * to the float label element.

```
`css
.e-input-group.e-control-wrapper.e-float-input .e-float-text::after {
content: "*";
color: red;
}
`
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
`css
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-
wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-
float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left)
.e-float-line::after {
background-color: #2319b8;
}

.e-multiselect.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top, .e-
float-input.e-control-wrapper:not(.e-error).e-input-focus input ~ label.e-float-text {
color: #2319b8;
}
`
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
`css
.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-
```

```
success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-
input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-
control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {
border-color: #b1bd15;
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
}
`
```

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```
`css
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
`
```

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```
`css
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px
}
`
```

Customizing the color of the checkbox

Use the following CSS to customize the color of checkbox.

```
`css
.e-popup .e-checkbox-wrapper .e-frame.e-check, .e-popup .e-checkbox-wrapper:hover .e-frame.e-check
{
background-color: green;
color: white;
}
`
```

Accessibility in React Multi select component

The MultiSelect component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties along with keyboard support. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who


use assistive technologies (AT) or those who completely rely on keyboard navigation.

The MultiSelect 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 MultiSelect component is outlined below.

| Accessibility Criteria | Compatibility |

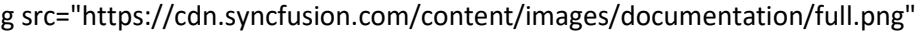
| -- | -- |

| [WCAG 2.2 Support](#) |  alt="Yes" data-bbox="279 348 300 361"/> |

| [Section 508 Support](#) |  alt="Yes" data-bbox="279 386 757 403"/> |

| [Screen Reader Support](#) |  alt="Yes" data-bbox="279 427 757 444"/> |

| [Right-To-Left Support](#) |  alt="Yes" data-bbox="279 469 757 486"/> |

| [Color Contrast](#) |  alt="Yes" data-bbox="279 514 300 526"/> |

| [Mobile Device Support](#) |  alt="Yes" data-bbox="279 551 757 568"/> |

| [Keyboard Navigation Support](#) |  alt="Yes" data-bbox="279 593 757 610"/> |

| [Accessibility Checker Validation](#) |  alt="Yes" data-bbox="279 634 757 651"/> |

| [Axe-core Accessibility Validation](#) |  alt="Yes" data-bbox="279 676 757 693"/> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> alt="Yes" data-bbox="279 868 533 881"/> - 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 MultiSelect component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the MultiSelect component:

| Properties | Functionalities |

| --- | --- |

| aria-haspopup | Indicates whether the MultiSelect input element has a popup list or not. |

| aria-expanded | Indicates whether the popup list has expanded or not. |

| aria-selected | Indicates the selected option. |

| aria-readonly | Indicates the readonly state of the MultiSelect element. |

| aria-disabled | Indicates whether the MultiSelect component is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

Keyboard interaction

You can use the following key shortcuts to access the MultiSelect without interruptions.

| Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | Set focus at the first item in the MultiSelect when no item selected. Otherwise, moves focus next to the currently selected item. |

| Arrow Up | Moves focus previous to the currently selected one. |

| Page Down | Scrolls down to the next page and set focus to the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and set focus to the first item when popup list opens. |

| Enter | Selects the focused item, and popup list closes when it is in open state. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Alt + Down | Opens the popup list. |

| Alt + Up | Closes the popup list. |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | set focus to the first item. |

| End | set focus to the last item. |

In the below sample, focus the MultiSelect component using alt+t keys.

[Class-component]

INDEX.JSX

```
{% raw %}
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // instance of MultiSelect component
  multiSelectObj;
  // defined the array of data
  gameList = [
    { id: 'Game1', game: 'Badminton' },
    { id: 'Game2', game: 'Basketball' },
    { id: 'Game3', game: 'Cricket' },
    { id: 'Game4', game: 'Football' },
    { id: 'Game5', game: 'Golf' },
    { id: 'Game6', game: 'Hockey' },
    { id: 'Game7', game: 'Rugby' },
    { id: 'Game8', game: 'Snooker' },
    { id: 'Game9', game: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'game', value: 'id' };
  componentDidMount() {
    const proxy = this;
    document.onkeyup = (e) => {
      if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        proxy.multiSelectObj.inputElement.focus();
      }
    };
  }
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement" ref={ (scope) => {
        this.multiSelectObj = scope; }} popupHeight='200px' fields={this.fields}
        dataSource={this.gameList} placeholder="Select a game"/>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
```



```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
    // instance of MultiSelect component
    public multiSelectObj: MultiSelectComponent | null;
    // defined the array of data
    private gameList: { [key: string]: Object }[] = [
        { id: 'Game1', game: 'Badminton' },
        { id: 'Game2', game: 'Basketball' },
        { id: 'Game3', game: 'Cricket' },
        { id: 'Game4', game: 'Football' },
        { id: 'Game5', game: 'Golf' },
        { id: 'Game6', game: 'Hockey' },
        { id: 'Game7', game: 'Rugby' },
        { id: 'Game8', game: 'Snooker' },
        { id: 'Game9', game: 'Tennis' }
    ];
    // maps the appropriate column to fields property
    private fields: object = { text: 'game', value: 'id' };
    public componentDidMount() {
        const proxy = this;
        document.onkeyup = (e) => {
            if (e.altKey && e.keyCode === 84 /* t */) {
                // press alt+t to focus the control.
                (proxy.multiSelectObj as any).inputElement.focus()
            }
        };
    }
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement" ref={(scope) => {
                this.multiSelectObj = scope; }} popupHeight='200px' fields={this.fields}
                dataSource={this.gameList} placeholder="Select a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // instance of MultiSelect component
    let multiSelectObj;
    // defined the array of data
    const gameList = [
        { id: 'Game1', game: 'Badminton' },
        { id: 'Game2', game: 'Basketball' },

```

```

    { id: 'Game3', game: 'Cricket' },
    { id: 'Game4', game: 'Football' },
    { id: 'Game5', game: 'Golf' },
    { id: 'Game6', game: 'Hockey' },
    { id: 'Game7', game: 'Rugby' },
    { id: 'Game8', game: 'Snooker' },
    { id: 'Game9', game: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  const fields = { text: 'game', value: 'id' };
  React.useEffect(() => {
    const proxy = this;
    document.onkeyup = (e) => {
      if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        proxy.multiSelectObj.inputElement.focus();
      }
    };
  }, []);
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" ref={(scope) => { multiSelectObj =
scope; }} popupHeight='200px' fields={fields} dataSource={gameList}
placeholder="Select a game"/>
  )
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // instance of MultiSelect component
  let multiSelectObj: MultiSelectComponent | null;
  // defined the array of data
  const gameList: { [key: string]: Object }[] = [
    { id: 'Game1', game: 'Badminton' },
    { id: 'Game2', game: 'Basketball' },
    { id: 'Game3', game: 'Cricket' },
    { id: 'Game4', game: 'Football' },
    { id: 'Game5', game: 'Golf' },
    { id: 'Game6', game: 'Hockey' },
    { id: 'Game7', game: 'Rugby' },
    { id: 'Game8', game: 'Snooker' },
    { id: 'Game9', game: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  const fields: object = { text: 'game', value: 'id' };
  React.useEffect(() => {
    document.onkeyup = (e) => {
      if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        (multiSelectObj as any).inputElement.focus()
      }
    };
  });
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

```

    }
    };
  }, []);
  return (
    // specifies the tag for render the MultiSelect component
    <MultiSelectComponent id="mtselement" ref={(scope) => {multiSelectObj
= scope; }} popupHeight='200px' fields={fields} dataSource={gameList}
placeholder="Select a game" />
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}

```

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

How To

Icons support in React Multi select component

You can render **icons** to the list items by mapping the [iconCss](#) field. This [iconCss](#) field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with [iconCss](#) field.

INDEX.JSX

```

import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of data
  sortFormatData = [
    { class: 'asc-sort', type: 'Sort A to Z', id: '1' },
    { class: 'dsc-sort', type: 'Sort Z to A ', id: '2' },
    { class: 'filter', type: 'Filter', id: '3' },
    { class: 'clear', type: 'Clear', id: '4' }
  ];
  // map the icon column to iconCSS field.
  fields = { text: 'type', iconCss: 'class', value: 'id' };
  render() {
    return (
      // specifies the tag for render the MultiSelect component
      <MultiSelectComponent id="mtselement"
dataSource={this.sortFormatData} fields={this.fields} placeholder="Select a
format"/>
    );
  }
}

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
    // define the array of data
    private sortFormatData: { [key: string]: Object }[] = [
        { class: 'asc-sort', type: 'Sort A to Z', id: '1' },
        { class: 'dsc-sort', type: 'Sort Z to A ', id: '2' },
        { class: 'filter', type: 'Filter', id: '3' },
        { class: 'clear', type: 'Clear', id: '4' }
    ];
    // map the icon column to iconCSS field.
    private fields: object = { text: 'type', iconCss: 'class', value: 'id' };
    public render() {
        return (
            // specifies the tag for render the MultiSelect component
            <MultiSelectComponent id="mtselement"
            dataSource={this.sortFormatData} fields={this.fields} placeholder="Select a
            format" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Cascading in React Multi select component

The cascading MultiSelect is a series of MultiSelect, where the value of one MultiSelect depends upon another's value. This can be configured by using the [change](#) event of the parent MultiSelect. Within that change event handler, data has to be loaded to the child MultiSelect based on the selected value of the parent MultiSelect.

The following example, shows the cascade behavior of country, state, and city MultiSelect. Here, the [Link to the Video](#) method is used to reflect the property changes immediately to the MultiSelect.

INDEX.JSX

```
{% raw %}
import { Predicate, Query } from '@syncfusion/ej2-data';
import { MultiSelectComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // country MultiSelect instance
    countryObj;
    // state MultiSelect instance
    stateObj;
    // city MultiSelect instance
    cityObj;
    // define the country MultiSelect data
    countryData = [
        { countryName: 'Australia', countryId: '2' },
```

```

    { countryName: 'United States', countryId: '1' }
  ];
  // define the state MultiSelect data
  stateData = [
    { stateName: 'New York', countryId: '1', stateId: '101' },
    { stateName: 'Virginia ', countryId: '1', stateId: '102' },
    { stateName: 'Tasmania ', countryId: '2', stateId: '105' }
  ];
  // define the city MultiSelect data
  cityData = [
    { cityName: 'Albany', stateId: '101', cityId: 201 },
    { cityName: 'Beacon ', stateId: '101', cityId: 202 },
    { cityName: 'Emporia', stateId: '102', cityId: 206 },
    { cityName: 'Hampton ', stateId: '102', cityId: 205 },
    { cityName: 'Hobart', stateId: '105', cityId: 213 },
    { cityName: 'Launceston ', stateId: '105', cityId: 214 }
  ];
  // maps the country column to fields property
  countryField = { value: 'countryId', text: 'countryName' };
  // maps the state column to fields property
  stateField = { value: 'stateId', text: 'stateName' };
  // maps the city column to fields property
  cityField = { text: 'cityName', value: 'cityId' };
  constructor(props) {
    super(props);
    this.onCountryChange = this.onCountryChange.bind(this);
    this.onStateChange = this.onStateChange.bind(this);
  }
  onCountryChange() {
    // Query the data source based on country MultiSelect selected value
    const state = this.stateObj;
    const city = this.cityObj;
    const country = this.countryObj;
    // disable the state DropDownList
    state.enabled = true;
    // frame the query based on selected value in country DropDownList.
    const tempQuery = this.getQueryForVal(country.value, country);
    // set the framed query based on selected value in country
    DropDownList.
    state.query = tempQuery;
    // set empty value to state DropDownList text property
    state.text = '';
    if (state.mainData || state.mainList) {
      state.mainData = null;
      state.mainList = null;
    }
    // bind the property changes to state DropDownList
    state.dataBind();
    // set empty value to city DropDownList text property
    city.text = '';
    // disable the city DropDownList
    city.enabled = false;
    // bind the property changes to City DropDownList
    city.dataBind();
  }
  onStateChange() {
    // Query the data source based on country MultiSelect selected value

```

```

    const city = this.cityObj;
    const state = this.stateObj;
    city.enabled = true;
    // Query the data source based on state DropDownList selected value
    const tempQuery1 = this.getQueryForVal(state.value, state);
    // set the framed query based on selected value in city DropDownList.
    city.query = tempQuery1;
    if (city.mainData || city.mainList) {
        city.mainData = null;
        city.mainList = null;
    }
    // clear the existing selection
    city.text = '';
    // bind the property change to city DropDownList
    city.dataBind();
}
getQueryForVal(valuecheck, mulObj) {
    const field = !(mulObj.fields.value) ? mulObj.fields.text :
mulObj.fields.value;
    let predicate = new Predicate(field, 'equal', '');
    for (let i = 0; i < valuecheck.length; i++) {
        if (i === 0) {
            predicate = new Predicate(field, 'equal', (valuecheck[i]));
        }
        else {
            predicate = predicate.or(field, 'equal', valuecheck[i]);
        }
    }
    return new Query().where(predicate);
}
render() {
    return (<div>
        /* specifies the tag for render the country MultiSelect
component */
        <MultiSelectComponent id="country-ms" ref={(scope) => {
this.countryObj = scope; }} fields={this.countryField}
dataSource={this.countryData} placeholder='Select a country'
change={this.onCountryChange}/>
        <br />
        /* specifies the tag for render the state MultiSelect
component */
        <MultiSelectComponent id="state-ms" ref={(scope) => {
this.stateObj = scope; }} enabled={false} fields={this.stateField}
dataSource={this.stateData} placeholder='Select a state'
change={this.onStateChange}/>
        <br />
        /* specifies the tag for render the city MultiSelect
component */
        <MultiSelectComponent id="city-ms" ref={(scope) => {
this.cityObj = scope; }} enabled={false} fields={this.cityField}
dataSource={this.cityData} placeholder='Select a city' />
        </div>);
}
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { Predicate, Query } from '@syncfusion/ej2-data';
import { MultiSelect, MultiSelectComponent } from '@syncfusion/ej2-react-
dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // country MultiSelect instance
    private countryObj: MultiSelect;
    // state MultiSelect instance
    private stateObj: MultiSelect;
    // city MultiSelect instance
    private cityObj: MultiSelect;
    // define the country MultiSelect data
    private countryData: { [key: string]: Object }[] = [
        { countryName: 'Australia', countryId: '2' },
        { countryName: 'United States', countryId: '1' }
    ];
    // define the state MultiSelect data
    private stateData: { [key: string]: Object }[] = [
        { stateName: 'New York', countryId: '1', stateId: '101' },
        { stateName: 'Virginia ', countryId: '1', stateId: '102' },
        { stateName: 'Tasmania ', countryId: '2', stateId: '105' }
    ];
    // define the city MultiSelect data
    private cityData: { [key: string]: Object }[] = [
        { cityName: 'Albany', stateId: '101', cityId: 201 },
        { cityName: 'Beacon ', stateId: '101', cityId: 202 },
        { cityName: 'Emporia', stateId: '102', cityId: 206 },
        { cityName: 'Hampton ', stateId: '102', cityId: 205 },
        { cityName: 'Hobart', stateId: '105', cityId: 213 },
        { cityName: 'Launceston ', stateId: '105', cityId: 214 }
    ];
    // maps the country column to fields property
    private countryField: object = { value: 'countryId', text: 'countryName'
};
    // maps the state column to fields property
    private stateField: object = { value: 'stateId', text: 'stateName' };
    // maps the city column to fields property
    private cityField: object = { text: 'cityName', value: 'cityId' };
    constructor(props: any) {
        super(props);
        this.onCountryChange = this.onCountryChange.bind(this);
        this.onStateChange = this.onStateChange.bind(this);
    }
    public onCountryChange() {
        // Query the data source based on country MultiSelect selected value
        const state = this.stateObj;
        const city = this.cityObj;
        const country = this.countryObj;
        // disable the state DropDownList
        state.enabled = true;
        // frame the query based on selected value in country DropDownList.

```

```

        const tempQuery = this.getQueryForVal(country.value, country);
        // set the framed query based on selected value in country
        DropDownList.
        state.query = tempQuery;
        // set empty value to state DropDownList text property
        state.text = '';
        if ((state as any).mainData || (state as any).mainList) {
            (state as any).mainData = null;
            (state as any).mainList = null
        }
        // bind the property changes to state DropDownList
        state.dataBind();
        // set empty value to city DropDownList text property
        city.text = '';
        // disable the city DropDownList
        city.enabled = false;
        // bind the property changes to City DropDownList
        city.dataBind();
    }
    public onChange() {
        // Query the data source based on country MultiSelect selected value
        const city = this.cityObj;
        const state = this.stateObj;
        city.enabled = true;
        // Query the data source based on state DropDownList selected value
        const tempQuery1 = this.getQueryForVal(state.value, state);
        // set the framed query based on selected value in city DropDownList.
        city.query = tempQuery1;
        if ((city as any).mainData || (city as any).mainList) {
            (city as any).mainData = null;
            (city as any).mainList = null
        }
        // clear the existing selection
        city.text = '';
        // bind the property change to city DropDownList
        city.dataBind();
    }
    public getQueryForVal(valuecheck: number[] | boolean[] | string[],
mulObj: MultiSelect) {
        const field: string | undefined = !(mulObj.fields.value) ?
mulObj.fields.text : mulObj.fields.value;
        let predicate: Predicate = new Predicate((field as string), 'equal',
        '');
        for (let i = 0; i < valuecheck.length; i++) {
            if (i === 0) {
                predicate = new Predicate((field as string), 'equal',
                (valuecheck[i]));
            } else {
                predicate = predicate.or((field as string), 'equal',
                (valuecheck[i] as string));
            }
        }
        return new Query().where(predicate);
    }
    public render() {
        return (
            <div>

```



```

        /* specifies the tag for render the country MultiSelect
component */
        <MultiSelectComponent id="country-ms" ref={(scope) => {
        (this.countryObj as MultiSelect | null) = scope; }}
        fields={this.countryField} dataSource={this.countryData} placeholder='Select
a country' change={this.onCountryChange} />
        <br />
        /* specifies the tag for render the state MultiSelect
component */
        <MultiSelectComponent id="state-ms" ref={(scope) => {
        (this.stateObj as MultiSelect | null) = scope; }} enabled={false}
        fields={this.stateField} dataSource={this.stateData} placeholder='Select a
state' change={this.onStateChange} />
        <br />
        /* specifies the tag for render the city MultiSelect
component */
        <MultiSelectComponent id="city-ms" ref={(scope) => {
        (this.cityObj as MultiSelect | null) = scope; }} enabled={false}
        fields={this.cityField} dataSource={this.cityData} placeholder='Select a
city' />
        </div>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

NumericTextBox

Getting Started

The following section explains the required steps to build the NumericTextBox component with its basic usage in step by step procedure.

To get start quickly with React Numerictextbox component, you can check on this video:

Dependencies

The following list of dependencies are required to use the NumericTextBox component in your application.

```

`javascript
|-- @syncfusion/ej2-react-inputs
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-react-base
`

```

Installation and configuration

You can use [Create-react-app](#) to setup the applications.

To install create-react-app run the following command.

```

`bash

```

```
npm install -g create-react-app
```

```
,
```

Start a new project using create-react-app command as follows

```
`bash
```

```
create-react-app quickstart --scripts-version=react-scripts-ts
```

```
cd quickstart
```

```
,
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry.

You can choose the component that you want to install. For this application, we are going to use NumericTextBox component.

To install NumericTextBox component, use the following command

```
`bash
```

```
npm install @syncfusion/ej2-react-inputs --save
```

```
,
```

Adding NumericTextBox to the application

Now, you can start adding NumericTextBox component to the application. We have added NumericTextBox component in `src/App.tsx` file using following code.

[Class-component]

```
`ts
```

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
```

```
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
```

```
// initializes NumericTextBox component
```

```
// sets value to the NumericTextBox
```

```
ReactDOM.render(<NumericTextBoxComponent value={10} />
```

```
,document.getElementById('numericContainer'));
```

```
,
```

[Functional-component]

```
`ts
```

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
```

```
import * as React from 'react';
```

```
import * as ReactDOM from 'react-dom';
```

```
// initializes NumericTextBox component
```

```
// sets value to the NumericTextBox
function App(){
  return(
    <NumericTextBoxComponent value={10} />
  );
}

ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

Adding CSS reference

Import the NumericTextBox component's required CSS references as follows in `src/App.css`.

```
`css
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-inputs/styles/material.css";
```

Note: If you want to refer the combined component styles, please make use of our [CRG](#) (Custom Resource Generator) in your application.

Run the application

Now use the `npm run start` command to run the application in the browser.

```
npm run start
```

The below example shows the NumericTextBox.

[Class-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets value to the NumericTextBox
ReactDOM.render(<NumericTextBoxComponent value={10}/>,
  document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets value to the NumericTextBox
```

```
ReactDOM.render(<NumericTextBoxComponent value={10} />,
document.getElementById('numericContainer'));
```

[Functional-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes NumericTextBox component
// sets value to the NumericTextBox
function App() {
    return <NumericTextBoxComponent value={10}/>;
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes NumericTextBox component
// sets value to the NumericTextBox
function App() {
    return <NumericTextBoxComponent value={10} />;
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

Range validation

You can set the minimum and maximum range of values in the NumericTextBox using the [min](#) and [max](#) properties, so the numeric value should be in the min and max range.

The validation behavior depends on the [strictMode](#) property.

[Class-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets the minimum and maximum range values
// strictMode has been enabled by default
ReactDOM.render(<NumericTextBoxComponent min={10} max={20} value={16}
step={2}/>, document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets the minimum and maximum range values
```

```
// strictMode has been enabled by default
ReactDOM.render(<NumericTextBoxComponent min={10} max={20} value={16}
step={2} />
, document.getElementById('numericContainer'));
```

[Functional-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes NumericTextBox component
// sets value to the NumericTextBox
function App() {
    return <NumericTextBoxComponent min={10} max={20} value={16} step={2}/>;
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes NumericTextBox component
// sets value to the NumericTextBox
function App() {
    return <NumericTextBoxComponent min={10} max={20} value={16} step={2} />;
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

Formatting the value

User can set the format of the NumericTextBox component using [format](#) property. The value will be displayed in the specified format, when the component is in focused out state. For more information about

formatting the value, refer to this [link](#).

The below example demonstrates format the value by using currency format value `c2`.

[Class-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes NumericTextBox component
// sets currency with 2 numbers of decimal places format
ReactDOM.render(<NumericTextBoxComponent format='c2' value={10}/>,
document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
```

```
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets currency with 2 numbers of decimal places format
ReactDOM.render(<NumericTextBoxComponent format='c2' value={10} />
, document.getElementById('numericContainer'));
```

[Functional-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes NumericTextBox component
// sets value to the NumericTextBox
function App() {
    return <NumericTextBoxComponent format="c2" value={10}/>;
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes NumericTextBox component
// sets value to the NumericTextBox
function App() {
    return <NumericTextBoxComponent format="c2" value={10} />;
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

Precision of numbers

You can restrict the number of decimals to be entered in the NumericTextBox by using the [decimals](#) and [validateDecimalOnType](#) properties.

So, you can't enter the number whose precision is greater than the mentioned decimals.

- If `validateDecimalOnType` is false, number of decimals will not be restricted.

Else, number of decimals will be restricted while typing in the NumericTextBox.

[Class-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// restricts number of decimals to be entered in the NumericTextBox by
// enabling validateDecimalOnType property
// sets number of decimal places to be allowed by the NumericTextBox
// sets number with 3 numbers of decimal places format
```

```
ReactDOM.render(<NumericTextBoxComponent validateDecimalOnType={true}
decimals={3} format='n3' value={10} placeholder='ValidateDecimalOnType
Enabled' floatLabelType='Auto'/>,
document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// validateDecimalOnType is false by default. So, number of decimals will not
be restricted.
// sets number of decimal places to be allowed by the NumericTextBox
// sets number with 3 numbers of decimal places format
ReactDOM.render(<NumericTextBoxComponent decimals={3} format='n3' value={10}
placeholder='ValidateDecimalOnType Disabled' floatLabelType='Auto'/>,
document.getElementById('numericContainer2'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// restricts number of decimals to be entered in the NumericTextBox by
enabling validateDecimalOnType property
// sets number of decimal places to be allowed by the NumericTextBox
// sets number with 3 numbers of decimal places format
ReactDOM.render(<NumericTextBoxComponent validateDecimalOnType={true}
decimals={3} format='n3' value={10} placeholder='ValidateDecimalOnType
Enabled' floatLabelType='Auto' />
,document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// validateDecimalOnType is false by default. So, number of decimals will not
be restricted.
// sets number of decimal places to be allowed by the NumericTextBox
// sets number with 3 numbers of decimal places format
ReactDOM.render(<NumericTextBoxComponent decimals={3} format='n3' value={10}
placeholder='ValidateDecimalOnType Disabled' floatLabelType='Auto' />
,document.getElementById('numericContainer2'));
```

[Functional-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes NumericTextBox component
// sets percentage with 2 numbers of decimal places format
function App1() {
    return (<NumericTextBoxComponent validateDecimalOnType={true}
decimals={3} format="n3" value={10} placeholder="ValidateDecimalOnType
Enabled" floatLabelType="Auto"/>);
}
ReactDOM.render(<App1 />, document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// sets currency with 2 numbers of decimal places format
function App2() {
    return (<NumericTextBoxComponent decimals={3} format="n3" value={10}
placeholder="ValidateDecimalOnType Disabled" floatLabelType="Auto"/>);
}
```

```
}
ReactDOM.render(<App2 />, document.getElementById('numericContainer2'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// initializes NumericTextBox component
// sets percentage with 2 numbers of decimal places format
function App1() {
  return (
    <NumericTextBoxComponent
      validateDecimalOnType={true}
      decimals={3}
      format="n3"
      value={10}
      placeholder="ValidateDecimalOnType Enabled"
      floatLabelType="Auto"
    />
  );
}
ReactDOM.render(<App1 />, document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// sets currency with 2 numbers of decimal places format
function App2() {
  return (
    <NumericTextBoxComponent
      decimals={3}
      format="n3"
      value={10}
      placeholder="ValidateDecimalOnType Disabled"
      floatLabelType="Auto"
    />
  );
}
ReactDOM.render(<App2 />, document.getElementById('numericContainer2'));
```

See Also

- [How to perform custom validation using FormValidator](#)
- [How to customize the UI appearance of the control](#)
- [How to customize the spin button's up and down arrow](#)
- [How to customize the step value and hide spin buttons](#)
- [How to prevent nullable input in NumericTextBox](#)
- [How to maintain trailing zeros in NumericTextBox](#)

Formats in React Numerictextbox component

You can format the value of NumericTextBox using [format](#) property. The value will be displayed in the specified format when the component is in focused out state. The format string supports both the [standard numeric format string](#) and [custom numeric format string](#).

Standard formats

From the [standard numeric formats](#), you can use the numeric related format specifiers such as **n**, **p** and **c** in the NumericTextBox component. By using these format specifiers, you can achieve the percentage and currency textbox behavior also.

The below example demonstrates percentage and currency formats.

[Class-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets percentage with 2 numbers of decimal places format
ReactDOM.render(<NumericTextBoxComponent format='p2' value={0.5} min={0}
max={1} step={0.01} placeholder='Percentage format' floatLabelType='Auto' />,
document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// sets currency with 2 numbers of decimal places format
ReactDOM.render(<NumericTextBoxComponent format='c2' value={10}
placeholder='Currency format' floatLabelType='Auto' />,
document.getElementById('numericContainer2'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets percentage with 2 numbers of decimal places format
ReactDOM.render(<NumericTextBoxComponent format='p2' value={0.5} min={0}
max={1} step={0.01} placeholder='Percentage format' floatLabelType='Auto' />,
document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// sets currency with 2 numbers of decimal places format
ReactDOM.render(<NumericTextBoxComponent format='c2' value={10}
placeholder='Currency format' floatLabelType='Auto' />,
document.getElementById('numericContainer2'));
```

[Functional-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets percentage with 2 numbers of decimal places format
function App1() {
    return (<NumericTextBoxComponent format="p2" value={0.5} min={0} max={1}
step={0.01} placeholder="Percentage format" floatLabelType="Auto"/>);
}
ReactDOM.render(<App1 />, document.getElementById('numericContainer1'));
// initializes NumericTextBox component
```

```
// sets currency with 2 numbers of decimal places format
function App2() {
    return (<NumericTextBoxComponent format="c2" value={10}
placeholder="Currency format" floatLabelType="Auto"/>);
}
ReactDOM.render(<App2 />, document.getElementById('numericContainer2'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets percentage with 2 numbers of decimal places format
function App1() {
    return (
        <NumericTextBoxComponent
            format="p2"
            value={0.5}
            min={0}
            max={1}
            step={0.01}
            placeholder="Percentage format"
            floatLabelType="Auto"
        />
    );
}
ReactDOM.render(<App1 />, document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// sets currency with 2 numbers of decimal places format
function App2() {
    return (
        <NumericTextBoxComponent
            format="c2"
            value={10}
            placeholder="Currency format"
            floatLabelType="Auto"
        />
    );
}
ReactDOM.render(<App2 />, document.getElementById('numericContainer2'));
```

Custom formats

From the [custom numeric format string](#), you can provide any custom format by combining one or more custom specifiers.

The below examples demonstrate format the value by using currency format string # and 0.

[Class-component]**INDEX.JSX**

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
```

```
// sets the format using custom format string `#`
ReactDOM.render(<NumericTextBoxComponent format='###.##' value={10}
placeholder='Custom format string #' floatLabelType='Auto' />,
document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// sets the format using custom format string `0`
ReactDOM.render(<NumericTextBoxComponent format='000.00' value={10}
placeholder='Custom format string 0' floatLabelType='Auto' />,
document.getElementById('numericContainer2'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets the format using custom format string `#`
ReactDOM.render(<NumericTextBoxComponent format='###.##' value={10}
placeholder='Custom format string #' floatLabelType='Auto' />,
document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// sets the format using custom format string `0`
ReactDOM.render(<NumericTextBoxComponent format='000.00' value={10}
placeholder='Custom format string 0' floatLabelType='Auto' />,
document.getElementById('numericContainer2'));
```

[Functional-component]**INDEX.JSX**

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets the format using custom format string `#`
function App1() {
    return (<NumericTextBoxComponent format="###.##" value={10}
placeholder="Custom format string #" floatLabelType="Auto"/>);
}
ReactDOM.render(<App1 />, document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// sets the format using custom format string `0`
function App2() {
    return (<NumericTextBoxComponent format="000.00" value={10}
placeholder="Custom format string 0" floatLabelType="Auto"/>);
}
ReactDOM.render(<App2 />, document.getElementById('numericContainer2'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets the format using custom format string `#`
```

```
function App1() {
  return (
    <NumericTextBoxComponent
      format="###.##"
      value={10}
      placeholder="Custom format string #"
      floatLabelType="Auto"
    />
  );
}
ReactDOM.render(<App1 />, document.getElementById('numericContainer1'));
// initializes NumericTextBox component
// sets the format using custom format string `0`
function App2() {
  return (
    <NumericTextBoxComponent
      format="000.00"
      value={10}
      placeholder="Custom format string 0"
      floatLabelType="Auto"
    />
  );
}
ReactDOM.render(<App2 />, document.getElementById('numericContainer2'));
```

Globalization in React Numerictextbox component

Localization

[Localization](#) library allows users to localize the default text contents of the NumericTextBox to different cultures using the [locale](#) property.

In NumericTextBox, spin buttons title for the tooltip will be localized based on the culture.

| Locale key | en-US (default) |

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

| incrementTitle | Increment value |

| decrementTitle | Decrement value |

Loading translations

To load translation object in your application use `load` function of `L10n` class.

The below example demonstrates the NumericTextBox in `German` culture with the spin buttons tooltip.

INDEX.JSX

```
import { L10n } from '@syncfusion/ej2-base';
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
L10n.load({
  'de': {
    'numerictextbox': { incrementTitle: 'Wert erhöhen', decrementTitle:
    'Dekrementwert' }
  }
});
```

```
// initializes NumericTextBox component
// sets `German` culture using the culture value 'de'
ReactDOM.render(<NumericTextBoxComponent locale='de' value={10}/>,
document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { L10n } from '@syncfusion/ej2-base';
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
L10n.load({
  'de': {
    'numerictextbox': { incrementTitle: 'Wert erhöhen', decrementTitle:
'Dekrementwert' }
  }
});
// initializes NumericTextBox component
// sets `German` culture using the culture value 'de'
ReactDOM.render(<NumericTextBoxComponent locale='de' value={10}
/>,document.getElementById('numericContainer'));
```

Internationalization

Internationalization library provides support for formatting and parsing the number by using the official [Unicode CLDR](#) JSON data and also provides the `loadCldr` method to load the culture specific CLDR JSON data. The NumericTextBox comes with built-in internationalization support to adapt based on culture. For more information about internationalization, refer to this [link](#).

By default, all the Essential JS 2 component are specific to English culture ('en-US'). If you want to go with the 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 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.tsx` file.
- Now import the required culture from the installed location to `app.tsx` file as like the below code snippets.

```
`ts
```

```
import * as currencies from 'cldr-data/main/de/currencies.json';
```

```
import * as numbers from 'cldr-data/main/de/numbers.json';
```

```
import * as currencyData from 'cldr-data/supplemental/currencyData.json';
import * as numberingSystems from 'cldr-data/supplemental/numberingSystems.json';
loadCldr(numberingSystems, currencies, numbers, currencyData);
`
```

if you are facing the error `/node_modules/cldr-data/main/de/*.json (1,1): unused expression, expected an assignment or function call` when you are adding the json files to render the culture sample, then add the below configuration in your `tslint.json` file

```
`ts
"linterOptions": {
"exclude": [
"*.json",
"/*.json"
]
}
```

- Set the culture by using the [locale](#) property.

The below example demonstrates the NumericTextBox in **German** culture with the **EUR** currency format.

[Class-component]

CURRENCIES.JSX

```
{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number":
            "$Revision: 13259 $",
          "_cldrVersion":
            "31";
        }
      },
      "language":
        "de";
    }
  },
  "numbers": {
    "currencies": {
      "ADP";
    }
  }
}
```

```

    {
      "displayName";
      "Andorranische Pesete",
      "displayName-count-one";
      "Andorranische Pesete",
      "displayName-count-other";
      "Andorranische Peseten",
      "symbol";
      "ADP";
    }
    "AED";
    {
      "displayName";
      "VAE-Dirham",
      "displayName-count-one";
      "VAE-Dirham",
      "displayName-count-other";
      "VAE-Dirham",
      "symbol";
      "AED";
    }
    "AFA";
    {
      "displayName";
      "Afghanische Afghani (1927-2002)",
      "displayName-count-one";
      "Afghanische Afghani (1927-2002)",
      "displayName-count-other";
      "Afghanische Afghani (1927-2002)",
      "symbol";
      "AFA";
    }
    "AFN";
    {
      "displayName";
      "Afghanischer Afghani",
      "displayName-count-one";
      "Afghanischer Afghani",
      "displayName-count-other";
      "Afghanische Afghani",
      "symbol";
      "AFN";
    }
    "ALK";
    {
      "displayName";
      "Albanischer Lek (1946-1965)",
      "displayName-count-one";
      "Albanischer Lek (1946-1965)",
      "displayName-count-other";
      "Albanische Lek (1946-1965)";
    }
    "ALL";
    {
      "displayName";
      "Albanischer Lek",
      "displayName-count-one";

```

```

        "Albanischer Lek",
        "displayName-count-other";
        "Albanische Lek",
        "symbol";
        "ALL";
    }
    "AMD";
    {
        "displayName";
        "Armenischer Dram",
        "displayName-count-one";
        "Armenischer Dram",
        "displayName-count-other";
        "Armenische Dram",
        "symbol";
        "AMD";
    }
    "ANG";
    {
        "displayName";
        "Niederländische-Antillen-Gulden",
        "displayName-count-one";
        "Niederländische-Antillen-Gulden",
        "displayName-count-other";
        "Niederländische-Antillen-Gulden",
        "symbol";
        "ANG";
    }
    "AOA";
    {
        "displayName";
        "Angolanischer Kwanza",
        "displayName-count-one";
        "Angolanischer Kwanza",
        "displayName-count-other";
        "Angolanische Kwanza",
        "symbol";
        "AOA",
        "symbol-alt-narrow";
        "Kz";
    }
    "AOK";
    {
        "displayName";
        "Angolanischer Kwanza (1977-1990)",
        "displayName-count-one";
        "Angolanischer Kwanza (1977-1990)",
        "displayName-count-other";
        "Angolanische Kwanza (1977-1990)",
        "symbol";
        "AOK";
    }
    "AON";
    {
        "displayName";
        "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-one";
    }

```



```

        "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-other";
        "Angolanische Neue Kwanza (1990-2000)",
        "symbol";
        "AON";
    }
    "AOR";
    {
        "displayName";
        "Angolanischer Kwanza Reajustado (1995-1999)",
        "displayName-count-one";
        "Angolanischer Kwanza Reajustado (1995-1999)",
        "displayName-count-other";
        "Angolanische Kwanza Reajustado (1995-1999)",
        "symbol";
        "AOR";
    }
    "ARA";
    {
        "displayName";
        "Argentinischer Austral",
        "displayName-count-one";
        "Argentinischer Austral",
        "displayName-count-other";
        "Argentinische Austral",
        "symbol";
        "ARA";
    }
    "ARL";
    {
        "displayName";
        "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-one";
        "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-other";
        "Argentinische Pesos Ley (1970-1983)",
        "symbol";
        "ARL";
    }
    "ARM";
    {
        "displayName";
        "Argentinischer Peso (1881-1970)",
        "displayName-count-one";
        "Argentinischer Peso (1881-1970)",
        "displayName-count-other";
        "Argentinische Pesos (1881-1970)",
        "symbol";
        "ARM";
    }
    "ARP";
    {
        "displayName";
        "Argentinischer Peso (1983-1985)",
        "displayName-count-one";
        "Argentinischer Peso (1983-1985)",
        "displayName-count-other";
    }

```

```

        "Argentinische Peso (1983-1985)",
        "symbol";
        "ARP";
    }
    "ARS";
    {
        "displayName";
        "Argentinischer Peso",
        "displayName-count-one";
        "Argentinischer Peso",
        "displayName-count-other";
        "Argentinische Pesos",
        "symbol";
        "ARS",
        "symbol-alt-narrow";
        "$";
    }
    "ATS";
    {
        "displayName";
        "Österreichischer Schilling",
        "displayName-count-one";
        "Österreichischer Schilling",
        "displayName-count-other";
        "Österreichische Schilling",
        "symbol";
        "öS";
    }
    "AUD";
    {
        "displayName";
        "Australischer Dollar",
        "displayName-count-one";
        "Australischer Dollar",
        "displayName-count-other";
        "Australische Dollar",
        "symbol";
        "AU$",
        "symbol-alt-narrow";
        "$";
    }
    "AWG";
    {
        "displayName";
        "Aruba-Florin",
        "displayName-count-one";
        "Aruba-Florin",
        "displayName-count-other";
        "Aruba-Florin",
        "symbol";
        "AWG";
    }
    "AZM";
    {
        "displayName";
        "Aserbaidshan-Manat (1993-2006)",
        "displayName-count-one";

```

```

        "Aserbaidtschan-Manat (1993-2006)",
        "displayName-count-other";
        "Aserbaidtschan-Manat (1993-2006)",
        "symbol";
        "AZM";
    }
    "AZN";
    {
        "displayName";
        "Aserbaidtschan-Manat",
        "displayName-count-one";
        "Aserbaidtschan-Manat",
        "displayName-count-other";
        "Aserbaidtschan-Manat",
        "symbol";
        "AZN";
    }
    "BAD";
    {
        "displayName";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-one";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-other";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "symbol";
        "BAD";
    }
    "BAM";
    {
        "displayName";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-one";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-other";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "symbol";
        "BAM",
        "symbol-alt-narrow";
        "KM";
    }
    "BAN";
    {
        "displayName";
        "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-one";
        "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-other";
        "Bosnien und Herzegowina Neue Dinar (1994-1997)",
        "symbol";
        "BAN";
    }
    "BBD";
    {
        "displayName";
        "Barbados-Dollar",
        "displayName-count-one";
    }

```

```

        "Barbados-Dollar",
        "displayName-count-other";
        "Barbados-Dollar",
        "symbol";
        "BBD",
        "symbol-alt-narrow";
        "$";
    }
    "BDT";
    {
        "displayName";
        "Bangladesch-Taka",
        "displayName-count-one";
        "Bangladesch-Taka",
        "displayName-count-other";
        "Bangladesch-Taka",
        "symbol";
        "BDT",
        "symbol-alt-narrow";
        "৳";
    }
    "BEC";
    {
        "displayName";
        "Belgischer Franc (konvertibel)",
        "displayName-count-one";
        "Belgischer Franc (konvertibel)",
        "displayName-count-other";
        "Belgische Franc (konvertibel)",
        "symbol";
        "BEC";
    }
    "BEF";
    {
        "displayName";
        "Belgischer Franc",
        "displayName-count-one";
        "Belgischer Franc",
        "displayName-count-other";
        "Belgische Franc",
        "symbol";
        "BEF";
    }
    "BEL";
    {
        "displayName";
        "Belgischer Finanz-Franc",
        "displayName-count-one";
        "Belgischer Finanz-Franc",
        "displayName-count-other";
        "Belgische Finanz-Franc",
        "symbol";
        "BEL";
    }
    "BGL";
    {

```

```

        "displayName";
        "Bulgarische Lew (1962-1999)",
        "displayName-count-one";
        "Bulgarische Lew (1962-1999)",
        "displayName-count-other";
        "Bulgarische Lew (1962-1999)",
        "symbol";
        "BGL";
    }
    "BGM";
    {
        "displayName";
        "Bulgarischer Lew (1952-1962)",
        "displayName-count-one";
        "Bulgarischer Lew (1952-1962)",
        "displayName-count-other";
        "Bulgarische Lew (1952-1962)",
        "symbol";
        "BGK";
    }
    "BGN";
    {
        "displayName";
        "Bulgarischer Lew",
        "displayName-count-one";
        "Bulgarischer Lew",
        "displayName-count-other";
        "Bulgarische Lew",
        "symbol";
        "BGN";
    }
    "BGO";
    {
        "displayName";
        "Bulgarischer Lew (1879-1952)",
        "displayName-count-one";
        "Bulgarischer Lew (1879-1952)",
        "displayName-count-other";
        "Bulgarische Lew (1879-1952)",
        "symbol";
        "BGJ";
    }
    "BHD";
    {
        "displayName";
        "Bahrain-Dinar",
        "displayName-count-one";
        "Bahrain-Dinar",
        "displayName-count-other";
        "Bahrain-Dinar",
        "symbol";
        "BHD";
    }
    "BIF";
    {
        "displayName";
        "Burundi-Franc",

```

```

        "displayName-count-one";
        "Burundi-Franc",
        "displayName-count-other";
        "Burundi-Francs",
        "symbol";
        "BIF";
    }
    "BMD";
    {
        "displayName";
        "Bermuda-Dollar",
        "displayName-count-one";
        "Bermuda-Dollar",
        "displayName-count-other";
        "Bermuda-Dollar",
        "symbol";
        "BMD",
        "symbol-alt-narrow";
        "$";
    }
    "BND";
    {
        "displayName";
        "Brunei-Dollar",
        "displayName-count-one";
        "Brunei-Dollar",
        "displayName-count-other";
        "Brunei-Dollar",
        "symbol";
        "BND",
        "symbol-alt-narrow";
        "$";
    }
    "BOB";
    {
        "displayName";
        "Bolivanischer Boliviano",
        "displayName-count-one";
        "Bolivanischer Boliviano",
        "displayName-count-other";
        "Bolivianische Bolivianos",
        "symbol";
        "BOB",
        "symbol-alt-narrow";
        "Bs";
    }
    "BOL";
    {
        "displayName";
        "Bolivianischer Boliviano (1863-1963)",
        "displayName-count-one";
        "Bolivianischer Boliviano (1863-1963)",
        "displayName-count-other";
        "Bolivianische Bolivianos (1863-1963)",
        "symbol";
        "BOL";
    }
}

```

```
"BOP";
{
  "displayName";
  "Bolivianischer Peso",
    "displayName-count-one";
  "Bolivianischer Peso",
    "displayName-count-other";
  "Bolivianische Peso",
    "symbol";
  "BOP";
}
"BOV";
{
  "displayName";
  "Boliviansiche Mvdol",
    "displayName-count-one";
  "Boliviansiche Mvdol",
    "displayName-count-other";
  "Bolivianische Mvdol",
    "symbol";
  "BOV";
}
"BRB";
{
  "displayName";
  "Brazilianischer Cruzeiro Novo (1967-1986)",
    "displayName-count-one";
  "Brazilianischer Cruzeiro Novo (1967-1986)",
    "displayName-count-other";
  "Brazilianische Cruzeiro Novo (1967-1986)",
    "symbol";
  "BRB";
}
"BRC";
{
  "displayName";
  "Brazilianischer Cruzado (1986-1989)",
    "displayName-count-one";
  "Brazilianischer Cruzado (1986-1989)",
    "displayName-count-other";
  "Brazilianische Cruzado (1986-1989)",
    "symbol";
  "BRC";
}
"BRE";
{
  "displayName";
  "Brazilianischer Cruzeiro (1990-1993)",
    "displayName-count-one";
  "Brazilianischer Cruzeiro (1990-1993)",
    "displayName-count-other";
  "Brazilianische Cruzeiro (1990-1993)",
    "symbol";
  "BRE";
}
"BRL";
{
```

```

        "displayName";
        "Brazilianischer Real",
        "displayName-count-one";
        "Brazilianischer Real",
        "displayName-count-other";
        "Brazilianische Real",
        "symbol";
        "R$",
        "symbol-alt-narrow";
        "R$";
    }
    "BRN";
    {
        "displayName";
        "Brazilianischer Cruzado Novo (1989-1990)",
        "displayName-count-one";
        "Brazilianischer Cruzado Novo (1989-1990)",
        "displayName-count-other";
        "Brazilianische Cruzado Novo (1989-1990)",
        "symbol";
        "BRN";
    }
    "BRR";
    {
        "displayName";
        "Brazilianischer Cruzeiro (1993-1994)",
        "displayName-count-one";
        "Brazilianischer Cruzeiro (1993-1994)",
        "displayName-count-other";
        "Brazilianische Cruzeiro (1993-1994)",
        "symbol";
        "BRR";
    }
    "BRZ";
    {
        "displayName";
        "Brazilianischer Cruzeiro (1942-1967)",
        "displayName-count-one";
        "Brazilianischer Cruzeiro (1942-1967)",
        "displayName-count-other";
        "Brazilianischer Cruzeiro (1942-1967)",
        "symbol";
        "BRZ";
    }
    "BSD";
    {
        "displayName";
        "Bahamas-Dollar",
        "displayName-count-one";
        "Bahamas-Dollar",
        "displayName-count-other";
        "Bahamas-Dollar",
        "symbol";
        "BSD",
        "symbol-alt-narrow";
        "$";
    }
}

```



```

"BTN";
{
  "displayName";
  "Bhutan-Ngultrum",
    "displayName-count-one";
  "Bhutan-Ngultrum",
    "displayName-count-other";
  "Bhutan-Ngultrum",
    "symbol";
  "BTN";
}
"BUK";
{
  "displayName";
  "Birmanischer Kyat",
    "displayName-count-one";
  "Birmanischer Kyat",
    "displayName-count-other";
  "Birmanische Kyat",
    "symbol";
  "BUK";
}
"BWP";
{
  "displayName";
  "Botswanischer Pula",
    "displayName-count-one";
  "Botswanischer Pula",
    "displayName-count-other";
  "Botswanische Pula",
    "symbol";
  "BWP",
    "symbol-alt-narrow";
  "P";
}
"BYB";
{
  "displayName";
  "Belarus-Rubel (1994-1999)",
    "displayName-count-one";
  "Belarus-Rubel (1994-1999)",
    "displayName-count-other";
  "Belarus-Rubel (1994-1999)",
    "symbol";
  "BYB";
}
"BYN";
{
  "displayName";
  "Weißrussischer Rubel",
    "displayName-count-one";
  "Weißrussischer Rubel",
    "displayName-count-other";
  "Weißrussische Rubel",
    "symbol";
  "BYN",
    "symbol-alt-narrow";
}

```

```

        "p.";
    }
    "BYR";
    {
        "displayName";
        "Weißrussischer Rubel (2000-2016)",
        "displayName-count-one";
        "Weißrussischer Rubel (2000-2016)",
        "displayName-count-other";
        "Weißrussische Rubel (2000-2016)",
        "symbol";
        "BYR";
    }
    "BZD";
    {
        "displayName";
        "Belize-Dollar",
        "displayName-count-one";
        "Belize-Dollar",
        "displayName-count-other";
        "Belize-Dollar",
        "symbol";
        "BZD",
        "symbol-alt-narrow";
        "$";
    }
    "CAD";
    {
        "displayName";
        "Kanadischer Dollar",
        "displayName-count-one";
        "Kanadischer Dollar",
        "displayName-count-other";
        "Kanadische Dollar",
        "symbol";
        "CA$",
        "symbol-alt-narrow";
        "$";
    }
    "CDF";
    {
        "displayName";
        "Kongo-Franc",
        "displayName-count-one";
        "Kongo-Franc",
        "displayName-count-other";
        "Kongo-Francs",
        "symbol";
        "CDF";
    }
    "CHE";
    {
        "displayName";
        "WIR-Euro",
        "displayName-count-one";
        "WIR-Euro",
        "displayName-count-other";
    }

```

```

        "WIR-Euro",
        "symbol";
        "CHE";
    }
    "CHF";
    {
        "displayName";
        "Schweizer Franken",
        "displayName-count-one";
        "Schweizer Franken",
        "displayName-count-other";
        "Schweizer Franken",
        "symbol";
        "CHF";
    }
    "CHW";
    {
        "displayName";
        "WIR Franken",
        "displayName-count-one";
        "WIR Franken",
        "displayName-count-other";
        "WIR Franken",
        "symbol";
        "CHW";
    }
    "CLE";
    {
        "displayName";
        "Chilenischer Escudo",
        "displayName-count-one";
        "Chilenischer Escudo",
        "displayName-count-other";
        "Chilenische Escudo",
        "symbol";
        "CLE";
    }
    "CLF";
    {
        "displayName";
        "Chilenische Unidades de Fomento",
        "displayName-count-one";
        "Chilenische Unidades de Fomento",
        "displayName-count-other";
        "Chilenische Unidades de Fomento",
        "symbol";
        "CLF";
    }
    "CLP";
    {
        "displayName";
        "Chilenischer Peso",
        "displayName-count-one";
        "Chilenischer Peso",
        "displayName-count-other";
        "Chilenische Pesos",
        "symbol";
    }

```

```

        "CLP",
        "symbol-alt-narrow";
        "$";
    }
    "CNX";
    {
        "displayName";
        "Dollar der Chinesischen Volksbank",
        "displayName-count-one";
        "Dollar der Chinesischen Volksbank",
        "displayName-count-other";
        "Dollar der Chinesischen Volksbank",
        "symbol";
        "CNX";
    }
    "CNY";
    {
        "displayName";
        "Renminbi Yuan",
        "displayName-count-one";
        "Chinesischer Yuan",
        "displayName-count-other";
        "Renminbi Yuan",
        "symbol";
        "CN¥",
        "symbol-alt-narrow";
        "¥";
    }
    "COP";
    {
        "displayName";
        "Kolumbianischer Peso",
        "displayName-count-one";
        "Kolumbianischer Peso",
        "displayName-count-other";
        "Kolumbianische Pesos",
        "symbol";
        "COP",
        "symbol-alt-narrow";
        "$";
    }
    "COU";
    {
        "displayName";
        "Kolumbianische Unidades de valor real",
        "displayName-count-one";
        "Kolumbianische Unidad de valor real",
        "displayName-count-other";
        "Kolumbianische Unidades de valor real",
        "symbol";
        "COU";
    }
    "CRC";
    {
        "displayName";
        "Costa-Rica-Colón",
        "displayName-count-one";

```

```

        "Costa-Rica-Colón",
        "displayName-count-other";
    "Costa-Rica-Colón",
        "symbol";
    "CRC",
        "symbol-alt-narrow";
    "₡";
}
"CSD";
{
    "displayName";
    "Serbischer Dinar (2002-2006)",
        "displayName-count-one";
    "Serbischer Dinar (2002-2006)",
        "displayName-count-other";
    "Serbische Dinar (2002-2006)",
        "symbol";
    "CSD";
}
"CSK";
{
    "displayName";
    "Tschechoslowakische Krone",
        "displayName-count-one";
    "Tschechoslowakische Kronen",
        "displayName-count-other";
    "Tschechoslowakische Kronen",
        "symbol";
    "CSK";
}
"CUC";
{
    "displayName";
    "Kubanischer Peso (konvertibel)",
        "displayName-count-one";
    "Kubanischer Peso (konvertibel)",
        "displayName-count-other";
    "Kubanische Pesos (konvertibel)",
        "symbol";
    "CUC",
        "symbol-alt-narrow";
    "Cub$";
}
"CUP";
{
    "displayName";
    "Kubanischer Peso",
        "displayName-count-one";
    "Kubanischer Peso",
        "displayName-count-other";
    "Kubanische Pesos",
        "symbol";
    "CUP",
        "symbol-alt-narrow";
    "$";
}
"CVE";

```

```

    {
      "displayName";
      "Cabo-Verde-Escudo",
        "displayName-count-one";
      "Cabo-Verde-Escudo",
        "displayName-count-other";
      "Cabo-Verde-Escudos",
        "symbol";
      "CVE";
    }
    "CYP";
    {
      "displayName";
      "Zypern-Pfund",
        "displayName-count-one";
      "Zypern Pfund",
        "displayName-count-other";
      "Zypern Pfund",
        "symbol";
      "CYP";
    }
    "CZK";
    {
      "displayName";
      "Tschechische Krone",
        "displayName-count-one";
      "Tschechische Krone",
        "displayName-count-other";
      "Tschechische Kronen",
        "symbol";
      "CZK",
        "symbol-alt-narrow";
      "Kč";
    }
    "DDM";
    {
      "displayName";
      "Mark der DDR",
        "displayName-count-one";
      "Mark der DDR",
        "displayName-count-other";
      "Mark der DDR",
        "symbol";
      "DDM";
    }
    "DEM";
    {
      "displayName";
      "Deutsche Mark",
        "displayName-count-one";
      "Deutsche Mark",
        "displayName-count-other";
      "Deutsche Mark",
        "symbol";
      "DM";
    }
    "DJF";

```

```

    {
      "displayName";
      "Dschibuti-Franc",
        "displayName-count-one";
      "Dschibuti-Franc",
        "displayName-count-other";
      "Dschibuti-Franc",
        "symbol";
      "DJF";
    }
    "DKK";
    {
      "displayName";
      "Dänische Krone",
        "displayName-count-one";
      "Dänische Krone",
        "displayName-count-other";
      "Dänische Kronen",
        "symbol";
      "DKK",
        "symbol-alt-narrow";
      "kr";
    }
    "DOP";
    {
      "displayName";
      "Dominikanischer Peso",
        "displayName-count-one";
      "Dominikanischer Peso",
        "displayName-count-other";
      "Dominikanische Pesos",
        "symbol";
      "DOP",
        "symbol-alt-narrow";
      "$";
    }
    "DZD";
    {
      "displayName";
      "Algerischer Dinar",
        "displayName-count-one";
      "Algerischer Dinar",
        "displayName-count-other";
      "Algerische Dinar",
        "symbol";
      "DZD";
    }
    "ECS";
    {
      "displayName";
      "Ecuadorianischer Sucre",
        "displayName-count-one";
      "Ecuadorianischer Sucre",
        "displayName-count-other";
      "Ecuadorianische Sucre",
        "symbol";
      "ECS";
    }

```

```

    }
    "ECV";
    {
        "displayName";
        "Verrechnungseinheit für Ecuador",
        "displayName-count-one";
        "Verrechnungseinheiten für Ecuador",
        "displayName-count-other";
        "Verrechnungseinheiten für Ecuador",
        "symbol";
        "ECV";
    }
    "EEK";
    {
        "displayName";
        "Estnische Krone",
        "displayName-count-one";
        "Estnische Krone",
        "displayName-count-other";
        "Estnische Kronen",
        "symbol";
        "EEK";
    }
    "EGP";
    {
        "displayName";
        "Ägyptisches Pfund",
        "displayName-count-one";
        "Ägyptisches Pfund",
        "displayName-count-other";
        "Ägyptische Pfund",
        "symbol";
        "EGP",
        "symbol-alt-narrow";
        "£";
    }
    "ERN";
    {
        "displayName";
        "Eritreischer Nakfa",
        "displayName-count-one";
        "Eritreischer Nakfa",
        "displayName-count-other";
        "Eritreische Nakfa",
        "symbol";
        "ERN";
    }
    "ESA";
    {
        "displayName";
        "Spanische Peseta (A-Konten)",
        "displayName-count-one";
        "Spanische Peseta (A-Konten)",
        "displayName-count-other";
        "Spanische Peseten (A-Konten)",
        "symbol";
        "ESA";
    }

```



```

    }
    "ESB";
    {
        "displayName";
        "Spanische Peseta (konvertibel)",
            "displayName-count-one";
        "Spanische Peseta (konvertibel)",
            "displayName-count-other";
        "Spanische Peseten (konvertibel)",
            "symbol";
        "ESB";
    }
    "ESP";
    {
        "displayName";
        "Spanische Peseta",
            "displayName-count-one";
        "Spanische Peseta",
            "displayName-count-other";
        "Spanische Peseten",
            "symbol";
        "ESP",
            "symbol-alt-narrow";
        "₧";
    }
    "ETB";
    {
        "displayName";
        "Äthiopischer Birr",
            "displayName-count-one";
        "Äthiopischer Birr",
            "displayName-count-other";
        "Äthiopische Birr",
            "symbol";
        "ETB";
    }
    "EUR";
    {
        "displayName";
        "Euro",
            "displayName-count-one";
        "Euro",
            "displayName-count-other";
        "Euro",
            "symbol";
        "€",
            "symbol-alt-narrow";
        "€";
    }
    "FIM";
    {
        "displayName";
        "Finnische Mark",
            "displayName-count-one";
        "Finnische Mark",
            "displayName-count-other";
        "Finnische Mark",

```

```

        "symbol";
        "FIM";
    }
    "FJD";
    {
        "displayName";
        "Fidschi-Dollar",
        "displayName-count-one";
        "Fidschi-Dollar",
        "displayName-count-other";
        "Fidschi-Dollar",
        "symbol";
        "FJD",
        "symbol-alt-narrow";
        "$";
    }
    "FKP";
    {
        "displayName";
        "Falkland-Pfund",
        "displayName-count-one";
        "Falkland-Pfund",
        "displayName-count-other";
        "Falkland-Pfund",
        "symbol";
        "FKP",
        "symbol-alt-narrow";
        "Fl£";
    }
    "FRF";
    {
        "displayName";
        "Französischer Franc",
        "displayName-count-one";
        "Französischer Franc",
        "displayName-count-other";
        "Französische Franc",
        "symbol";
        "FRF";
    }
    "GBP";
    {
        "displayName";
        "Britisches Pfund",
        "displayName-count-one";
        "Britisches Pfund",
        "displayName-count-other";
        "Britische Pfund",
        "symbol";
        "£",
        "symbol-alt-narrow";
        "£";
    }
    "GEK";
    {
        "displayName";
        "Georgischer Kupon Larit",

```

```

        "displayName-count-one";
        "Georgischer Kupon Larit",
        "displayName-count-other";
        "Georgische Kupon Larit",
        "symbol";
        "GEK";
    }
    "GEL";
    {
        "displayName";
        "Georgischer Lari",
        "displayName-count-one";
        "Georgischer Lari",
        "displayName-count-other";
        "Georgische Lari",
        "symbol";
        "GEL",
        "symbol-alt-narrow";
        "ლ",
        "symbol-alt-variant";
        "ლ";
    }
    "GHC";
    {
        "displayName";
        "Ghanaischer Cedi (1979-2007)",
        "displayName-count-one";
        "Ghanaischer Cedi (1979-2007)",
        "displayName-count-other";
        "Ghanaische Cedi (1979-2007)",
        "symbol";
        "GHC";
    }
    "GHS";
    {
        "displayName";
        "Ghanaischer Cedi",
        "displayName-count-one";
        "Ghanaischer Cedi",
        "displayName-count-other";
        "Ghanaische Cedi",
        "symbol";
        "GHS";
    }
    "GIP";
    {
        "displayName";
        "Gibraltar-Pfund",
        "displayName-count-one";
        "Gibraltar-Pfund",
        "displayName-count-other";
        "Gibraltar Pfund",
        "symbol";
        "GIP",
        "symbol-alt-narrow";
        "£";
    }
}

```

```
"GMD";
{
  "displayName";
  "Gambia-Dalasi",
    "displayName-count-one";
  "Gambia-Dalasi",
    "displayName-count-other";
  "Gambia-Dalasi",
    "symbol";
  "GMD";
}
"GNF";
{
  "displayName";
  "Guinea-Franc",
    "displayName-count-one";
  "Guinea-Franc",
    "displayName-count-other";
  "Guinea-Franc",
    "symbol";
  "GNF",
    "symbol-alt-narrow";
  "F.G.";
}
"GNS";
{
  "displayName";
  "Guineischer Syli",
    "displayName-count-one";
  "Guineischer Syli",
    "displayName-count-other";
  "Guineische Syli",
    "symbol";
  "GNS";
}
"GQE";
{
  "displayName";
  "Äquatorialguinea-Ekwele",
    "displayName-count-one";
  "Äquatorialguinea-Ekwele",
    "displayName-count-other";
  "Äquatorialguinea-Ekwele",
    "symbol";
  "GQE";
}
"GRD";
{
  "displayName";
  "Griechische Drachme",
    "displayName-count-one";
  "Griechische Drachme",
    "displayName-count-other";
  "Griechische Drachmen",
    "symbol";
  "GRD";
}
```

```
"GTQ";
{
  "displayName";
  "Guatemaltekinscher Quetzal",
    "displayName-count-one";
  "Guatemaltekinscher Quetzal",
    "displayName-count-other";
  "Guatemaltekinsche Quetzales",
    "symbol";
  "GTQ",
    "symbol-alt-narrow";
  "Q";
}
"GWE";
{
  "displayName";
  "Portugiesisch Guinea Escudo",
    "displayName-count-one";
  "Portugiesisch Guinea Escudo",
    "displayName-count-other";
  "Portugiesisch Guinea Escudo",
    "symbol";
  "GWE";
}
"GWP";
{
  "displayName";
  "Guinea-Bissau Peso",
    "displayName-count-one";
  "Guinea-Bissau Peso",
    "displayName-count-other";
  "Guinea-Bissau Pesos",
    "symbol";
  "GWP";
}
"GYD";
{
  "displayName";
  "Guyana-Dollar",
    "displayName-count-one";
  "Guyana-Dollar",
    "displayName-count-other";
  "Guyana-Dollar",
    "symbol";
  "GYD",
    "symbol-alt-narrow";
  "$";
}
"HKD";
{
  "displayName";
  "Hongkong-Dollar",
    "displayName-count-one";
  "Hongkong-Dollar",
    "displayName-count-other";
  "Hongkong-Dollar",
    "symbol";
}
```

```

        "HK$",
        "symbol-alt-narrow";
        "$";
    }
    "HNL";
    {
        "displayName";
        "Honduras-Lempira",
        "displayName-count-one";
        "Honduras-Lempira",
        "displayName-count-other";
        "Honduras-Lempira",
        "symbol";
        "HNL",
        "symbol-alt-narrow";
        "L";
    }
    "HRD";
    {
        "displayName";
        "Kroatischer Dinar",
        "displayName-count-one";
        "Kroatischer Dinar",
        "displayName-count-other";
        "Kroatische Dinar",
        "symbol";
        "HRD";
    }
    "HRK";
    {
        "displayName";
        "Kroatischer Kuna",
        "displayName-count-one";
        "Kroatischer Kuna",
        "displayName-count-other";
        "Kroatische Kuna",
        "symbol";
        "HRK",
        "symbol-alt-narrow";
        "kn";
    }
    "HTG";
    {
        "displayName";
        "Haitianische Gourde",
        "displayName-count-one";
        "Haitianische Gourde",
        "displayName-count-other";
        "Haitianische Gourdes",
        "symbol";
        "HTG";
    }
    "HUF";
    {
        "displayName";
        "Ungarischer Forint",
        "displayName-count-one";

```

```

        "Ungarischer Forint",
        "displayName-count-other";
        "Ungarische Forint",
        "symbol";
        "HUF",
        "symbol-alt-narrow";
        "Ft";
    }
    "IDR";
    {
        "displayName";
        "Indonesische Rupiah",
        "displayName-count-one";
        "Indonesische Rupiah",
        "displayName-count-other";
        "Indonesische Rupiah",
        "symbol";
        "IDR",
        "symbol-alt-narrow";
        "Rp";
    }
    "IEP";
    {
        "displayName";
        "Irisches Pfund",
        "displayName-count-one";
        "Irisches Pfund",
        "displayName-count-other";
        "Irische Pfund",
        "symbol";
        "IEP";
    }
    "ILP";
    {
        "displayName";
        "Israelisches Pfund",
        "displayName-count-one";
        "Israelisches Pfund",
        "displayName-count-other";
        "Israelische Pfund",
        "symbol";
        "ILP";
    }
    "ILR";
    {
        "displayName";
        "Israelischer Schekel (1980-1985)",
        "displayName-count-one";
        "Israelischer Schekel (1980-1985)",
        "displayName-count-other";
        "Israelische Schekel (1980-1985)";
    }
    "ILS";
    {
        "displayName";
        "Israelischer Neuer Schekel",
        "displayName-count-one";
    }

```

```

        "Israelischer Neuer Schekel",
        "displayName-count-other";
        "Israelische Neue Schekel",
        "symbol";
        "₪",
        "symbol-alt-narrow";
        "₪";
    }
    "INR";
    {
        "displayName";
        "Indische Rupie",
        "displayName-count-one";
        "Indische Rupie",
        "displayName-count-other";
        "Indische Rupien",
        "symbol";
        "₹",
        "symbol-alt-narrow";
        "₹";
    }
    "IQD";
    {
        "displayName";
        "Irakischer Dinar",
        "displayName-count-one";
        "Irakischer Dinar",
        "displayName-count-other";
        "Irakische Dinar",
        "symbol";
        "IQD";
    }
    "IRR";
    {
        "displayName";
        "Iranischer Rial",
        "displayName-count-one";
        "Iranischer Rial",
        "displayName-count-other";
        "Iranische Rial",
        "symbol";
        "IRR";
    }
    "ISJ";
    {
        "displayName";
        "Isländische Krone (1918-1981)",
        "displayName-count-one";
        "Isländische Krone (1918-1981)",
        "displayName-count-other";
        "Isländische Kronen (1918-1981)";
    }
    "ISK";
    {
        "displayName";
        "Isländische Krone",
        "displayName-count-one";
    }

```



```

        "Isländische Krone",
        "displayName-count-other";
    "Isländische Kronen",
        "symbol";
    "ISK",
        "symbol-alt-narrow";
    "kr";
}
"ITL";
{
    "displayName";
    "Italienische Lira",
        "displayName-count-one";
    "Italienische Lira",
        "displayName-count-other";
    "Italienische Lire",
        "symbol";
    "ITL";
}
"JMD";
{
    "displayName";
    "Jamaika-Dollar",
        "displayName-count-one";
    "Jamaika-Dollar",
        "displayName-count-other";
    "Jamaika-Dollar",
        "symbol";
    "JMD",
        "symbol-alt-narrow";
    "$";
}
"JOD";
{
    "displayName";
    "Jordanischer Dinar",
        "displayName-count-one";
    "Jordanischer Dinar",
        "displayName-count-other";
    "Jordanische Dinar",
        "symbol";
    "JOD";
}
"JPY";
{
    "displayName";
    "Japanischer Yen",
        "displayName-count-one";
    "Japanischer Yen",
        "displayName-count-other";
    "Japanische Yen",
        "symbol";
    "¥",
        "symbol-alt-narrow";
    "¥";
}
"KES";

```

```

    {
      "displayName";
      "Kenia-Schilling",
        "displayName-count-one";
      "Kenia-Schilling",
        "displayName-count-other";
      "Kenia-Schilling",
        "symbol";
      "KES";
    }
    "KGS";
    {
      "displayName";
      "Kirgisischer Som",
        "displayName-count-one";
      "Kirgisischer Som",
        "displayName-count-other";
      "Kirgisische Som",
        "symbol";
      "KGS";
    }
    "KHR";
    {
      "displayName";
      "Kambodschanischer Riel",
        "displayName-count-one";
      "Kambodschanischer Riel",
        "displayName-count-other";
      "Kambodschanische Riel",
        "symbol";
      "KHR",
        "symbol-alt-narrow";
      "៛";
    }
    "KMF";
    {
      "displayName";
      "Komoren-Franc",
        "displayName-count-one";
      "Komoren-Franc",
        "displayName-count-other";
      "Komoren-Francs",
        "symbol";
      "KMF",
        "symbol-alt-narrow";
      "FC";
    }
    "KPW";
    {
      "displayName";
      "Nordkoreanischer Won",
        "displayName-count-one";
      "Nordkoreanischer Won",
        "displayName-count-other";
      "Nordkoreanische Won",
        "symbol";
    }

```

```

        "KPW",
        "symbol-alt-narrow";
        "₩";
    }
    "KRH";
    {
        "displayName";
        "Südkoreanischer Hwan (1953-1962)",
        "displayName-count-one";
        "Südkoreanischer Hwan (1953-1962)",
        "displayName-count-other";
        "Südkoreanischer Hwan (1953-1962)",
        "symbol";
        "KRH";
    }
    "KRO";
    {
        "displayName";
        "Südkoreanischer Won (1945-1953)",
        "displayName-count-one";
        "Südkoreanischer Won (1945-1953)",
        "displayName-count-other";
        "Südkoreanischer Won (1945-1953)",
        "symbol";
        "KRO";
    }
    "KRW";
    {
        "displayName";
        "Südkoreanischer Won",
        "displayName-count-one";
        "Südkoreanischer Won",
        "displayName-count-other";
        "Südkoreanische Won",
        "symbol";
        "₩",
        "symbol-alt-narrow";
        "₩";
    }
    "KWD";
    {
        "displayName";
        "Kuwait-Dinar",
        "displayName-count-one";
        "Kuwait-Dinar",
        "displayName-count-other";
        "Kuwait-Dinar",
        "symbol";
        "KWD";
    }
    "KYD";
    {
        "displayName";
        "Kaiman-Dollar",
        "displayName-count-one";
        "Kaiman-Dollar",
        "displayName-count-other";
    }

```

```

        "Kaiman-Dollar",
        "symbol";
        "KYD",
        "symbol-alt-narrow";
        "$";
    }
    "KZT";
    {
        "displayName";
        "Kasachischer Tenge",
        "displayName-count-one";
        "Kasachischer Tenge",
        "displayName-count-other";
        "Kasachische Tenge",
        "symbol";
        "KZT",
        "symbol-alt-narrow";
        "₸";
    }
    "LAK";
    {
        "displayName";
        "Laotischer Kip",
        "displayName-count-one";
        "Laotischer Kip",
        "displayName-count-other";
        "Laotische Kip",
        "symbol";
        "LAK",
        "symbol-alt-narrow";
        "₭";
    }
    "LBP";
    {
        "displayName";
        "Libanesisches Pfund",
        "displayName-count-one";
        "Libanesisches Pfund",
        "displayName-count-other";
        "Libanesische Pfund",
        "symbol";
        "LBP",
        "symbol-alt-narrow";
        "L£";
    }
    "LKR";
    {
        "displayName";
        "Sri-Lanka-Rupie",
        "displayName-count-one";
        "Sri-Lanka-Rupie",
        "displayName-count-other";
        "Sri-Lanka-Rupien",
        "symbol";
        "LKR",
        "symbol-alt-narrow";
        "Rs";
    }

```

```
}
"LRD";
{
  "displayName";
  "Liberianischer Dollar",
    "displayName-count-one";
  "Liberianischer Dollar",
    "displayName-count-other";
  "Liberianische Dollar",
    "symbol";
  "LRD",
    "symbol-alt-narrow";
  "$";
}
"LSL";
{
  "displayName";
  "Loti",
    "displayName-count-one";
  "Loti",
    "displayName-count-other";
  "Loti",
    "symbol";
  "LSL";
}
"LTL";
{
  "displayName";
  "Litauischer Litas",
    "displayName-count-one";
  "Litauischer Litas",
    "displayName-count-other";
  "Litauische Litas",
    "symbol";
  "LTL",
    "symbol-alt-narrow";
  "Lt";
}
"LTT";
{
  "displayName";
  "Litauischer Talonas",
    "displayName-count-one";
  "Litauische Talonas",
    "displayName-count-other";
  "Litauische Talonas",
    "symbol";
  "LTT";
}
"LUC";
{
  "displayName";
  "Luxemburgischer Franc (konvertibel)",
    "displayName-count-one";
  "Luxemburgische Franc (konvertibel)",
    "displayName-count-other";
  "Luxemburgische Franc (konvertibel)",
```

```

        "symbol";
        "LUC";
    }
    "LUF";
    {
        "displayName";
        "Luxemburgischer Franc",
        "displayName-count-one";
        "Luxemburgische Franc",
        "displayName-count-other";
        "Luxemburgische Franc",
        "symbol";
        "LUF";
    }
    "LUL";
    {
        "displayName";
        "Luxemburgischer Finanz-Franc",
        "displayName-count-one";
        "Luxemburgische Finanz-Franc",
        "displayName-count-other";
        "Luxemburgische Finanz-Franc",
        "symbol";
        "LUL";
    }
    "LVL";
    {
        "displayName";
        "Lettischer Lats",
        "displayName-count-one";
        "Lettischer Lats",
        "displayName-count-other";
        "Lettische Lats",
        "symbol";
        "LVL",
        "symbol-alt-narrow";
        "Ls";
    }
    "LVR";
    {
        "displayName";
        "Lettischer Rubel",
        "displayName-count-one";
        "Lettische Rubel",
        "displayName-count-other";
        "Lettische Rubel",
        "symbol";
        "LVR";
    }
    "LYD";
    {
        "displayName";
        "Libyscher Dinar",
        "displayName-count-one";
        "Libyscher Dinar",
        "displayName-count-other";
        "Libysche Dinar",

```

```

        "symbol";
        "LYD";
    }
    "MAD";
    {
        "displayName";
        "Marokkanischer Dirham",
        "displayName-count-one";
        "Marokkanischer Dirham",
        "displayName-count-other";
        "Marokkanische Dirham",
        "symbol";
        "MAD";
    }
    "MAF";
    {
        "displayName";
        "Marokkanischer Franc",
        "displayName-count-one";
        "Marokkanische Franc",
        "displayName-count-other";
        "Marokkanische Franc",
        "symbol";
        "MAF";
    }
    "MCF";
    {
        "displayName";
        "Monegassischer Franc",
        "displayName-count-one";
        "Monegassischer Franc",
        "displayName-count-other";
        "Monegassische Franc",
        "symbol";
        "MCF";
    }
    "MDC";
    {
        "displayName";
        "Moldau-Cupon",
        "displayName-count-one";
        "Moldau-Cupon",
        "displayName-count-other";
        "Moldau-Cupon",
        "symbol";
        "MDC";
    }
    "MDL";
    {
        "displayName";
        "Moldau-Leu",
        "displayName-count-one";
        "Moldau-Leu",
        "displayName-count-other";
        "Moldau-Leu",
        "symbol";
        "MDL";
    }

```

```

    }
    "MGA";
    {
        "displayName";
        "Madagaskar-Ariary",
        "displayName-count-one";
        "Madagaskar-Ariary",
        "displayName-count-other";
        "Madagaskar-Ariary",
        "symbol";
        "MGA",
        "symbol-alt-narrow";
        "Ar";
    }
    "MGF";
    {
        "displayName";
        "Madagaskar-Franc",
        "displayName-count-one";
        "Madagaskar-Franc",
        "displayName-count-other";
        "Madagaskar-Franc",
        "symbol";
        "MGF";
    }
    "MKD";
    {
        "displayName";
        "Mazedonischer Denar",
        "displayName-count-one";
        "Mazedonischer Denar",
        "displayName-count-other";
        "Mazedonische Denari",
        "symbol";
        "MKD";
    }
    "MKN";
    {
        "displayName";
        "Mazedonischer Denar (1992-1993)",
        "displayName-count-one";
        "Mazedonischer Denar (1992-1993)",
        "displayName-count-other";
        "Mazedonische Denar (1992-1993)",
        "symbol";
        "MKN";
    }
    "MLF";
    {
        "displayName";
        "Malischer Franc",
        "displayName-count-one";
        "Malische Franc",
        "displayName-count-other";
        "Malische Franc",
        "symbol";
        "MLF";
    }

```



```

    }
    "MMK";
    {
        "displayName";
        "Myanmarischer Kyat",
        "displayName-count-one";
        "Myanmarischer Kyat",
        "displayName-count-other";
        "Myanmarische Kyat",
        "symbol";
        "MMK",
        "symbol-alt-narrow";
        "K";
    }
    "MNT";
    {
        "displayName";
        "Mongolischer Tögrög",
        "displayName-count-one";
        "Mongolischer Tögrög",
        "displayName-count-other";
        "Mongolische Tögrög",
        "symbol";
        "MNT",
        "symbol-alt-narrow";
        "₮";
    }
    "MOP";
    {
        "displayName";
        "Macao-Pataca",
        "displayName-count-one";
        "Macao-Pataca",
        "displayName-count-other";
        "Macao-Pataca",
        "symbol";
        "MOP";
    }
    "MRO";
    {
        "displayName";
        "Mauretanischer Ouguiya",
        "displayName-count-one";
        "Mauretanischer Ouguiya",
        "displayName-count-other";
        "Mauretanische Ouguiya",
        "symbol";
        "MRO";
    }
    "MTL";
    {
        "displayName";
        "Maltesische Lira",
        "displayName-count-one";
        "Maltesische Lira",
        "displayName-count-other";
        "Maltesische Lira",

```

```

        "symbol";
        "MTL";
    }
    "MTP";
    {
        "displayName";
        "Maltesisches Pfund",
        "displayName-count-one";
        "Maltesische Pfund",
        "displayName-count-other";
        "Maltesische Pfund",
        "symbol";
        "MTP";
    }
    "MUR";
    {
        "displayName";
        "Mauritius-Rupie",
        "displayName-count-one";
        "Mauritius-Rupie",
        "displayName-count-other";
        "Mauritius-Rupien",
        "symbol";
        "MUR",
        "symbol-alt-narrow";
        "Rs";
    }
    "MVP";
    {
        "displayName";
        "Malediven-Rupie (alt)",
        "displayName-count-one";
        "Malediven-Rupie (alt)",
        "displayName-count-other";
        "Malediven-Rupien (alt)";
    }
    "MVR";
    {
        "displayName";
        "Malediven-Rufiyaa",
        "displayName-count-one";
        "Malediven-Rufiyaa",
        "displayName-count-other";
        "Malediven-Rupien",
        "symbol";
        "MVR";
    }
    "MWK";
    {
        "displayName";
        "Malawi-Kwacha",
        "displayName-count-one";
        "Malawi-Kwacha",
        "displayName-count-other";
        "Malawi-Kwacha",
        "symbol";
        "MWK";
    }

```

```

    }
    "MXN";
    {
        "displayName";
        "Mexikanischer Peso",
        "displayName-count-one";
        "Mexikanischer Peso",
        "displayName-count-other";
        "Mexikanische Pesos",
        "symbol";
        "MX$",
        "symbol-alt-narrow";
        "$";
    }
    "MXP";
    {
        "displayName";
        "Mexikanischer Silber-Peso (1861-1992)",
        "displayName-count-one";
        "Mexikanische Silber-Peso (1861-1992)",
        "displayName-count-other";
        "Mexikanische Silber-Pesos (1861-1992)",
        "symbol";
        "MXP";
    }
    "MXV";
    {
        "displayName";
        "Mexicanischer Unidad de Inversion (UDI)",
        "displayName-count-one";
        "Mexicanischer Unidad de Inversion (UDI)",
        "displayName-count-other";
        "Mexikanische Unidad de Inversion (UDI)",
        "symbol";
        "MXV";
    }
    "MYR";
    {
        "displayName";
        "Malaysischer Ringgit",
        "displayName-count-one";
        "Malaysischer Ringgit",
        "displayName-count-other";
        "Malaysische Ringgit",
        "symbol";
        "MYR",
        "symbol-alt-narrow";
        "RM";
    }
    "MZE";
    {
        "displayName";
        "Mosambikanischer Escudo",
        "displayName-count-one";
        "Mozambikanische Escudo",
        "displayName-count-other";
        "Mozambikanische Escudo",

```

```

        "symbol";
        "MZE";
    }
    "MZM";
    {
        "displayName";
        "Mosambikanischer Metical (1980-2006)",
        "displayName-count-one";
        "Mosambikanischer Metical (1980-2006)",
        "displayName-count-other";
        "Mosambikanische Meticaïs (1980-2006)",
        "symbol";
        "MZM";
    }
    "MZN";
    {
        "displayName";
        "Mosambikanischer Metical",
        "displayName-count-one";
        "Mosambikanischer Metical",
        "displayName-count-other";
        "Mosambikanische Meticaïs",
        "symbol";
        "MZN";
    }
    "NAD";
    {
        "displayName";
        "Namibia-Dollar",
        "displayName-count-one";
        "Namibia-Dollar",
        "displayName-count-other";
        "Namibia-Dollar",
        "symbol";
        "NAD",
        "symbol-alt-narrow";
        "$";
    }
    "NGN";
    {
        "displayName";
        "Nigerianischer Naira",
        "displayName-count-one";
        "Nigerianischer Naira",
        "displayName-count-other";
        "Nigerianische Naira",
        "symbol";
        "NGN",
        "symbol-alt-narrow";
        "₦";
    }
    "NIC";
    {
        "displayName";
        "Nicaraguanischer Córdoba (1988-1991)",
        "displayName-count-one";
        "Nicaraguanischer Córdoba (1988-1991)",

```

```

        "displayName-count-other";
        "Nicaraguanische Córdoba (1988-1991)",
        "symbol";
        "NIC";
    }
    "NIO";
    {
        "displayName";
        "Nicaragua-Córdoba",
        "displayName-count-one";
        "Nicaragua-Córdoba",
        "displayName-count-other";
        "Nicaragua-Córdobas",
        "symbol";
        "NIO",
        "symbol-alt-narrow";
        "C$";
    }
    "NLG";
    {
        "displayName";
        "Niederländischer Gulden",
        "displayName-count-one";
        "Niederländischer Gulden",
        "displayName-count-other";
        "Niederländische Gulden",
        "symbol";
        "NLG";
    }
    "NOK";
    {
        "displayName";
        "Norwegische Krone",
        "displayName-count-one";
        "Norwegische Krone",
        "displayName-count-other";
        "Norwegische Kronen",
        "symbol";
        "NOK",
        "symbol-alt-narrow";
        "kr";
    }
    "NPR";
    {
        "displayName";
        "Nepalesische Rupie",
        "displayName-count-one";
        "Nepalesische Rupie",
        "displayName-count-other";
        "Nepalesische Rupien",
        "symbol";
        "NPR",
        "symbol-alt-narrow";
        "Rs";
    }
    "NZD";
    {

```

```

        "displayName";
        "Neuseeland-Dollar",
        "displayName-count-one";
        "Neuseeland-Dollar",
        "displayName-count-other";
        "Neuseeland-Dollar",
        "symbol";
        "NZ$",
        "symbol-alt-narrow";
        "$";
    }
    "OMR";
    {
        "displayName";
        "Omanischer Rial",
        "displayName-count-one";
        "Omanischer Rial",
        "displayName-count-other";
        "Omanische Rials",
        "symbol";
        "OMR";
    }
    "PAB";
    {
        "displayName";
        "Panamaischer Balboa",
        "displayName-count-one";
        "Panamaischer Balboa",
        "displayName-count-other";
        "Panamaische Balboas",
        "symbol";
        "PAB";
    }
    "PEI";
    {
        "displayName";
        "Peruanischer Inti",
        "displayName-count-one";
        "Peruanische Inti",
        "displayName-count-other";
        "Peruanische Inti",
        "symbol";
        "PEI";
    }
    "PEN";
    {
        "displayName";
        "Peruanischer Sol",
        "displayName-count-one";
        "Peruanischer Sol",
        "displayName-count-other";
        "Peruanische Sol",
        "symbol";
        "PEN";
    }
    "PES";
    {

```

```

        "displayName";
        "Peruanischer Sol (1863-1965)",
        "displayName-count-one";
        "Peruanischer Sol (1863-1965)",
        "displayName-count-other";
        "Peruanische Sol (1863-1965)",
        "symbol";
        "PES";
    }
    "PGK";
    {
        "displayName";
        "Papua-Neuguineischer Kina",
        "displayName-count-one";
        "Papua-Neuguineischer Kina",
        "displayName-count-other";
        "Papua-Neuguineische Kina",
        "symbol";
        "PGK";
    }
    "PHP";
    {
        "displayName";
        "Philippinischer Peso",
        "displayName-count-one";
        "Philippinischer Peso",
        "displayName-count-other";
        "Philippinische Pesos",
        "symbol";
        "PHP",
        "symbol-alt-narrow";
        "₱";
    }
    "PKR";
    {
        "displayName";
        "Pakistanische Rupie",
        "displayName-count-one";
        "Pakistanische Rupie",
        "displayName-count-other";
        "Pakistanische Rupien",
        "symbol";
        "PKR",
        "symbol-alt-narrow";
        "Rs";
    }
    "PLN";
    {
        "displayName";
        "Polnischer Złoty",
        "displayName-count-one";
        "Polnischer Złoty",
        "displayName-count-other";
        "Polnische Złoty",
        "symbol";
        "PLN",
        "symbol-alt-narrow";
    }

```

```

        "zł";
    }
    "PLZ";
    {
        "displayName";
        "Polnischer Zloty (1950-1995)",
        "displayName-count-one";
        "Polnischer Zloty (1950-1995)",
        "displayName-count-other";
        "Polnische Zloty (1950-1995)",
        "symbol";
        "PLZ";
    }
    "PTE";
    {
        "displayName";
        "Portugiesischer Escudo",
        "displayName-count-one";
        "Portugiesische Escudo",
        "displayName-count-other";
        "Portugiesische Escudo",
        "symbol";
        "PTE";
    }
    "PYG";
    {
        "displayName";
        "Paraguayischer Guaraní",
        "displayName-count-one";
        "Paraguayischer Guaraní",
        "displayName-count-other";
        "Paraguayische Guaraníes",
        "symbol";
        "PYG",
        "symbol-alt-narrow";
        "₲";
    }
    "QAR";
    {
        "displayName";
        "Katar-Riyal",
        "displayName-count-one";
        "Katar-Riyal",
        "displayName-count-other";
        "Katar-Riyal",
        "symbol";
        "QAR";
    }
    "RHD";
    {
        "displayName";
        "Rhodesischer Dollar",
        "displayName-count-one";
        "Rhodesische Dollar",
        "displayName-count-other";
        "Rhodesische Dollar",
        "symbol";
    }

```



```

        "RHD";
    }
    "ROL";
    {
        "displayName";
        "Rumänischer Leu (1952-2006)",
        "displayName-count-one";
        "Rumänischer Leu (1952-2006)",
        "displayName-count-other";
        "Rumänische Leu (1952-2006)",
        "symbol";
        "ROL";
    }
    "RON";
    {
        "displayName";
        "Rumänischer Leu",
        "displayName-count-one";
        "Rumänischer Leu",
        "displayName-count-other";
        "Rumänische Leu",
        "symbol";
        "RON",
        "symbol-alt-narrow";
        "L";
    }
    "RSD";
    {
        "displayName";
        "Serbischer Dinar",
        "displayName-count-one";
        "Serbischer Dinar",
        "displayName-count-other";
        "Serbische Dinaren",
        "symbol";
        "RSD";
    }
    "RUB";
    {
        "displayName";
        "Russischer Rubel",
        "displayName-count-one";
        "Russischer Rubel",
        "displayName-count-other";
        "Russische Rubel",
        "symbol";
        "RUB",
        "symbol-alt-narrow";
        "₽";
    }
    "RUR";
    {
        "displayName";
        "Russischer Rubel (1991-1998)",
        "displayName-count-one";
        "Russischer Rubel (1991-1998)",
        "displayName-count-other";
    }

```

```

        "Russische Rubel (1991-1998)",
        "symbol";
        "RUR",
        "symbol-alt-narrow";
        "p.";
    }
    "RWF";
    {
        "displayName";
        "Ruanda-Franc",
        "displayName-count-one";
        "Ruanda-Franc",
        "displayName-count-other";
        "Ruanda-Francs",
        "symbol";
        "RWF",
        "symbol-alt-narrow";
        "F.Rw";
    }
    "SAR";
    {
        "displayName";
        "Saudi-Rial",
        "displayName-count-one";
        "Saudi-Rial",
        "displayName-count-other";
        "Saudi-Rial",
        "symbol";
        "SAR";
    }
    "SBD";
    {
        "displayName";
        "Salomonen-Dollar",
        "displayName-count-one";
        "Salomonen-Dollar",
        "displayName-count-other";
        "Salomonen-Dollar",
        "symbol";
        "SBD",
        "symbol-alt-narrow";
        "$";
    }
    "SCR";
    {
        "displayName";
        "Seychellen-Rupie",
        "displayName-count-one";
        "Seychellen-Rupie",
        "displayName-count-other";
        "Seychellen-Rupien",
        "symbol";
        "SCR";
    }
    "SDD";
    {
        "displayName";

```

```

        "Sudanesischer Dinar (1992-2007)",
        "displayName-count-one";
        "Sudanesischer Dinar (1992-2007)",
        "displayName-count-other";
        "Sudanesische Dinar (1992-2007)",
        "symbol";
        "SDD";
    }
    "SDG";
    {
        "displayName";
        "Sudanesisches Pfund",
        "displayName-count-one";
        "Sudanesisches Pfund",
        "displayName-count-other";
        "Sudanesische Pfund",
        "symbol";
        "SDG";
    }
    "SDP";
    {
        "displayName";
        "Sudanesisches Pfund (1957-1998)",
        "displayName-count-one";
        "Sudanesisches Pfund (1957-1998)",
        "displayName-count-other";
        "Sudanesische Pfund (1957-1998)",
        "symbol";
        "SDP";
    }
    "SEK";
    {
        "displayName";
        "Schwedische Krone",
        "displayName-count-one";
        "Schwedische Krone",
        "displayName-count-other";
        "Schwedische Kronen",
        "symbol";
        "SEK",
        "symbol-alt-narrow";
        "kr";
    }
    "SGD";
    {
        "displayName";
        "Singapur-Dollar",
        "displayName-count-one";
        "Singapur-Dollar",
        "displayName-count-other";
        "Singapur-Dollar",
        "symbol";
        "SGD",
        "symbol-alt-narrow";
        "$";
    }
    "SHP";

```

```

    {
      "displayName";
      "St. Helena-Pfund",
        "displayName-count-one";
      "St. Helena-Pfund",
        "displayName-count-other";
      "St. Helena-Pfund",
        "symbol";
      "SHP",
        "symbol-alt-narrow";
      "£";
    }
    "SIT";
    {
      "displayName";
      "Slowenischer Tolar",
        "displayName-count-one";
      "Slowenischer Tolar",
        "displayName-count-other";
      "Slowenische Tolar",
        "symbol";
      "SIT";
    }
    "SKK";
    {
      "displayName";
      "Slowakische Krone",
        "displayName-count-one";
      "Slowakische Kronen",
        "displayName-count-other";
      "Slowakische Kronen",
        "symbol";
      "SKK";
    }
    "SLL";
    {
      "displayName";
      "Sierra-leonischer Leone",
        "displayName-count-one";
      "Sierra-leonischer Leone",
        "displayName-count-other";
      "Sierra-leonische Leones",
        "symbol";
      "SLL";
    }
    "SOS";
    {
      "displayName";
      "Somalia-Schilling",
        "displayName-count-one";
      "Somalia-Schilling",
        "displayName-count-other";
      "Somalia-Schilling",
        "symbol";
      "SOS";
    }
    "SRD";

```

```

    {
      "displayName";
      "Suriname-Dollar",
        "displayName-count-one";
      "Suriname-Dollar",
        "displayName-count-other";
      "Suriname-Dollar",
        "symbol";
      "SRD",
        "symbol-alt-narrow";
      "$";
    }
    "SRG";
    {
      "displayName";
      "Suriname Gulden",
        "displayName-count-one";
      "Suriname-Gulden",
        "displayName-count-other";
      "Suriname-Gulden",
        "symbol";
      "SRG";
    }
    "SSP";
    {
      "displayName";
      "Südsudanesisches Pfund",
        "displayName-count-one";
      "Südsudanesisches Pfund",
        "displayName-count-other";
      "Südsudanesische Pfund",
        "symbol";
      "SSP",
        "symbol-alt-narrow";
      "£";
    }
    "STD";
    {
      "displayName";
      "São-toméischer Dobra",
        "displayName-count-one";
      "São-toméischer Dobra",
        "displayName-count-other";
      "São-toméische Dobra",
        "symbol";
      "STD",
        "symbol-alt-narrow";
      "Db";
    }
    "SUR";
    {
      "displayName";
      "Sowjetischer Rubel",
        "displayName-count-one";
      "Sowjetische Rubel",
        "displayName-count-other";
      "Sowjetische Rubel",

```

```

        "symbol";
        "SUR";
    }
    "SVC";
    {
        "displayName";
        "El Salvador Colon",
        "displayName-count-one";
        "El Salvador-Colon",
        "displayName-count-other";
        "El Salvador-Colon",
        "symbol";
        "SVC";
    }
    "SYP";
    {
        "displayName";
        "Syrisches Pfund",
        "displayName-count-one";
        "Syrisches Pfund",
        "displayName-count-other";
        "Syrische Pfund",
        "symbol";
        "SYP",
        "symbol-alt-narrow";
        "SYP";
    }
    "SZL";
    {
        "displayName";
        "Swasiländischer Lilangeni",
        "displayName-count-one";
        "Swasiländischer Lilangeni",
        "displayName-count-other";
        "Swasiländische Emalangeni",
        "symbol";
        "SZL";
    }
    "THB";
    {
        "displayName";
        "Thailändischer Baht",
        "displayName-count-one";
        "Thailändischer Baht",
        "displayName-count-other";
        "Thailändische Baht",
        "symbol";
        "฿",
        "symbol-alt-narrow";
        "฿";
    }
    "TJR";
    {
        "displayName";
        "Tadschikistan Rubel",
        "displayName-count-one";
        "Tadschikistan-Rubel",

```

```

        "displayName-count-other";
        "Tadschikistan-Rubel",
        "symbol";
        "TJR";
    }
    "TJS";
    {
        "displayName";
        "Tadschikistan-Somoni",
        "displayName-count-one";
        "Tadschikistan-Somoni",
        "displayName-count-other";
        "Tadschikistan-Somoni",
        "symbol";
        "TJS";
    }
    "TMM";
    {
        "displayName";
        "Turkmenistan-Manat (1993-2009)",
        "displayName-count-one";
        "Turkmenistan-Manat (1993-2009)",
        "displayName-count-other";
        "Turkmenistan-Manat (1993-2009)",
        "symbol";
        "TMM";
    }
    "TMT";
    {
        "displayName";
        "Turkmenistan-Manat",
        "displayName-count-one";
        "Turkmenistan-Manat",
        "displayName-count-other";
        "Turkmenistan-Manat",
        "symbol";
        "TMT";
    }
    "TND";
    {
        "displayName";
        "Tunesischer Dinar",
        "displayName-count-one";
        "Tunesischer Dinar",
        "displayName-count-other";
        "Tunesische Dinar",
        "symbol";
        "TND";
    }
    "TOP";
    {
        "displayName";
        "Tongaischer Pa'anga",
        "displayName-count-one";
        "Tongaischer Pa'anga",
        "displayName-count-other";
        "Tongaische Pa'anga",

```

```

        "symbol";
        "TOP",
        "symbol-alt-narrow";
        "T$";
    }
    "TPE";
    {
        "displayName";
        "Timor-Escudo",
        "displayName-count-one";
        "Timor-Escudo",
        "displayName-count-other";
        "Timor-Escudo",
        "symbol";
        "TPE";
    }
    "TRL";
    {
        "displayName";
        "Türkische Lira (1922-2005)",
        "displayName-count-one";
        "Türkische Lira (1922-2005)",
        "displayName-count-other";
        "Türkische Lira (1922-2005)",
        "symbol";
        "TRL";
    }
    "TRY";
    {
        "displayName";
        "Türkische Lira",
        "displayName-count-one";
        "Türkische Lira",
        "displayName-count-other";
        "Türkische Lira",
        "symbol";
        "TRY",
        "symbol-alt-narrow";
        "₺",
        "symbol-alt-variant";
        "TL";
    }
    "TTD";
    {
        "displayName";
        "Trinidad und Tobago-Dollar",
        "displayName-count-one";
        "Trinidad und Tobago-Dollar",
        "displayName-count-other";
        "Trinidad und Tobago-Dollar",
        "symbol";
        "TTD",
        "symbol-alt-narrow";
        "$";
    }
    "TWD";
    {

```



```

        "displayName";
        "Neuer Taiwan-Dollar",
        "displayName-count-one";
        "Neuer Taiwan-Dollar",
        "displayName-count-other";
        "Neue Taiwan-Dollar",
        "symbol";
        "NT$",
        "symbol-alt-narrow";
        "NT$";
    }
    "TZS";
    {
        "displayName";
        "Tansania-Schilling",
        "displayName-count-one";
        "Tansania-Schilling",
        "displayName-count-other";
        "Tansania-Schilling",
        "symbol";
        "TZS";
    }
    "UAH";
    {
        "displayName";
        "Ukrainische Hrywnja",
        "displayName-count-one";
        "Ukrainische Hrywnja",
        "displayName-count-other";
        "Ukrainische Hrywen",
        "symbol";
        "UAH",
        "symbol-alt-narrow";
        "₴";
    }
    "UAK";
    {
        "displayName";
        "Ukrainischer Karbovanetz",
        "displayName-count-one";
        "Ukrainische Karbovanetz",
        "displayName-count-other";
        "Ukrainische Karbovanetz",
        "symbol";
        "UAK";
    }
    "UGS";
    {
        "displayName";
        "Uganda-Schilling (1966-1987)",
        "displayName-count-one";
        "Uganda-Schilling (1966-1987)",
        "displayName-count-other";
        "Uganda-Schilling (1966-1987)",
        "symbol";
        "UGS";
    }
}

```

```

    "UGX";
    {
        "displayName";
        "Uganda-Schilling",
        "displayName-count-one";
        "Uganda-Schilling",
        "displayName-count-other";
        "Uganda-Schilling",
        "symbol";
        "UGX";
    }
    "USD";
    {
        "displayName";
        "US-Dollar",
        "displayName-count-one";
        "US-Dollar",
        "displayName-count-other";
        "US-Dollar",
        "symbol";
        "$",
        "symbol-alt-narrow";
        "$";
    }
    "USN";
    {
        "displayName";
        "US Dollar (Nächster Tag)",
        "displayName-count-one";
        "US-Dollar (Nächster Tag)",
        "displayName-count-other";
        "US-Dollar (Nächster Tag)",
        "symbol";
        "USN";
    }
    "USS";
    {
        "displayName";
        "US Dollar (Gleicher Tag)",
        "displayName-count-one";
        "US-Dollar (Gleicher Tag)",
        "displayName-count-other";
        "US-Dollar (Gleicher Tag)",
        "symbol";
        "USS";
    }
    "UYI";
    {
        "displayName";
        "Uruguayischer Peso (Indexierte Rechnungseinheiten)",
        "displayName-count-one";
        "Uruguayischer Peso (Indexierte Rechnungseinheiten)",
        "displayName-count-other";
        "Uruguayische Pesos (Indexierte Rechnungseinheiten)",
        "symbol";
        "UYI";
    }
}

```

```
"UYP";
{
  "displayName";
  "Uruguayischer Peso (1975-1993)",
    "displayName-count-one";
  "Uruguayischer Peso (1975-1993)",
    "displayName-count-other";
  "Uruguayische Pesos (1975-1993)",
    "symbol";
  "UYP";
}
"UYU";
{
  "displayName";
  "Uruguayischer Peso",
    "displayName-count-one";
  "Uruguayischer Peso",
    "displayName-count-other";
  "Uruguayische Pesos",
    "symbol";
  "UYU",
    "symbol-alt-narrow";
  "$";
}
"UZS";
{
  "displayName";
  "Usbekistan-Sum",
    "displayName-count-one";
  "Usbekistan-Sum",
    "displayName-count-other";
  "Usbekistan-Sum",
    "symbol";
  "UZS";
}
"VEB";
{
  "displayName";
  "Venezolanischer Bolívar (1871-2008)",
    "displayName-count-one";
  "Venezolanischer Bolívar (1871-2008)",
    "displayName-count-other";
  "Venezolanische Bolívares (1871-2008)",
    "symbol";
  "VEB";
}
"VEF";
{
  "displayName";
  "Venezolanischer Bolívar",
    "displayName-count-one";
  "Venezolanischer Bolívar",
    "displayName-count-other";
  "Venezolanische Bolívares",
    "symbol";
  "VEF",
    "symbol-alt-narrow";
}
```

```

        "Bs";
    }
    "VND";
    {
        "displayName";
        "Vietnamesischer Dong",
        "displayName-count-one";
        "Vietnamesischer Dong",
        "displayName-count-other";
        "Vietnamesische Dong",
        "symbol";
        "₫",
        "symbol-alt-narrow";
        "₫";
    }
    "VNN";
    {
        "displayName";
        "Vietnamesischer Dong (1978-1985) ",
        "displayName-count-one";
        "Vietnamesischer Dong (1978-1985) ",
        "displayName-count-other";
        "Vietnamesische Dong (1978-1985) ",
        "symbol";
        "VNN";
    }
    "VUV";
    {
        "displayName";
        "Vanuatu-Vatu",
        "displayName-count-one";
        "Vanuatu-Vatu",
        "displayName-count-other";
        "Vanuatu-Vatu",
        "symbol";
        "VUV";
    }
    "WST";
    {
        "displayName";
        "Samoanischer Tala",
        "displayName-count-one";
        "Samoanischer Tala",
        "displayName-count-other";
        "Samoanische Tala",
        "symbol";
        "WST";
    }
    "XAF";
    {
        "displayName";
        "CFA-Franc (BEAC) ",
        "displayName-count-one";
        "CFA-Franc (BEAC) ",
        "displayName-count-other";
        "CFA-Franc (BEAC) ",
        "symbol";
    }

```

```

        "FCFA";
    }
    "XAG";
    {
        "displayName";
        "Unze Silber",
        "displayName-count-one";
        "Unze Silber",
        "displayName-count-other";
        "Unzen Silber",
        "symbol";
        "XAG";
    }
    "XAU";
    {
        "displayName";
        "Unze Gold",
        "displayName-count-one";
        "Unze Gold",
        "displayName-count-other";
        "Unzen Gold",
        "symbol";
        "XAU";
    }
    "XBA";
    {
        "displayName";
        "Europäische Rechnungseinheit",
        "displayName-count-one";
        "Europäische Rechnungseinheiten",
        "displayName-count-other";
        "Europäische Rechnungseinheiten",
        "symbol";
        "XBA";
    }
    "XBB";
    {
        "displayName";
        "Europäische Währungseinheit (XBB)",
        "displayName-count-one";
        "Europäische Währungseinheiten (XBB)",
        "displayName-count-other";
        "Europäische Währungseinheiten (XBB)",
        "symbol";
        "XBB";
    }
    "XBC";
    {
        "displayName";
        "Europäische Rechnungseinheit (XBC)",
        "displayName-count-one";
        "Europäische Rechnungseinheiten (XBC)",
        "displayName-count-other";
        "Europäische Rechnungseinheiten (XBC)",
        "symbol";
        "XBC";
    }
}

```

```

    "XBD";
    {
        "displayName";
        "Europäische Rechnungseinheit (XBD)",
            "displayName-count-one";
        "Europäische Rechnungseinheiten (XBD)",
            "displayName-count-other";
        "Europäische Rechnungseinheiten (XBD)",
            "symbol";
        "XBD";
    }
    "XCD";
    {
        "displayName";
        "Ostkaribischer Dollar",
            "displayName-count-one";
        "Ostkaribischer Dollar",
            "displayName-count-other";
        "Ostkaribische Dollar",
            "symbol";
        "EC$",
            "symbol-alt-narrow";
        "$";
    }
    "XDR";
    {
        "displayName";
        "Sonderziehungsrechte",
            "displayName-count-one";
        "Sonderziehungsrechte",
            "displayName-count-other";
        "Sonderziehungsrechte",
            "symbol";
        "XDR";
    }
    "XEU";
    {
        "displayName";
        "Europäische Währungseinheit (XEU)",
            "displayName-count-one";
        "Europäische Währungseinheiten (XEU)",
            "displayName-count-other";
        "Europäische Währungseinheiten (XEU)",
            "symbol";
        "XEU";
    }
    "XFO";
    {
        "displayName";
        "Französischer Gold-Franc",
            "displayName-count-one";
        "Französische Gold-Franc",
            "displayName-count-other";
        "Französische Gold-Franc",
            "symbol";
        "XFO";
    }
}

```

```

"XFU";
{
  "displayName";
  "Französischer UIC-Franc",
    "displayName-count-one";
  "Französische UIC-Franc",
    "displayName-count-other";
  "Französische UIC-Franc",
    "symbol";
  "XFU";
}
"XOF";
{
  "displayName";
  "CFA-Franc (BCEAO) ",
    "displayName-count-one";
  "CFA-Franc (BCEAO) ",
    "displayName-count-other";
  "CFA-Francs (BCEAO) ",
    "symbol";
  "CFA";
}
"XPD";
{
  "displayName";
  "Unze Palladium",
    "displayName-count-one";
  "Unze Palladium",
    "displayName-count-other";
  "Unzen Palladium",
    "symbol";
  "XPD";
}
"XPF";
{
  "displayName";
  "CFP-Franc",
    "displayName-count-one";
  "CFP-Franc",
    "displayName-count-other";
  "CFP-Franc",
    "symbol";
  "CFPF";
}
"XPT";
{
  "displayName";
  "Unze Platin",
    "displayName-count-one";
  "Unze Platin",
    "displayName-count-other";
  "Unzen Platin",
    "symbol";
  "XPT";
}
"XRE";
{

```

```

        "displayName";
        "RINET Funds",
        "displayName-count-one";
        "RINET Funds",
        "displayName-count-other";
        "RINET Funds",
        "symbol";
        "XRE";
    }
    "XSU";
    {
        "displayName";
        "SUCRE",
        "displayName-count-one";
        "SUCRE",
        "displayName-count-other";
        "SUCRE",
        "symbol";
        "XSU";
    }
    "XTS";
    {
        "displayName";
        "Testwährung",
        "displayName-count-one";
        "Testwährung",
        "displayName-count-other";
        "Testwährung",
        "symbol";
        "XTS";
    }
    "XUA";
    {
        "displayName";
        "Rechnungseinheit der AfEB",
        "displayName-count-one";
        "Rechnungseinheit der AfEB",
        "displayName-count-other";
        "Rechnungseinheiten der AfEB",
        "symbol";
        "XUA";
    }
    "XXX";
    {
        "displayName";
        "Unbekannte Währung",
        "displayName-count-one";
        "(unbekannte Währung)",
        "displayName-count-other";
        "(unbekannte Währung)",
        "symbol";
        "XXX";
    }
    "YDD";
    {
        "displayName";
        "Jemen-Dinar",

```



```

        "displayName-count-one";
        "Jemen-Dinar",
        "displayName-count-other";
        "Jemen-Dinar",
        "symbol";
        "YDD";
    }
    "YER";
    {
        "displayName";
        "Jemen-Rial",
        "displayName-count-one";
        "Jemen-Rial",
        "displayName-count-other";
        "Jemen-Rial",
        "symbol";
        "YER";
    }
    "YUD";
    {
        "displayName";
        "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-one";
        "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-other";
        "Jugoslawische Dinar (1966-1990)",
        "symbol";
        "YUD";
    }
    "YUM";
    {
        "displayName";
        "Jugoslawischer Neuer Dinar (1994-2002)",
        "displayName-count-one";
        "Jugoslawischer Neuer Dinar (1994-2002)",
        "displayName-count-other";
        "Jugoslawische Neue Dinar (1994-2002)",
        "symbol";
        "YUM";
    }
    "YUN";
    {
        "displayName";
        "Jugoslawischer Dinar (konvertibel)",
        "displayName-count-one";
        "Jugoslawische Dinar (konvertibel)",
        "displayName-count-other";
        "Jugoslawische Dinar (konvertibel)",
        "symbol";
        "YUN";
    }
    "YUR";
    {
        "displayName";
        "Jugoslawischer reformierter Dinar (1992-1993)",
        "displayName-count-one";
        "Jugoslawischer reformierter Dinar (1992-1993)",

```

```

        "displayName-count-other";
        "Jugoslawische reformierte Dinar (1992-1993)",
        "symbol";
        "YUR";
    }
    "ZAL";
    {
        "displayName";
        "Südafrikanischer Rand (Finanz)",
        "displayName-count-one";
        "Südafrikanischer Rand (Finanz)",
        "displayName-count-other";
        "Südafrikanischer Rand (Finanz)",
        "symbol";
        "ZAL";
    }
    "ZAR";
    {
        "displayName";
        "Südafrikanischer Rand",
        "displayName-count-one";
        "Südafrikanischer Rand",
        "displayName-count-other";
        "Südafrikanische Rand",
        "symbol";
        "ZAR",
        "symbol-alt-narrow";
        "R";
    }
    "ZMK";
    {
        "displayName";
        "Kwacha (1968-2012)",
        "displayName-count-one";
        "Kwacha (1968-2012)",
        "displayName-count-other";
        "Kwacha (1968-2012)",
        "symbol";
        "ZMK";
    }
    "ZMW";
    {
        "displayName";
        "Kwacha",
        "displayName-count-one";
        "Kwacha",
        "displayName-count-other";
        "Kwacha",
        "symbol";
        "ZMW",
        "symbol-alt-narrow";
        "K";
    }
    "ZRN";
    {
        "displayName";
        "Zaire-Neuer Zaïre (1993-1998)",

```

```

        "displayName-count-one";
        "Zaire-Neuer Zaïre (1993-1998)",
            "displayName-count-other";
        "Zaire-Neue Zaïre (1993-1998)",
            "symbol";
        "ZRN";
    }
    "ZRZ";
    {
        "displayName";
        "Zaire-Zaire (1971-1993)",
            "displayName-count-one";
        "Zaire-Zaire (1971-1993)",
            "displayName-count-other";
        "Zaire-Zaïre (1971-1993)",
            "symbol";
        "ZRZ";
    }
    "ZWD";
    {
        "displayName";
        "Simbabwe-Dollar (1980-2008)",
            "displayName-count-one";
        "Simbabwe-Dollar (1980-2008)",
            "displayName-count-other";
        "Simbabwe-Dollar (1980-2008)",
            "symbol";
        "ZWD";
    }
    "ZWL";
    {
        "displayName";
        "Simbabwe-Dollar (2009)",
            "displayName-count-one";
        "Simbabwe-Dollar (2009)",
            "displayName-count-other";
        "Simbabwe-Dollar (2009)",
            "symbol";
        "ZWL";
    }
    "ZWR";
    {
        "displayName";
        "Simbabwe-Dollar (2008)",
            "displayName-count-one";
        "Simbabwe-Dollar (2008)",
            "displayName-count-other";
        "Simbabwe-Dollar (2008)",
            "symbol";
        "ZWR";
    }
}
}
}
}
}

```

CURRENCYDATA.JSX

```
{
  "supplemental";
  {
    "version";
    {
      "_number";
      "$Revision: 13254 $",
      "_unicodeVersion";
      "9.0.0",
      "_cldrVersion";
      "31";
    }
  }
  "currencyData";
  {
    "fractions";
    {
      "ADP";
      {
        "_rounding";
        "0",
        "_digits";
        "0";
      }
      "AFN";
      {
        "_rounding";
        "0",
        "_digits";
        "0";
      }
      "ALL";
      {
        "_rounding";
        "0",
        "_digits";
        "0";
      }
      "AMD";
      {
        "_rounding";
        "0",
        "_digits";
        "0";
      }
      "BHD";
      {
        "_rounding";
        "0",
        "_digits";
        "3";
      }
      "BIF";
      {
```

```
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "BYN";
    {
        "_rounding";
        "0",
        "_digits";
        "2";
    }
    "BYR";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "CAD";
    {
        "_rounding";
        "0",
        "_digits";
        "2",
        "_cashRounding";
        "5";
    }
    "CHF";
    {
        "_rounding";
        "0",
        "_digits";
        "2",
        "_cashRounding";
        "5";
    }
    "CLF";
    {
        "_rounding";
        "0",
        "_digits";
        "4";
    }
    "CLP";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "COP";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
}
```

```
}
"CRC";
{
  "_rounding";
  "0",
  "_digits";
  "2",
  "_cashRounding";
  "0",
  "_cashDigits";
  "0";
}
"CZK";
{
  "_rounding";
  "0",
  "_digits";
  "2",
  "_cashRounding";
  "0",
  "_cashDigits";
  "0";
}
"DEFAULT";
{
  "_rounding";
  "0",
  "_digits";
  "2";
}
"DJF";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
"ESP";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
"GNF";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
"GYD";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
```

```
}
"HUF";
{
  "_rounding";
  "0",
  "_digits";
  "2",
  "_cashRounding";
  "0",
  "_cashDigits";
  "0";
}
>IDR";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
>IQD";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
>IRR";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
>ISK";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
>ITL";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
>JOD";
{
  "_rounding";
  "0",
  "_digits";
  "3";
}
>JPY";
{
  "_rounding";
```

```
        "0",
        "_digits";
    "0";
}
"KMF";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"KPW";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"KRW";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"KWD";
{
    "_rounding";
    "0",
    "_digits";
    "3";
}
"LAK";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"LBP";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"LUF";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"LYD";
{
    "_rounding";
    "0",
```



```

        "_digits";
        "3";
    }
    "MGA";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "MGF";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "MMK";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "MNT";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "MRO";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "MUR";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "OMR";
    {
        "_rounding";
        "0",
        "_digits";
        "3";
    }
    "PKR";
    {
        "_rounding";
        "0",
        "_digits";

```

```
        "0";
    }
    "PYG";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "RSD";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "RWF";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "SLL";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "SOS";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "STD";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "SYP";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "TMM";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
}
```

```
}
"TND";
{
  "_rounding";
  "0",
  "_digits";
  "3";
}
"TRL";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
"TWD";
{
  "_rounding";
  "0",
  "_digits";
  "2",
  "_cashRounding";
  "0",
  "_cashDigits";
  "0";
}
"TZS";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
"UGX";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
"UYI";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
"UZS";
{
  "_rounding";
  "0",
  "_digits";
  "0";
}
"VND";
{
  "_rounding";
```

```
        "0",
        "_digits";
    "0";
}
"VUV";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"XAF";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"XOF";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"XPF";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"YER";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"ZMK";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"ZWD";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
}
"region";
{
    "AC";
```

```

[
  {
    "SHP": {
      "_from": "1976-01-01"
    }
  },
  "AD";
  [
    {
      "ESP": {
        "_from": "1873-01-01",
        "_to": "2002-02-28"
      }
    },
    {
      "ADP": {
        "_from": "1936-01-01",
        "_to": "2001-12-31"
      }
    },
    {
      "FRF": {
        "_from": "1960-01-01",
        "_to": "2002-02-17"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "AE";
  [
    {
      "AED": {
        "_from": "1973-05-19"
      }
    }
  ],
  "AF";
  [
    {
      "AFA": {
        "_from": "1927-03-14",
        "_to": "2002-12-31"
      }
    },
    {
      "AFN": {
        "_from": "2002-10-07"
      }
    }
  ],
  "AG";
  [

```

```

        {
            "XCD": {
                "_from": "1965-10-06"
            }
        },
        "AI";
    [
        {
            "XCD": {
                "_from": "1965-10-06"
            }
        },
        "AL";
    [
        {
            "ALK": {
                "_from": "1946-11-01",
                "_to": "1965-08-16"
            }
        },
        {
            "ALL": {
                "_from": "1965-08-16"
            }
        }
    ],
    "AM";
    [
        {
            "SUR": {
                "_from": "1961-01-01",
                "_to": "1991-12-25"
            }
        },
        {
            "RUR": {
                "_from": "1991-12-25",
                "_to": "1993-11-22"
            }
        },
        {
            "AMD": {
                "_from": "1993-11-22"
            }
        }
    ],
    "AO";
    [
        {
            "AOK": {
                "_from": "1977-01-08",
                "_to": "1991-03-01"
            }
        },
    ]

```

```

        "AON": {
            "_from": "1990-09-25",
            "_to": "2000-02-01"
        }
    },
    {
        "AOR": {
            "_from": "1995-07-01",
            "_to": "2000-02-01"
        }
    },
    {
        "AOA": {
            "_from": "1999-12-13"
        }
    }
],
"AQ";
[
    {
        "XXX": {
            "_tender": "false"
        }
    }
],
"AR";
[
    {
        "ARM": {
            "_from": "1881-11-05",
            "_to": "1970-01-01"
        }
    },
    {
        "ARL": {
            "_from": "1970-01-01",
            "_to": "1983-06-01"
        }
    },
    {
        "ARP": {
            "_from": "1983-06-01",
            "_to": "1985-06-14"
        }
    },
    {
        "ARA": {
            "_from": "1985-06-14",
            "_to": "1992-01-01"
        }
    },
    {
        "ARS": {
            "_from": "1992-01-01"
        }
    }
],

```

```
        "AS";
    [
      {
        "USD": {
          "_from": "1904-07-16"
        }
      }
    ],
    "AT";
    [
      {
        "ATS": {
          "_from": "1947-12-04",
          "_to": "2002-02-28"
        }
      },
      {
        "EUR": {
          "_from": "1999-01-01"
        }
      }
    ],
    "AU";
    [
      {
        "AUD": {
          "_from": "1966-02-14"
        }
      }
    ],
    "AW";
    [
      {
        "ANG": {
          "_from": "1940-05-10",
          "_to": "1986-01-01"
        }
      },
      {
        "AWG": {
          "_from": "1986-01-01"
        }
      }
    ],
    "AX";
    [
      {
        "EUR": {
          "_from": "1999-01-01"
        }
      }
    ],
    "AZ";
    [
      {
        "SUR": {
          "_from": "1961-01-01",
```



```
        "_to": "1991-12-25"
      },
    },
    {
      "RUR": {
        "_from": "1991-12-25",
        "_to": "1994-01-01"
      }
    },
    {
      "AZM": {
        "_from": "1993-11-22",
        "_to": "2006-12-31"
      }
    },
    {
      "AZN": {
        "_from": "2006-01-01"
      }
    }
  ],
  "BA";
[
  {
    "YUD": {
      "_from": "1966-01-01",
      "_to": "1990-01-01"
    }
  },
  {
    "YUN": {
      "_from": "1990-01-01",
      "_to": "1992-07-01"
    }
  },
  {
    "YUR": {
      "_from": "1992-07-01",
      "_to": "1993-10-01"
    }
  },
  {
    "BAD": {
      "_from": "1992-07-01",
      "_to": "1994-08-15"
    }
  },
  {
    "BAN": {
      "_from": "1994-08-15",
      "_to": "1997-07-01"
    }
  },
  {
    "BAM": {
      "_from": "1995-01-01"
    }
  }
]
```

```

    },
    "BB";
    [
      {
        "XCD": {
          "_from": "1965-10-06",
          "_to": "1973-12-03"
        }
      },
      {
        "BBD": {
          "_from": "1973-12-03"
        }
      }
    ],
    "BD";
    [
      {
        "INR": {
          "_from": "1835-08-17",
          "_to": "1948-04-01"
        }
      },
      {
        "PKR": {
          "_from": "1948-04-01",
          "_to": "1972-01-01"
        }
      },
      {
        "BDT": {
          "_from": "1972-01-01"
        }
      }
    ],
    "BE";
    [
      {
        "NLG": {
          "_from": "1816-12-15",
          "_to": "1831-02-07"
        }
      },
      {
        "BEF": {
          "_from": "1831-02-07",
          "_to": "2002-02-28"
        }
      },
      {
        "BEC": {
          "_tender": "false",
          "_from": "1970-01-01",
          "_to": "1990-03-05"
        }
      }
    ],
  ],

```

```

        {
            "BEL": {
                "_tender": "false",
                "_from": "1970-01-01",
                "_to": "1990-03-05"
            }
        },
        {
            "EUR": {
                "_from": "1999-01-01"
            }
        }
    ],
    "BF";
    [
        {
            "XOF": {
                "_from": "1984-08-04"
            }
        }
    ],
    "BG";
    [
        {
            "BGO": {
                "_from": "1879-07-08",
                "_to": "1952-05-12"
            }
        },
        {
            "BGM": {
                "_from": "1952-05-12",
                "_to": "1962-01-01"
            }
        },
        {
            "BGL": {
                "_from": "1962-01-01",
                "_to": "1999-07-05"
            }
        },
        {
            "BGN": {
                "_from": "1999-07-05"
            }
        }
    ],
    "BH";
    [
        {
            "BHD": {
                "_from": "1965-10-16"
            }
        }
    ],
    "BI";
    [

```

```
        {
          "BIF": {
            "_from": "1964-05-19"
          }
        },
        "BJ";
        [
          {
            "XOF": {
              "_from": "1975-11-30"
            }
          }
        ],
        "BL";
        [
          {
            "FRF": {
              "_from": "1960-01-01",
              "_to": "2002-02-17"
            }
          },
          {
            "EUR": {
              "_from": "1999-01-01"
            }
          }
        ],
        "BM";
        [
          {
            "BMD": {
              "_from": "1970-02-06"
            }
          }
        ],
        "BN";
        [
          {
            "MYR": {
              "_from": "1963-09-16",
              "_to": "1967-06-12"
            }
          },
          {
            "BND": {
              "_from": "1967-06-12"
            }
          }
        ],
        "BO";
        [
          {
            "BOV": {
              "_tender": "false"
            }
          },
          {
```

```

        {
            "BOL": {
                "_from": "1863-06-23",
                "_to": "1963-01-01"
            }
        },
        {
            "BOP": {
                "_from": "1963-01-01",
                "_to": "1986-12-31"
            }
        },
        {
            "BOB": {
                "_from": "1987-01-01"
            }
        }
    ],
    "BQ";
    [
        {
            "ANG": {
                "_from": "2010-10-10",
                "_to": "2011-01-01"
            }
        },
        {
            "USD": {
                "_from": "2011-01-01"
            }
        }
    ],
    "BR";
    [
        {
            "BRZ": {
                "_from": "1942-11-01",
                "_to": "1967-02-13"
            }
        },
        {
            "BRB": {
                "_from": "1967-02-13",
                "_to": "1986-02-28"
            }
        },
        {
            "BRC": {
                "_from": "1986-02-28",
                "_to": "1989-01-15"
            }
        },
        {
            "BRN": {
                "_from": "1989-01-15",
                "_to": "1990-03-16"
            }
        }
    ]

```

```
    },
    {
      "BRE": {
        "_from": "1990-03-16",
        "_to": "1993-08-01"
      }
    },
    {
      "BRR": {
        "_from": "1993-08-01",
        "_to": "1994-07-01"
      }
    },
    {
      "BRL": {
        "_from": "1994-07-01"
      }
    }
  ],
  "BS";
  [
    {
      "BSD": {
        "_from": "1966-05-25"
      }
    }
  ],
  "BT";
  [
    {
      "INR": {
        "_from": "1907-01-01"
      }
    },
    {
      "BTN": {
        "_from": "1974-04-16"
      }
    }
  ],
  "BU";
  [
    {
      "BUK": {
        "_from": "1952-07-01",
        "_to": "1989-06-18"
      }
    }
  ],
  "BV";
  [
    {
      "NOK": {
        "_from": "1905-06-07"
      }
    }
  ],
  ],
```

```
    "BW";
    [
      {
        "ZAR": {
          "_from": "1961-02-14",
          "_to": "1976-08-23"
        }
      },
      {
        "BWP": {
          "_from": "1976-08-23"
        }
      }
    ],
    "BY";
    [
      {
        "SUR": {
          "_from": "1961-01-01",
          "_to": "1991-12-25"
        }
      },
      {
        "RUR": {
          "_from": "1991-12-25",
          "_to": "1994-11-08"
        }
      },
      {
        "BYB": {
          "_from": "1994-08-01",
          "_to": "2000-12-31"
        }
      },
      {
        "BYR": {
          "_from": "2000-01-01",
          "_to": "2017-01-01"
        }
      },
      {
        "BYN": {
          "_from": "2016-07-01"
        }
      }
    ],
    "BZ";
    [
      {
        "BZD": {
          "_from": "1974-01-01"
        }
      }
    ],
    "CA";
    [
      {
```

```

        "CAD": {
            "_from": "1858-01-01"
        }
    ],
    "CC";
    [
        {
            "AUD": {
                "_from": "1966-02-14"
            }
        }
    ],
    "CD";
    [
        {
            "ZRZ": {
                "_from": "1971-10-27",
                "_to": "1993-11-01"
            }
        },
        {
            "ZRN": {
                "_from": "1993-11-01",
                "_to": "1998-07-01"
            }
        },
        {
            "CDF": {
                "_from": "1998-07-01"
            }
        }
    ],
    "CF";
    [
        {
            "XAF": {
                "_from": "1993-01-01"
            }
        }
    ],
    "CG";
    [
        {
            "XAF": {
                "_from": "1993-01-01"
            }
        }
    ],
    "CH";
    [
        {
            "CHE": {
                "_tender": "false"
            }
        },
        {

```



```

        "CHW": {
            "_tender": "false"
        }
    },
    {
        "CHF": {
            "_from": "1799-03-17"
        }
    }
],
"CI";
[
    {
        "XOF": {
            "_from": "1958-12-04"
        }
    }
],
"CK";
[
    {
        "NZD": {
            "_from": "1967-07-10"
        }
    }
],
"CL";
[
    {
        "CLF": {
            "_tender": "false"
        }
    },
    {
        "CLE": {
            "_from": "1960-01-01",
            "_to": "1975-09-29"
        }
    },
    {
        "CLP": {
            "_from": "1975-09-29"
        }
    }
],
"CM";
[
    {
        "XAF": {
            "_from": "1973-04-01"
        }
    }
],
"CN";
[
    {
        "CNY": {

```

```

        "_from": "1953-03-01"
      }
    },
    {
      "CNX": {
        "_tender": "false",
        "_from": "1979-01-01",
        "_to": "1998-12-31"
      }
    }
  ],
  "CO";
  [
    {
      "COU": {
        "_tender": "false"
      }
    },
    {
      "COP": {
        "_from": "1905-01-01"
      }
    }
  ],
  "CP";
  [
    {
      "XXX": {
        "_tender": "false"
      }
    }
  ],
  "CR";
  [
    {
      "CRC": {
        "_from": "1896-10-26"
      }
    }
  ],
  "CS";
  [
    {
      "YUM": {
        "_from": "1994-01-24",
        "_to": "2002-05-15"
      }
    },
    {
      "CSD": {
        "_from": "2002-05-15",
        "_to": "2006-06-03"
      }
    },
    {
      "EUR": {
        "_from": "2003-02-04",

```

```

        "_to": "2006-06-03"
      }
    }
  ],
  "CU";
  [
    {
      "CUP": {
        "_from": "1859-01-01"
      }
    },
    {
      "USD": {
        "_from": "1899-01-01",
        "_to": "1959-01-01"
      }
    },
    {
      "CUC": {
        "_from": "1994-01-01"
      }
    }
  ],
  "CV";
  [
    {
      "PTE": {
        "_from": "1911-05-22",
        "_to": "1975-07-05"
      }
    },
    {
      "CVE": {
        "_from": "1914-01-01"
      }
    }
  ],
  "CW";
  [
    {
      "ANG": {
        "_from": "2010-10-10"
      }
    }
  ],
  "CX";
  [
    {
      "AUD": {
        "_from": "1966-02-14"
      }
    }
  ],
  "CY";
  [
    {
      "CYP": {

```

```

        "_from": "1914-09-10",
        "_to": "2008-01-31"
      }
    },
    {
      "EUR": {
        "_from": "2008-01-01"
      }
    }
  ],
  "CZ";
  [
    {
      "CSK": {
        "_from": "1953-06-01",
        "_to": "1993-03-01"
      }
    },
    {
      "CZK": {
        "_from": "1993-01-01"
      }
    }
  ],
  "DD";
  [
    {
      "DDM": {
        "_from": "1948-07-20",
        "_to": "1990-10-02"
      }
    }
  ],
  "DE";
  [
    {
      "DEM": {
        "_from": "1948-06-20",
        "_to": "2002-02-28"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "DG";
  [
    {
      "USD": {
        "_from": "1965-11-08"
      }
    }
  ],
  "DJ";
  [

```

```
        {
          "DJF": {
            "_from": "1977-06-27"
          }
        },
        "DK";
        [
          {
            "DKK": {
              "_from": "1873-05-27"
            }
          }
        ],
        "DM";
        [
          {
            "XCD": {
              "_from": "1965-10-06"
            }
          }
        ],
        "DO";
        [
          {
            "USD": {
              "_from": "1905-06-21",
              "_to": "1947-10-01"
            }
          },
          {
            "DOP": {
              "_from": "1947-10-01"
            }
          }
        ],
        "DZ";
        [
          {
            "DZD": {
              "_from": "1964-04-01"
            }
          }
        ],
        "EA";
        [
          {
            "EUR": {
              "_from": "1999-01-01"
            }
          }
        ],
        "EC";
        [
          {
            "ECS": {
              "_from": "1884-04-01",
```

```

        "_to": "2000-10-02"
      }
    },
    {
      "ECV": {
        "_tender": "false",
        "_from": "1993-05-23",
        "_to": "2000-01-09"
      }
    },
    {
      "USD": {
        "_from": "2000-10-02"
      }
    }
  ],
  "EE";
  [
    {
      "SUR": {
        "_from": "1961-01-01",
        "_to": "1992-06-20"
      }
    },
    {
      "EEK": {
        "_from": "1992-06-21",
        "_to": "2010-12-31"
      }
    },
    {
      "EUR": {
        "_from": "2011-01-01"
      }
    }
  ],
  "EG";
  [
    {
      "EGP": {
        "_from": "1885-11-14"
      }
    }
  ],
  "EH";
  [
    {
      "MAD": {
        "_from": "1976-02-26"
      }
    }
  ],
  "ER";
  [
    {
      "ETB": {
        "_from": "1993-05-24",

```

```

        "_to": "1997-11-08"
      }
    },
    {
      "ERN": {
        "_from": "1997-11-08"
      }
    }
  ],
  "ES";
  [
    {
      "ESP": {
        "_from": "1868-10-19",
        "_to": "2002-02-28"
      }
    },
    {
      "ESB": {
        "_tender": "false",
        "_from": "1975-01-01",
        "_to": "1994-12-31"
      }
    },
    {
      "ESA": {
        "_tender": "false",
        "_from": "1978-01-01",
        "_to": "1981-12-31"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "ET";
  [
    {
      "ETB": {
        "_from": "1976-09-15"
      }
    }
  ],
  "EU";
  [
    {
      "XEU": {
        "_tender": "false",
        "_from": "1979-01-01",
        "_to": "1998-12-31"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ]

```

```

    }
],
    "FI";
[
    {
        "FIM": {
            "_from": "1963-01-01",
            "_to": "2002-02-28"
        }
    },
    {
        "EUR": {
            "_from": "1999-01-01"
        }
    }
],
    "FJ";
[
    {
        "FJD": {
            "_from": "1969-01-13"
        }
    }
],
    "FK";
[
    {
        "FKP": {
            "_from": "1901-01-01"
        }
    }
],
    "FM";
[
    {
        "JPY": {
            "_from": "1914-10-03",
            "_to": "1944-01-01"
        }
    },
    {
        "USD": {
            "_from": "1944-01-01"
        }
    }
],
    "FO";
[
    {
        "DKK": {
            "_from": "1948-01-01"
        }
    }
],
    "FR";
[

```



```
{
  "FRF": {
    "_from": "1960-01-01",
    "_to": "2002-02-17"
  },
  {
    "EUR": {
      "_from": "1999-01-01"
    }
  },
],
"GA";
[
  {
    "XAF": {
      "_from": "1993-01-01"
    }
  },
],
"GB";
[
  {
    "GBP": {
      "_from": "1694-07-27"
    }
  },
],
"GD";
[
  {
    "XCD": {
      "_from": "1967-02-27"
    }
  },
],
"GE";
[
  {
    "SUR": {
      "_from": "1961-01-01",
      "_to": "1991-12-25"
    }
  },
  {
    "RUR": {
      "_from": "1991-12-25",
      "_to": "1993-06-11"
    }
  },
  {
    "GEK": {
      "_from": "1993-04-05",
      "_to": "1995-09-25"
    }
  },
],
{
```

```
        "GEL": {
            "_from": "1995-09-23"
        }
    ],
    "GF";
    [
        {
            "FRF": {
                "_from": "1960-01-01",
                "_to": "2002-02-17"
            }
        },
        {
            "EUR": {
                "_from": "1999-01-01"
            }
        }
    ],
    "GG";
    [
        {
            "GBP": {
                "_from": "1830-01-01"
            }
        }
    ],
    "GH";
    [
        {
            "GHC": {
                "_from": "1979-03-09",
                "_to": "2007-12-31"
            }
        },
        {
            "GHS": {
                "_from": "2007-07-03"
            }
        }
    ],
    "GI";
    [
        {
            "GIP": {
                "_from": "1713-01-01"
            }
        }
    ],
    "GL";
    [
        {
            "DKK": {
                "_from": "1873-05-27"
            }
        }
    ],
    ],
```

```
      "GM";
    [
      {
        "GMD": {
          "_from": "1971-07-01"
        }
      }
    ],
    "GN";
    [
      {
        "GNS": {
          "_from": "1972-10-02",
          "_to": "1986-01-06"
        }
      },
      {
        "GNF": {
          "_from": "1986-01-06"
        }
      }
    ],
    "GP";
    [
      {
        "FRF": {
          "_from": "1960-01-01",
          "_to": "2002-02-17"
        }
      },
      {
        "EUR": {
          "_from": "1999-01-01"
        }
      }
    ],
    "GQ";
    [
      {
        "GQE": {
          "_from": "1975-07-07",
          "_to": "1986-06-01"
        }
      },
      {
        "XAF": {
          "_from": "1993-01-01"
        }
      }
    ],
    "GR";
    [
      {
        "GRD": {
          "_from": "1954-05-01",
          "_to": "2002-02-28"
        }
      }
    ]
  ]
}
```

```
    },  
    {  
      "EUR": {  
        "_from": "2001-01-01"  
      }  
    }  
  ],  
  "GS";  
  [  
    {  
      "GBP": {  
        "_from": "1908-01-01"  
      }  
    }  
  ],  
  "GT";  
  [  
    {  
      "GTQ": {  
        "_from": "1925-05-27"  
      }  
    }  
  ],  
  "GU";  
  [  
    {  
      "USD": {  
        "_from": "1944-08-21"  
      }  
    }  
  ],  
  "GW";  
  [  
    {  
      "GWE": {  
        "_from": "1914-01-01",  
        "_to": "1976-02-28"  
      }  
    },  
    {  
      "GWP": {  
        "_from": "1976-02-28",  
        "_to": "1997-03-31"  
      }  
    },  
    {  
      "XOF": {  
        "_from": "1997-03-31"  
      }  
    }  
  ],  
  "GY";  
  [  
    {  
      "GYD": {  
        "_from": "1966-05-26"  
      }  
    }  
  ]  
];
```

```

    },
    "HK";
    [
      {
        "HKD": {
          "_from": "1895-02-02"
        }
      }
    ],
    "HM";
    [
      {
        "AUD": {
          "_from": "1967-02-16"
        }
      }
    ],
    "HN";
    [
      {
        "HNL": {
          "_from": "1926-04-03"
        }
      }
    ],
    "HR";
    [
      {
        "YUD": {
          "_from": "1966-01-01",
          "_to": "1990-01-01"
        }
      },
      {
        "YUN": {
          "_from": "1990-01-01",
          "_to": "1991-12-23"
        }
      },
      {
        "HRD": {
          "_from": "1991-12-23",
          "_to": "1995-01-01"
        }
      },
      {
        "HRK": {
          "_from": "1994-05-30"
        }
      }
    ],
    "HT";
    [
      {
        "HTG": {
          "_from": "1872-08-26"
        }
      }
    ]
  ],
  "HT";
  [
    {
      "HTG": {
        "_from": "1872-08-26"
      }
    }
  ]

```

```

    },
    {
      "USD": {
        "_from": "1915-01-01"
      }
    }
  ],
  "HU";
  [
    {
      "HUF": {
        "_from": "1946-07-23"
      }
    }
  ],
  "IC";
  [
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "ID";
  [
    {
      "IDR": {
        "_from": "1965-12-13"
      }
    }
  ],
  "IE";
  [
    {
      "GBP": {
        "_from": "1800-01-01",
        "_to": "1922-01-01"
      }
    },
    {
      "IEP": {
        "_from": "1922-01-01",
        "_to": "2002-02-09"
      }
    }
  ],
  {
    "EUR": {
      "_from": "1999-01-01"
    }
  }
],
  "IL";
  [
    {
      "ILP": {
        "_from": "1948-08-16",

```

```
        "_to": "1980-02-22"
      }
    },
    {
      "ILR": {
        "_from": "1980-02-22",
        "_to": "1985-09-04"
      }
    },
    {
      "ILS": {
        "_from": "1985-09-04"
      }
    }
  ],
  "IM";
[
  {
    "GBP": {
      "_from": "1840-01-03"
    }
  }
],
  "IN";
[
  {
    "INR": {
      "_from": "1835-08-17"
    }
  }
],
  "IO";
[
  {
    "USD": {
      "_from": "1965-11-08"
    }
  }
],
  "IQ";
[
  {
    "EGP": {
      "_from": "1920-11-11",
      "_to": "1931-04-19"
    }
  },
  {
    "INR": {
      "_from": "1920-11-11",
      "_to": "1931-04-19"
    }
  },
  {
    "IQD": {
      "_from": "1931-04-19"
    }
  }
]
```

```
[
  "IR";
  [
    {
      "IRR": {
        "_from": "1932-05-13"
      }
    }
  ],
  "IS";
  [
    {
      "DKK": {
        "_from": "1873-05-27",
        "_to": "1918-12-01"
      }
    },
    {
      "ISJ": {
        "_from": "1918-12-01",
        "_to": "1981-01-01"
      }
    },
    {
      "ISK": {
        "_from": "1981-01-01"
      }
    }
  ],
  "IT";
  [
    {
      "ITL": {
        "_from": "1862-08-24",
        "_to": "2002-02-28"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "JE";
  [
    {
      "GBP": {
        "_from": "1837-01-01"
      }
    }
  ],
  "JM";
  [
    {
      "JMD": {
        "_from": "1969-09-08"
```



```

    }
  ],
  "JO";
  [
    {
      "JOD": {
        "_from": "1950-07-01"
      }
    }
  ],
  "JP";
  [
    {
      "JPY": {
        "_from": "1871-06-01"
      }
    }
  ],
  "KE";
  [
    {
      "KES": {
        "_from": "1966-09-14"
      }
    }
  ],
  "KG";
  [
    {
      "SUR": {
        "_from": "1961-01-01",
        "_to": "1991-12-25"
      }
    },
    {
      "RUR": {
        "_from": "1991-12-25",
        "_to": "1993-05-10"
      }
    },
    {
      "KGS": {
        "_from": "1993-05-10"
      }
    }
  ],
  "KH";
  [
    {
      "KHR": {
        "_from": "1980-03-20"
      }
    }
  ],
  "KI";
  [

```

```
    {
      "AUD": {
        "_from": "1966-02-14"
      }
    },
    "KM";
    [
      {
        "KMF": {
          "_from": "1975-07-06"
        }
      }
    ],
    "KN";
    [
      {
        "XCD": {
          "_from": "1965-10-06"
        }
      }
    ],
    "KP";
    [
      {
        "KPW": {
          "_from": "1959-04-17"
        }
      }
    ],
    "KR";
    [
      {
        "KRO": {
          "_from": "1945-08-15",
          "_to": "1953-02-15"
        }
      },
      {
        "KRH": {
          "_from": "1953-02-15",
          "_to": "1962-06-10"
        }
      },
      {
        "KRW": {
          "_from": "1962-06-10"
        }
      }
    ],
    "KW";
    [
      {
        "KWD": {
          "_from": "1961-04-01"
        }
      }
    ]
  ]
}
```

```
],
  "KY";
[
  {
    "JMD": {
      "_from": "1969-09-08",
      "_to": "1971-01-01"
    }
  },
  {
    "KYD": {
      "_from": "1971-01-01"
    }
  }
],
  "KZ";
[
  {
    "KZT": {
      "_from": "1993-11-05"
    }
  }
],
  "LA";
[
  {
    "LAK": {
      "_from": "1979-12-10"
    }
  }
],
  "LB";
[
  {
    "LBP": {
      "_from": "1948-02-02"
    }
  }
],
  "LC";
[
  {
    "XCD": {
      "_from": "1965-10-06"
    }
  }
],
  "LI";
[
  {
    "CHF": {
      "_from": "1921-02-01"
    }
  }
],
  "LK";
[
```

```

        {
            "LKR": {
                "_from": "1978-05-22"
            }
        },
        "LR";
        [
            {
                "LRD": {
                    "_from": "1944-01-01"
                }
            }
        ],
        "LS";
        [
            {
                "ZAR": {
                    "_from": "1961-02-14"
                }
            },
            {
                "LSL": {
                    "_from": "1980-01-22"
                }
            }
        ],
        "LT";
        [
            {
                "SUR": {
                    "_from": "1961-01-01",
                    "_to": "1992-10-01"
                }
            },
            {
                "LTT": {
                    "_from": "1992-10-01",
                    "_to": "1993-06-25"
                }
            },
            {
                "LTL": {
                    "_from": "1993-06-25",
                    "_to": "2014-12-31"
                }
            },
            {
                "EUR": {
                    "_from": "2015-01-01"
                }
            }
        ],
        "LU";
        [
            {
                "LUF": {

```

```

        "_from": "1944-09-04",
        "_to": "2002-02-28"
      }
    },
    {
      "LUC": {
        "_tender": "false",
        "_from": "1970-01-01",
        "_to": "1990-03-05"
      }
    },
    {
      "LUL": {
        "_tender": "false",
        "_from": "1970-01-01",
        "_to": "1990-03-05"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "LV";
  [
    {
      "SUR": {
        "_from": "1961-01-01",
        "_to": "1992-07-20"
      }
    },
    {
      "LVR": {
        "_from": "1992-05-07",
        "_to": "1993-10-17"
      }
    },
    {
      "LVL": {
        "_from": "1993-06-28",
        "_to": "2013-12-31"
      }
    },
    {
      "EUR": {
        "_from": "2014-01-01"
      }
    }
  ],
  "LY";
  [
    {
      "LYD": {
        "_from": "1971-09-01"
      }
    }
  ]

```

```
],
  "MA";
[
  {
    "MAF": {
      "_from": "1881-01-01",
      "_to": "1959-10-17"
    }
  },
  {
    "MAD": {
      "_from": "1959-10-17"
    }
  }
],
  "MC";
[
  {
    "FRF": {
      "_from": "1960-01-01",
      "_to": "2002-02-17"
    }
  },
  {
    "MCF": {
      "_from": "1960-01-01",
      "_to": "2002-02-17"
    }
  },
  {
    "EUR": {
      "_from": "1999-01-01"
    }
  }
],
  "MD";
[
  {
    "MDC": {
      "_from": "1992-06-01",
      "_to": "1993-11-29"
    }
  },
  {
    "MDL": {
      "_from": "1993-11-29"
    }
  }
],
  "ME";
[
  {
    "YUM": {
      "_from": "1994-01-24",
      "_to": "2002-05-15"
    }
  },
],
```

```

        {
            "DEM": {
                "_from": "1999-10-02",
                "_to": "2002-05-15"
            }
        },
        {
            "EUR": {
                "_from": "2002-01-01"
            }
        }
    ],
    "MF";
    [
        {
            "FRF": {
                "_from": "1960-01-01",
                "_to": "2002-02-17"
            }
        },
        {
            "EUR": {
                "_from": "1999-01-01"
            }
        }
    ],
    "MG";
    [
        {
            "MGF": {
                "_from": "1963-07-01",
                "_to": "2004-12-31"
            }
        },
        {
            "MGA": {
                "_from": "1983-11-01"
            }
        }
    ],
    "MH";
    [
        {
            "USD": {
                "_from": "1944-01-01"
            }
        }
    ],
    "MK";
    [
        {
            "MKN": {
                "_from": "1992-04-26",
                "_to": "1993-05-20"
            }
        },
    ],
    {

```

```
        "MKD": {
            "_from": "1993-05-20"
        }
    ],
    "ML";
    [
        {
            "XOF": {
                "_from": "1958-11-24",
                "_to": "1962-07-02"
            }
        },
        {
            "MLF": {
                "_from": "1962-07-02",
                "_to": "1984-08-31"
            }
        },
        {
            "XOF": {
                "_from": "1984-06-01"
            }
        }
    ],
    "MM";
    [
        {
            "BUK": {
                "_from": "1952-07-01",
                "_to": "1989-06-18"
            }
        },
        {
            "MMK": {
                "_from": "1989-06-18"
            }
        }
    ],
    "MN";
    [
        {
            "MNT": {
                "_from": "1915-03-01"
            }
        }
    ],
    "MO";
    [
        {
            "MOP": {
                "_from": "1901-01-01"
            }
        }
    ],
    "MP";
    [
```



```
{
  "USD": {
    "_from": "1944-01-01"
  }
},
"MQ";
[
  {
    "FRF": {
      "_from": "1960-01-01",
      "_to": "2002-02-17"
    }
  },
  {
    "EUR": {
      "_from": "1999-01-01"
    }
  }
],
"MR";
[
  {
    "XOF": {
      "_from": "1958-11-28",
      "_to": "1973-06-29"
    }
  },
  {
    "MRO": {
      "_from": "1973-06-29"
    }
  }
],
"MS";
[
  {
    "XCD": {
      "_from": "1967-02-27"
    }
  }
],
"MT";
[
  {
    "MTP": {
      "_from": "1914-08-13",
      "_to": "1968-06-07"
    }
  },
  {
    "MTL": {
      "_from": "1968-06-07",
      "_to": "2008-01-31"
    }
  }
],
{
```

```

        "EUR": {
            "_from": "2008-01-01"
        }
    ],
    "MU";
    [
        {
            "MUR": {
                "_from": "1934-04-01"
            }
        }
    ],
    "MV";
    [
        {
            "MVR": {
                "_from": "1981-07-01"
            }
        }
    ],
    "MW";
    [
        {
            "MWK": {
                "_from": "1971-02-15"
            }
        }
    ],
    "MX";
    [
        {
            "MXV": {
                "_tender": "false"
            }
        },
        {
            "MXP": {
                "_from": "1822-01-01",
                "_to": "1992-12-31"
            }
        },
        {
            "MXN": {
                "_from": "1993-01-01"
            }
        }
    ],
    "MY";
    [
        {
            "MYR": {
                "_from": "1963-09-16"
            }
        }
    ],
    "MZ";

```

```
[
  {
    "MZE": {
      "_from": "1975-06-25",
      "_to": "1980-06-16"
    }
  },
  {
    "MZM": {
      "_from": "1980-06-16",
      "_to": "2006-12-31"
    }
  },
  {
    "MZN": {
      "_from": "2006-07-01"
    }
  }
],
"NA";
[
  {
    "ZAR": {
      "_from": "1961-02-14"
    }
  },
  {
    "NAD": {
      "_from": "1993-01-01"
    }
  }
],
"NC";
[
  {
    "XPF": {
      "_from": "1985-01-01"
    }
  }
],
"NE";
[
  {
    "XOF": {
      "_from": "1958-12-19"
    }
  }
],
"NF";
[
  {
    "AUD": {
      "_from": "1966-02-14"
    }
  }
],
"NG";
```

```
[
  {
    "NGN": {
      "_from": "1973-01-01"
    }
  },
  "NI";
  [
    {
      "NIC": {
        "_from": "1988-02-15",
        "_to": "1991-04-30"
      }
    },
    {
      "NIO": {
        "_from": "1991-04-30"
      }
    }
  ],
  "NL";
  [
    {
      "NLG": {
        "_from": "1813-01-01",
        "_to": "2002-02-28"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "NO";
  [
    {
      "SEK": {
        "_from": "1873-05-27",
        "_to": "1905-06-07"
      }
    },
    {
      "NOK": {
        "_from": "1905-06-07"
      }
    }
  ],
  "NP";
  [
    {
      "INR": {
        "_from": "1870-01-01",
        "_to": "1966-10-17"
      }
    },
  ],
```

```

        {
            "NPR": {
                "_from": "1933-01-01"
            }
        },
        "NR";
    [
        {
            "AUD": {
                "_from": "1966-02-14"
            }
        },
        "NU";
    [
        {
            "NZD": {
                "_from": "1967-07-10"
            }
        },
        "NZ";
    [
        {
            "NZD": {
                "_from": "1967-07-10"
            }
        },
        "OM";
    [
        {
            "OMR": {
                "_from": "1972-11-11"
            }
        },
        "PA";
    [
        {
            "PAB": {
                "_from": "1903-11-04"
            }
        },
        {
            "USD": {
                "_from": "1903-11-18"
            }
        }
    ],
    "PE";
    [
        {
            "PES": {
                "_from": "1863-02-14",
                "_to": "1985-02-01"
            }
        }
    ]

```

```
    },
    {
      "PEI": {
        "_from": "1985-02-01",
        "_to": "1991-07-01"
      }
    },
    {
      "PEN": {
        "_from": "1991-07-01"
      }
    }
  ],
  "PF";
[
  {
    "XPF": {
      "_from": "1945-12-26"
    }
  }
],
  "PG";
[
  {
    "AUD": {
      "_from": "1966-02-14",
      "_to": "1975-09-16"
    }
  },
  {
    "PGK": {
      "_from": "1975-09-16"
    }
  }
],
  "PH";
[
  {
    "PHP": {
      "_from": "1946-07-04"
    }
  }
],
  "PK";
[
  {
    "INR": {
      "_from": "1835-08-17",
      "_to": "1947-08-15"
    }
  },
  {
    "PKR": {
      "_from": "1948-04-01"
    }
  }
]
```

```
],
  "PL";
[
  {
    "PLZ": {
      "_from": "1950-10-28",
      "_to": "1994-12-31"
    }
  },
  {
    "PLN": {
      "_from": "1995-01-01"
    }
  }
],
  "PM";
[
  {
    "FRF": {
      "_from": "1972-12-21",
      "_to": "2002-02-17"
    }
  },
  {
    "EUR": {
      "_from": "1999-01-01"
    }
  }
],
  "PN";
[
  {
    "NZD": {
      "_from": "1969-01-13"
    }
  }
],
  "PR";
[
  {
    "ESP": {
      "_from": "1800-01-01",
      "_to": "1898-12-10"
    }
  },
  {
    "USD": {
      "_from": "1898-12-10"
    }
  }
],
  "PS";
[
  {
    "JOD": {
      "_from": "1950-07-01",
      "_to": "1967-06-01"
    }
  }
]
```

```
    },
    {
      "ILP": {
        "_from": "1967-06-01",
        "_to": "1980-02-22"
      }
    },
    {
      "ILS": {
        "_from": "1985-09-04"
      }
    },
    {
      "JOD": {
        "_from": "1996-02-12"
      }
    }
  ],
  "PT";
  [
    {
      "PTE": {
        "_from": "1911-05-22",
        "_to": "2002-02-28"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "PW";
  [
    {
      "USD": {
        "_from": "1944-01-01"
      }
    }
  ],
  "PY";
  [
    {
      "PYG": {
        "_from": "1943-11-01"
      }
    }
  ],
  "QA";
  [
    {
      "QAR": {
        "_from": "1973-05-19"
      }
    }
  ],
  ],
```



```
        "RE";
    [
        {
            "FRF": {
                "_from": "1975-01-01",
                "_to": "2002-02-17"
            }
        },
        {
            "EUR": {
                "_from": "1999-01-01"
            }
        }
    ],
    "RO";
    [
        {
            "ROL": {
                "_from": "1952-01-28",
                "_to": "2006-12-31"
            }
        },
        {
            "RON": {
                "_from": "2005-07-01"
            }
        }
    ],
    "RS";
    [
        {
            "YUM": {
                "_from": "1994-01-24",
                "_to": "2002-05-15"
            }
        },
        {
            "CSD": {
                "_from": "2002-05-15",
                "_to": "2006-10-25"
            }
        },
        {
            "RSD": {
                "_from": "2006-10-25"
            }
        }
    ],
    "RU";
    [
        {
            "RUR": {
                "_from": "1991-12-25",
                "_to": "1998-12-31"
            }
        },
        {
```

```
        "RUB": {
          "_from": "1999-01-01"
        }
      ],
      "RW";
      [
        {
          "RWF": {
            "_from": "1964-05-19"
          }
        }
      ],
      "SA";
      [
        {
          "SAR": {
            "_from": "1952-10-22"
          }
        }
      ],
      "SB";
      [
        {
          "AUD": {
            "_from": "1966-02-14",
            "_to": "1978-06-30"
          }
        },
        {
          "SBD": {
            "_from": "1977-10-24"
          }
        }
      ],
      "SC";
      [
        {
          "SCR": {
            "_from": "1903-11-01"
          }
        }
      ],
      "SD";
      [
        {
          "EGP": {
            "_from": "1889-01-19",
            "_to": "1958-01-01"
          }
        },
        {
          "GBP": {
            "_from": "1889-01-19",
            "_to": "1958-01-01"
          }
        }
      ],
    ],
  ],
}
```

```

        {
            "SDP": {
                "_from": "1957-04-08",
                "_to": "1998-06-01"
            }
        },
        {
            "SDD": {
                "_from": "1992-06-08",
                "_to": "2007-06-30"
            }
        },
        {
            "SDG": {
                "_from": "2007-01-10"
            }
        }
    ],
    "SE";
    [
        {
            "SEK": {
                "_from": "1873-05-27"
            }
        }
    ],
    "SG";
    [
        {
            "MYR": {
                "_from": "1963-09-16",
                "_to": "1967-06-12"
            }
        },
        {
            "SGD": {
                "_from": "1967-06-12"
            }
        }
    ],
    "SH";
    [
        {
            "SHP": {
                "_from": "1917-02-15"
            }
        }
    ],
    "SI";
    [
        {
            "SIT": {
                "_from": "1992-10-07",
                "_to": "2007-01-14"
            }
        },
    ],
    {

```

```

        "EUR": {
            "_from": "2007-01-01"
        }
    ],
    "SJ";
    [
        {
            "NOK": {
                "_from": "1905-06-07"
            }
        }
    ],
    "SK";
    [
        {
            "CSK": {
                "_from": "1953-06-01",
                "_to": "1992-12-31"
            }
        },
        {
            "SKK": {
                "_from": "1992-12-31",
                "_to": "2009-01-01"
            }
        },
        {
            "EUR": {
                "_from": "2009-01-01"
            }
        }
    ],
    "SL";
    [
        {
            "GBP": {
                "_from": "1808-11-30",
                "_to": "1966-02-04"
            }
        },
        {
            "SLL": {
                "_from": "1964-08-04"
            }
        }
    ],
    "SM";
    [
        {
            "ITL": {
                "_from": "1865-12-23",
                "_to": "2001-02-28"
            }
        },
        {
            "EUR": {

```

```

        "_from": "1999-01-01"
      }
    }
  ],
  "SN";
  [
    {
      "XOF": {
        "_from": "1959-04-04"
      }
    }
  ],
  "SO";
  [
    {
      "SOS": {
        "_from": "1960-07-01"
      }
    }
  ],
  "SR";
  [
    {
      "NLG": {
        "_from": "1815-11-20",
        "_to": "1940-05-10"
      }
    },
    {
      "SRG": {
        "_from": "1940-05-10",
        "_to": "2003-12-31"
      }
    },
    {
      "SRD": {
        "_from": "2004-01-01"
      }
    }
  ],
  "SS";
  [
    {
      "SDG": {
        "_from": "2007-01-10",
        "_to": "2011-09-01"
      }
    },
    {
      "SSP": {
        "_from": "2011-07-18"
      }
    }
  ],
  "ST";
  [
    {

```

```
        "STD": {
          "_from": "1977-09-08"
        }
      ],
      "SU";
      [
        {
          "SUR": {
            "_from": "1961-01-01",
            "_to": "1991-12-25"
          }
        }
      ],
      "SV";
      [
        {
          "SVC": {
            "_from": "1919-11-11",
            "_to": "2001-01-01"
          }
        },
        {
          "USD": {
            "_from": "2001-01-01"
          }
        }
      ],
      "SX";
      [
        {
          "ANG": {
            "_from": "2010-10-10"
          }
        }
      ],
      "SY";
      [
        {
          "SYP": {
            "_from": "1948-01-01"
          }
        }
      ],
      "SZ";
      [
        {
          "SZL": {
            "_from": "1974-09-06"
          }
        }
      ],
      "TA";
      [
        {
          "GBP": {
            "_from": "1938-01-12"
          }
        }
      ]
    ]
  }
}
```

```
    }  
  },  
  ],  
  "TC";  
  [  
    {  
      "USD": {  
        "_from": "1969-09-08"  
      }  
    }  
  ],  
  "TD";  
  [  
    {  
      "XAF": {  
        "_from": "1993-01-01"  
      }  
    }  
  ],  
  "TF";  
  [  
    {  
      "FRF": {  
        "_from": "1959-01-01",  
        "_to": "2002-02-17"  
      }  
    },  
    {  
      "EUR": {  
        "_from": "1999-01-01"  
      }  
    }  
  ],  
  "TG";  
  [  
    {  
      "XOF": {  
        "_from": "1958-11-28"  
      }  
    }  
  ],  
  "TH";  
  [  
    {  
      "THB": {  
        "_from": "1928-04-15"  
      }  
    }  
  ],  
  "TJ";  
  [  
    {  
      "RUR": {  
        "_from": "1991-12-25",  
        "_to": "1995-05-10"  
      }  
    },  
  ],  
],  
}
```

```
{
  "TJR": {
    "_from": "1995-05-10",
    "_to": "2000-10-25"
  },
  {
    "TJS": {
      "_from": "2000-10-26"
    }
  },
  "TK";
[
  {
    "NZD": {
      "_from": "1967-07-10"
    }
  },
  "TL";
[
  {
    "TPE": {
      "_from": "1959-01-02",
      "_to": "2002-05-20"
    }
  },
  {
    "IDR": {
      "_from": "1975-12-07",
      "_to": "2002-05-20"
    }
  },
  {
    "USD": {
      "_from": "1999-10-20"
    }
  },
  "TM";
[
  {
    "SUR": {
      "_from": "1961-01-01",
      "_to": "1991-12-25"
    }
  },
  {
    "RUR": {
      "_from": "1991-12-25",
      "_to": "1993-11-01"
    }
  },
  {
    "TMM": {
      "_from": "1993-11-01",
```



```
        "_to": "2009-01-01"
      }
    },
    {
      "TMT": {
        "_from": "2009-01-01"
      }
    }
  ],
  "TN";
[
  {
    "TND": {
      "_from": "1958-11-01"
    }
  }
],
  "TO";
[
  {
    "TOP": {
      "_from": "1966-02-14"
    }
  }
],
  "TP";
[
  {
    "TPE": {
      "_from": "1959-01-02",
      "_to": "2002-05-20"
    }
  },
  {
    "IDR": {
      "_from": "1975-12-07",
      "_to": "2002-05-20"
    }
  }
],
  "TR";
[
  {
    "TRL": {
      "_from": "1922-11-01",
      "_to": "2005-12-31"
    }
  },
  {
    "TRY": {
      "_from": "2005-01-01"
    }
  }
],
  "TT";
[
  {
```

```

        "TTD": {
            "_from": "1964-01-01"
        }
    ],
    "TV";
    [
        {
            "AUD": {
                "_from": "1966-02-14"
            }
        }
    ],
    "TW";
    [
        {
            "TWD": {
                "_from": "1949-06-15"
            }
        }
    ],
    "TZ";
    [
        {
            "TZS": {
                "_from": "1966-06-14"
            }
        }
    ],
    "UA";
    [
        {
            "SUR": {
                "_from": "1961-01-01",
                "_to": "1991-12-25"
            }
        },
        {
            "RUR": {
                "_from": "1991-12-25",
                "_to": "1992-11-13"
            }
        },
        {
            "UAK": {
                "_from": "1992-11-13",
                "_to": "1993-10-17"
            }
        },
        {
            "UAH": {
                "_from": "1996-09-02"
            }
        }
    ],
    "UG";
    [

```

```
{
  "UGS": {
    "_from": "1966-08-15",
    "_to": "1987-05-15"
  },
  {
    "UGX": {
      "_from": "1987-05-15"
    }
  },
  "UM";
[
  {
    "USD": {
      "_from": "1944-01-01"
    }
  },
  "US";
[
  {
    "USN": {
      "_tender": "false"
    }
  },
  {
    "USS": {
      "_tender": "false",
      "_to": "2014-03-01"
    }
  },
  {
    "USD": {
      "_from": "1792-01-01"
    }
  },
  "UY";
[
  {
    "UYI": {
      "_tender": "false"
    }
  },
  {
    "UYP": {
      "_from": "1975-07-01",
      "_to": "1993-03-01"
    }
  },
  {
    "UYU": {
      "_from": "1993-03-01"
    }
  }
]
```

```
],
  "UZ";
[
  {
    "UZS": {
      "_from": "1994-07-01"
    }
  }
],
  "VA";
[
  {
    "ITL": {
      "_from": "1870-10-19",
      "_to": "2002-02-28"
    }
  },
  {
    "EUR": {
      "_from": "1999-01-01"
    }
  }
],
  "VC";
[
  {
    "XCD": {
      "_from": "1965-10-06"
    }
  }
],
  "VE";
[
  {
    "VEB": {
      "_from": "1871-05-11",
      "_to": "2008-06-30"
    }
  },
  {
    "VEF": {
      "_from": "2008-01-01"
    }
  }
],
  "VG";
[
  {
    "USD": {
      "_from": "1833-01-01"
    }
  },
  {
    "GBP": {
      "_from": "1833-01-01",
      "_to": "1959-01-01"
    }
  }
]
```

```
    },
    [
      {
        "USD": {
          "_from": "1837-01-01"
        }
      }
    ],
    "VN";
    [
      {
        "VNN": {
          "_from": "1978-05-03",
          "_to": "1985-09-14"
        }
      },
      {
        "VND": {
          "_from": "1985-09-14"
        }
      }
    ],
    "VU";
    [
      {
        "VUV": {
          "_from": "1981-01-01"
        }
      }
    ],
    "WF";
    [
      {
        "XPF": {
          "_from": "1961-07-30"
        }
      }
    ],
    "WS";
    [
      {
        "WST": {
          "_from": "1967-07-10"
        }
      }
    ],
    "XK";
    [
      {
        "YUM": {
          "_from": "1994-01-24",
          "_to": "1999-09-30"
        }
      },
      {
```

```

        "DEM": {
            "_from": "1999-09-01",
            "_to": "2002-03-09"
        }
    },
    {
        "EUR": {
            "_from": "2002-01-01"
        }
    }
],
"YD";
[
    {
        "YDD": {
            "_from": "1965-04-01",
            "_to": "1996-01-01"
        }
    }
],
"YE";
[
    {
        "YER": {
            "_from": "1990-05-22"
        }
    }
],
"YT";
[
    {
        "KMF": {
            "_from": "1975-01-01",
            "_to": "1976-02-23"
        }
    },
    {
        "FRF": {
            "_from": "1976-02-23",
            "_to": "2002-02-17"
        }
    },
    {
        "EUR": {
            "_from": "1999-01-01"
        }
    }
],
"YU";
[
    {
        "YUD": {
            "_from": "1966-01-01",
            "_to": "1990-01-01"
        }
    },
    {

```

```

        "YUN": {
            "_from": "1990-01-01",
            "_to": "1992-07-24"
        }
    },
    {
        "YUM": {
            "_from": "1994-01-24",
            "_to": "2002-05-15"
        }
    }
],
"ZA";
[
    {
        "ZAR": {
            "_from": "1961-02-14"
        }
    },
    {
        "ZAL": {
            "_tender": "false",
            "_from": "1985-09-01",
            "_to": "1995-03-13"
        }
    }
],
"ZM";
[
    {
        "ZMK": {
            "_from": "1968-01-16",
            "_to": "2013-01-01"
        }
    },
    {
        "ZMW": {
            "_from": "2013-01-01"
        }
    }
],
"ZR";
[
    {
        "ZRZ": {
            "_from": "1971-10-27",
            "_to": "1993-11-01"
        }
    },
    {
        "ZRN": {
            "_from": "1993-11-01",
            "_to": "1998-07-31"
        }
    }
],
"ZW";

```

```
[
  {
    "RHD": {
      "_from": "1970-02-17",
      "_to": "1980-04-18"
    }
  },
  {
    "ZWD": {
      "_from": "1980-04-18",
      "_to": "2008-08-01"
    }
  },
  {
    "ZWR": {
      "_from": "2008-08-01",
      "_to": "2009-02-02"
    }
  },
  {
    "ZWL": {
      "_from": "2009-02-02",
      "_to": "2009-04-12"
    }
  },
  {
    "USD": {
      "_from": "2009-04-12"
    }
  }
],
  "ZZ";
[
  {
    "XAG": {
      "_tender": "false"
    }
  },
  {
    "XAU": {
      "_tender": "false"
    }
  },
  {
    "XBA": {
      "_tender": "false"
    }
  },
  {
    "XBB": {
      "_tender": "false"
    }
  },
  {
    "XBC": {
      "_tender": "false"
    }
  }
]
```



```
    },  
    {  
      "XBD": {  
        "_tender": "false"  
      }  
    },  
    {  
      "XDR": {  
        "_tender": "false"  
      }  
    },  
    {  
      "XPD": {  
        "_tender": "false"  
      }  
    },  
    {  
      "XPT": {  
        "_tender": "false"  
      }  
    },  
    {  
      "XSU": {  
        "_tender": "false"  
      }  
    },  
    {  
      "XTS": {  
        "_tender": "false"  
      }  
    },  
    {  
      "XUA": {  
        "_tender": "false"  
      }  
    },  
    {  
      "XXX": {  
        "_tender": "false"  
      }  
    },  
    {  
      "XRE": {  
        "_tender": "false",  
        "_to": "1999-11-30"  
      }  
    },  
    {  
      "XFU": {  
        "_tender": "false",  
        "_to": "2013-11-30"  
      }  
    },  
    {  
      "XFO": {  
        "_tender": "false",  
        "_from": "1930-01-01",
```

```
}  
}  
}  
    }  
];  
}  
}_to": "2003-04-01"  
}
```

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as currencyData from './currencyData.json';
import * as numbers from './numbers.json';
import * as currencies from './currencies.json';
loadCldr(numberingSystems, currencyData, numbers, currencies);
L10n.load({
    'de': {
        'numerictextbox': { incrementTitle: 'Wert erhöhen', decrementTitle:
'Dekrementwert' }
    }
});
// initializes NumericTextBox component
// sets `German` culture using the culture value 'de'
// sets the 'EUR' currency format
ReactDOM.render(<NumericTextBoxComponent locale='de' currency='EUR'
format='c2' value={100}>
</NumericTextBoxComponent>, document.getElementById('numericContainer'));
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as currencyData from './currencyData.json';
import * as numbers from './numbers.json';
import * as currencies from './currencies.json';
loadCldr(numberingSystems, currencyData, numbers, currencies);
L10n.load({
    'de': {
        'numerictextbox': { incrementTitle: 'Wert erhöhen', decrementTitle:
'Dekrementwert' }
    }
});
// initializes NumericTextBox component
// sets `German` culture using the culture value 'de'
// sets the 'EUR' currency format
```

```
ReactDOM.render(<NumericTextBoxComponent locale='de' currency='EUR'
format='c2' value={100} >
</NumericTextBoxComponent>,document.getElementById('numericContainer'));
```

NUMBERINGSYSTEMS.JSX

```
{
  "supplemental";
  {
    "version";
    {
      "_number";
      "$Revision: 12732 $",
      "_unicodeVersion";
      "9.0.0",
      "_cldrVersion";
      "31";
    }
    "numberingSystems";
    {
      "adlm";
      {
        "_digits";
        "ᲠᲡᲢᲣᲤᲥᲦᲧᲨᲩ",
        "_type";
        "numeric";
      }
      "ahom";
      {
        "_digits";
        "᩠ᩡᩢᩣᩤᩥᩦᩧᩨᩩ",
        "_type";
        "numeric";
      }
      "arab";
      {
        "_digits";
        "٠١٢٣٤٥٦٧٨٩",
        "_type";
        "numeric";
      }
      "arabext";
      {
        "_digits";
        "٠١٢٣٤٥٦٧٨٩",
        "_type";
        "numeric";
      }
      "armn";
      {
        "_rules";
        "armenian-upper",
        "_type";
        "algorithmic";
      }
      "armnlow";
```

```

    {
        "_rules";
        "armenian-lower",
        "_type";
        "algorithmic";
    }
    "bali";
    {
        "_digits";
        "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",
        "_type";
        "numeric";
    }
    "beng";
    {
        "_digits";
        "০১২৩৪৫৬৭৮৯",
        "_type";
        "numeric";
    }
    "bhks";
    {
        "_digits";
        "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",
        "_type";
        "numeric";
    }
    "brah";
    {
        "_digits";
        "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",
        "_type";
        "numeric";
    }
    "cakm";
    {
        "_digits";
        "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",
        "_type";
        "numeric";
    }
    "cham";
    {
        "_digits";
        "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",
        "_type";
        "numeric";
    }
    "cyr1";
    {
        "_rules";
        "cyrillic-lower",
        "_type";
        "algorithmic";
    }
    "deva";

```

```
{
  "_digits";
  "ፀፁፃፄፅፆፇፈ",
  "_type";
  "numeric";
}
"ethi";
{
  "_rules";
  "ethiopic",
  "_type";
  "algorithmic";
}
"fullwide";
{
  "_digits";
  "0 1 2 3 4 5 6 7 8 9",
  "_type";
  "numeric";
}
"geor";
{
  "_rules";
  "georgian",
  "_type";
  "algorithmic";
}
"grek";
{
  "_rules";
  "greek-upper",
  "_type";
  "algorithmic";
}
"greklow";
{
  "_rules";
  "greek-lower",
  "_type";
  "algorithmic";
}
"gujr";
{
  "_digits";
  "૦૧૨૩૪૫૬૭૮૯",
  "_type";
  "numeric";
}
"guru";
{
  "_digits";
  "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉᱊᱋᱌ᱍᱎᱏ᱐᱑᱒᱓᱔᱕᱖᱗᱘᱙ᱚᱛᱜᱝᱞᱟᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱮᱹᱺᱻᱼᱽ᱾᱿",
  "_type";
  "numeric";
}
"hanidays";
```

```

    {
        "_rules";
        "zh/SpelloutRules/spellout-numbering-days",
        "_type";
        "algorithmic";
    }
    "hanidec";
    {
        "_digits";
        "〇一三四五六七八九",
        "_type";
        "numeric";
    }
    "hans";
    {
        "_rules";
        "zh/SpelloutRules/spellout-cardinal",
        "_type";
        "algorithmic";
    }
    "hansfin";
    {
        "_rules";
        "zh/SpelloutRules/spellout-cardinal-financial",
        "_type";
        "algorithmic";
    }
    "hant";
    {
        "_rules";
        "zh_Hant/SpelloutRules/spellout-cardinal",
        "_type";
        "algorithmic";
    }
    "hantfin";
    {
        "_rules";
        "zh_Hant/SpelloutRules/spellout-cardinal-financial",
        "_type";
        "algorithmic";
    }
    "hebr";
    {
        "_rules";
        "hebrew",
        "_type";
        "algorithmic";
    }
    "hmng";
    {
        "_digits";
        "□□□□□□□□□□",
        "_type";
        "numeric";
    }
    "java";

```

```

    {
      "_digits";
      "၀၁၂၃၄၅၆၇၈၉၁၀၁၁",
      "_type";
      "numeric";
    }
    "jpan";
    {
      "_rules";
      "ja/SpelloutRules/spellout-cardinal",
      "_type";
      "algorithmic";
    }
    "jpanfin";
    {
      "_rules";
      "ja/SpelloutRules/spellout-cardinal-financial",
      "_type";
      "algorithmic";
    }
    "kali";
    {
      "_digits";
      "၀၀၀၀၀၀၀၀၀၀",
      "_type";
      "numeric";
    }
    "khmr";
    {
      "_digits";
      "០១២៣៤៥៦៧៨៩",
      "_type";
      "numeric";
    }
    "knda";
    {
      "_digits";
      "೦೧೨೩೪೫೬೭೮೯",
      "_type";
      "numeric";
    }
    "lana";
    {
      "_digits";
      "၀၀၀၀၀၀၀၀၀၀",
      "_type";
      "numeric";
    }
    "lanatham";
    {
      "_digits";
      "၀၀၀၀၀၀၀၀၀၀",
      "_type";
      "numeric";
    }
  }

```

```
"lao";
{
  "_digits";
  "໐໑໒໓໔໕໖໗໘໑",
  "_type";
  "numeric";
}
"latn";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
"lepc";
{
  "_digits";
  "໐໑໒໓໔໕໖໗໘໑",
  "_type";
  "numeric";
}
"limb";
{
  "_digits";
  "໐໑໒໓໔໕໖໗໘໑",
  "_type";
  "numeric";
}
"mathbold";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
"mathdbl";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
"mathmono";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
"mathsanb";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
```



```
"mathsans";
{
    "_digits";
    "0123456789",
        "_type";
    "numeric";
}
"mlym";
{
    "_digits";
    "ഐനമർത്ത്യവൃന്ത",
        "_type";
    "numeric";
}
"modi";
{
    "_digits";
    "□□□□□□□□□□",
        "_type";
    "numeric";
}
"mong";
{
    "_digits";
    "᠎ᠠᠨᠮᠤᠩᠭᠡ",
        "_type";
    "numeric";
}
"mroo";
{
    "_digits";
    "□□□□□□□□□□",
        "_type";
    "numeric";
}
"mtei";
{
    "_digits";
    "၆၆၆၆၆၆၆၆၆၆",
        "_type";
    "numeric";
}
"mymr";
{
    "_digits";
    "၀၁၂၃၄၅၆၇၈",
        "_type";
    "numeric";
}
"mymrshan";
{
    "_digits";
    "0123456789",
        "_type";
    "numeric";
}
```

```

}
"mymrtlng";
{
    "_digits";
    "0ᄇᄃ3ᄃ6ᄃᄃᄃᄃ",
    "_type";
    "numeric";
}
"newa";
{
    "_digits";
    "□□□□□□□□□□",
    "_type";
    "numeric";
}
"nkoo";
{
    "_digits";
    "ᄃᄃᄃᄃᄃᄃᄃᄃᄃᄃ",
    "_type";
    "numeric";
}
"olck";
{
    "_digits";
    "0ᄃᄃᄃᄃᄃᄃᄃᄃᄃ",
    "_type";
    "numeric";
}
"orya";
{
    "_digits";
    "0ᄃᄃᄃᄃᄃᄃᄃᄃᄃ",
    "_type";
    "numeric";
}
"osma";
{
    "_digits";
    "0ᄃᄃᄃᄃᄃᄃᄃᄃᄃ",
    "_type";
    "numeric";
}
"roman";
{
    "_rules";
    "roman-upper",
    "_type";
    "algorithmic";
}
"romanlow";
{
    "_rules";
    "roman-lower",
    "type";
}

```

```

        "algorithmic";
    }
    "saur";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "shrd";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "sind";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "sinh";
    {
        "_digits";
        "⁂௮௮௮௮௮௮௮௮",
        "_type";
        "numeric";
    }
    "sora";
    {
        "_digits";
        "0௭9௮௮௮௮௮",
        "_type";
        "numeric";
    }
    "sund";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "takr";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "talv";
    {
        "_digits";
        "௮9௮9௮9௮9",
        "_type";
    }

```

```

"numeric";
}
"taml";
{
    "_rules";
    "tamil",
        "_type";
    "algorithmic";
}
"tamldec";
{
    "_digits";
    "௦௧௨௩௪௫௬௭௮௯",
        "_type";
    "numeric";
}
"telu";
{
    "_digits";
    "౦౧౨౩౪౫౬౭౮౯",
        "_type";
    "numeric";
}
"thai";
{
    "_digits";
    "๐๑๒๓๔๕๖๗๘๙",
        "_type";
    "numeric";
}
"tibtb";
{
    "_digits";
    "༠༡༢༣༤༥༦༧༨༩",
        "_type";
    "numeric";
}
"tirh";
{
    "_digits";
    "□□□□□□□□□□",
        "_type";
    "numeric";
}
"vair";
{
    "_digits";
    "୦୧୨୩୪୫୬୭୮୯",
        "_type";
    "numeric";
}
"warab";
{
    "_digits";
    "□□□□□□□□□□",
        "_type";

```

```

        "numeric";
    }
}
}
}

```

NUMBERS.JSX

```

{
  "main";
  {
    "de";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 13259 $",
          "_cldrVersion";
          "31";
        }
        "language";
        "de";
      }
      "numbers";
      {
        "defaultNumberingSystem";
        "latn",
        "otherNumberingSystems";
        {
          "native";
          "latn";
        }
        "minimumGroupingDigits";
        "1",
        "symbols-numberSystem-latn";
        {
          "decimal";
          ",",
          "group";
          ".",
          "list";
          ";",
          "percentSign";
          "%",
          "plusSign";
          "+",
          "minusSign";
          "-",
          "exponential";
          "E",
          "superscriptingExponent";
          ". ",
          "perMille";
          "‰",

```

```

        "infinity";
        "∞",
        "nan";
        "NaN",
        "timeSeparator";
        ":";
    }
    "decimalFormats-numberSystem-latn";
    {
        "standard";
        "#,##0.###",
        "long";
        {
            "decimalFormat";
            {
                "1000-count-one";
                "0 Tausend",
                "1000-count-other";
                "0 Tausend",
                "10000-count-one";
                "00 Tausend",
                "10000-count-other";
                "00 Tausend",
                "100000-count-one";
                "000 Tausend",
                "100000-count-other";
                "000 Tausend",
                "1000000-count-one";
                "0 Million",
                "1000000-count-other";
                "0 Millionen",
                "10000000-count-one";
                "00 Millionen",
                "10000000-count-other";
                "00 Millionen",
                "100000000-count-one";
                "000 Millionen",
                "100000000-count-other";
                "000 Millionen",
                "1000000000-count-one";
                "0 Milliarde",
                "1000000000-count-other";
                "0 Milliarden",
                "10000000000-count-one";
                "00 Milliarden",
                "10000000000-count-other";
                "00 Milliarden",
                "100000000000-count-one";
                "000 Milliarden",
                "100000000000-count-other";
                "000 Milliarden",
                "1000000000000-count-one";
                "0 Billion",
                "1000000000000-count-other";
                "0 Billionen",
                "10000000000000-count-one";
                "00 Billionen",
            }
        }
    }

```

```

        "10000000000000-count-other";
        "00 Billionen",
        "10000000000000-count-one";
        "000 Billionen",
        "10000000000000-count-other";
        "000 Billionen";
    }
}
"short";
{
    "decimalFormat";
    {
        "1000-count-one";
        "0",
        "1000-count-other";
        "0",
        "10000-count-one";
        "0",
        "10000-count-other";
        "0",
        "100000-count-one";
        "0",
        "100000-count-other";
        "0",
        "1000000-count-one";
        "0 Mio'.'",
        "1000000-count-other";
        "0 Mio'.'",
        "10000000-count-one";
        "00 Mio'.'",
        "10000000-count-other";
        "00 Mio'.'",
        "100000000-count-one";
        "000 Mio'.'",
        "100000000-count-other";
        "000 Mio'.'",
        "1000000000-count-one";
        "0 Mrd'.'",
        "1000000000-count-other";
        "0 Mrd'.'",
        "10000000000-count-one";
        "00 Mrd'.'",
        "10000000000-count-other";
        "00 Mrd'.'",
        "100000000000-count-one";
        "000 Mrd'.'",
        "100000000000-count-other";
        "000 Mrd'.'",
        "1000000000000-count-one";
        "0 Bio'.'",
        "1000000000000-count-other";
        "0 Bio'.'",
        "10000000000000-count-one";
        "00 Bio'.'",
        "10000000000000-count-other";
        "00 Bio'.'",
        "100000000000000-count-one";
    }
}

```

```

        "000 Bio'.'",
        "100000000000000-count-other";
        "000 Bio'.'";
    }
}
}
"scientificFormats-numberSystem-latn";
{
    "standard";
    "#E0";
}
"percentFormats-numberSystem-latn";
{
    "standard";
    "#,##0 %";
}
"currencyFormats-numberSystem-latn";
{
    "currencySpacing";
    {
        "beforeCurrency";
        {
            "currencyMatch";
            "[:^S:]",
            "surroundingMatch";
            "[:digit:]",
            "insertBetween";
            " ";
        }
        "afterCurrency";
        {
            "currencyMatch";
            "[:^S:]",
            "surroundingMatch";
            "[:digit:]",
            "insertBetween";
            " ";
        }
    }
}
"standard";
"#,##0.00 ¤",
    "accounting";
"#,##0.00 ¤",
    "short";
{
    "standard";
    {
        "1000-count-one";
        "0 Tsd'.' ¤",
        "1000-count-other";
        "0 Tsd'.' ¤",
        "10000-count-one";
        "00 Tsd'.' ¤",
        "10000-count-other";
        "00 Tsd'.' ¤",
        "100000-count-one";
        "000 Tsd'.' ¤",
    }
}

```



```

        "100000-count-other";
        "000 Tsd'.' ¤",
        "1000000-count-one";
        "0 Mio'.' ¤",
        "1000000-count-other";
        "0 Mio'.' ¤",
        "10000000-count-one";
        "00 Mio'.' ¤",
        "10000000-count-other";
        "00 Mio'.' ¤",
        "100000000-count-one";
        "000 Mio'.' ¤",
        "100000000-count-other";
        "000 Mio'.' ¤",
        "1000000000-count-one";
        "0 Mrd'.' ¤",
        "1000000000-count-other";
        "0 Mrd'.' ¤",
        "10000000000-count-one";
        "00 Mrd'.' ¤",
        "10000000000-count-other";
        "00 Mrd'.' ¤",
        "100000000000-count-one";
        "000 Mrd'.' ¤",
        "100000000000-count-other";
        "000 Mrd'.' ¤",
        "1000000000000-count-one";
        "0 Bio'.' ¤",
        "1000000000000-count-other";
        "0 Bio'.' ¤",
        "10000000000000-count-one";
        "00 Bio'.' ¤",
        "10000000000000-count-other";
        "00 Bio'.' ¤",
        "100000000000000-count-one";
        "000 Bio'.' ¤",
        "100000000000000-count-other";
        "000 Bio'.' ¤";
    }
}
"unitPattern-count-one";
"{0} {1}",
    "unitPattern-count-other";
"{0} {1}";
}
"miscPatterns-numberSystem-latn";
{
    "atLeast";
    "{0}+",
        "range";
    "{0}-{1}";
}
"minimalPairs";
{
    "pluralMinimalPairs";
    "{0} Tag",
        "pluralMinimalPairs";
}

```

```

    "{0} Tage",
    "other";
    "{0}. Abzweigung nach rechts nehmen";
  }
}
}
}
}

```

SETTINGS.JSX

```
{
  "file";
  "app/index.tsx",
    "line";
  -1;
}
```

[Functional-component]

CURRENCIES.JSX

```
{
  "main";
  {
    "de";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 13259 $",
          "_cldrVersion";
          "31";
        }
        "language";
        "de";
      }
      "numbers";
      {
        "currencies";
        {
          "ADP";
          {
            "displayName";
            "Andorranische Pesete",
            "displayName-count-one";
            "Andorranische Pesete",
            "displayName-count-other";
            "Andorranische Peseten",
            "symbol";
            "ADP";
          }
          "AED";
          {
```

```

        "displayName";
        "VAE-Dirham",
        "displayName-count-one";
        "VAE-Dirham",
        "displayName-count-other";
        "VAE-Dirham",
        "symbol";
        "AED";
    }
    "AFA";
    {
        "displayName";
        "Afghanische Afghani (1927-2002)",
        "displayName-count-one";
        "Afghanische Afghani (1927-2002)",
        "displayName-count-other";
        "Afghanische Afghani (1927-2002)",
        "symbol";
        "AFA";
    }
    "AFN";
    {
        "displayName";
        "Afghanischer Afghani",
        "displayName-count-one";
        "Afghanischer Afghani",
        "displayName-count-other";
        "Afghanische Afghani",
        "symbol";
        "AFN";
    }
    "ALK";
    {
        "displayName";
        "Albanischer Lek (1946-1965)",
        "displayName-count-one";
        "Albanischer Lek (1946-1965)",
        "displayName-count-other";
        "Albanische Lek (1946-1965)";
    }
    "ALL";
    {
        "displayName";
        "Albanischer Lek",
        "displayName-count-one";
        "Albanischer Lek",
        "displayName-count-other";
        "Albanische Lek",
        "symbol";
        "ALL";
    }
    "AMD";
    {
        "displayName";
        "Armenischer Dram",
        "displayName-count-one";
        "Armenischer Dram",

```

```

        "displayName-count-other";
        "Armenische Dram",
        "symbol";
        "AMD";
    }
    "ANG";
    {
        "displayName";
        "Niederländische-Antillen-Gulden",
        "displayName-count-one";
        "Niederländische-Antillen-Gulden",
        "displayName-count-other";
        "Niederländische-Antillen-Gulden",
        "symbol";
        "ANG";
    }
    "AOA";
    {
        "displayName";
        "Angolanischer Kwanza",
        "displayName-count-one";
        "Angolanischer Kwanza",
        "displayName-count-other";
        "Angolanische Kwanza",
        "symbol";
        "AOA",
        "symbol-alt-narrow";
        "Kz";
    }
    "AOK";
    {
        "displayName";
        "Angolanischer Kwanza (1977-1990)",
        "displayName-count-one";
        "Angolanischer Kwanza (1977-1990)",
        "displayName-count-other";
        "Angolanische Kwanza (1977-1990)",
        "symbol";
        "AOK";
    }
    "AON";
    {
        "displayName";
        "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-one";
        "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-other";
        "Angolanische Neue Kwanza (1990-2000)",
        "symbol";
        "AON";
    }
    "AOR";
    {
        "displayName";
        "Angolanischer Kwanza Reajustado (1995-1999)",
        "displayName-count-one";
        "Angolanischer Kwanza Reajustado (1995-1999)",

```

```

        "displayName-count-other";
        "Angolanische Kwanza Reajustado (1995-1999)",
        "symbol";
        "AOR";
    }
    "ARA";
    {
        "displayName";
        "Argentinischer Austral",
        "displayName-count-one";
        "Argentinischer Austral",
        "displayName-count-other";
        "Argentinische Austral",
        "symbol";
        "ARA";
    }
    "ARL";
    {
        "displayName";
        "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-one";
        "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-other";
        "Argentinische Pesos Ley (1970-1983)",
        "symbol";
        "ARL";
    }
    "ARM";
    {
        "displayName";
        "Argentinischer Peso (1881-1970)",
        "displayName-count-one";
        "Argentinischer Peso (1881-1970)",
        "displayName-count-other";
        "Argentinische Pesos (1881-1970)",
        "symbol";
        "ARM";
    }
    "ARP";
    {
        "displayName";
        "Argentinischer Peso (1983-1985)",
        "displayName-count-one";
        "Argentinischer Peso (1983-1985)",
        "displayName-count-other";
        "Argentinische Peso (1983-1985)",
        "symbol";
        "ARP";
    }
    "ARS";
    {
        "displayName";
        "Argentinischer Peso",
        "displayName-count-one";
        "Argentinischer Peso",
        "displayName-count-other";
        "Argentinische Pesos",

```

```

        "symbol";
        "ARS",
        "symbol-alt-narrow";
        "$";
    }
    "ATS";
    {
        "displayName";
        "Österreichischer Schilling",
        "displayName-count-one";
        "Österreichischer Schilling",
        "displayName-count-other";
        "Österreichische Schilling",
        "symbol";
        "öS";
    }
    "AUD";
    {
        "displayName";
        "Australischer Dollar",
        "displayName-count-one";
        "Australischer Dollar",
        "displayName-count-other";
        "Australische Dollar",
        "symbol";
        "AU$",
        "symbol-alt-narrow";
        "$";
    }
    "AWG";
    {
        "displayName";
        "Aruba-Florin",
        "displayName-count-one";
        "Aruba-Florin",
        "displayName-count-other";
        "Aruba-Florin",
        "symbol";
        "AWG";
    }
    "AZM";
    {
        "displayName";
        "Aserbaidtschan-Manat (1993-2006)",
        "displayName-count-one";
        "Aserbaidtschan-Manat (1993-2006)",
        "displayName-count-other";
        "Aserbaidtschan-Manat (1993-2006)",
        "symbol";
        "AZM";
    }
    "AZN";
    {
        "displayName";
        "Aserbaidtschan-Manat",
        "displayName-count-one";
        "Aserbaidtschan-Manat",

```

```

        "displayName-count-other";
        "Aserbaidtschan-Manat",
        "symbol";
        "AZN";
    }
    "BAD";
    {
        "displayName";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-one";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-other";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "symbol";
        "BAD";
    }
    "BAM";
    {
        "displayName";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-one";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-other";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "symbol";
        "BAM",
        "symbol-alt-narrow";
        "KM";
    }
    "BAN";
    {
        "displayName";
        "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-one";
        "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-other";
        "Bosnien und Herzegowina Neue Dinar (1994-1997)",
        "symbol";
        "BAN";
    }
    "BBD";
    {
        "displayName";
        "Barbados-Dollar",
        "displayName-count-one";
        "Barbados-Dollar",
        "displayName-count-other";
        "Barbados-Dollar",
        "symbol";
        "BBD",
        "symbol-alt-narrow";
        "$";
    }
    "BDT";
    {
        "displayName";
        "Bangladesch-Taka",

```

```

        "displayName-count-one";
        "Bangladesch-Taka",
        "displayName-count-other";
        "Bangladesch-Taka",
        "symbol";
        "BDT",
        "symbol-alt-narrow";
        "৳";
    }
    "BEC";
    {
        "displayName";
        "Belgischer Franc (konvertibel)",
        "displayName-count-one";
        "Belgischer Franc (konvertibel)",
        "displayName-count-other";
        "Belgische Franc (konvertibel)",
        "symbol";
        "BEC";
    }
    "BEF";
    {
        "displayName";
        "Belgischer Franc",
        "displayName-count-one";
        "Belgischer Franc",
        "displayName-count-other";
        "Belgische Franc",
        "symbol";
        "BEF";
    }
    "BEL";
    {
        "displayName";
        "Belgischer Finanz-Franc",
        "displayName-count-one";
        "Belgischer Finanz-Franc",
        "displayName-count-other";
        "Belgische Finanz-Franc",
        "symbol";
        "BEL";
    }
    "BGL";
    {
        "displayName";
        "Bulgarische Lew (1962-1999)",
        "displayName-count-one";
        "Bulgarische Lew (1962-1999)",
        "displayName-count-other";
        "Bulgarische Lew (1962-1999)",
        "symbol";
        "BGL";
    }
    "BGM";
    {
        "displayName";

```



```

        "Bulgarischer Lew (1952-1962)",
        "displayName-count-one";
        "Bulgarischer Lew (1952-1962)",
        "displayName-count-other";
        "Bulgarische Lew (1952-1962)",
        "symbol";
        "BGK";
    }
    "BGN";
    {
        "displayName";
        "Bulgarischer Lew",
        "displayName-count-one";
        "Bulgarischer Lew",
        "displayName-count-other";
        "Bulgarische Lew",
        "symbol";
        "BGN";
    }
    "BGO";
    {
        "displayName";
        "Bulgarischer Lew (1879-1952)",
        "displayName-count-one";
        "Bulgarischer Lew (1879-1952)",
        "displayName-count-other";
        "Bulgarische Lew (1879-1952)",
        "symbol";
        "BGJ";
    }
    "BHD";
    {
        "displayName";
        "Bahrain-Dinar",
        "displayName-count-one";
        "Bahrain-Dinar",
        "displayName-count-other";
        "Bahrain-Dinar",
        "symbol";
        "BHD";
    }
    "BIF";
    {
        "displayName";
        "Burundi-Franc",
        "displayName-count-one";
        "Burundi-Franc",
        "displayName-count-other";
        "Burundi-Francis",
        "symbol";
        "BIF";
    }
    "BMD";
    {
        "displayName";
        "Bermuda-Dollar",
        "displayName-count-one";

```

```

        "Bermuda-Dollar",
        "displayName-count-other";
    "Bermuda-Dollar",
        "symbol";
    "BMD",
        "symbol-alt-narrow";
    "$";
}
"BND";
{
    "displayName";
    "Brunei-Dollar",
        "displayName-count-one";
    "Brunei-Dollar",
        "displayName-count-other";
    "Brunei-Dollar",
        "symbol";
    "BND",
        "symbol-alt-narrow";
    "$";
}
"BOB";
{
    "displayName";
    "Bolivanischer Boliviano",
        "displayName-count-one";
    "Bolivanischer Boliviano",
        "displayName-count-other";
    "Bolivianische Bolivianos",
        "symbol";
    "BOB",
        "symbol-alt-narrow";
    "Bs";
}
"BOL";
{
    "displayName";
    "Bolivianischer Boliviano (1863-1963)",
        "displayName-count-one";
    "Bolivianischer Boliviano (1863-1963)",
        "displayName-count-other";
    "Bolivianische Bolivianos (1863-1963)",
        "symbol";
    "BOL";
}
"BOP";
{
    "displayName";
    "Bolivianischer Peso",
        "displayName-count-one";
    "Bolivianischer Peso",
        "displayName-count-other";
    "Bolivianische Peso",
        "symbol";
    "BOP";
}
"BOV";

```

```

        {
            "displayName";
            "Boliviansiche Mvdol",
            "displayName-count-one";
            "Boliviansiche Mvdol",
            "displayName-count-other";
            "Bolivianische Mvdol",
            "symbol";
            "BOV";
        }
        "BRB";
        {
            "displayName";
            "Brasilianischer Cruzeiro Novo (1967-1986)",
            "displayName-count-one";
            "Brasilianischer Cruzeiro Novo (1967-1986)",
            "displayName-count-other";
            "Brasilianische Cruzeiro Novo (1967-1986)",
            "symbol";
            "BRB";
        }
        "BRC";
        {
            "displayName";
            "Brasilianischer Cruzado (1986-1989)",
            "displayName-count-one";
            "Brasilianischer Cruzado (1986-1989)",
            "displayName-count-other";
            "Brasilianische Cruzado (1986-1989)",
            "symbol";
            "BRC";
        }
        "BRE";
        {
            "displayName";
            "Brasilianischer Cruzeiro (1990-1993)",
            "displayName-count-one";
            "Brasilianischer Cruzeiro (1990-1993)",
            "displayName-count-other";
            "Brasilianische Cruzeiro (1990-1993)",
            "symbol";
            "BRE";
        }
        "BRL";
        {
            "displayName";
            "Brasilianischer Real",
            "displayName-count-one";
            "Brasilianischer Real",
            "displayName-count-other";
            "Brasilianische Real",
            "symbol";
            "R$",
            "symbol-alt-narrow";
            "R$";
        }
        "BRN";
    
```

```

    {
      "displayName";
      "Brasilianischer Cruzado Novo (1989-1990)",
      "displayName-count-one";
      "Brasilianischer Cruzado Novo (1989-1990)",
      "displayName-count-other";
      "Brasilianische Cruzado Novo (1989-1990)",
      "symbol";
      "BRN";
    }
    "BRR";
    {
      "displayName";
      "Brasilianischer Cruzeiro (1993-1994)",
      "displayName-count-one";
      "Brasilianischer Cruzeiro (1993-1994)",
      "displayName-count-other";
      "Brasilianische Cruzeiro (1993-1994)",
      "symbol";
      "BRR";
    }
    "BRZ";
    {
      "displayName";
      "Brasilianischer Cruzeiro (1942-1967)",
      "displayName-count-one";
      "Brasilianischer Cruzeiro (1942-1967)",
      "displayName-count-other";
      "Brasilianischer Cruzeiro (1942-1967)",
      "symbol";
      "BRZ";
    }
    "BSD";
    {
      "displayName";
      "Bahamas-Dollar",
      "displayName-count-one";
      "Bahamas-Dollar",
      "displayName-count-other";
      "Bahamas-Dollar",
      "symbol";
      "BSD",
      "symbol-alt-narrow";
      "$";
    }
    "BTN";
    {
      "displayName";
      "Bhutan-Ngultrum",
      "displayName-count-one";
      "Bhutan-Ngultrum",
      "displayName-count-other";
      "Bhutan-Ngultrum",
      "symbol";
      "BTN";
    }
    "BUK";

```

```

    {
      "displayName";
      "Birmanischer Kyat",
      "displayName-count-one";
      "Birmanischer Kyat",
      "displayName-count-other";
      "Birmanische Kyat",
      "symbol";
      "BUK";
    }
    "BWP";
    {
      "displayName";
      "Botswanischer Pula",
      "displayName-count-one";
      "Botswanischer Pula",
      "displayName-count-other";
      "Botswanische Pula",
      "symbol";
      "BWP",
      "symbol-alt-narrow";
      "P";
    }
    "BYB";
    {
      "displayName";
      "Belarus-Rubel (1994-1999)",
      "displayName-count-one";
      "Belarus-Rubel (1994-1999)",
      "displayName-count-other";
      "Belarus-Rubel (1994-1999)",
      "symbol";
      "BYB";
    }
    "BYN";
    {
      "displayName";
      "Weißrussischer Rubel",
      "displayName-count-one";
      "Weißrussischer Rubel",
      "displayName-count-other";
      "Weißrussische Rubel",
      "symbol";
      "BYN",
      "symbol-alt-narrow";
      "p.";
    }
    "BYR";
    {
      "displayName";
      "Weißrussischer Rubel (2000-2016)",
      "displayName-count-one";
      "Weißrussischer Rubel (2000-2016)",
      "displayName-count-other";
      "Weißrussische Rubel (2000-2016)",
      "symbol";
      "BYR";
    }

```

```

    }
    "BZD";
    {
        "displayName";
        "Belize-Dollar",
            "displayName-count-one";
        "Belize-Dollar",
            "displayName-count-other";
        "Belize-Dollar",
            "symbol";
        "BZD",
            "symbol-alt-narrow";
        "$";
    }
    "CAD";
    {
        "displayName";
        "Kanadischer Dollar",
            "displayName-count-one";
        "Kanadischer Dollar",
            "displayName-count-other";
        "Kanadische Dollar",
            "symbol";
        "CA$",
            "symbol-alt-narrow";
        "$";
    }
    "CDF";
    {
        "displayName";
        "Kongo-Franc",
            "displayName-count-one";
        "Kongo-Franc",
            "displayName-count-other";
        "Kongo-Francs",
            "symbol";
        "CDF";
    }
    "CHE";
    {
        "displayName";
        "WIR-Euro",
            "displayName-count-one";
        "WIR-Euro",
            "displayName-count-other";
        "WIR-Euro",
            "symbol";
        "CHE";
    }
    "CHF";
    {
        "displayName";
        "Schweizer Franken",
            "displayName-count-one";
        "Schweizer Franken",
            "displayName-count-other";
        "Schweizer Franken",

```

```

        "symbol";
        "CHF";
    }
    "CHW";
    {
        "displayName";
        "WIR Franken",
        "displayName-count-one";
        "WIR Franken",
        "displayName-count-other";
        "WIR Franken",
        "symbol";
        "CHW";
    }
    "CLE";
    {
        "displayName";
        "Chilenischer Escudo",
        "displayName-count-one";
        "Chilenischer Escudo",
        "displayName-count-other";
        "Chilenische Escudo",
        "symbol";
        "CLE";
    }
    "CLF";
    {
        "displayName";
        "Chilenische Unidades de Fomento",
        "displayName-count-one";
        "Chilenische Unidades de Fomento",
        "displayName-count-other";
        "Chilenische Unidades de Fomento",
        "symbol";
        "CLF";
    }
    "CLP";
    {
        "displayName";
        "Chilenischer Peso",
        "displayName-count-one";
        "Chilenischer Peso",
        "displayName-count-other";
        "Chilenische Pesos",
        "symbol";
        "CLP",
        "symbol-alt-narrow";
        "$";
    }
    "CNX";
    {
        "displayName";
        "Dollar der Chinesischen Volksbank",
        "displayName-count-one";
        "Dollar der Chinesischen Volksbank",
        "displayName-count-other";
        "Dollar der Chinesischen Volksbank",

```

```

        "symbol";
        "CNX";
    }
    "CNY";
    {
        "displayName";
        "Renminbi Yuan",
        "displayName-count-one";
        "Chinesischer Yuan",
        "displayName-count-other";
        "Renminbi Yuan",
        "symbol";
        "CN¥",
        "symbol-alt-narrow";
        "¥";
    }
    "COP";
    {
        "displayName";
        "Kolumbianischer Peso",
        "displayName-count-one";
        "Kolumbianischer Peso",
        "displayName-count-other";
        "Kolumbianische Pesos",
        "symbol";
        "COP",
        "symbol-alt-narrow";
        "$";
    }
    "COU";
    {
        "displayName";
        "Kolumbianische Unidades de valor real",
        "displayName-count-one";
        "Kolumbianische Unidad de valor real",
        "displayName-count-other";
        "Kolumbianische Unidades de valor real",
        "symbol";
        "COU";
    }
    "CRC";
    {
        "displayName";
        "Costa-Rica-Colón",
        "displayName-count-one";
        "Costa-Rica-Colón",
        "displayName-count-other";
        "Costa-Rica-Colón",
        "symbol";
        "CRC",
        "symbol-alt-narrow";
        "₡";
    }
    "CSD";
    {
        "displayName";
        "Serbischer Dinar (2002-2006)",

```



```

        "displayName-count-one";
        "Serbischer Dinar (2002-2006)",
        "displayName-count-other";
        "Serbische Dinar (2002-2006)",
        "symbol";
        "CSD";
    }
    "CSK";
    {
        "displayName";
        "Tschechoslowakische Krone",
        "displayName-count-one";
        "Tschechoslowakische Kronen",
        "displayName-count-other";
        "Tschechoslowakische Kronen",
        "symbol";
        "CSK";
    }
    "CUC";
    {
        "displayName";
        "Kubanischer Peso (konvertibel)",
        "displayName-count-one";
        "Kubanischer Peso (konvertibel)",
        "displayName-count-other";
        "Kubanische Pesos (konvertibel)",
        "symbol";
        "CUC",
        "symbol-alt-narrow";
        "Cub$";
    }
    "CUP";
    {
        "displayName";
        "Kubanischer Peso",
        "displayName-count-one";
        "Kubanischer Peso",
        "displayName-count-other";
        "Kubanische Pesos",
        "symbol";
        "CUP",
        "symbol-alt-narrow";
        "$";
    }
    "CVE";
    {
        "displayName";
        "Cabo-Verde-Escudo",
        "displayName-count-one";
        "Cabo-Verde-Escudo",
        "displayName-count-other";
        "Cabo-Verde-Escudos",
        "symbol";
        "CVE";
    }
    "CYP";
    {

```

```

        "displayName";
        "Zypern-Pfund",
        "displayName-count-one";
        "Zypern Pfund",
        "displayName-count-other";
        "Zypern Pfund",
        "symbol";
        "CYP";
    }
    "CZK";
    {
        "displayName";
        "Tschechische Krone",
        "displayName-count-one";
        "Tschechische Krone",
        "displayName-count-other";
        "Tschechische Kronen",
        "symbol";
        "CZK",
        "symbol-alt-narrow";
        "Kč";
    }
    "DDM";
    {
        "displayName";
        "Mark der DDR",
        "displayName-count-one";
        "Mark der DDR",
        "displayName-count-other";
        "Mark der DDR",
        "symbol";
        "DDM";
    }
    "DEM";
    {
        "displayName";
        "Deutsche Mark",
        "displayName-count-one";
        "Deutsche Mark",
        "displayName-count-other";
        "Deutsche Mark",
        "symbol";
        "DM";
    }
    "DJF";
    {
        "displayName";
        "Dschibuti-Franc",
        "displayName-count-one";
        "Dschibuti-Franc",
        "displayName-count-other";
        "Dschibuti-Franc",
        "symbol";
        "DJF";
    }
    "DKK";
    {

```

```

        "displayName";
        "Dänische Krone",
            "displayName-count-one";
        "Dänische Krone",
            "displayName-count-other";
        "Dänische Kronen",
            "symbol";
        "DKK",
            "symbol-alt-narrow";
        "kr";
    }
    "DOP";
    {
        "displayName";
        "Dominikanischer Peso",
            "displayName-count-one";
        "Dominikanischer Peso",
            "displayName-count-other";
        "Dominikanische Pesos",
            "symbol";
        "DOP",
            "symbol-alt-narrow";
        "$";
    }
    "DZD";
    {
        "displayName";
        "Algerischer Dinar",
            "displayName-count-one";
        "Algerischer Dinar",
            "displayName-count-other";
        "Algerische Dinar",
            "symbol";
        "DZD";
    }
    "ECS";
    {
        "displayName";
        "Ecuadorianischer Sucre",
            "displayName-count-one";
        "Ecuadorianischer Sucre",
            "displayName-count-other";
        "Ecuadorianische Sucre",
            "symbol";
        "ECS";
    }
    "ECV";
    {
        "displayName";
        "Verrechnungseinheit für Ecuador",
            "displayName-count-one";
        "Verrechnungseinheiten für Ecuador",
            "displayName-count-other";
        "Verrechnungseinheiten für Ecuador",
            "symbol";
        "ECV";
    }
}

```

```

"EEK";
{
  "displayName";
  "Estnische Krone",
    "displayName-count-one";
  "Estnische Krone",
    "displayName-count-other";
  "Estnische Kronen",
    "symbol";
  "EEK";
}
"EGP";
{
  "displayName";
  "Ägyptisches Pfund",
    "displayName-count-one";
  "Ägyptisches Pfund",
    "displayName-count-other";
  "Ägyptische Pfund",
    "symbol";
  "EGP",
    "symbol-alt-narrow";
  "E£";
}
"ERN";
{
  "displayName";
  "Eritreischer Nakfa",
    "displayName-count-one";
  "Eritreischer Nakfa",
    "displayName-count-other";
  "Eritreische Nakfa",
    "symbol";
  "ERN";
}
"ESA";
{
  "displayName";
  "Spanische Peseta (A-Konten)",
    "displayName-count-one";
  "Spanische Peseta (A-Konten)",
    "displayName-count-other";
  "Spanische Peseten (A-Konten)",
    "symbol";
  "ESA";
}
"ESB";
{
  "displayName";
  "Spanische Peseta (konvertibel)",
    "displayName-count-one";
  "Spanische Peseta (konvertibel)",
    "displayName-count-other";
  "Spanische Peseten (konvertibel)",
    "symbol";
  "ESB";
}

```

```
"ESP";
{
  "displayName";
  "Spanische Peseta",
    "displayName-count-one";
  "Spanische Peseta",
    "displayName-count-other";
  "Spanische Peseten",
    "symbol";
  "ESP",
    "symbol-alt-narrow";
  "₧";
}
"ETB";
{
  "displayName";
  "Äthiopischer Birr",
    "displayName-count-one";
  "Äthiopischer Birr",
    "displayName-count-other";
  "Äthiopische Birr",
    "symbol";
  "ETB";
}
"EUR";
{
  "displayName";
  "Euro",
    "displayName-count-one";
  "Euro",
    "displayName-count-other";
  "Euro",
    "symbol";
  "€",
    "symbol-alt-narrow";
  "€";
}
"FIM";
{
  "displayName";
  "Finnische Mark",
    "displayName-count-one";
  "Finnische Mark",
    "displayName-count-other";
  "Finnische Mark",
    "symbol";
  "FIM";
}
"FJD";
{
  "displayName";
  "Fidschi-Dollar",
    "displayName-count-one";
  "Fidschi-Dollar",
    "displayName-count-other";
  "Fidschi-Dollar",
    "symbol";
}
```

```

        "FJD",
        "symbol-alt-narrow";
        "$";
    }
    "FKP";
    {
        "displayName";
        "Falkland-Pfund",
        "displayName-count-one";
        "Falkland-Pfund",
        "displayName-count-other";
        "Falkland-Pfund",
        "symbol";
        "FKP",
        "symbol-alt-narrow";
        "Fl£";
    }
    "FRF";
    {
        "displayName";
        "Französischer Franc",
        "displayName-count-one";
        "Französischer Franc",
        "displayName-count-other";
        "Französische Franc",
        "symbol";
        "FRF";
    }
    "GBP";
    {
        "displayName";
        "Britisches Pfund",
        "displayName-count-one";
        "Britisches Pfund",
        "displayName-count-other";
        "Britische Pfund",
        "symbol";
        "£",
        "symbol-alt-narrow";
        "£";
    }
    "GEK";
    {
        "displayName";
        "Georgischer Kupon Larit",
        "displayName-count-one";
        "Georgischer Kupon Larit",
        "displayName-count-other";
        "Georgische Kupon Larit",
        "symbol";
        "GEK";
    }
    "GEL";
    {
        "displayName";
        "Georgischer Lari",
        "displayName-count-one";

```

```

        "Georgischer Lari",
        "displayName-count-other";
        "Georgische Lari",
        "symbol";
        "GEL",
        "symbol-alt-narrow";
        "ლ",
        "symbol-alt-variant";
        "ლ";
    }
    "GHC";
    {
        "displayName";
        "Ghanaischer Cedi (1979-2007)",
        "displayName-count-one";
        "Ghanaischer Cedi (1979-2007)",
        "displayName-count-other";
        "Ghanaische Cedi (1979-2007)",
        "symbol";
        "GHC";
    }
    "GHS";
    {
        "displayName";
        "Ghanaischer Cedi",
        "displayName-count-one";
        "Ghanaischer Cedi",
        "displayName-count-other";
        "Ghanaische Cedi",
        "symbol";
        "GHS";
    }
    "GIP";
    {
        "displayName";
        "Gibraltar-Pfund",
        "displayName-count-one";
        "Gibraltar-Pfund",
        "displayName-count-other";
        "Gibraltar Pfund",
        "symbol";
        "GIP",
        "symbol-alt-narrow";
        "£";
    }
    "GMD";
    {
        "displayName";
        "Gambia-Dalasi",
        "displayName-count-one";
        "Gambia-Dalasi",
        "displayName-count-other";
        "Gambia-Dalasi",
        "symbol";
        "GMD";
    }
    "GNF";

```

```

    {
      "displayName";
      "Guinea-Franc",
        "displayName-count-one";
      "Guinea-Franc",
        "displayName-count-other";
      "Guinea-Franc",
        "symbol";
      "GNF",
        "symbol-alt-narrow";
      "F.G.";
    }
    "GNS";
    {
      "displayName";
      "Guineischer Syli",
        "displayName-count-one";
      "Guineischer Syli",
        "displayName-count-other";
      "Guineische Syli",
        "symbol";
      "GNS";
    }
  }
  "GQE";
  {
    "displayName";
    "Äquatorialguinea-Ekwele",
      "displayName-count-one";
    "Äquatorialguinea-Ekwele",
      "displayName-count-other";
    "Äquatorialguinea-Ekwele",
      "symbol";
    "GQE";
  }
  "GRD";
  {
    "displayName";
    "Griechische Drachme",
      "displayName-count-one";
    "Griechische Drachme",
      "displayName-count-other";
    "Griechische Drachmen",
      "symbol";
    "GRD";
  }
  "GTQ";
  {
    "displayName";
    "Guatemaltekinscher Quetzal",
      "displayName-count-one";
    "Guatemaltekinscher Quetzal",
      "displayName-count-other";
    "Guatemaltekinsche Quetzales",
      "symbol";
    "GTQ",
      "symbol-alt-narrow";
    "Q";
  }

```



```

    }
    "GWE";
    {
        "displayName";
        "Portugiesisch Guinea Escudo",
            "displayName-count-one";
        "Portugiesisch Guinea Escudo",
            "displayName-count-other";
        "Portugiesisch Guinea Escudo",
            "symbol";
        "GWE";
    }
    "GWP";
    {
        "displayName";
        "Guinea-Bissau Peso",
            "displayName-count-one";
        "Guinea-Bissau Peso",
            "displayName-count-other";
        "Guinea-Bissau Pesos",
            "symbol";
        "GWP";
    }
    "GYD";
    {
        "displayName";
        "Guyana-Dollar",
            "displayName-count-one";
        "Guyana-Dollar",
            "displayName-count-other";
        "Guyana-Dollar",
            "symbol";
        "GYD",
            "symbol-alt-narrow";
        "$";
    }
    "HKD";
    {
        "displayName";
        "Hongkong-Dollar",
            "displayName-count-one";
        "Hongkong-Dollar",
            "displayName-count-other";
        "Hongkong-Dollar",
            "symbol";
        "HK$",
            "symbol-alt-narrow";
        "$";
    }
    "HNL";
    {
        "displayName";
        "Honduras-Lempira",
            "displayName-count-one";
        "Honduras-Lempira",
            "displayName-count-other";
        "Honduras-Lempira",

```

```

        "symbol";
        "HNL",
        "symbol-alt-narrow";
        "L";
    }
    "HRD";
    {
        "displayName";
        "Kroatischer Dinar",
        "displayName-count-one";
        "Kroatischer Dinar",
        "displayName-count-other";
        "Kroatische Dinar",
        "symbol";
        "HRD";
    }
    "HRK";
    {
        "displayName";
        "Kroatischer Kuna",
        "displayName-count-one";
        "Kroatischer Kuna",
        "displayName-count-other";
        "Kroatische Kuna",
        "symbol";
        "HRK",
        "symbol-alt-narrow";
        "kn";
    }
    "HTG";
    {
        "displayName";
        "Haitianische Gourde",
        "displayName-count-one";
        "Haitianische Gourde",
        "displayName-count-other";
        "Haitianische Gourdes",
        "symbol";
        "HTG";
    }
    "HUF";
    {
        "displayName";
        "Ungarischer Forint",
        "displayName-count-one";
        "Ungarischer Forint",
        "displayName-count-other";
        "Ungarische Forint",
        "symbol";
        "HUF",
        "symbol-alt-narrow";
        "Ft";
    }
    "IDR";
    {
        "displayName";
        "Indonesische Rupiah",

```

```

        "displayName-count-one";
        "Indonesische Rupiah",
        "displayName-count-other";
        "Indonesische Rupiah",
        "symbol";
        "IDR",
        "symbol-alt-narrow";
        "Rp";
    }
    "IEP";
    {
        "displayName";
        "Irisches Pfund",
        "displayName-count-one";
        "Irisches Pfund",
        "displayName-count-other";
        "Irische Pfund",
        "symbol";
        "IEP";
    }
    "ILP";
    {
        "displayName";
        "Israelisches Pfund",
        "displayName-count-one";
        "Israelisches Pfund",
        "displayName-count-other";
        "Israelische Pfund",
        "symbol";
        "ILP";
    }
    "ILR";
    {
        "displayName";
        "Israelischer Schekel (1980-1985)",
        "displayName-count-one";
        "Israelischer Schekel (1980-1985)",
        "displayName-count-other";
        "Israelische Schekel (1980-1985)";
    }
    "ILS";
    {
        "displayName";
        "Israelischer Neuer Schekel",
        "displayName-count-one";
        "Israelischer Neuer Schekel",
        "displayName-count-other";
        "Israelische Neue Schekel",
        "symbol";
        "₪",
        "symbol-alt-narrow";
        "₪";
    }
    "INR";
    {
        "displayName";
        "Indische Rupie",

```

```

        "displayName-count-one";
        "Indische Rupie",
        "displayName-count-other";
        "Indische Rupien",
        "symbol";
        "₹",
        "symbol-alt-narrow";
        "₹";
    }
    "IQD";
    {
        "displayName";
        "Irakischer Dinar",
        "displayName-count-one";
        "Irakischer Dinar",
        "displayName-count-other";
        "Irakische Dinar",
        "symbol";
        "IQD";
    }
    "IRR";
    {
        "displayName";
        "Iranischer Rial",
        "displayName-count-one";
        "Iranischer Rial",
        "displayName-count-other";
        "Iranische Rial",
        "symbol";
        "IRR";
    }
    "ISJ";
    {
        "displayName";
        "Isländische Krone (1918-1981)",
        "displayName-count-one";
        "Isländische Krone (1918-1981)",
        "displayName-count-other";
        "Isländische Kronen (1918-1981)";
    }
    "ISK";
    {
        "displayName";
        "Isländische Krone",
        "displayName-count-one";
        "Isländische Krone",
        "displayName-count-other";
        "Isländische Kronen",
        "symbol";
        "ISK",
        "symbol-alt-narrow";
        "kr";
    }
    "ITL";
    {
        "displayName";
        "Italienische Lira",

```

```

        "displayName-count-one";
        "Italienische Lira",
        "displayName-count-other";
        "Italienische Lire",
        "symbol";
        "ITL";
    }
    "JMD";
    {
        "displayName";
        "Jamaika-Dollar",
        "displayName-count-one";
        "Jamaika-Dollar",
        "displayName-count-other";
        "Jamaika-Dollar",
        "symbol";
        "JMD",
        "symbol-alt-narrow";
        "$";
    }
    "JOD";
    {
        "displayName";
        "Jordanischer Dinar",
        "displayName-count-one";
        "Jordanischer Dinar",
        "displayName-count-other";
        "Jordanische Dinar",
        "symbol";
        "JOD";
    }
    "JPY";
    {
        "displayName";
        "Japanischer Yen",
        "displayName-count-one";
        "Japanischer Yen",
        "displayName-count-other";
        "Japanische Yen",
        "symbol";
        "¥",
        "symbol-alt-narrow";
        "¥";
    }
    "KES";
    {
        "displayName";
        "Kenia-Schilling",
        "displayName-count-one";
        "Kenia-Schilling",
        "displayName-count-other";
        "Kenia-Schilling",
        "symbol";
        "KES";
    }
    "KGS";
    {

```

```

        "displayName";
        "Kirgisischer Som",
        "displayName-count-one";
        "Kirgisischer Som",
        "displayName-count-other";
        "Kirgisische Som",
        "symbol";
        "KGS";
    }
    "KHR";
    {
        "displayName";
        "Kambodschanischer Riel",
        "displayName-count-one";
        "Kambodschanischer Riel",
        "displayName-count-other";
        "Kambodschanische Riel",
        "symbol";
        "KHR",
        "symbol-alt-narrow";
        "៛";
    }
    "KMF";
    {
        "displayName";
        "Komoren-Franc",
        "displayName-count-one";
        "Komoren-Franc",
        "displayName-count-other";
        "Komoren-Francis",
        "symbol";
        "KMF",
        "symbol-alt-narrow";
        "FC";
    }
    "KPW";
    {
        "displayName";
        "Nordkoreanischer Won",
        "displayName-count-one";
        "Nordkoreanischer Won",
        "displayName-count-other";
        "Nordkoreanische Won",
        "symbol";
        "KPW",
        "symbol-alt-narrow";
        "₩";
    }
    "KRH";
    {
        "displayName";
        "Südkoreanischer Hwan (1953-1962)",
        "displayName-count-one";
        "Südkoreanischer Hwan (1953-1962)",
        "displayName-count-other";
        "Südkoreanischer Hwan (1953-1962)",

```

```

        "symbol";
        "KRH";
    }
    "KRO";
    {
        "displayName";
        "Südkoreanischer Won (1945-1953)",
        "displayName-count-one";
        "Südkoreanischer Won (1945-1953)",
        "displayName-count-other";
        "Südkoreanischer Won (1945-1953)",
        "symbol";
        "KRO";
    }
    "KRW";
    {
        "displayName";
        "Südkoreanischer Won",
        "displayName-count-one";
        "Südkoreanischer Won",
        "displayName-count-other";
        "Südkoreanische Won",
        "symbol";
        "₩",
        "symbol-alt-narrow";
        "₩";
    }
    "KWD";
    {
        "displayName";
        "Kuwait-Dinar",
        "displayName-count-one";
        "Kuwait-Dinar",
        "displayName-count-other";
        "Kuwait-Dinar",
        "symbol";
        "KWD";
    }
    "KYD";
    {
        "displayName";
        "Kaiman-Dollar",
        "displayName-count-one";
        "Kaiman-Dollar",
        "displayName-count-other";
        "Kaiman-Dollar",
        "symbol";
        "KYD",
        "symbol-alt-narrow";
        "$";
    }
    "KZT";
    {
        "displayName";
        "Kasachischer Tenge",
        "displayName-count-one";
        "Kasachischer Tenge",

```

```

        "displayName-count-other";
        "Kasachische Tenge",
        "symbol";
        "KZT",
        "symbol-alt-narrow";
        "₸";
    }
    "LAK";
    {
        "displayName";
        "Laotischer Kip",
        "displayName-count-one";
        "Laotischer Kip",
        "displayName-count-other";
        "Laotische Kip",
        "symbol";
        "LAK",
        "symbol-alt-narrow";
        "₭";
    }
    "LBP";
    {
        "displayName";
        "Libanesisches Pfund",
        "displayName-count-one";
        "Libanesisches Pfund",
        "displayName-count-other";
        "Libanesische Pfund",
        "symbol";
        "LBP",
        "symbol-alt-narrow";
        "L£";
    }
    "LKR";
    {
        "displayName";
        "Sri-Lanka-Rupie",
        "displayName-count-one";
        "Sri-Lanka-Rupie",
        "displayName-count-other";
        "Sri-Lanka-Rupien",
        "symbol";
        "LKR",
        "symbol-alt-narrow";
        "Rs";
    }
    "LRD";
    {
        "displayName";
        "Liberianischer Dollar",
        "displayName-count-one";
        "Liberianischer Dollar",
        "displayName-count-other";
        "Liberianische Dollar",
        "symbol";
        "LRD",
        "symbol-alt-narrow";
    }

```



```

        "$";
    }
    "LSL";
    {
        "displayName";
        "Loti",
        "displayName-count-one";
        "Loti",
        "displayName-count-other";
        "Loti",
        "symbol";
        "LSL";
    }
    "LTL";
    {
        "displayName";
        "Litauischer Litas",
        "displayName-count-one";
        "Litauischer Litas",
        "displayName-count-other";
        "Litauische Litas",
        "symbol";
        "LTL",
        "symbol-alt-narrow";
        "Lt";
    }
    "LTT";
    {
        "displayName";
        "Litauischer Talonas",
        "displayName-count-one";
        "Litauische Talonas",
        "displayName-count-other";
        "Litauische Talonas",
        "symbol";
        "LTT";
    }
    "LUC";
    {
        "displayName";
        "Luxemburgischer Franc (konvertibel)",
        "displayName-count-one";
        "Luxemburgische Franc (konvertibel)",
        "displayName-count-other";
        "Luxemburgische Franc (konvertibel)",
        "symbol";
        "LUC";
    }
    "LUF";
    {
        "displayName";
        "Luxemburgischer Franc",
        "displayName-count-one";
        "Luxemburgische Franc",
        "displayName-count-other";
        "Luxemburgische Franc",
        "symbol";
    }

```

```

        "LUF";
    }
    "LUL";
    {
        "displayName";
        "Luxemburgischer Finanz-Franc",
        "displayName-count-one";
        "Luxemburgische Finanz-Franc",
        "displayName-count-other";
        "Luxemburgische Finanz-Franc",
        "symbol";
        "LUL";
    }
    "LVL";
    {
        "displayName";
        "Lettischer Lats",
        "displayName-count-one";
        "Lettischer Lats",
        "displayName-count-other";
        "Lettische Lats",
        "symbol";
        "LVL",
        "symbol-alt-narrow";
        "Ls";
    }
    "LVR";
    {
        "displayName";
        "Lettischer Rubel",
        "displayName-count-one";
        "Lettische Rubel",
        "displayName-count-other";
        "Lettische Rubel",
        "symbol";
        "LVR";
    }
    "LYD";
    {
        "displayName";
        "Libyscher Dinar",
        "displayName-count-one";
        "Libyscher Dinar",
        "displayName-count-other";
        "Libysche Dinar",
        "symbol";
        "LYD";
    }
    "MAD";
    {
        "displayName";
        "Marokkanischer Dirham",
        "displayName-count-one";
        "Marokkanischer Dirham",
        "displayName-count-other";
        "Marokkanische Dirham",
        "symbol";
    }

```

```

        "MAD";
    }
    "MAF";
    {
        "displayName";
        "Marokkanischer Franc",
        "displayName-count-one";
        "Marokkanische Franc",
        "displayName-count-other";
        "Marokkanische Franc",
        "symbol";
        "MAF";
    }
    "MCF";
    {
        "displayName";
        "Monegassischer Franc",
        "displayName-count-one";
        "Monegassischer Franc",
        "displayName-count-other";
        "Monegassische Franc",
        "symbol";
        "MCF";
    }
    "MDC";
    {
        "displayName";
        "Moldau-Cupon",
        "displayName-count-one";
        "Moldau-Cupon",
        "displayName-count-other";
        "Moldau-Cupon",
        "symbol";
        "MDC";
    }
    "MDL";
    {
        "displayName";
        "Moldau-Leu",
        "displayName-count-one";
        "Moldau-Leu",
        "displayName-count-other";
        "Moldau-Leu",
        "symbol";
        "MDL";
    }
    "MGA";
    {
        "displayName";
        "Madagaskar-Ariary",
        "displayName-count-one";
        "Madagaskar-Ariary",
        "displayName-count-other";
        "Madagaskar-Ariary",
        "symbol";
        "MGA",
        "symbol-alt-narrow";
    }

```

```

        "Ar";
    }
    "MGF";
    {
        "displayName";
        "Madagaskar-Franc",
        "displayName-count-one";
        "Madagaskar-Franc",
        "displayName-count-other";
        "Madagaskar-Franc",
        "symbol";
        "MGF";
    }
    "MKD";
    {
        "displayName";
        "Mazedonischer Denar",
        "displayName-count-one";
        "Mazedonischer Denar",
        "displayName-count-other";
        "Mazedonische Denari",
        "symbol";
        "MKD";
    }
    "MKN";
    {
        "displayName";
        "Mazedonischer Denar (1992-1993)",
        "displayName-count-one";
        "Mazedonischer Denar (1992-1993)",
        "displayName-count-other";
        "Mazedonische Denar (1992-1993)",
        "symbol";
        "MKN";
    }
    "MLF";
    {
        "displayName";
        "Malischer Franc",
        "displayName-count-one";
        "Malische Franc",
        "displayName-count-other";
        "Malische Franc",
        "symbol";
        "MLF";
    }
    "MMK";
    {
        "displayName";
        "Myanmarischer Kyat",
        "displayName-count-one";
        "Myanmarischer Kyat",
        "displayName-count-other";
        "Myanmarische Kyat",
        "symbol";
        "MMK",
        "symbol-alt-narrow";
    }

```

```
        "K";
    }
    "MNT";
    {
        "displayName";
        "Mongolischer Tögrög",
        "displayName-count-one";
        "Mongolischer Tögrög",
        "displayName-count-other";
        "Mongolische Tögrög",
        "symbol";
        "MNT",
        "symbol-alt-narrow";
        "₮";
    }
    "MOP";
    {
        "displayName";
        "Macao-Pataca",
        "displayName-count-one";
        "Macao-Pataca",
        "displayName-count-other";
        "Macao-Pataca",
        "symbol";
        "MOP";
    }
    "MRO";
    {
        "displayName";
        "Mauretanischer Ouguiya",
        "displayName-count-one";
        "Mauretanischer Ouguiya",
        "displayName-count-other";
        "Mauretanische Ouguiya",
        "symbol";
        "MRO";
    }
    "MTL";
    {
        "displayName";
        "Maltesische Lira",
        "displayName-count-one";
        "Maltesische Lira",
        "displayName-count-other";
        "Maltesische Lira",
        "symbol";
        "MTL";
    }
    "MTP";
    {
        "displayName";
        "Maltesisches Pfund",
        "displayName-count-one";
        "Maltesische Pfund",
        "displayName-count-other";
        "Maltesische Pfund",
        "symbol";
    }
```

```

        "MTP";
    }
    "MUR";
    {
        "displayName";
        "Mauritius-Rupie",
            "displayName-count-one";
        "Mauritius-Rupie",
            "displayName-count-other";
        "Mauritius-Rupien",
            "symbol";
        "MUR",
            "symbol-alt-narrow";
        "Rs";
    }
    "MVP";
    {
        "displayName";
        "Malediven-Rupie (alt)",
            "displayName-count-one";
        "Malediven-Rupie (alt)",
            "displayName-count-other";
        "Malediven-Rupien (alt)";
    }
    "MVR";
    {
        "displayName";
        "Malediven-Rufiyaa",
            "displayName-count-one";
        "Malediven-Rufiyaa",
            "displayName-count-other";
        "Malediven-Rupien",
            "symbol";
        "MVR";
    }
    "MWK";
    {
        "displayName";
        "Malawi-Kwacha",
            "displayName-count-one";
        "Malawi-Kwacha",
            "displayName-count-other";
        "Malawi-Kwacha",
            "symbol";
        "MWK";
    }
    "MXN";
    {
        "displayName";
        "Mexikanischer Peso",
            "displayName-count-one";
        "Mexikanischer Peso",
            "displayName-count-other";
        "Mexikanische Pesos",
            "symbol";
        "MX$",
            "symbol-alt-narrow";
    }

```

```

        "$";
    }
    "MXP";
    {
        "displayName";
        "Mexikanischer Silber-Peso (1861-1992)",
        "displayName-count-one";
        "Mexikanische Silber-Peso (1861-1992)",
        "displayName-count-other";
        "Mexikanische Silber-Pesos (1861-1992)",
        "symbol";
        "MXP";
    }
    "MXV";
    {
        "displayName";
        "Mexicanischer Unidad de Inversion (UDI)",
        "displayName-count-one";
        "Mexicanischer Unidad de Inversion (UDI)",
        "displayName-count-other";
        "Mexikanische Unidad de Inversion (UDI)",
        "symbol";
        "MXV";
    }
    "MYR";
    {
        "displayName";
        "Malaysischer Ringgit",
        "displayName-count-one";
        "Malaysischer Ringgit",
        "displayName-count-other";
        "Malaysische Ringgit",
        "symbol";
        "MYR",
        "symbol-alt-narrow";
        "RM";
    }
    "MZE";
    {
        "displayName";
        "Mosambikanischer Escudo",
        "displayName-count-one";
        "Mozambikanische Escudo",
        "displayName-count-other";
        "Mozambikanische Escudo",
        "symbol";
        "MZE";
    }
    "MZM";
    {
        "displayName";
        "Mosambikanischer Metical (1980-2006)",
        "displayName-count-one";
        "Mosambikanischer Metical (1980-2006)",
        "displayName-count-other";
        "Mosambikanische Meticais (1980-2006)",
        "symbol";
    }

```

```

        "MZM";
    }
    "MZN";
    {
        "displayName";
        "Mosambikanischer Metical",
        "displayName-count-one";
        "Mosambikanischer Metical",
        "displayName-count-other";
        "Mosambikanische Meticais",
        "symbol";
        "MZN";
    }
    "NAD";
    {
        "displayName";
        "Namibia-Dollar",
        "displayName-count-one";
        "Namibia-Dollar",
        "displayName-count-other";
        "Namibia-Dollar",
        "symbol";
        "NAD",
        "symbol-alt-narrow";
        "$";
    }
    "NGN";
    {
        "displayName";
        "Nigerianischer Naira",
        "displayName-count-one";
        "Nigerianischer Naira",
        "displayName-count-other";
        "Nigerianische Naira",
        "symbol";
        "NGN",
        "symbol-alt-narrow";
        "₦";
    }
    "NIC";
    {
        "displayName";
        "Nicaraguanischer Córdoba (1988-1991)",
        "displayName-count-one";
        "Nicaraguanischer Córdoba (1988-1991)",
        "displayName-count-other";
        "Nicaraguanische Córdoba (1988-1991)",
        "symbol";
        "NIC";
    }
    "NIO";
    {
        "displayName";
        "Nicaragua-Córdoba",
        "displayName-count-one";
        "Nicaragua-Córdoba",
        "displayName-count-other";
    }

```



```

        "Nicaragua-Córdobas",
        "symbol";
        "NIO",
        "symbol-alt-narrow";
        "C$";
    }
    "NLG";
    {
        "displayName";
        "Niederländischer Gulden",
        "displayName-count-one";
        "Niederländischer Gulden",
        "displayName-count-other";
        "Niederländische Gulden",
        "symbol";
        "NLG";
    }
    "NOK";
    {
        "displayName";
        "Norwegische Krone",
        "displayName-count-one";
        "Norwegische Krone",
        "displayName-count-other";
        "Norwegische Kronen",
        "symbol";
        "NOK",
        "symbol-alt-narrow";
        "kr";
    }
    "NPR";
    {
        "displayName";
        "Nepalesische Rupie",
        "displayName-count-one";
        "Nepalesische Rupie",
        "displayName-count-other";
        "Nepalesische Rupien",
        "symbol";
        "NPR",
        "symbol-alt-narrow";
        "Rs";
    }
    "NZD";
    {
        "displayName";
        "Neuseeland-Dollar",
        "displayName-count-one";
        "Neuseeland-Dollar",
        "displayName-count-other";
        "Neuseeland-Dollar",
        "symbol";
        "NZ$",
        "symbol-alt-narrow";
        "$";
    }
    "OMR";

```

```

    {
      "displayName";
      "Omanischer Rial",
      "displayName-count-one";
      "Omanischer Rial",
      "displayName-count-other";
      "Omanische Rials",
      "symbol";
      "OMR";
    }
    "PAB";
    {
      "displayName";
      "Panamaischer Balboa",
      "displayName-count-one";
      "Panamaischer Balboa",
      "displayName-count-other";
      "Panamaische Balboas",
      "symbol";
      "PAB";
    }
    "PEI";
    {
      "displayName";
      "Peruanischer Inti",
      "displayName-count-one";
      "Peruanische Inti",
      "displayName-count-other";
      "Peruanische Inti",
      "symbol";
      "PEI";
    }
    "PEN";
    {
      "displayName";
      "Peruanischer Sol",
      "displayName-count-one";
      "Peruanischer Sol",
      "displayName-count-other";
      "Peruanische Sol",
      "symbol";
      "PEN";
    }
    "PES";
    {
      "displayName";
      "Peruanischer Sol (1863-1965)",
      "displayName-count-one";
      "Peruanischer Sol (1863-1965)",
      "displayName-count-other";
      "Peruanische Sol (1863-1965)",
      "symbol";
      "PES";
    }
    "PGK";
    {
      "displayName";

```

```

        "Papua-Neuguineischer Kina",
        "displayName-count-one";
        "Papua-Neuguineischer Kina",
        "displayName-count-other";
        "Papua-Neuguineische Kina",
        "symbol";
        "PGK";
    }
    "PHP";
    {
        "displayName";
        "Philippinischer Peso",
        "displayName-count-one";
        "Philippinischer Peso",
        "displayName-count-other";
        "Philippinische Pesos",
        "symbol";
        "PHP",
        "symbol-alt-narrow";
        "₱";
    }
    "PKR";
    {
        "displayName";
        "Pakistanische Rupie",
        "displayName-count-one";
        "Pakistanische Rupie",
        "displayName-count-other";
        "Pakistanische Rupien",
        "symbol";
        "PKR",
        "symbol-alt-narrow";
        "Rs";
    }
    "PLN";
    {
        "displayName";
        "Polnischer Złoty",
        "displayName-count-one";
        "Polnischer Złoty",
        "displayName-count-other";
        "Polnische Złoty",
        "symbol";
        "PLN",
        "symbol-alt-narrow";
        "zł";
    }
    "PLZ";
    {
        "displayName";
        "Polnischer Zloty (1950-1995)",
        "displayName-count-one";
        "Polnischer Zloty (1950-1995)",
        "displayName-count-other";
        "Polnische Zloty (1950-1995)",
        "symbol";
        "PLZ";
    }

```

```

    }
    "PTE";
    {
        "displayName";
        "Portugiesischer Escudo",
        "displayName-count-one";
        "Portugiesische Escudo",
        "displayName-count-other";
        "Portugiesische Escudo",
        "symbol";
        "PTE";
    }
    "PYG";
    {
        "displayName";
        "Paraguayischer Guaraní",
        "displayName-count-one";
        "Paraguayischer Guaraní",
        "displayName-count-other";
        "Paraguayische Guaraníes",
        "symbol";
        "PYG",
        "symbol-alt-narrow";
        "₲";
    }
    "QAR";
    {
        "displayName";
        "Katar-Riyal",
        "displayName-count-one";
        "Katar-Riyal",
        "displayName-count-other";
        "Katar-Riyal",
        "symbol";
        "QAR";
    }
    "RHD";
    {
        "displayName";
        "Rhodesischer Dollar",
        "displayName-count-one";
        "Rhodesische Dollar",
        "displayName-count-other";
        "Rhodesische Dollar",
        "symbol";
        "RHD";
    }
    "ROL";
    {
        "displayName";
        "Rumänischer Leu (1952-2006)",
        "displayName-count-one";
        "Rumänischer Leu (1952-2006)",
        "displayName-count-other";
        "Rumänische Leu (1952-2006)",
        "symbol";
        "ROL";
    }

```

```

    }
    "RON";
    {
        "displayName";
        "Rumänischer Leu",
        "displayName-count-one";
        "Rumänischer Leu",
        "displayName-count-other";
        "Rumänische Leu",
        "symbol";
        "RON",
        "symbol-alt-narrow";
        "L";
    }
    "RSD";
    {
        "displayName";
        "Serbischer Dinar",
        "displayName-count-one";
        "Serbischer Dinar",
        "displayName-count-other";
        "Serbische Dinaren",
        "symbol";
        "RSD";
    }
    "RUB";
    {
        "displayName";
        "Russischer Rubel",
        "displayName-count-one";
        "Russischer Rubel",
        "displayName-count-other";
        "Russische Rubel",
        "symbol";
        "RUB",
        "symbol-alt-narrow";
        "₽";
    }
    "RUR";
    {
        "displayName";
        "Russischer Rubel (1991-1998)",
        "displayName-count-one";
        "Russischer Rubel (1991-1998)",
        "displayName-count-other";
        "Russische Rubel (1991-1998)",
        "symbol";
        "RUR",
        "symbol-alt-narrow";
        "p.";
    }
    "RWF";
    {
        "displayName";
        "Ruanda-Franc",
        "displayName-count-one";
        "Ruanda-Franc",

```

```

        "displayName-count-other";
        "Ruanda-Francs",
        "symbol";
        "RWF",
        "symbol-alt-narrow";
        "F.Rw";
    }
    "SAR";
    {
        "displayName";
        "Saudi-Rial",
        "displayName-count-one";
        "Saudi-Rial",
        "displayName-count-other";
        "Saudi-Rial",
        "symbol";
        "SAR";
    }
    "SBD";
    {
        "displayName";
        "Salomonen-Dollar",
        "displayName-count-one";
        "Salomonen-Dollar",
        "displayName-count-other";
        "Salomonen-Dollar",
        "symbol";
        "SBD",
        "symbol-alt-narrow";
        "$";
    }
    "SCR";
    {
        "displayName";
        "Seychellen-Rupie",
        "displayName-count-one";
        "Seychellen-Rupie",
        "displayName-count-other";
        "Seychellen-Rupien",
        "symbol";
        "SCR";
    }
    "SDD";
    {
        "displayName";
        "Sudanesischer Dinar (1992-2007)",
        "displayName-count-one";
        "Sudanesischer Dinar (1992-2007)",
        "displayName-count-other";
        "Sudanesische Dinar (1992-2007)",
        "symbol";
        "SDD";
    }
    "SDG";
    {
        "displayName";
        "Sudanesisches Pfund",

```

```

        "displayName-count-one";
        "Sudanesisches Pfund",
        "displayName-count-other";
        "Sudanesische Pfund",
        "symbol";
        "SDG";
    }
    "SDP";
    {
        "displayName";
        "Sudanesisches Pfund (1957-1998)",
        "displayName-count-one";
        "Sudanesisches Pfund (1957-1998)",
        "displayName-count-other";
        "Sudanesische Pfund (1957-1998)",
        "symbol";
        "SDP";
    }
    "SEK";
    {
        "displayName";
        "Schwedische Krone",
        "displayName-count-one";
        "Schwedische Krone",
        "displayName-count-other";
        "Schwedische Kronen",
        "symbol";
        "SEK",
        "symbol-alt-narrow";
        "kr";
    }
    "SGD";
    {
        "displayName";
        "Singapur-Dollar",
        "displayName-count-one";
        "Singapur-Dollar",
        "displayName-count-other";
        "Singapur-Dollar",
        "symbol";
        "SGD",
        "symbol-alt-narrow";
        "$";
    }
    "SHP";
    {
        "displayName";
        "St. Helena-Pfund",
        "displayName-count-one";
        "St. Helena-Pfund",
        "displayName-count-other";
        "St. Helena-Pfund",
        "symbol";
        "SHP",
        "symbol-alt-narrow";
        "£";
    }
}

```

```
"SIT";
{
  "displayName";
  "Slowenischer Tolar",
    "displayName-count-one";
  "Slowenischer Tolar",
    "displayName-count-other";
  "Slowenische Tolar",
    "symbol";
  "SIT";
}
"SKK";
{
  "displayName";
  "Slowakische Krone",
    "displayName-count-one";
  "Slowakische Kronen",
    "displayName-count-other";
  "Slowakische Kronen",
    "symbol";
  "SKK";
}
"SLL";
{
  "displayName";
  "Sierra-leonischer Leone",
    "displayName-count-one";
  "Sierra-leonischer Leone",
    "displayName-count-other";
  "Sierra-leonische Leones",
    "symbol";
  "SLL";
}
"SOS";
{
  "displayName";
  "Somalia-Schilling",
    "displayName-count-one";
  "Somalia-Schilling",
    "displayName-count-other";
  "Somalia-Schilling",
    "symbol";
  "SOS";
}
"SRD";
{
  "displayName";
  "Suriname-Dollar",
    "displayName-count-one";
  "Suriname-Dollar",
    "displayName-count-other";
  "Suriname-Dollar",
    "symbol";
  "SRD",
    "symbol-alt-narrow";
  "$";
}
```



```

"SRG";
{
  "displayName";
  "Suriname Gulden",
    "displayName-count-one";
  "Suriname-Gulden",
    "displayName-count-other";
  "Suriname-Gulden",
    "symbol";
  "SRG";
}
"SSP";
{
  "displayName";
  "Südsudanesisches Pfund",
    "displayName-count-one";
  "Südsudanesisches Pfund",
    "displayName-count-other";
  "Südsudanesische Pfund",
    "symbol";
  "SSP",
    "symbol-alt-narrow";
  "£";
}
"STD";
{
  "displayName";
  "São-toméischer Dobra",
    "displayName-count-one";
  "São-toméischer Dobra",
    "displayName-count-other";
  "São-toméische Dobra",
    "symbol";
  "STD",
    "symbol-alt-narrow";
  "Db";
}
"SUR";
{
  "displayName";
  "Sowjetischer Rubel",
    "displayName-count-one";
  "Sowjetische Rubel",
    "displayName-count-other";
  "Sowjetische Rubel",
    "symbol";
  "SUR";
}
"SVC";
{
  "displayName";
  "El Salvador Colon",
    "displayName-count-one";
  "El Salvador-Colon",
    "displayName-count-other";
  "El Salvador-Colon",
    "symbol";
}

```

```

        "SVC";
    }
    "SYP";
    {
        "displayName";
        "Syrisches Pfund",
            "displayName-count-one";
        "Syrisches Pfund",
            "displayName-count-other";
        "Syrische Pfund",
            "symbol";
        "SYP",
            "symbol-alt-narrow";
        "SYP";
    }
    "SZL";
    {
        "displayName";
        "Swasiländischer Lilangeni",
            "displayName-count-one";
        "Swasiländischer Lilangeni",
            "displayName-count-other";
        "Swasiländische Emalangeni",
            "symbol";
        "SZL";
    }
    "THB";
    {
        "displayName";
        "Thailändischer Baht",
            "displayName-count-one";
        "Thailändischer Baht",
            "displayName-count-other";
        "Thailändische Baht",
            "symbol";
        "฿",
            "symbol-alt-narrow";
        "฿";
    }
    "TJR";
    {
        "displayName";
        "Tadschikistan Rubel",
            "displayName-count-one";
        "Tadschikistan-Rubel",
            "displayName-count-other";
        "Tadschikistan-Rubel",
            "symbol";
        "TJR";
    }
    "TJS";
    {
        "displayName";
        "Tadschikistan-Somoni",
            "displayName-count-one";
        "Tadschikistan-Somoni",
            "displayName-count-other";
    }

```

```

        "Tadschikistan-Somoni",
        "symbol";
        "TJS";
    }
    "TMM";
    {
        "displayName";
        "Turkmenistan-Manat (1993-2009)",
        "displayName-count-one";
        "Turkmenistan-Manat (1993-2009)",
        "displayName-count-other";
        "Turkmenistan-Manat (1993-2009)",
        "symbol";
        "TMM";
    }
    "TMT";
    {
        "displayName";
        "Turkmenistan-Manat",
        "displayName-count-one";
        "Turkmenistan-Manat",
        "displayName-count-other";
        "Turkmenistan-Manat",
        "symbol";
        "TMT";
    }
    "TND";
    {
        "displayName";
        "Tunesischer Dinar",
        "displayName-count-one";
        "Tunesischer Dinar",
        "displayName-count-other";
        "Tunesische Dinar",
        "symbol";
        "TND";
    }
    "TOP";
    {
        "displayName";
        "Tongaischer Pa'anga",
        "displayName-count-one";
        "Tongaischer Pa'anga",
        "displayName-count-other";
        "Tongaische Pa'anga",
        "symbol";
        "TOP",
        "symbol-alt-narrow";
        "T$";
    }
    "TPE";
    {
        "displayName";
        "Timor-Escudo",
        "displayName-count-one";
        "Timor-Escudo",
        "displayName-count-other";
    }

```

```

        "Timor-Escudo",
        "symbol";
        "TPE";
    }
    "TRL";
    {
        "displayName";
        "Türkische Lira (1922-2005)",
        "displayName-count-one";
        "Türkische Lira (1922-2005)",
        "displayName-count-other";
        "Türkische Lira (1922-2005)",
        "symbol";
        "TRL";
    }
    "TRY";
    {
        "displayName";
        "Türkische Lira",
        "displayName-count-one";
        "Türkische Lira",
        "displayName-count-other";
        "Türkische Lira",
        "symbol";
        "TRY",
        "symbol-alt-narrow";
        "₺",
        "symbol-alt-variant";
        "TL";
    }
    "TTD";
    {
        "displayName";
        "Trinidad und Tobago-Dollar",
        "displayName-count-one";
        "Trinidad und Tobago-Dollar",
        "displayName-count-other";
        "Trinidad und Tobago-Dollar",
        "symbol";
        "TTD",
        "symbol-alt-narrow";
        "$";
    }
    "TWD";
    {
        "displayName";
        "Neuer Taiwan-Dollar",
        "displayName-count-one";
        "Neuer Taiwan-Dollar",
        "displayName-count-other";
        "Neue Taiwan-Dollar",
        "symbol";
        "NT$",
        "symbol-alt-narrow";
        "NT$";
    }
    "TZS";

```

```

    {
      "displayName";
      "Tansania-Schilling",
      "displayName-count-one";
      "Tansania-Schilling",
      "displayName-count-other";
      "Tansania-Schilling",
      "symbol";
      "TZS";
    }
    "UAH";
    {
      "displayName";
      "Ukrainische Hrywnja",
      "displayName-count-one";
      "Ukrainische Hrywnja",
      "displayName-count-other";
      "Ukrainische Hrywen",
      "symbol";
      "UAH",
      "symbol-alt-narrow";
      "₴";
    }
    "UAK";
    {
      "displayName";
      "Ukrainischer Karbovanetz",
      "displayName-count-one";
      "Ukrainische Karbovanetz",
      "displayName-count-other";
      "Ukrainische Karbovanetz",
      "symbol";
      "UAK";
    }
    "UGS";
    {
      "displayName";
      "Uganda-Schilling (1966-1987)",
      "displayName-count-one";
      "Uganda-Schilling (1966-1987)",
      "displayName-count-other";
      "Uganda-Schilling (1966-1987)",
      "symbol";
      "UGS";
    }
    "UGX";
    {
      "displayName";
      "Uganda-Schilling",
      "displayName-count-one";
      "Uganda-Schilling",
      "displayName-count-other";
      "Uganda-Schilling",
      "symbol";
      "UGX";
    }
    "USD";

```

```

    {
      "displayName";
      "US-Dollar",
        "displayName-count-one";
      "US-Dollar",
        "displayName-count-other";
      "US-Dollar",
        "symbol";
      "$",
        "symbol-alt-narrow";
      "$";
    }
    "USN";
    {
      "displayName";
      "US Dollar (Nächster Tag)",
        "displayName-count-one";
      "US-Dollar (Nächster Tag)",
        "displayName-count-other";
      "US-Dollar (Nächster Tag)",
        "symbol";
      "USN";
    }
    "USS";
    {
      "displayName";
      "US Dollar (Gleicher Tag)",
        "displayName-count-one";
      "US-Dollar (Gleicher Tag)",
        "displayName-count-other";
      "US-Dollar (Gleicher Tag)",
        "symbol";
      "USS";
    }
    "UYI";
    {
      "displayName";
      "Uruguayischer Peso (Indexierte Rechnungseinheiten)",
        "displayName-count-one";
      "Uruguayischer Peso (Indexierte Rechnungseinheiten)",
        "displayName-count-other";
      "Uruguayische Pesos (Indexierte Rechnungseinheiten)",
        "symbol";
      "UYI";
    }
    "UYP";
    {
      "displayName";
      "Uruguayischer Peso (1975-1993)",
        "displayName-count-one";
      "Uruguayischer Peso (1975-1993)",
        "displayName-count-other";
      "Uruguayische Pesos (1975-1993)",
        "symbol";
      "UYP";
    }
    "UYU";

```

```

    {
      "displayName";
      "Uruguayischer Peso",
      "displayName-count-one";
      "Uruguayischer Peso",
      "displayName-count-other";
      "Uruguayische Pesos",
      "symbol";
      "UYU",
      "symbol-alt-narrow";
      "$";
    }
    "UZS";
    {
      "displayName";
      "Usbekistan-Sum",
      "displayName-count-one";
      "Usbekistan-Sum",
      "displayName-count-other";
      "Usbekistan-Sum",
      "symbol";
      "UZS";
    }
    "VEB";
    {
      "displayName";
      "Venezolanischer Bolívar (1871-2008)",
      "displayName-count-one";
      "Venezolanischer Bolívar (1871-2008)",
      "displayName-count-other";
      "Venezolanische Bolívares (1871-2008)",
      "symbol";
      "VEB";
    }
    "VEF";
    {
      "displayName";
      "Venezolanischer Bolívar",
      "displayName-count-one";
      "Venezolanischer Bolívar",
      "displayName-count-other";
      "Venezolanische Bolívares",
      "symbol";
      "VEF",
      "symbol-alt-narrow";
      "Bs";
    }
    "VND";
    {
      "displayName";
      "Vietnamesischer Dong",
      "displayName-count-one";
      "Vietnamesischer Dong",
      "displayName-count-other";
      "Vietnamesische Dong",
      "symbol";
      "₫",
    }
  }

```

```

        "symbol-alt-narrow";
        "₫";
    }
    "VNN";
    {
        "displayName";
        "Vietnamesischer Dong (1978-1985) ",
        "displayName-count-one";
        "Vietnamesischer Dong (1978-1985) ",
        "displayName-count-other";
        "Vietnamesische Dong (1978-1985) ",
        "symbol";
        "VNN";
    }
    "VUV";
    {
        "displayName";
        "Vanuatu-Vatu",
        "displayName-count-one";
        "Vanuatu-Vatu",
        "displayName-count-other";
        "Vanuatu-Vatu",
        "symbol";
        "VUV";
    }
    "WST";
    {
        "displayName";
        "Samoanischer Tala",
        "displayName-count-one";
        "Samoanischer Tala",
        "displayName-count-other";
        "Samoanische Tala",
        "symbol";
        "WST";
    }
    "XAF";
    {
        "displayName";
        "CFA-Franc (BEAC) ",
        "displayName-count-one";
        "CFA-Franc (BEAC) ",
        "displayName-count-other";
        "CFA-Franc (BEAC) ",
        "symbol";
        "FCFA";
    }
    "XAG";
    {
        "displayName";
        "Unze Silber",
        "displayName-count-one";
        "Unze Silber",
        "displayName-count-other";
        "Unzen Silber",
        "symbol";
        "XAG";
    }

```



```

    }
    "XAU";
    {
        "displayName";
        "Unze Gold",
            "displayName-count-one";
        "Unze Gold",
            "displayName-count-other";
        "Unzen Gold",
            "symbol";
        "XAU";
    }
    "XBA";
    {
        "displayName";
        "Europäische Rechnungseinheit",
            "displayName-count-one";
        "Europäische Rechnungseinheiten",
            "displayName-count-other";
        "Europäische Rechnungseinheiten",
            "symbol";
        "XBA";
    }
    "XBB";
    {
        "displayName";
        "Europäische Währungseinheit (XBB)",
            "displayName-count-one";
        "Europäische Währungseinheiten (XBB)",
            "displayName-count-other";
        "Europäische Währungseinheiten (XBB)",
            "symbol";
        "XBB";
    }
    "XBC";
    {
        "displayName";
        "Europäische Rechnungseinheit (XBC)",
            "displayName-count-one";
        "Europäische Rechnungseinheiten (XBC)",
            "displayName-count-other";
        "Europäische Rechnungseinheiten (XBC)",
            "symbol";
        "XBC";
    }
    "XBD";
    {
        "displayName";
        "Europäische Rechnungseinheit (XBD)",
            "displayName-count-one";
        "Europäische Rechnungseinheiten (XBD)",
            "displayName-count-other";
        "Europäische Rechnungseinheiten (XBD)",
            "symbol";
        "XBD";
    }
    "XCD";

```

```

    {
      "displayName";
      "Ostkaribischer Dollar",
        "displayName-count-one";
      "Ostkaribischer Dollar",
        "displayName-count-other";
      "Ostkaribische Dollar",
        "symbol";
      "EC$",
        "symbol-alt-narrow";
      "$";
    }
    "XDR";
    {
      "displayName";
      "Sonderziehungsrechte",
        "displayName-count-one";
      "Sonderziehungsrechte",
        "displayName-count-other";
      "Sonderziehungsrechte",
        "symbol";
      "XDR";
    }
    "XEU";
    {
      "displayName";
      "Europäische Währungseinheit (XEU)",
        "displayName-count-one";
      "Europäische Währungseinheiten (XEU)",
        "displayName-count-other";
      "Europäische Währungseinheiten (XEU)",
        "symbol";
      "XEU";
    }
    "XFO";
    {
      "displayName";
      "Französischer Gold-Franc",
        "displayName-count-one";
      "Französische Gold-Franc",
        "displayName-count-other";
      "Französische Gold-Franc",
        "symbol";
      "XFO";
    }
    "XFU";
    {
      "displayName";
      "Französischer UIC-Franc",
        "displayName-count-one";
      "Französische UIC-Franc",
        "displayName-count-other";
      "Französische UIC-Franc",
        "symbol";
      "XFU";
    }
    "XOF";

```

```

    {
      "displayName";
      "CFA-Franc (BCEAO)",
      "displayName-count-one";
      "CFA-Franc (BCEAO)",
      "displayName-count-other";
      "CFA-Francs (BCEAO)",
      "symbol";
      "CFA";
    }
    "XPD";
    {
      "displayName";
      "Unze Palladium",
      "displayName-count-one";
      "Unze Palladium",
      "displayName-count-other";
      "Unzen Palladium",
      "symbol";
      "XPD";
    }
    "XPF";
    {
      "displayName";
      "CFP-Franc",
      "displayName-count-one";
      "CFP-Franc",
      "displayName-count-other";
      "CFP-Franc",
      "symbol";
      "CFPF";
    }
    "XPT";
    {
      "displayName";
      "Unze Platin",
      "displayName-count-one";
      "Unze Platin",
      "displayName-count-other";
      "Unzen Platin",
      "symbol";
      "XPT";
    }
    "XRE";
    {
      "displayName";
      "RINET Funds",
      "displayName-count-one";
      "RINET Funds",
      "displayName-count-other";
      "RINET Funds",
      "symbol";
      "XRE";
    }
    "XSU";
    {
      "displayName";

```

```

        "SUCRE",
        "displayName-count-one";
        "SUCRE",
        "displayName-count-other";
        "SUCRE",
        "symbol";
        "XSU";
    }
    "XTS";
    {
        "displayName";
        "Testwährung",
        "displayName-count-one";
        "Testwährung",
        "displayName-count-other";
        "Testwährung",
        "symbol";
        "XTS";
    }
    "XUA";
    {
        "displayName";
        "Rechnungseinheit der AfEB",
        "displayName-count-one";
        "Rechnungseinheit der AfEB",
        "displayName-count-other";
        "Rechnungseinheiten der AfEB",
        "symbol";
        "XUA";
    }
    "XXX";
    {
        "displayName";
        "Unbekannte Währung",
        "displayName-count-one";
        "(unbekannte Währung)",
        "displayName-count-other";
        "(unbekannte Währung)",
        "symbol";
        "XXX";
    }
    "YDD";
    {
        "displayName";
        "Jemen-Dinar",
        "displayName-count-one";
        "Jemen-Dinar",
        "displayName-count-other";
        "Jemen-Dinar",
        "symbol";
        "YDD";
    }
    "YER";
    {
        "displayName";
        "Jemen-Rial",
        "displayName-count-one";

```

```

        "Jemen-Rial",
        "displayName-count-other";
        "Jemen-Rial",
        "symbol";
        "YER";
    }
    "YUD";
    {
        "displayName";
        "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-one";
        "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-other";
        "Jugoslawische Dinar (1966-1990)",
        "symbol";
        "YUD";
    }
    "YUM";
    {
        "displayName";
        "Jugoslawischer Neuer Dinar (1994-2002)",
        "displayName-count-one";
        "Jugoslawischer Neuer Dinar (1994-2002)",
        "displayName-count-other";
        "Jugoslawische Neue Dinar (1994-2002)",
        "symbol";
        "YUM";
    }
    "YUN";
    {
        "displayName";
        "Jugoslawischer Dinar (konvertibel)",
        "displayName-count-one";
        "Jugoslawische Dinar (konvertibel)",
        "displayName-count-other";
        "Jugoslawische Dinar (konvertibel)",
        "symbol";
        "YUN";
    }
    "YUR";
    {
        "displayName";
        "Jugoslawischer reformierter Dinar (1992-1993)",
        "displayName-count-one";
        "Jugoslawischer reformierter Dinar (1992-1993)",
        "displayName-count-other";
        "Jugoslawische reformierte Dinar (1992-1993)",
        "symbol";
        "YUR";
    }
    "ZAL";
    {
        "displayName";
        "Südafrikanischer Rand (Finanz)",
        "displayName-count-one";
        "Südafrikanischer Rand (Finanz)",
        "displayName-count-other";
    }

```

```

        "Südafrikanischer Rand (Finanz)",
        "symbol";
        "ZAL";
    }
    "ZAR";
    {
        "displayName";
        "Südafrikanischer Rand",
        "displayName-count-one";
        "Südafrikanischer Rand",
        "displayName-count-other";
        "Südafrikanische Rand",
        "symbol";
        "ZAR",
        "symbol-alt-narrow";
        "R";
    }
    "ZMK";
    {
        "displayName";
        "Kwacha (1968-2012)",
        "displayName-count-one";
        "Kwacha (1968-2012)",
        "displayName-count-other";
        "Kwacha (1968-2012)",
        "symbol";
        "ZMK";
    }
    "ZMW";
    {
        "displayName";
        "Kwacha",
        "displayName-count-one";
        "Kwacha",
        "displayName-count-other";
        "Kwacha",
        "symbol";
        "ZMW",
        "symbol-alt-narrow";
        "K";
    }
    "ZRN";
    {
        "displayName";
        "Zaire-Neuer Zaïre (1993-1998)",
        "displayName-count-one";
        "Zaire-Neuer Zaïre (1993-1998)",
        "displayName-count-other";
        "Zaire-Neue Zaïre (1993-1998)",
        "symbol";
        "ZRN";
    }
    "ZRZ";
    {
        "displayName";
        "Zaire-Zaïre (1971-1993)",
        "displayName-count-one";
    }

```

CURRENCYDATA.JSX

Copyright © 2001 -2024 Syncfusion Inc.

```
        "_cldrVersion";
        "31";
    }
    "currencyData";
    {
        "fractions";
        {
            "ADP";
            {
                "_rounding";
                "0",
                "_digits";
                "0";
            }
            "AFN";
            {
                "_rounding";
                "0",
                "_digits";
                "0";
            }
            "ALL";
            {
                "_rounding";
                "0",
                "_digits";
                "0";
            }
            "AMD";
            {
                "_rounding";
                "0",
                "_digits";
                "0";
            }
            "BHD";
            {
                "_rounding";
                "0",
                "_digits";
                "3";
            }
            "BIF";
            {
                "_rounding";
                "0",
                "_digits";
                "0";
            }
            "BYN";
            {
                "_rounding";
                "0",
                "_digits";
                "2";
            }
            "BYR";
```



```
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"CAD";
{
    "_rounding";
    "0",
    "_digits";
    "2",
    "_cashRounding";
    "5";
}
"CHF";
{
    "_rounding";
    "0",
    "_digits";
    "2",
    "_cashRounding";
    "5";
}
"CLF";
{
    "_rounding";
    "0",
    "_digits";
    "4";
}
"CLP";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"COP";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"CRC";
{
    "_rounding";
    "0",
    "_digits";
    "2",
    "_cashRounding";
    "0",
    "_cashDigits";
    "0";
}
"CZK";
```

```

    {
        "_rounding";
        "0",
        "_digits";
        "2",
        "_cashRounding";
        "0",
        "_cashDigits";
        "0";
    }
    "DEFAULT";
    {
        "_rounding";
        "0",
        "_digits";
        "2";
    }
    "DJF";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "ESP";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "GNF";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "GYD";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "HUF";
    {
        "_rounding";
        "0",
        "_digits";
        "2",
        "_cashRounding";
        "0",
        "_cashDigits";
        "0";
    }
    "IDR";

```

```
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"IQD";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"IRR";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"ISK";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"ITL";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"JOD";
{
    "_rounding";
    "0",
    "_digits";
    "3";
}
"JPY";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"KMF";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"KPW";
{
```

```
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "KRW";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "KWD";
    {
        "_rounding";
        "0",
        "_digits";
        "3";
    }
    "LAK";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "LBP";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "LUF";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "LYD";
    {
        "_rounding";
        "0",
        "_digits";
        "3";
    }
    "MGA";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "MGF";
    {
        "_rounding";
```

```
        "0",
        "_digits";
    "0";
}
"MMK";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"MNT";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"MRO";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"MUR";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"OMR";
{
    "_rounding";
    "0",
    "_digits";
    "3";
}
"PKR";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"PYG";
{
    "_rounding";
    "0",
    "_digits";
    "0";
}
"RSD";
{
    "_rounding";
    "0",
```

```

        "_digits";
        "0";
    }
    "RWF";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "SLL";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "SOS";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "STD";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "SYP";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "TMM";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "TND";
    {
        "_rounding";
        "0",
        "_digits";
        "3";
    }
    "TRL";
    {
        "_rounding";
        "0",
        "_digits";

```

```
        "0";
    }
    "TWD";
    {
        "_rounding";
        "0",
        "_digits";
        "2",
        "_cashRounding";
        "0",
        "_cashDigits";
        "0";
    }
    "TZS";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "UGX";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "UYI";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "UZS";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "VND";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "VUV";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "XAF";
    {
```

```

        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "XOF";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "XPF";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "YER";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "ZMK";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
    "ZWD";
    {
        "_rounding";
        "0",
        "_digits";
        "0";
    }
}
"region";
{
    "AC";
    [
        {
            "SHP": {
                "_from": "1976-01-01"
            }
        }
    ],
    "AD";
    [
        {
            "ESP": {
                "_from": "1873-01-01",
                "_to": "2002-02-28"
            }
        }
    ]
}

```



```

    },
    {
      "ADP": {
        "_from": "1936-01-01",
        "_to": "2001-12-31"
      }
    },
    {
      "FRF": {
        "_from": "1960-01-01",
        "_to": "2002-02-17"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "AE";
  [
    {
      "AED": {
        "_from": "1973-05-19"
      }
    }
  ],
  "AF";
  [
    {
      "AFA": {
        "_from": "1927-03-14",
        "_to": "2002-12-31"
      }
    },
    {
      "AFN": {
        "_from": "2002-10-07"
      }
    }
  ],
  "AG";
  [
    {
      "XCD": {
        "_from": "1965-10-06"
      }
    }
  ],
  "AI";
  [
    {
      "XCD": {
        "_from": "1965-10-06"
      }
    }
  ]
}

```

```
],
  "AL";
[
  {
    "ALK": {
      "_from": "1946-11-01",
      "_to": "1965-08-16"
    }
  },
  {
    "ALL": {
      "_from": "1965-08-16"
    }
  }
],
  "AM";
[
  {
    "SUR": {
      "_from": "1961-01-01",
      "_to": "1991-12-25"
    }
  },
  {
    "RUR": {
      "_from": "1991-12-25",
      "_to": "1993-11-22"
    }
  },
  {
    "AMD": {
      "_from": "1993-11-22"
    }
  }
],
  "AO";
[
  {
    "AOK": {
      "_from": "1977-01-08",
      "_to": "1991-03-01"
    }
  },
  {
    "AON": {
      "_from": "1990-09-25",
      "_to": "2000-02-01"
    }
  },
  {
    "AOR": {
      "_from": "1995-07-01",
      "_to": "2000-02-01"
    }
  },
  {
    "AOA": {
```

```
        "_from": "1999-12-13"
      }
    },
    ],
    "AQ";
  [
    {
      "XXX": {
        "_tender": "false"
      }
    }
  ],
  "AR";
  [
    {
      "ARM": {
        "_from": "1881-11-05",
        "_to": "1970-01-01"
      }
    },
    {
      "ARL": {
        "_from": "1970-01-01",
        "_to": "1983-06-01"
      }
    },
    {
      "ARP": {
        "_from": "1983-06-01",
        "_to": "1985-06-14"
      }
    },
    {
      "ARA": {
        "_from": "1985-06-14",
        "_to": "1992-01-01"
      }
    },
    {
      "ARS": {
        "_from": "1992-01-01"
      }
    }
  ],
  "AS";
  [
    {
      "USD": {
        "_from": "1904-07-16"
      }
    }
  ],
  "AT";
  [
    {
      "ATS": {
        "_from": "1947-12-04",
```

```

        "_to": "2002-02-28"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "AU";
  [
    {
      "AUD": {
        "_from": "1966-02-14"
      }
    }
  ],
  "AW";
  [
    {
      "ANG": {
        "_from": "1940-05-10",
        "_to": "1986-01-01"
      }
    },
    {
      "AWG": {
        "_from": "1986-01-01"
      }
    }
  ],
  "AX";
  [
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "AZ";
  [
    {
      "SUR": {
        "_from": "1961-01-01",
        "_to": "1991-12-25"
      }
    },
    {
      "RUR": {
        "_from": "1991-12-25",
        "_to": "1994-01-01"
      }
    },
    {
      "AZM": {
        "_from": "1993-11-22",
        "_to": "2006-12-31"
      }
    }
  ]

```

```

    },
    {
      "AZN": {
        "_from": "2006-01-01"
      }
    }
  ],
  "BA";
  [
    {
      "YUD": {
        "_from": "1966-01-01",
        "_to": "1990-01-01"
      }
    },
    {
      "YUN": {
        "_from": "1990-01-01",
        "_to": "1992-07-01"
      }
    },
    {
      "YUR": {
        "_from": "1992-07-01",
        "_to": "1993-10-01"
      }
    },
    {
      "BAD": {
        "_from": "1992-07-01",
        "_to": "1994-08-15"
      }
    },
    {
      "BAN": {
        "_from": "1994-08-15",
        "_to": "1997-07-01"
      }
    },
    {
      "BAM": {
        "_from": "1995-01-01"
      }
    }
  ],
  "BB";
  [
    {
      "XCD": {
        "_from": "1965-10-06",
        "_to": "1973-12-03"
      }
    },
    {
      "BBD": {
        "_from": "1973-12-03"
      }
    }
  ]

```

```
    }  
  },  
  "BD";  
  [  
    {  
      "INR": {  
        "_from": "1835-08-17",  
        "_to": "1948-04-01"  
      }  
    },  
    {  
      "PKR": {  
        "_from": "1948-04-01",  
        "_to": "1972-01-01"  
      }  
    },  
    {  
      "BDT": {  
        "_from": "1972-01-01"  
      }  
    }  
  ],  
  "BE";  
  [  
    {  
      "NLG": {  
        "_from": "1816-12-15",  
        "_to": "1831-02-07"  
      }  
    },  
    {  
      "BEF": {  
        "_from": "1831-02-07",  
        "_to": "2002-02-28"  
      }  
    },  
    {  
      "BEC": {  
        "_tender": "false",  
        "_from": "1970-01-01",  
        "_to": "1990-03-05"  
      }  
    },  
    {  
      "BEL": {  
        "_tender": "false",  
        "_from": "1970-01-01",  
        "_to": "1990-03-05"  
      }  
    },  
    {  
      "EUR": {  
        "_from": "1999-01-01"  
      }  
    }  
  ],  
],
```

```
        "BF";
    [
        {
            "XOF": {
                "_from": "1984-08-04"
            }
        }
    ],
    "BG";
    [
        {
            "BGO": {
                "_from": "1879-07-08",
                "_to": "1952-05-12"
            }
        },
        {
            "BGM": {
                "_from": "1952-05-12",
                "_to": "1962-01-01"
            }
        },
        {
            "BGL": {
                "_from": "1962-01-01",
                "_to": "1999-07-05"
            }
        },
        {
            "BGN": {
                "_from": "1999-07-05"
            }
        }
    ],
    "BH";
    [
        {
            "BHD": {
                "_from": "1965-10-16"
            }
        }
    ],
    "BI";
    [
        {
            "BIF": {
                "_from": "1964-05-19"
            }
        }
    ],
    "BJ";
    [
        {
            "XOF": {
                "_from": "1975-11-30"
            }
        }
    ]
}
```

```

    ],
    "BL";
    [
      {
        "FRF": {
          "_from": "1960-01-01",
          "_to": "2002-02-17"
        }
      },
      {
        "EUR": {
          "_from": "1999-01-01"
        }
      }
    ],
    "BM";
    [
      {
        "BMD": {
          "_from": "1970-02-06"
        }
      }
    ],
    "BN";
    [
      {
        "MYR": {
          "_from": "1963-09-16",
          "_to": "1967-06-12"
        }
      },
      {
        "BND": {
          "_from": "1967-06-12"
        }
      }
    ],
    "BO";
    [
      {
        "BOV": {
          "_tender": "false"
        }
      },
      {
        "BOL": {
          "_from": "1863-06-23",
          "_to": "1963-01-01"
        }
      },
      {
        "BOP": {
          "_from": "1963-01-01",
          "_to": "1986-12-31"
        }
      }
    ],
    {

```



```

        "BOB": {
            "_from": "1987-01-01"
        }
    ],
    "BQ";
    [
        {
            "ANG": {
                "_from": "2010-10-10",
                "_to": "2011-01-01"
            }
        },
        {
            "USD": {
                "_from": "2011-01-01"
            }
        }
    ],
    "BR";
    [
        {
            "BRZ": {
                "_from": "1942-11-01",
                "_to": "1967-02-13"
            }
        },
        {
            "BRB": {
                "_from": "1967-02-13",
                "_to": "1986-02-28"
            }
        },
        {
            "BRC": {
                "_from": "1986-02-28",
                "_to": "1989-01-15"
            }
        },
        {
            "BRN": {
                "_from": "1989-01-15",
                "_to": "1990-03-16"
            }
        },
        {
            "BRE": {
                "_from": "1990-03-16",
                "_to": "1993-08-01"
            }
        },
        {
            "BRR": {
                "_from": "1993-08-01",
                "_to": "1994-07-01"
            }
        }
    ],

```

```
    {
      "BRL": {
        "_from": "1994-07-01"
      }
    },
    "BS";
    [
      {
        "BSD": {
          "_from": "1966-05-25"
        }
      }
    ],
    "BT";
    [
      {
        "INR": {
          "_from": "1907-01-01"
        }
      },
      {
        "BTN": {
          "_from": "1974-04-16"
        }
      }
    ],
    "BU";
    [
      {
        "BUK": {
          "_from": "1952-07-01",
          "_to": "1989-06-18"
        }
      }
    ],
    "BV";
    [
      {
        "NOK": {
          "_from": "1905-06-07"
        }
      }
    ],
    "BW";
    [
      {
        "ZAR": {
          "_from": "1961-02-14",
          "_to": "1976-08-23"
        }
      },
      {
        "BWP": {
          "_from": "1976-08-23"
        }
      }
    ]
  ]
}
```

```
],
  "BY";
[
  {
    "SUR": {
      "_from": "1961-01-01",
      "_to": "1991-12-25"
    }
  },
  {
    "RUR": {
      "_from": "1991-12-25",
      "_to": "1994-11-08"
    }
  },
  {
    "BYB": {
      "_from": "1994-08-01",
      "_to": "2000-12-31"
    }
  },
  {
    "BYR": {
      "_from": "2000-01-01",
      "_to": "2017-01-01"
    }
  },
  {
    "BYN": {
      "_from": "2016-07-01"
    }
  }
],
  "BZ";
[
  {
    "BZD": {
      "_from": "1974-01-01"
    }
  }
],
  "CA";
[
  {
    "CAD": {
      "_from": "1858-01-01"
    }
  }
],
  "CC";
[
  {
    "AUD": {
      "_from": "1966-02-14"
    }
  }
],
```

```
        "CD";
    [
        {
            "ZRZ": {
                "_from": "1971-10-27",
                "_to": "1993-11-01"
            }
        },
        {
            "ZRN": {
                "_from": "1993-11-01",
                "_to": "1998-07-01"
            }
        },
        {
            "CDF": {
                "_from": "1998-07-01"
            }
        }
    ],
    "CF";
    [
        {
            "XAF": {
                "_from": "1993-01-01"
            }
        }
    ],
    "CG";
    [
        {
            "XAF": {
                "_from": "1993-01-01"
            }
        }
    ],
    "CH";
    [
        {
            "CHE": {
                "_tender": "false"
            }
        },
        {
            "CHW": {
                "_tender": "false"
            }
        },
        {
            "CHF": {
                "_from": "1799-03-17"
            }
        }
    ],
    "CI";
    [
        {
```

```

        "XOF": {
            "_from": "1958-12-04"
        }
    ],
    "CK";
    [
        {
            "NZD": {
                "_from": "1967-07-10"
            }
        }
    ],
    "CL";
    [
        {
            "CLF": {
                "_tender": "false"
            }
        },
        {
            "CLE": {
                "_from": "1960-01-01",
                "_to": "1975-09-29"
            }
        },
        {
            "CLP": {
                "_from": "1975-09-29"
            }
        }
    ],
    "CM";
    [
        {
            "XAF": {
                "_from": "1973-04-01"
            }
        }
    ],
    "CN";
    [
        {
            "CNY": {
                "_from": "1953-03-01"
            }
        },
        {
            "CNX": {
                "_tender": "false",
                "_from": "1979-01-01",
                "_to": "1998-12-31"
            }
        }
    ],
    "CO";
    [

```

```

        {
            "COU": {
                "_tender": "false"
            }
        },
        {
            "COP": {
                "_from": "1905-01-01"
            }
        }
    ],
    "CP";
    [
        {
            "XXX": {
                "_tender": "false"
            }
        }
    ],
    "CR";
    [
        {
            "CRC": {
                "_from": "1896-10-26"
            }
        }
    ],
    "CS";
    [
        {
            "YUM": {
                "_from": "1994-01-24",
                "_to": "2002-05-15"
            }
        },
        {
            "CSD": {
                "_from": "2002-05-15",
                "_to": "2006-06-03"
            }
        },
        {
            "EUR": {
                "_from": "2003-02-04",
                "_to": "2006-06-03"
            }
        }
    ],
    "CU";
    [
        {
            "CUP": {
                "_from": "1859-01-01"
            }
        },
        {
            "USD": {

```

```
        "_from": "1899-01-01",
        "_to": "1959-01-01"
      }
    },
    {
      "CUC": {
        "_from": "1994-01-01"
      }
    }
  ],
  "CV";
[
  {
    "PTE": {
      "_from": "1911-05-22",
      "_to": "1975-07-05"
    }
  },
  {
    "CVE": {
      "_from": "1914-01-01"
    }
  }
],
  "CW";
[
  {
    "ANG": {
      "_from": "2010-10-10"
    }
  }
],
  "CX";
[
  {
    "AUD": {
      "_from": "1966-02-14"
    }
  }
],
  "CY";
[
  {
    "CYP": {
      "_from": "1914-09-10",
      "_to": "2008-01-31"
    }
  },
  {
    "EUR": {
      "_from": "2008-01-01"
    }
  }
],
  "CZ";
[
  {
```

```
        "CSK": {
            "_from": "1953-06-01",
            "_to": "1993-03-01"
        }
    },
    {
        "CZK": {
            "_from": "1993-01-01"
        }
    }
],
"DD";
[
    {
        "DDM": {
            "_from": "1948-07-20",
            "_to": "1990-10-02"
        }
    }
],
"DE";
[
    {
        "DEM": {
            "_from": "1948-06-20",
            "_to": "2002-02-28"
        }
    },
    {
        "EUR": {
            "_from": "1999-01-01"
        }
    }
],
"DG";
[
    {
        "USD": {
            "_from": "1965-11-08"
        }
    }
],
"DJ";
[
    {
        "DJF": {
            "_from": "1977-06-27"
        }
    }
],
"DK";
[
    {
        "DKK": {
            "_from": "1873-05-27"
        }
    }
]
```



```
],
  "DM";
[
  {
    "XCD": {
      "_from": "1965-10-06"
    }
  }
],
  "DO";
[
  {
    "USD": {
      "_from": "1905-06-21",
      "_to": "1947-10-01"
    }
  },
  {
    "DOP": {
      "_from": "1947-10-01"
    }
  }
],
  "DZ";
[
  {
    "DZD": {
      "_from": "1964-04-01"
    }
  }
],
  "EA";
[
  {
    "EUR": {
      "_from": "1999-01-01"
    }
  }
],
  "EC";
[
  {
    "ECS": {
      "_from": "1884-04-01",
      "_to": "2000-10-02"
    }
  },
  {
    "ECV": {
      "_tender": "false",
      "_from": "1993-05-23",
      "_to": "2000-01-09"
    }
  },
  {
    "USD": {
      "_from": "2000-10-02"
    }
  }
]
```

```
    }  
  },  
  ],  
  "EE";  
  [  
    {  
      "SUR": {  
        "_from": "1961-01-01",  
        "_to": "1992-06-20"  
      }  
    },  
    {  
      "EEK": {  
        "_from": "1992-06-21",  
        "_to": "2010-12-31"  
      }  
    },  
    {  
      "EUR": {  
        "_from": "2011-01-01"  
      }  
    }  
  ],  
  "EG";  
  [  
    {  
      "EGP": {  
        "_from": "1885-11-14"  
      }  
    }  
  ],  
  "EH";  
  [  
    {  
      "MAD": {  
        "_from": "1976-02-26"  
      }  
    }  
  ],  
  "ER";  
  [  
    {  
      "ETB": {  
        "_from": "1993-05-24",  
        "_to": "1997-11-08"  
      }  
    },  
    {  
      "ERN": {  
        "_from": "1997-11-08"  
      }  
    }  
  ],  
  "ES";  
  [  
    {  
      "ESP": {
```

```

        "_from": "1868-10-19",
        "_to": "2002-02-28"
      }
    },
    {
      "ESB": {
        "_tender": "false",
        "_from": "1975-01-01",
        "_to": "1994-12-31"
      }
    },
    {
      "ESA": {
        "_tender": "false",
        "_from": "1978-01-01",
        "_to": "1981-12-31"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "ET";
  [
    {
      "ETB": {
        "_from": "1976-09-15"
      }
    }
  ],
  "EU";
  [
    {
      "XEU": {
        "_tender": "false",
        "_from": "1979-01-01",
        "_to": "1998-12-31"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "FI";
  [
    {
      "FIM": {
        "_from": "1963-01-01",
        "_to": "2002-02-28"
      }
    },
    {
      "EUR": {

```

```
        "_from": "1999-01-01"
      }
    },
    ],
    "FJ";
  [
    {
      "FJD": {
        "_from": "1969-01-13"
      }
    }
  ],
  "FK";
  [
    {
      "FKP": {
        "_from": "1901-01-01"
      }
    }
  ],
  "FM";
  [
    {
      "JPY": {
        "_from": "1914-10-03",
        "_to": "1944-01-01"
      }
    },
    {
      "USD": {
        "_from": "1944-01-01"
      }
    }
  ],
  "FO";
  [
    {
      "DKK": {
        "_from": "1948-01-01"
      }
    }
  ],
  "FR";
  [
    {
      "FRF": {
        "_from": "1960-01-01",
        "_to": "2002-02-17"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "GA";
```

```
[
  {
    "XAF": {
      "_from": "1993-01-01"
    }
  },
  "GB";
  [
    {
      "GBP": {
        "_from": "1694-07-27"
      }
    }
  ],
  "GD";
  [
    {
      "XCD": {
        "_from": "1967-02-27"
      }
    }
  ],
  "GE";
  [
    {
      "SUR": {
        "_from": "1961-01-01",
        "_to": "1991-12-25"
      }
    },
    {
      "RUR": {
        "_from": "1991-12-25",
        "_to": "1993-06-11"
      }
    },
    {
      "GEK": {
        "_from": "1993-04-05",
        "_to": "1995-09-25"
      }
    },
    {
      "GEL": {
        "_from": "1995-09-23"
      }
    }
  ],
  "GF";
  [
    {
      "FRF": {
        "_from": "1960-01-01",
        "_to": "2002-02-17"
      }
    },
  ],
```

```
        {
          "EUR": {
            "_from": "1999-01-01"
          }
        },
        "GG";
        [
          {
            "GBP": {
              "_from": "1830-01-01"
            }
          }
        ],
        "GH";
        [
          {
            "GHC": {
              "_from": "1979-03-09",
              "_to": "2007-12-31"
            }
          },
          {
            "GHS": {
              "_from": "2007-07-03"
            }
          }
        ],
        "GI";
        [
          {
            "GIP": {
              "_from": "1713-01-01"
            }
          }
        ],
        "GL";
        [
          {
            "DKK": {
              "_from": "1873-05-27"
            }
          }
        ],
        "GM";
        [
          {
            "GMD": {
              "_from": "1971-07-01"
            }
          }
        ],
        "GN";
        [
          {
            "GNS": {
              "_from": "1972-10-02",
```

```
        "_to": "1986-01-06"
      }
    },
    {
      "GNF": {
        "_from": "1986-01-06"
      }
    }
  ],
  "GP";
  [
    {
      "FRF": {
        "_from": "1960-01-01",
        "_to": "2002-02-17"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "GQ";
  [
    {
      "GQE": {
        "_from": "1975-07-07",
        "_to": "1986-06-01"
      }
    },
    {
      "XAF": {
        "_from": "1993-01-01"
      }
    }
  ],
  "GR";
  [
    {
      "GRD": {
        "_from": "1954-05-01",
        "_to": "2002-02-28"
      }
    },
    {
      "EUR": {
        "_from": "2001-01-01"
      }
    }
  ],
  "GS";
  [
    {
      "GBP": {
        "_from": "1908-01-01"
      }
    }
  ]
}
```

```
    },
    [
      {
        "GTQ": {
          "_from": "1925-05-27"
        }
      }
    ],
    "GU";
    [
      {
        "USD": {
          "_from": "1944-08-21"
        }
      }
    ],
    "GW";
    [
      {
        "GWE": {
          "_from": "1914-01-01",
          "_to": "1976-02-28"
        }
      },
      {
        "GWP": {
          "_from": "1976-02-28",
          "_to": "1997-03-31"
        }
      },
      {
        "XOF": {
          "_from": "1997-03-31"
        }
      }
    ],
    "GY";
    [
      {
        "GYD": {
          "_from": "1966-05-26"
        }
      }
    ],
    "HK";
    [
      {
        "HKD": {
          "_from": "1895-02-02"
        }
      }
    ],
    "HM";
    [
      {
```



```

        "AUD": {
            "_from": "1967-02-16"
        }
    ],
    "HN";
    [
        {
            "HNL": {
                "_from": "1926-04-03"
            }
        }
    ],
    "HR";
    [
        {
            "YUD": {
                "_from": "1966-01-01",
                "_to": "1990-01-01"
            }
        },
        {
            "YUN": {
                "_from": "1990-01-01",
                "_to": "1991-12-23"
            }
        },
        {
            "HRD": {
                "_from": "1991-12-23",
                "_to": "1995-01-01"
            }
        },
        {
            "HRK": {
                "_from": "1994-05-30"
            }
        }
    ],
    "HT";
    [
        {
            "HTG": {
                "_from": "1872-08-26"
            }
        },
        {
            "USD": {
                "_from": "1915-01-01"
            }
        }
    ],
    "HU";
    [
        {
            "HUF": {
                "_from": "1946-07-23"
            }
        }
    ]

```

```
    }  
  },  
  ],  
  "IC";  
  [  
    {  
      "EUR": {  
        "_from": "1999-01-01"  
      }  
    }  
  ],  
  "ID";  
  [  
    {  
      "IDR": {  
        "_from": "1965-12-13"  
      }  
    }  
  ],  
  "IE";  
  [  
    {  
      "GBP": {  
        "_from": "1800-01-01",  
        "_to": "1922-01-01"  
      }  
    },  
    {  
      "IEP": {  
        "_from": "1922-01-01",  
        "_to": "2002-02-09"  
      }  
    },  
    {  
      "EUR": {  
        "_from": "1999-01-01"  
      }  
    }  
  ],  
  "IL";  
  [  
    {  
      "ILP": {  
        "_from": "1948-08-16",  
        "_to": "1980-02-22"  
      }  
    },  
    {  
      "ILR": {  
        "_from": "1980-02-22",  
        "_to": "1985-09-04"  
      }  
    },  
    {  
      "ILS": {  
        "_from": "1985-09-04"  
      }  
    }  
  ]  
}
```

```
    },
    [
      {
        "GBP": {
          "_from": "1840-01-03"
        }
      }
    ],
    "IN";
    [
      {
        "INR": {
          "_from": "1835-08-17"
        }
      }
    ],
    "IO";
    [
      {
        "USD": {
          "_from": "1965-11-08"
        }
      }
    ],
    "IQ";
    [
      {
        "EGP": {
          "_from": "1920-11-11",
          "_to": "1931-04-19"
        }
      },
      {
        "INR": {
          "_from": "1920-11-11",
          "_to": "1931-04-19"
        }
      },
      {
        "IQD": {
          "_from": "1931-04-19"
        }
      }
    ],
    "IR";
    [
      {
        "IRR": {
          "_from": "1932-05-13"
        }
      }
    ],
    "IS";
    [
      {
```

```
        "DKK": {
            "_from": "1873-05-27",
            "_to": "1918-12-01"
        }
    },
    {
        "ISJ": {
            "_from": "1918-12-01",
            "_to": "1981-01-01"
        }
    },
    {
        "ISK": {
            "_from": "1981-01-01"
        }
    }
],
"IT";
[
    {
        "ITL": {
            "_from": "1862-08-24",
            "_to": "2002-02-28"
        }
    },
    {
        "EUR": {
            "_from": "1999-01-01"
        }
    }
],
"JE";
[
    {
        "GBP": {
            "_from": "1837-01-01"
        }
    }
],
"JM";
[
    {
        "JMD": {
            "_from": "1969-09-08"
        }
    }
],
"JO";
[
    {
        "JOD": {
            "_from": "1950-07-01"
        }
    }
],
"JP";
[
```

```
{
  "JPY": {
    "_from": "1871-06-01"
  }
},
"KE";
[
  {
    "KES": {
      "_from": "1966-09-14"
    }
  },
"KG";
[
  {
    "SUR": {
      "_from": "1961-01-01",
      "_to": "1991-12-25"
    }
  },
  {
    "RUR": {
      "_from": "1991-12-25",
      "_to": "1993-05-10"
    }
  },
  {
    "KGS": {
      "_from": "1993-05-10"
    }
  }
],
"KH";
[
  {
    "KHR": {
      "_from": "1980-03-20"
    }
  }
],
"KI";
[
  {
    "AUD": {
      "_from": "1966-02-14"
    }
  }
],
"KM";
[
  {
    "KMF": {
      "_from": "1975-07-06"
    }
  }
]
```

```
],
  "KN";
[
  {
    "XCD": {
      "_from": "1965-10-06"
    }
  }
],
  "KP";
[
  {
    "KPW": {
      "_from": "1959-04-17"
    }
  }
],
  "KR";
[
  {
    "KRO": {
      "_from": "1945-08-15",
      "_to": "1953-02-15"
    }
  },
  {
    "KRH": {
      "_from": "1953-02-15",
      "_to": "1962-06-10"
    }
  },
  {
    "KRW": {
      "_from": "1962-06-10"
    }
  }
],
  "KW";
[
  {
    "KWD": {
      "_from": "1961-04-01"
    }
  }
],
  "KY";
[
  {
    "JMD": {
      "_from": "1969-09-08",
      "_to": "1971-01-01"
    }
  },
  {
    "KYD": {
      "_from": "1971-01-01"
    }
  }
]
```

```
    },
    "KZ";
    [
      {
        "KZT": {
          "_from": "1993-11-05"
        }
      }
    ],
    "LA";
    [
      {
        "LAK": {
          "_from": "1979-12-10"
        }
      }
    ],
    "LB";
    [
      {
        "LBP": {
          "_from": "1948-02-02"
        }
      }
    ],
    "LC";
    [
      {
        "XCD": {
          "_from": "1965-10-06"
        }
      }
    ],
    "LI";
    [
      {
        "CHF": {
          "_from": "1921-02-01"
        }
      }
    ],
    "LK";
    [
      {
        "LKR": {
          "_from": "1978-05-22"
        }
      }
    ],
    "LR";
    [
      {
        "LRD": {
          "_from": "1944-01-01"
        }
      }
    ]
  ]
}
```

```
],
  "LS";
[
  {
    "ZAR": {
      "_from": "1961-02-14"
    }
  },
  {
    "LSL": {
      "_from": "1980-01-22"
    }
  }
],
  "LT";
[
  {
    "SUR": {
      "_from": "1961-01-01",
      "_to": "1992-10-01"
    }
  },
  {
    "LTT": {
      "_from": "1992-10-01",
      "_to": "1993-06-25"
    }
  },
  {
    "LTL": {
      "_from": "1993-06-25",
      "_to": "2014-12-31"
    }
  },
  {
    "EUR": {
      "_from": "2015-01-01"
    }
  }
],
  "LU";
[
  {
    "LUF": {
      "_from": "1944-09-04",
      "_to": "2002-02-28"
    }
  },
  {
    "LUC": {
      "_tender": "false",
      "_from": "1970-01-01",
      "_to": "1990-03-05"
    }
  },
  {
    "LUL": {
```



```
        "_tender": "false",
        "_from": "1970-01-01",
        "_to": "1990-03-05"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "LV";
[
  {
    "SUR": {
      "_from": "1961-01-01",
      "_to": "1992-07-20"
    }
  },
  {
    "LVR": {
      "_from": "1992-05-07",
      "_to": "1993-10-17"
    }
  },
  {
    "LVL": {
      "_from": "1993-06-28",
      "_to": "2013-12-31"
    }
  },
  {
    "EUR": {
      "_from": "2014-01-01"
    }
  }
],
  "LY";
[
  {
    "LYD": {
      "_from": "1971-09-01"
    }
  }
],
  "MA";
[
  {
    "MAF": {
      "_from": "1881-01-01",
      "_to": "1959-10-17"
    }
  },
  {
    "MAD": {
      "_from": "1959-10-17"
    }
  }
]
```

```

    },
    "MC";
    [
      {
        "FRF": {
          "_from": "1960-01-01",
          "_to": "2002-02-17"
        }
      },
      {
        "MCF": {
          "_from": "1960-01-01",
          "_to": "2002-02-17"
        }
      },
      {
        "EUR": {
          "_from": "1999-01-01"
        }
      }
    ],
    "MD";
    [
      {
        "MDC": {
          "_from": "1992-06-01",
          "_to": "1993-11-29"
        }
      },
      {
        "MDL": {
          "_from": "1993-11-29"
        }
      }
    ],
    "ME";
    [
      {
        "YUM": {
          "_from": "1994-01-24",
          "_to": "2002-05-15"
        }
      },
      {
        "DEM": {
          "_from": "1999-10-02",
          "_to": "2002-05-15"
        }
      },
      {
        "EUR": {
          "_from": "2002-01-01"
        }
      }
    ],
    "MF";

```

```

[
  {
    "FRF": {
      "_from": "1960-01-01",
      "_to": "2002-02-17"
    }
  },
  {
    "EUR": {
      "_from": "1999-01-01"
    }
  }
],
"MG";
[
  {
    "MGF": {
      "_from": "1963-07-01",
      "_to": "2004-12-31"
    }
  },
  {
    "MGA": {
      "_from": "1983-11-01"
    }
  }
],
"MH";
[
  {
    "USD": {
      "_from": "1944-01-01"
    }
  }
],
"MK";
[
  {
    "MKN": {
      "_from": "1992-04-26",
      "_to": "1993-05-20"
    }
  },
  {
    "MKD": {
      "_from": "1993-05-20"
    }
  }
],
"ML";
[
  {
    "XOF": {
      "_from": "1958-11-24",
      "_to": "1962-07-02"
    }
  }
],

```

```
{
  "MLF": {
    "_from": "1962-07-02",
    "_to": "1984-08-31"
  },
  {
    "XOF": {
      "_from": "1984-06-01"
    }
  },
],
"MM";
[
  {
    "BUK": {
      "_from": "1952-07-01",
      "_to": "1989-06-18"
    }
  },
  {
    "MMK": {
      "_from": "1989-06-18"
    }
  }
],
"MN";
[
  {
    "MNT": {
      "_from": "1915-03-01"
    }
  }
],
"MO";
[
  {
    "MOP": {
      "_from": "1901-01-01"
    }
  }
],
"MP";
[
  {
    "USD": {
      "_from": "1944-01-01"
    }
  }
],
"MQ";
[
  {
    "FRF": {
      "_from": "1960-01-01",
      "_to": "2002-02-17"
    }
  }
]
```

```
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "MR";
  [
    {
      "XOF": {
        "_from": "1958-11-28",
        "_to": "1973-06-29"
      }
    },
    {
      "MRO": {
        "_from": "1973-06-29"
      }
    }
  ],
  "MS";
  [
    {
      "XCD": {
        "_from": "1967-02-27"
      }
    }
  ],
  "MT";
  [
    {
      "MTP": {
        "_from": "1914-08-13",
        "_to": "1968-06-07"
      }
    },
    {
      "MTL": {
        "_from": "1968-06-07",
        "_to": "2008-01-31"
      }
    }
  ],
  {
    "EUR": {
      "_from": "2008-01-01"
    }
  }
],
  "MU";
  [
    {
      "MUR": {
        "_from": "1934-04-01"
      }
    }
  ],
  ],
```

```
        "MV";
    [
        {
            "MVR": {
                "_from": "1981-07-01"
            }
        }
    ],
    "MW";
    [
        {
            "MWK": {
                "_from": "1971-02-15"
            }
        }
    ],
    "MX";
    [
        {
            "MXV": {
                "_tender": "false"
            }
        },
        {
            "MXP": {
                "_from": "1822-01-01",
                "_to": "1992-12-31"
            }
        },
        {
            "MXN": {
                "_from": "1993-01-01"
            }
        }
    ],
    "MY";
    [
        {
            "MYR": {
                "_from": "1963-09-16"
            }
        }
    ],
    "MZ";
    [
        {
            "MZE": {
                "_from": "1975-06-25",
                "_to": "1980-06-16"
            }
        },
        {
            "MZM": {
                "_from": "1980-06-16",
                "_to": "2006-12-31"
            }
        }
    ],
    ],
```

```
    {
      "MZN": {
        "_from": "2006-07-01"
      }
    },
    "NA";
    [
      {
        "ZAR": {
          "_from": "1961-02-14"
        }
      },
      {
        "NAD": {
          "_from": "1993-01-01"
        }
      }
    ],
    "NC";
    [
      {
        "XPF": {
          "_from": "1985-01-01"
        }
      }
    ],
    "NE";
    [
      {
        "XOF": {
          "_from": "1958-12-19"
        }
      }
    ],
    "NF";
    [
      {
        "AUD": {
          "_from": "1966-02-14"
        }
      }
    ],
    "NG";
    [
      {
        "NGN": {
          "_from": "1973-01-01"
        }
      }
    ],
    "NI";
    [
      {
        "NIC": {
          "_from": "1988-02-15",
          "_to": "1991-04-30"
        }
      }
    ]
  ]
}
```

```
    },
    {
      "NIO": {
        "_from": "1991-04-30"
      }
    }
  ],
  "NL";
  [
    {
      "NLG": {
        "_from": "1813-01-01",
        "_to": "2002-02-28"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "NO";
  [
    {
      "SEK": {
        "_from": "1873-05-27",
        "_to": "1905-06-07"
      }
    },
    {
      "NOK": {
        "_from": "1905-06-07"
      }
    }
  ],
  "NP";
  [
    {
      "INR": {
        "_from": "1870-01-01",
        "_to": "1966-10-17"
      }
    },
    {
      "NPR": {
        "_from": "1933-01-01"
      }
    }
  ],
  "NR";
  [
    {
      "AUD": {
        "_from": "1966-02-14"
      }
    }
  ]
}
```



```

    ],
    "NU";
    [
      {
        "NZD": {
          "_from": "1967-07-10"
        }
      }
    ],
    "NZ";
    [
      {
        "NZD": {
          "_from": "1967-07-10"
        }
      }
    ],
    "OM";
    [
      {
        "OMR": {
          "_from": "1972-11-11"
        }
      }
    ],
    "PA";
    [
      {
        "PAB": {
          "_from": "1903-11-04"
        }
      },
      {
        "USD": {
          "_from": "1903-11-18"
        }
      }
    ],
    "PE";
    [
      {
        "PES": {
          "_from": "1863-02-14",
          "_to": "1985-02-01"
        }
      },
      {
        "PEI": {
          "_from": "1985-02-01",
          "_to": "1991-07-01"
        }
      },
      {
        "PEN": {
          "_from": "1991-07-01"
        }
      }
    ]
  ],

```

```
],
  "PF";
[
  {
    "XPF": {
      "_from": "1945-12-26"
    }
  }
],
  "PG";
[
  {
    "AUD": {
      "_from": "1966-02-14",
      "_to": "1975-09-16"
    }
  },
  {
    "PGK": {
      "_from": "1975-09-16"
    }
  }
],
  "PH";
[
  {
    "PHP": {
      "_from": "1946-07-04"
    }
  }
],
  "PK";
[
  {
    "INR": {
      "_from": "1835-08-17",
      "_to": "1947-08-15"
    }
  },
  {
    "PKR": {
      "_from": "1948-04-01"
    }
  }
],
  "PL";
[
  {
    "PLZ": {
      "_from": "1950-10-28",
      "_to": "1994-12-31"
    }
  },
  {
    "PLN": {
      "_from": "1995-01-01"
    }
  }
]
```

```

    },
    "PM";
    [
      {
        "FRF": {
          "_from": "1972-12-21",
          "_to": "2002-02-17"
        }
      },
      {
        "EUR": {
          "_from": "1999-01-01"
        }
      }
    ],
    "PN";
    [
      {
        "NZD": {
          "_from": "1969-01-13"
        }
      }
    ],
    "PR";
    [
      {
        "ESP": {
          "_from": "1800-01-01",
          "_to": "1898-12-10"
        }
      },
      {
        "USD": {
          "_from": "1898-12-10"
        }
      }
    ],
    "PS";
    [
      {
        "JOD": {
          "_from": "1950-07-01",
          "_to": "1967-06-01"
        }
      },
      {
        "ILP": {
          "_from": "1967-06-01",
          "_to": "1980-02-22"
        }
      },
      {
        "ILS": {
          "_from": "1985-09-04"
        }
      }
    ],

```

```
    {
      "JOD": {
        "_from": "1996-02-12"
      }
    },
    "PT";
    [
      {
        "PTE": {
          "_from": "1911-05-22",
          "_to": "2002-02-28"
        }
      },
      {
        "EUR": {
          "_from": "1999-01-01"
        }
      }
    ],
    "PW";
    [
      {
        "USD": {
          "_from": "1944-01-01"
        }
      }
    ],
    "PY";
    [
      {
        "PYG": {
          "_from": "1943-11-01"
        }
      }
    ],
    "QA";
    [
      {
        "QAR": {
          "_from": "1973-05-19"
        }
      }
    ],
    "RE";
    [
      {
        "FRF": {
          "_from": "1975-01-01",
          "_to": "2002-02-17"
        }
      },
      {
        "EUR": {
          "_from": "1999-01-01"
        }
      }
    ]
  ]
}
```

```
],
  "RO";
[
  {
    "ROL": {
      "_from": "1952-01-28",
      "_to": "2006-12-31"
    }
  },
  {
    "RON": {
      "_from": "2005-07-01"
    }
  }
],
  "RS";
[
  {
    "YUM": {
      "_from": "1994-01-24",
      "_to": "2002-05-15"
    }
  },
  {
    "CSD": {
      "_from": "2002-05-15",
      "_to": "2006-10-25"
    }
  },
  {
    "RSD": {
      "_from": "2006-10-25"
    }
  }
],
  "RU";
[
  {
    "RUR": {
      "_from": "1991-12-25",
      "_to": "1998-12-31"
    }
  },
  {
    "RUB": {
      "_from": "1999-01-01"
    }
  }
],
  "RW";
[
  {
    "RWF": {
      "_from": "1964-05-19"
    }
  }
],
```

```
        "SA";
    [
        {
            "SAR": {
                "_from": "1952-10-22"
            }
        }
    ],
    "SB";
    [
        {
            "AUD": {
                "_from": "1966-02-14",
                "_to": "1978-06-30"
            }
        },
        {
            "SBD": {
                "_from": "1977-10-24"
            }
        }
    ],
    "SC";
    [
        {
            "SCR": {
                "_from": "1903-11-01"
            }
        }
    ],
    "SD";
    [
        {
            "EGP": {
                "_from": "1889-01-19",
                "_to": "1958-01-01"
            }
        },
        {
            "GBP": {
                "_from": "1889-01-19",
                "_to": "1958-01-01"
            }
        },
        {
            "SDP": {
                "_from": "1957-04-08",
                "_to": "1998-06-01"
            }
        },
        {
            "SDD": {
                "_from": "1992-06-08",
                "_to": "2007-06-30"
            }
        }
    ],
    {
```

```
        "SDG": {
            "_from": "2007-01-10"
        }
    ],
    "SE";
    [
        {
            "SEK": {
                "_from": "1873-05-27"
            }
        }
    ],
    "SG";
    [
        {
            "MYR": {
                "_from": "1963-09-16",
                "_to": "1967-06-12"
            }
        },
        {
            "SGD": {
                "_from": "1967-06-12"
            }
        }
    ],
    "SH";
    [
        {
            "SHP": {
                "_from": "1917-02-15"
            }
        }
    ],
    "SI";
    [
        {
            "SIT": {
                "_from": "1992-10-07",
                "_to": "2007-01-14"
            }
        },
        {
            "EUR": {
                "_from": "2007-01-01"
            }
        }
    ],
    "SJ";
    [
        {
            "NOK": {
                "_from": "1905-06-07"
            }
        }
    ],
    ],
```

```
    "SK";
    [
      {
        "CSK": {
          "_from": "1953-06-01",
          "_to": "1992-12-31"
        }
      },
      {
        "SKK": {
          "_from": "1992-12-31",
          "_to": "2009-01-01"
        }
      },
      {
        "EUR": {
          "_from": "2009-01-01"
        }
      }
    ],
    "SL";
    [
      {
        "GBP": {
          "_from": "1808-11-30",
          "_to": "1966-02-04"
        }
      },
      {
        "SLL": {
          "_from": "1964-08-04"
        }
      }
    ],
    "SM";
    [
      {
        "ITL": {
          "_from": "1865-12-23",
          "_to": "2001-02-28"
        }
      },
      {
        "EUR": {
          "_from": "1999-01-01"
        }
      }
    ],
    "SN";
    [
      {
        "XOF": {
          "_from": "1959-04-04"
        }
      }
    ],
    "SO";
```



```

[
  {
    "SOS": {
      "_from": "1960-07-01"
    }
  },
  "SR";
  [
    {
      "NLG": {
        "_from": "1815-11-20",
        "_to": "1940-05-10"
      }
    },
    {
      "SRG": {
        "_from": "1940-05-10",
        "_to": "2003-12-31"
      }
    },
    {
      "SRD": {
        "_from": "2004-01-01"
      }
    }
  ],
  "SS";
  [
    {
      "SDG": {
        "_from": "2007-01-10",
        "_to": "2011-09-01"
      }
    },
    {
      "SSP": {
        "_from": "2011-07-18"
      }
    }
  ],
  "ST";
  [
    {
      "STD": {
        "_from": "1977-09-08"
      }
    }
  ],
  "SU";
  [
    {
      "SUR": {
        "_from": "1961-01-01",
        "_to": "1991-12-25"
      }
    }
  ]
]

```

```
],
  "SV";
[
  {
    "SVC": {
      "_from": "1919-11-11",
      "_to": "2001-01-01"
    }
  },
  {
    "USD": {
      "_from": "2001-01-01"
    }
  }
],
  "SX";
[
  {
    "ANG": {
      "_from": "2010-10-10"
    }
  }
],
  "SY";
[
  {
    "SYP": {
      "_from": "1948-01-01"
    }
  }
],
  "SZ";
[
  {
    "SZL": {
      "_from": "1974-09-06"
    }
  }
],
  "TA";
[
  {
    "GBP": {
      "_from": "1938-01-12"
    }
  }
],
  "TC";
[
  {
    "USD": {
      "_from": "1969-09-08"
    }
  }
],
  "TD";
[
```

```
{
  "XAF": {
    "_from": "1993-01-01"
  }
},
"TF";
[
  {
    "FRF": {
      "_from": "1959-01-01",
      "_to": "2002-02-17"
    }
  },
  {
    "EUR": {
      "_from": "1999-01-01"
    }
  }
],
"TG";
[
  {
    "XOF": {
      "_from": "1958-11-28"
    }
  }
],
"TH";
[
  {
    "THB": {
      "_from": "1928-04-15"
    }
  }
],
"TJ";
[
  {
    "RUR": {
      "_from": "1991-12-25",
      "_to": "1995-05-10"
    }
  },
  {
    "TJR": {
      "_from": "1995-05-10",
      "_to": "2000-10-25"
    }
  },
  {
    "TJS": {
      "_from": "2000-10-26"
    }
  }
],
"TK";
```

```
[
  {
    "NZD": {
      "_from": "1967-07-10"
    }
  },
  "TL";
[
  {
    "TPE": {
      "_from": "1959-01-02",
      "_to": "2002-05-20"
    }
  },
  {
    "IDR": {
      "_from": "1975-12-07",
      "_to": "2002-05-20"
    }
  },
  {
    "USD": {
      "_from": "1999-10-20"
    }
  }
],
"TM";
[
  {
    "SUR": {
      "_from": "1961-01-01",
      "_to": "1991-12-25"
    }
  },
  {
    "RUR": {
      "_from": "1991-12-25",
      "_to": "1993-11-01"
    }
  },
  {
    "TMM": {
      "_from": "1993-11-01",
      "_to": "2009-01-01"
    }
  },
  {
    "TMT": {
      "_from": "2009-01-01"
    }
  }
],
"TN";
[
  {
    "TND": {
```

```
        "_from": "1958-11-01"
      }
    },
    ],
    "TO";
  [
    {
      "TOP": {
        "_from": "1966-02-14"
      }
    }
  ],
  "TP";
  [
    {
      "TPE": {
        "_from": "1959-01-02",
        "_to": "2002-05-20"
      }
    },
    {
      "IDR": {
        "_from": "1975-12-07",
        "_to": "2002-05-20"
      }
    }
  ],
  "TR";
  [
    {
      "TRL": {
        "_from": "1922-11-01",
        "_to": "2005-12-31"
      }
    },
    {
      "TRY": {
        "_from": "2005-01-01"
      }
    }
  ],
  "TT";
  [
    {
      "TTD": {
        "_from": "1964-01-01"
      }
    }
  ],
  "TV";
  [
    {
      "AUD": {
        "_from": "1966-02-14"
      }
    }
  ],
  ],
```

```
    "TW";
    [
      {
        "TWD": {
          "_from": "1949-06-15"
        }
      }
    ],
    "TZ";
    [
      {
        "TZS": {
          "_from": "1966-06-14"
        }
      }
    ],
    "UA";
    [
      {
        "SUR": {
          "_from": "1961-01-01",
          "_to": "1991-12-25"
        }
      },
      {
        "RUR": {
          "_from": "1991-12-25",
          "_to": "1992-11-13"
        }
      },
      {
        "UAK": {
          "_from": "1992-11-13",
          "_to": "1993-10-17"
        }
      },
      {
        "UAH": {
          "_from": "1996-09-02"
        }
      }
    ],
    "UG";
    [
      {
        "UGS": {
          "_from": "1966-08-15",
          "_to": "1987-05-15"
        }
      },
      {
        "UGX": {
          "_from": "1987-05-15"
        }
      }
    ],
    "UM";
```

```
[
  {
    "USD": {
      "_from": "1944-01-01"
    }
  },
  "US";
[
  {
    "USN": {
      "_tender": "false"
    }
  },
  {
    "USS": {
      "_tender": "false",
      "_to": "2014-03-01"
    }
  },
  {
    "USD": {
      "_from": "1792-01-01"
    }
  }
],
"UY";
[
  {
    "UYI": {
      "_tender": "false"
    }
  },
  {
    "UYP": {
      "_from": "1975-07-01",
      "_to": "1993-03-01"
    }
  },
  {
    "UYU": {
      "_from": "1993-03-01"
    }
  }
],
"UZ";
[
  {
    "UZS": {
      "_from": "1994-07-01"
    }
  }
],
"VA";
[
  {
    "ITL": {
```

```

        "_from": "1870-10-19",
        "_to": "2002-02-28"
      }
    },
    {
      "EUR": {
        "_from": "1999-01-01"
      }
    }
  ],
  "VC";
  [
    {
      "XCD": {
        "_from": "1965-10-06"
      }
    }
  ],
  "VE";
  [
    {
      "VEB": {
        "_from": "1871-05-11",
        "_to": "2008-06-30"
      }
    },
    {
      "VEF": {
        "_from": "2008-01-01"
      }
    }
  ],
  "VG";
  [
    {
      "USD": {
        "_from": "1833-01-01"
      }
    },
    {
      "GBP": {
        "_from": "1833-01-01",
        "_to": "1959-01-01"
      }
    }
  ],
  "VI";
  [
    {
      "USD": {
        "_from": "1837-01-01"
      }
    }
  ],
  "VN";
  [
    {

```



```
        "VNN": {
          "_from": "1978-05-03",
          "_to": "1985-09-14"
        },
      ],
      {
        "VND": {
          "_from": "1985-09-14"
        }
      }
    ],
    "VU";
  [
    {
      "VUV": {
        "_from": "1981-01-01"
      }
    }
  ],
  "WF";
  [
    {
      "XPF": {
        "_from": "1961-07-30"
      }
    }
  ],
  "WS";
  [
    {
      "WST": {
        "_from": "1967-07-10"
      }
    }
  ],
  "XK";
  [
    {
      "YUM": {
        "_from": "1994-01-24",
        "_to": "1999-09-30"
      }
    },
    {
      "DEM": {
        "_from": "1999-09-01",
        "_to": "2002-03-09"
      }
    },
    {
      "EUR": {
        "_from": "2002-01-01"
      }
    }
  ],
  "YD";
  [
```

```

        {
            "YDD": {
                "_from": "1965-04-01",
                "_to": "1996-01-01"
            }
        },
        "YE";
    [
        {
            "YER": {
                "_from": "1990-05-22"
            }
        },
        "YT";
    [
        {
            "KMF": {
                "_from": "1975-01-01",
                "_to": "1976-02-23"
            }
        },
        {
            "FRF": {
                "_from": "1976-02-23",
                "_to": "2002-02-17"
            }
        },
        {
            "EUR": {
                "_from": "1999-01-01"
            }
        }
    ],
    "YU";
    [
        {
            "YUD": {
                "_from": "1966-01-01",
                "_to": "1990-01-01"
            }
        },
        {
            "YUN": {
                "_from": "1990-01-01",
                "_to": "1992-07-24"
            }
        },
        {
            "YUM": {
                "_from": "1994-01-24",
                "_to": "2002-05-15"
            }
        }
    ],
    "ZA";

```

```
[
  {
    "ZAR": {
      "_from": "1961-02-14"
    }
  },
  {
    "ZAL": {
      "_tender": "false",
      "_from": "1985-09-01",
      "_to": "1995-03-13"
    }
  }
],
"ZM";
[
  {
    "ZMK": {
      "_from": "1968-01-16",
      "_to": "2013-01-01"
    }
  },
  {
    "ZMW": {
      "_from": "2013-01-01"
    }
  }
],
"ZR";
[
  {
    "ZRZ": {
      "_from": "1971-10-27",
      "_to": "1993-11-01"
    }
  },
  {
    "ZRN": {
      "_from": "1993-11-01",
      "_to": "1998-07-31"
    }
  }
],
"ZW";
[
  {
    "RHD": {
      "_from": "1970-02-17",
      "_to": "1980-04-18"
    }
  },
  {
    "ZWD": {
      "_from": "1980-04-18",
      "_to": "2008-08-01"
    }
  }
],
```

```

        {
            "ZWR": {
                "_from": "2008-08-01",
                "_to": "2009-02-02"
            }
        },
        {
            "ZWL": {
                "_from": "2009-02-02",
                "_to": "2009-04-12"
            }
        },
        {
            "USD": {
                "_from": "2009-04-12"
            }
        }
    ],
    "ZZ";
    [
        {
            "XAG": {
                "_tender": "false"
            }
        },
        {
            "XAU": {
                "_tender": "false"
            }
        },
        {
            "XBA": {
                "_tender": "false"
            }
        },
        {
            "XBB": {
                "_tender": "false"
            }
        },
        {
            "XBC": {
                "_tender": "false"
            }
        },
        {
            "XBD": {
                "_tender": "false"
            }
        },
        {
            "XDR": {
                "_tender": "false"
            }
        },
        {
            "XPD": {

```

```

        "_tender": "false"
    },
    {
        "XPT": {
            "_tender": "false"
        }
    },
    {
        "XSU": {
            "_tender": "false"
        }
    },
    {
        "XTS": {
            "_tender": "false"
        }
    },
    {
        "XUA": {
            "_tender": "false"
        }
    },
    {
        "XXX": {
            "_tender": "false"
        }
    },
    {
        "XRE": {
            "_tender": "false",
            "_to": "1999-11-30"
        }
    },
    {
        "XFU": {
            "_tender": "false",
            "_to": "2013-11-30"
        }
    },
    {
        "XFO": {
            "_tender": "false",
            "_from": "1930-01-01",
            "_to": "2003-04-01"
        }
    }
}
];
}
}
}

```

INDEX.JSX

```
import * as React from "react";
```

```

import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as currencyData from './currencyData.json';
import * as numbers from './numbers.json';
import * as currencies from './currencies.json';
loadCldr(numberingSystems, currencyData, numbers, currencies);
L10n.load({
    'de': {
        'numerictextbox': { incrementTitle: 'Wert erhöhen', decrementTitle:
'Dekrementwert' }
    }
});
// initializes NumericTextBox component
// sets `German` culture using the culture value 'de'
// sets the 'EUR' currency format
function App() {
    return (<NumericTextBoxComponent locale="de" currency="EUR" format="c2"
value={100}/>);
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { loadCldr,L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as currencyData from './currencyData.json';
import * as numbers from './numbers.json';
import * as currencies from './currencies.json';
loadCldr(numberingSystems, currencyData, numbers, currencies);
L10n.load({
    'de': {
        'numerictextbox': { incrementTitle: 'Wert erhöhen', decrementTitle:
'Dekrementwert' }
    }
});
// initializes NumericTextBox component
// sets `German` culture using the culture value 'de'
// sets the 'EUR' currency format
function App() {
    return (
        <NumericTextBoxComponent
            locale="de"
            currency="EUR"
            format="c2"
            value={100}
        />
    );
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));

```

NUMBERINGSYSTEMS.JSX

```

{
  "supplemental";
  {
    "version";
    {
      "_number";
      "$Revision: 12732 $",
      "_unicodeVersion";
      "9.0.0",
      "_cldrVersion";
      "31";
    }
  }
  "numberingSystems";
  {
    "adlm";
    {
      "_digits";
      "ᲠᲡᲢᲣᲤᲥᲦᲧᲨᲩ",
      "_type";
      "numeric";
    }
    "ahom";
    {
      "_digits";
      "ᩌᩍᩎᩏᩐᩑᩒᩓᩔᩕᩖ",
      "_type";
      "numeric";
    }
    "arab";
    {
      "_digits";
      "٠١٢٣٤٥٦٧٨٩",
      "_type";
      "numeric";
    }
    "arabext";
    {
      "_digits";
      "٠١٢٣٤٥٦٧٨٩",
      "_type";
      "numeric";
    }
    "armn";
    {
      "_rules";
      "armenian-upper",
      "_type";
      "algorithmic";
    }
    "armnlow";
    {
      "_rules";
      "armenian-lower",

```

```

        "_type";
        "algorithmic";
    }
    "bali";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "beng";
    {
        "_digits";
        "০১২৩৪৫৬৭৮৯",
        "_type";
        "numeric";
    }
    "bhks";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "brah";
    {
        "_digits";
        "·ᳵ᳜᳝᳞᳟᳚᳛᳠᳡",
        "_type";
        "numeric";
    }
    "cakm";
    {
        "_digits";
        "ᦅᦑᦒᦓᦔᦕᦖᦗ",
        "_type";
        "numeric";
    }
    "cham";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "cyrl";
    {
        "_rules";
        "cyrillic-lower",
        "_type";
        "algorithmic";
    }
    "deva";
    {
        "_digits";
        "०१२३४५६७८९",

```



```

        "_type";
        "numeric";
    }
    "ethi";
    {
        "_rules";
        "ethiopic",
        "_type";
        "algorithmic";
    }
    "fullwide";
    {
        "_digits";
        "0 1 2 3 4 5 6 7 8 9",
        "_type";
        "numeric";
    }
    "geor";
    {
        "_rules";
        "georgian",
        "_type";
        "algorithmic";
    }
    "grek";
    {
        "_rules";
        "greek-upper",
        "_type";
        "algorithmic";
    }
    "greklow";
    {
        "_rules";
        "greek-lower",
        "_type";
        "algorithmic";
    }
    "gujr";
    {
        "_digits";
        "૦૧૨૩૪૫૬૭૮૯",
        "_type";
        "numeric";
    }
    "guru";
    {
        "_digits";
        "੦੧੨੩੪੫੬੭੮੯",
        "_type";
        "numeric";
    }
    "hanidays";
    {
        "_rules";
        "zh/SpelloutRules/spellout-numbering-days",

```

```

        "_type";
        "algorithmic";
    }
    "hanidec";
    {
        "_digits";
        "〇一二三四五六七八九",
        "_type";
        "numeric";
    }
    "hans";
    {
        "_rules";
        "zh/SpelloutRules/spellout-cardinal",
        "_type";
        "algorithmic";
    }
    "hansfin";
    {
        "_rules";
        "zh/SpelloutRules/spellout-cardinal-financial",
        "_type";
        "algorithmic";
    }
    "hant";
    {
        "_rules";
        "zh_Hant/SpelloutRules/spellout-cardinal",
        "_type";
        "algorithmic";
    }
    "hantfin";
    {
        "_rules";
        "zh_Hant/SpelloutRules/spellout-cardinal-financial",
        "_type";
        "algorithmic";
    }
    "hebr";
    {
        "_rules";
        "hebrew",
        "_type";
        "algorithmic";
    }
    "hmng";
    {
        "_digits";
        "□□□□□□□□□□",
        "_type";
        "numeric";
    }
    "java";
    {
        "_digits";

```

Copyright © 2001 -2024 Syncfusion Inc.

```

        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "latn";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "lepc";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "limb";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "mathbold";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathdbl";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathmono";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathsanb";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathsans";
    {

```

```

    "_digits";
    "0123456789",
    "_type";
    "numeric";
}
"mlym";
{
    "_digits";
    "൦൧൨൩൪൫൬൭൮൯",
    "_type";
    "numeric";
}
"modi";
{
    "_digits";
    "□□□□□□□□□□",
    "_type";
    "numeric";
}
"mong";
{
    "_digits";
    "0᠑᠒᠓᠔᠕᠖᠗᠘᠙",
    "_type";
    "numeric";
}
"mroo";
{
    "_digits";
    "□□□□□□□□□□",
    "_type";
    "numeric";
}
"mtei";
{
    "_digits";
    "ႤႬႭᆑᆒᆓᆔᆕᆖᆗᆘᆙᆚᆛᆜᆝᆞᆟᆠᆡᆢᆣᆤᆥᆦᆧᆨᆩᆪᆫᆬᆭᆮᆯᆰᆱᆲᆳᆴᆵᆶᆷᆸᆹᆺᆻᆼᆽᆾᆿ",
    "_type";
    "numeric";
}
"mymr";
{
    "_digits";
    "၀၁၂၃၄၅၆၇၈",
    "_type";
    "numeric";
}
"mymrshan";
{
    "_digits";
    "0123456789",
    "_type";
    "numeric";
}
"mymrtlng";

```

```

    {
      "_digits";
      "᠐᠑᠒᠓᠔᠕᠖᠗᠘᠐",
      "_type";
      "numeric";
    }
    "newa";
    {
      "_digits";
      "᠐᠑᠒᠓᠔᠕᠖᠗᠘᠐",
      "_type";
      "numeric";
    }
    "nkoo";
    {
      "_digits";
      "᠐᠑᠒᠓᠔᠕᠖᠗᠘᠐",
      "_type";
      "numeric";
    }
    "olck";
    {
      "_digits";
      "᠐᠑᠒᠓᠔᠕᠖᠗᠘᠐",
      "_type";
      "numeric";
    }
    "orya";
    {
      "_digits";
      "᠐᠑᠒᠓᠔᠕᠖᠗᠘᠐",
      "_type";
      "numeric";
    }
    "osma";
    {
      "_digits";
      "᠐᠑᠒᠓᠔᠕᠖᠗᠘᠐",
      "_type";
      "numeric";
    }
    "roman";
    {
      "_rules";
      "roman-upper",
      "_type";
      "algorithmic";
    }
    "romanlow";
    {
      "_rules";
      "roman-lower",
      "_type";
      "algorithmic";
    }
  }

```

```

"saur";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"shrd";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"sind";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"sinh";
{
  "_digits";
  "⁂⁂⁂⁂⁂⁂⁂⁂⁂⁂",
  "_type";
  "numeric";
}
"sora";
{
  "_digits";
  "0⁂⁂⁂⁂⁂⁂⁂⁂",
  "_type";
  "numeric";
}
"sund";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"takr";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"talv";
{
  "_digits";
  "⁂⁂⁂⁂⁂⁂⁂⁂⁂⁂",
  "_type";
  "numeric";
}

```

```

    "taml";
    {
        "_rules";
        "tamil",
        "_type";
        "algorithmic";
    }
    "tamldec";
    {
        "_digits";
        "0௧௨௩௪௫௬௭௮௯",
        "_type";
        "numeric";
    }
    "telu";
    {
        "_digits";
        "౦౧౨౩౪౫౬౭౮౯",
        "_type";
        "numeric";
    }
    "thai";
    {
        "_digits";
        "๐๑๒๓๔๕๖๗๘๙",
        "_type";
        "numeric";
    }
    "tibtb";
    {
        "_digits";
        "༠༡༢༣༤༥༦༧༨༩",
        "_type";
        "numeric";
    }
    "tirh";
    {
        "_digits";
        "୦୧୨୩୪୫୬୭୮୯",
        "_type";
        "numeric";
    }
    "vair";
    {
        "_digits";
        "౦|౧౨౩౪౫౬౭౮౯",
        "_type";
        "numeric";
    }
    "wara";
    {
        "_digits";
        "ᳵᳶ᳷᳸᳹ᳺ᳻᳼᳽᳾᳿",
        "_type";
        "numeric";
    }
}

```



```
    }  
  }  
}
```

NUMBERS.JSX

```
{  
  "main";  
  {  
    "de";  
    {  
      "identity";  
      {  
        "version";  
        {  
          "_number";  
          "$Revision: 13259 $",  
          "_cldrVersion";  
          "31";  
        }  
        "language";  
        "de";  
      }  
    }  
    "numbers";  
    {  
      "defaultNumberingSystem";  
      "latn",  
      "otherNumberingSystems";  
      {  
        "native";  
        "latn";  
      }  
      "minimumGroupingDigits";  
      "1",  
      "symbols-numberSystem-latn";  
      {  
        "decimal";  
        ",",  
        "group";  
        ".",  
        "list";  
        ";",  
        "percentSign";  
        "%",  
        "plusSign";  
        "+",  
        "minusSign";  
        "-",  
        "exponential";  
        "E",  
        "superscriptingExponent";  
        ".",  
        "perMille";  
        "‰",  
        "infinity";  
        "∞",  
      }  
    }  
  }  
}
```

```

        "nan";
        "NaN",
        "timeSeparator";
        ":";
    }
    "decimalFormats-numberSystem-latn";
    {
        "standard";
        "#,##0.###",
        "long";
        {
            "decimalFormat";
            {
                "1000-count-one";
                "0 Tausend",
                "1000-count-other";
                "0 Tausend",
                "10000-count-one";
                "00 Tausend",
                "10000-count-other";
                "00 Tausend",
                "100000-count-one";
                "000 Tausend",
                "100000-count-other";
                "000 Tausend",
                "1000000-count-one";
                "0 Million",
                "1000000-count-other";
                "0 Millionen",
                "10000000-count-one";
                "00 Millionen",
                "10000000-count-other";
                "00 Millionen",
                "100000000-count-one";
                "000 Millionen",
                "100000000-count-other";
                "000 Millionen",
                "1000000000-count-one";
                "0 Milliarde",
                "1000000000-count-other";
                "0 Milliarden",
                "10000000000-count-one";
                "00 Milliarden",
                "10000000000-count-other";
                "00 Milliarden",
                "100000000000-count-one";
                "000 Milliarden",
                "100000000000-count-other";
                "000 Milliarden",
                "1000000000000-count-one";
                "0 Billion",
                "1000000000000-count-other";
                "0 Billionen",
                "10000000000000-count-one";
                "00 Billionen",
                "10000000000000-count-other";
                "00 Billionen",
            }
        }
    }

```

```

        "100000000000000-count-one";
        "000 Billionen",
        "100000000000000-count-other";
        "000 Billionen";
    }
}
"short";
{
    "decimalFormat";
    {
        "1000-count-one";
        "0",
        "1000-count-other";
        "0",
        "10000-count-one";
        "0",
        "10000-count-other";
        "0",
        "100000-count-one";
        "0",
        "100000-count-other";
        "0",
        "1000000-count-one";
        "0",
        "1000000-count-other";
        "0",
        "1000000-count-one";
        "0 Mio'. '",
        "1000000-count-other";
        "0 Mio'. '",
        "10000000-count-one";
        "00 Mio'. '",
        "10000000-count-other";
        "00 Mio'. '",
        "100000000-count-one";
        "000 Mio'. '",
        "100000000-count-other";
        "000 Mio'. '",
        "1000000000-count-one";
        "0 Mrd'. '",
        "1000000000-count-other";
        "0 Mrd'. '",
        "10000000000-count-one";
        "00 Mrd'. '",
        "10000000000-count-other";
        "00 Mrd'. '",
        "100000000000-count-one";
        "000 Mrd'. '",
        "100000000000-count-other";
        "000 Mrd'. '",
        "1000000000000-count-one";
        "0 Bio'. '",
        "1000000000000-count-other";
        "0 Bio'. '",
        "10000000000000-count-one";
        "00 Bio'. '",
        "10000000000000-count-other";
        "00 Bio'. '",
        "100000000000000-count-one";
        "000 Bio'. '",
        "100000000000000-count-other";
    }
}

```

```

        "000 Tsd'.' ¤";
    }
}
}
"scientificFormats-numberSystem-latn";
{
    "standard";
    "#E0";
}
"percentFormats-numberSystem-latn";
{
    "standard";
    "#,##0 %";
}
"currencyFormats-numberSystem-latn";
{
    "currencySpacing";
    {
        "beforeCurrency";
        {
            "currencyMatch";
            "[:^S:]",
            "surroundingMatch";
            "[:digit:]",
            "insertBetween";
            " ";
        }
        "afterCurrency";
        {
            "currencyMatch";
            "[:^S:]",
            "surroundingMatch";
            "[:digit:]",
            "insertBetween";
            " ";
        }
    }
}
"standard";
"#,##0.00 ¤",
"accounting";
"#,##0.00 ¤",
"short";
{
    "standard";
    {
        "1000-count-one";
        "0 Tsd'.' ¤",
        "1000-count-other";
        "0 Tsd'.' ¤",
        "10000-count-one";
        "00 Tsd'.' ¤",
        "10000-count-other";
        "00 Tsd'.' ¤",
        "100000-count-one";
        "000 Tsd'.' ¤",
        "100000-count-other";
        "000 Tsd'.' ¤",
    }
}

```

```

        "1000000-count-one";
        "0 Mio'.' ¤",
        "1000000-count-other";
        "0 Mio'.' ¤",
        "10000000-count-one";
        "00 Mio'.' ¤",
        "10000000-count-other";
        "00 Mio'.' ¤",
        "100000000-count-one";
        "000 Mio'.' ¤",
        "100000000-count-other";
        "000 Mio'.' ¤",
        "1000000000-count-one";
        "0 Mrd'.' ¤",
        "1000000000-count-other";
        "0 Mrd'.' ¤",
        "10000000000-count-one";
        "00 Mrd'.' ¤",
        "10000000000-count-other";
        "00 Mrd'.' ¤",
        "100000000000-count-one";
        "000 Mrd'.' ¤",
        "100000000000-count-other";
        "000 Mrd'.' ¤",
        "1000000000000-count-one";
        "0 Bio'.' ¤",
        "1000000000000-count-other";
        "0 Bio'.' ¤",
        "10000000000000-count-one";
        "00 Bio'.' ¤",
        "10000000000000-count-other";
        "00 Bio'.' ¤",
        "100000000000000-count-one";
        "000 Bio'.' ¤",
        "100000000000000-count-other";
        "000 Bio'.' ¤";
    }
}
"unitPattern-count-one";
"{0} {1}",
    "unitPattern-count-other";
"{0} {1}";
}
"miscPatterns-numberSystem-latn";
{
    "atLeast";
    "{0}+",
    "range";
    "{0}-{1}";
}
"minimalPairs";
{
    "pluralMinimalPairs";
    "{0} Tag",
    "pluralMinimalPairs";
    "{0} Tage",
    "other";
}

```

```

        "{0}. Abzweigung nach rechts nehmen";
    }
    }
    }
}

```

SETTINGS.JSX

```

{
    "file";
    "app/index.tsx",
    "line";
    -1;
}

```

Right to Left(RTL)

RTL provides an option to switch the text direction and layout of the NumericTextBox 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 NumericTextBox, set the [enableRtl](#) to true.

[Class-component]

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
    'ar-AE': {
        'numerictextbox': { incrementTitle: 'قيمة الزيادة', decrementTitle:
'قيمة تناقص' }
    }
});
// initializes NumericTextBox component
// sets `German` culture using the culture value 'de'
// sets the 'EUR' currency format
ReactDOM.render(<NumericTextBoxComponent locale='ar-AE' enableRtl='true'
floatLabelType='Auto' placeholder='أدخل القيمة' value={100}>
</NumericTextBoxComponent>, document.getElementById('numericContainer'));

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { loadCldr,L10n } from '@syncfusion/ej2-base';
L10n.load({
    'ar-AE': {
        'numerictextbox': { incrementTitle: 'قيمة الزيادة', decrementTitle:
'قيمة تناقص' }
    }
});
// initializes NumericTextBox component

```

```
// sets `German` culture using the culture value 'de'
// sets the 'EUR' currency format
ReactDOM.render(<NumericTextBoxComponent locale='ar-AE' enableRtl='true'
floatLabelType='Auto' placeholder='أدخل القيمة' value={100} >
</NumericTextBoxComponent>, document.getElementById('numericContainer'));
```

SETTINGS.JSX

```
{
  "file";
  "app/index.tsx",
  "line";
  -1;
}
```

[Functional-component]

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
  'ar-AE': {
    'numerictextbox': { incrementTitle: 'قيمة الزيادة', decrementTitle:
'قيمة تناقص' }
  }
});
// initializes NumericTextBox component
// sets `German` culture using the culture value 'de'
// sets the 'EUR' currency format
function App() {
  return (<NumericTextBoxComponent locale="ar-AE" enableRtl="true"
floatLabelType="Auto" placeholder="أدخل القيمة" value={100}/>);
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { loadCldr,L10n } from '@syncfusion/ej2-base';
L10n.load({
  'ar-AE': {
    'numerictextbox': { incrementTitle: 'قيمة الزيادة', decrementTitle:
'قيمة تناقص' }
  }
});
// initializes NumericTextBox component
// sets `German` culture using the culture value 'de'
// sets the 'EUR' currency format
function App() {
  return (
```

```

    <NumericTextBoxComponent
      locale="ar-AE"
      enableRtl="true"
      floatLabelType="Auto"
      placeholder="أدخل القيمة"
      value={100}
    />
  );
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));

```

SETTINGS.JSX

```

{
  "file";
  "app/index.tsx",
  "line";
  -1;
}

```

Accessibility in React Numerictextbox component

The Numerictextbox 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 Numerictextbox 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 NumericTextBox characterized with complete ARIA Accessibility support which helps to accessible by on-screen readers and other assistive technology devices. This component designed with the reference of the guidelines document given in [WAI ARAI Accessibility practices](#).

The NumericTextBox uses the `spinbutton` role and following ARIA properties to its element based on its state.

| Property | Functionality |

| --- | --- |

| aria-live | The `aria-live` attribute indicates the priority of updates to a live region |

| aria-valuemin | The `aria-valuemin` property specifies the minimum allowable range of the NumericTextBox. |

| aria-valuemax | The `aria-valuemax` property specifies the maximum allowable range of the NumericTextBox. |

| aria-disabled | The `aria-disabled` property indicates disabled state of the NumericTextBox. |

| aria-readonly | The `aria-readonly` property indicates the read-only state of the NumericTextBox. |

| aria-valuenow | The `aria-valuenow` property specifies the current value of the NumericTextBox. |

| aria-invalid | The `aria-invalid` property indicates that the user input is incorrect or not within acceptable ranges. |

| aria-label | The `aria-label` property indicates a string value that labels the NumericTextBox. |

Keyboard interaction

Keyboard interaction of the NumericTextBox component has been designed based on [WAI-ARIA Practices](#) described for the NumericTextBox and it is an alternative to mouse actions to interact with the NumericTextBox.

The below table shows shortcut keys and its corresponding usage.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| **Arrow Down** | **Increments the value.** |

| **Arrow Up** | **Decrements the value** |

[Class-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets value to the NumericTextBox
ReactDOM.render(<NumericTextBoxComponent value={10}/>,
document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets value to the NumericTextBox
ReactDOM.render(<NumericTextBoxComponent value={10} />
,document.getElementById('numericContainer'));
```

[Functional-component]

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    return <NumericTextBoxComponent value={10}/>;
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    return <NumericTextBoxComponent value={10} />;
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

React functional component in React Numerictextbox component

This section explains how to render the numeric textbox component in the functional component with react hooks methods. Please find the list of basic hook methods in the following table.

Hooks	Description
-----	-----
useState	useState is a Hook that allows you to define the state in the functional components. If you pass the initial state to this function, then it will return a state variable with value and function to update this state value. Here, useState is used to store the value of total amount field.
useEffect	useEffect is a Hook that allows you to execute code after rendering and re-rendering the component. Here, defined the set of rules for the price of the item, number of items, and total amount fields and initiated the form validation to those fields.
useRef	useRef is a Hook function that allows to directly create a reference to the DOM element in the functional component. Here, assigned the price of the item field reference for updating the predefined price on initial loading.
useReducer	useReducer is a Hook function that accepts a reducer function and an initial state as argument. It returns a state value and another function to dispatch an action. Here, dispatched the update action to update the price and count state values.

The following example shows how to render a simple numeric textbox with react hooks.

APP.JSX

```
import { useState, useEffect, useRef, useReducer } from 'react';
import * as React from 'react';
import { FormValidator } from '@syncfusion/ej2-inputs';
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
let formObject;
function App() {
  const [totalAmount, setTotalAmount] = useState('');
  const initialState = { price: '', count: '' };
  const priceValueRef = useRef(null);
  const reducer = (state, action) => {
    switch (action.type) {
      case 'update':
        return { ...state, [action.field]: action.value };
      default:
        return initialState;
    }
  };
  const [state, dispatch] = useReducer(reducer, initialState);
  const changeHandler = (event) => {
    setTotalAmount(event.value);
  };
}
```

```

};
const update = (field) => (event) => {
  //update action is dispatched and triggered state changes
  dispatch({ type: 'update', field, value: event.value });
};
useEffect(() => {
  priceValueRef.current.value = 100;
  const options = {
    // validation rules
    rules: {
      price: {
        required: [true, '* Please enter your price'],
      },
      count: {
        required: [true, '* Please enter your count'],
      },
      amount: {
        required: [true, '* Please enter your amount'],
      },
    },
  };
  // Initialize the form validator
  formObject = new FormValidator('#form1', options);
}, []);
return (<div>
  <div className="control_wrapper" id="control_wrapper">
    <h3 className="form-title">Purchase</h3>
    <div className="control_wrapper textbox-form">
      <form id="form1" method="post">
        <div className="form-group">
          <NumericTextBoxComponent ref={priceValueRef} name="price"
value={state.price} change={update('price')} placeholder="Price of the item"
floatLabelType="Auto" data-msg-containerid="errroForPrice"/>
          <div id="errroForPrice"/>
        </div>
        <div className="form-group">
          <NumericTextBoxComponent name="count" value={state.count}
change={update('count')} placeholder="Number of items" floatLabelType="Auto"
data-msg-containerid="errroForCount"/>
          <div id="errroForCount"/>
        </div>
        <div className="form-group">
          <NumericTextBoxComponent name="amount" value={totalAmount}
change={changeHandler} multiline="true" placeholder="Total amount"
floatLabelType="Auto" data-msg-containerid="errroForAmount"/>
          <div id="errroForAmount"/>
        </div>
      </form>
    </div>
    <br />
    <br />
  </div>
</div>);
}
export default App;

```

APP.TSX

```

import { useState, useEffect, useRef, useReducer } from 'react';
import * as React from 'react';
import { FormValidator } from '@syncfusion/ej2-inputs';
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
let formObject;
function App() {
  const [totalAmount, setTotalAmount] = useState('');
  const initialState = { price: '', count: '' };
  const priceValueRef = useRef(null);
  const reducer = (state, action) => {
    switch (action.type) {
      case 'update':
        return { ...state, [action.field]: action.value };
      default:
        return initialState;
    }
  };
  [state, dispatch] = useReducer(reducer, initialState);
  const changeHandler = (event) => {
    setTotalAmount(event.value);
  };
  const update = (field) => (event) => {
    //update action is dispatched and triggered state changes
    dispatch({ type: 'update', field, value: event.value });
  };
  useEffect(() => {
    priceValueRef.current.value = 100;
    const options = {
      // validation rules
      rules: {
        price: {
          required: [true, '* Please enter your price'],
        },
        count: {
          required: [true, '* Please enter your count'],
        },
        amount: {
          required: [true, '* Please enter your amount'],
        },
      },
    };
    // Initialize the form validator
    formObject = new FormValidator('#form1', options);
  }, []);
  return (
    <div>
      <div className="control_wrapper" id="control_wrapper">
        <h3 className="form-title">Purchase</h3>
        <div className="control_wrapper textbox-form">
          <form id="form1" method="post">
            <div className="form-group">
              <NumericTextBoxComponent
                ref={priceValueRef}
                name="price"
                value={state.price}

```

```

        change={update('price')}
        placeholder="Price of the item"
        floatLabelType="Auto"
        data-msg-containerid="errroForPrice"
      />
      <div id="errroForPrice" />
    </div>
    <div className="form-group">
      <NumericTextBoxComponent
        name="count"
        value={state.count}
        change={update('count')}
        placeholder="Number of items"
        floatLabelType="Auto"
        data-msg-containerid="errroForCount"
      />
      <div id="errroForCount" />
    </div>
    <div className="form-group">
      <NumericTextBoxComponent
        name="amount"
        value={totalAmount}
        change={changeHandler}
        multiline="true"
        placeholder="Total amount"
        floatLabelType="Auto"
        data-msg-containerid="errroForAmount"
      />
      <div id="errroForAmount" />
    </div>
  </form>
</div>
<br />
<br />
</div>
</div>
);
}
export default App;

```

Style appearance in React Numerictextbox component

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 NumericTextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
```

```
/ 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 {
```

```
height: 40px;
```

```
font-size: 20px;
```

```
}
```

```
,
```

Customizing the NumericTextBox icons

Use the following CSS to customize the Numeric TextBox icons

```
`css
```

```
/ To specify font size and background color /
```

```
.e-numeric.e-control-wrapper.e-input-group .e-input-group-icon {
```

```
font-size: 20px;
```

```
background-color: beige;
```

```
}
```

```
,
```

How To

Customize the ui appearance of the control in React Numerictextbox component

You can change the appearance of the NumericTextBox by adding custom `cssClass` to the component and enabling styles. Refer to the following example to change the appearance of the NumericTextBox.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
// initializes NumericTextBox component
export default class App extends React.Component {
  render() {
    return (<NumericTextBoxComponent id="sample" value={10} cssClass={'e-style'} placeholder="Enter value" floatLabelType={'Always'}/>);
  }
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
// initializes NumericTextBox component
export default class App extends React.Component<{}, {}> {
  render() {
    return (
      <NumericTextBoxComponent id="sample" value={10} cssClass={'e-style'}
        placeholder="Enter value" floatLabelType={'Always'}/>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('numericContainer'));
```

Customize the spin buttons up and down arrow in React Numerictextbox component

This section explains about how to change/customize spin up and down icons. You can customize spin button icons using `e-spin-up` and `e-spin-down` classes of those buttons.

You can override the default icons of `e-spin-up` and `e-spin-down` classes using the following CSS code snippets.

```
`css
.e-numeric .e-input-group-icon.e-spin-up:before {
content: "\e823";
color: rgba(0, 0, 0, 0.54);
}
.e-numeric .e-input-group-icon.e-spin-down:before {
content: "\e934";
color: rgba(0, 0, 0, 0.54);
}
`
```

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets value to the NumericTextBox
ReactDOM.render(<NumericTextBoxComponent value={10}/>,
document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets value to the NumericTextBox
ReactDOM.render(<NumericTextBoxComponent value={10}/>,
document.getElementById('numericContainer'));
```

Customize the step value and hide spin buttons in React Numerictextbox component

The spin buttons allow you to increase or decrease the value with the predefined [step](#) value. The visibility of spin buttons can be set using the [showSpinButton](#) property.

INDEX.JSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets the step value as '2' to increase/decrease the value by '2'
```



```
// sets the showSpinButton value as `false` to hide the spin buttons
ReactDOM.render(<NumericTextBoxComponent step={2} showSpinButton={false}
min={10} max={100} value={16}/>,
document.getElementById('numericContainer'));
```

INDEX.TSX

```
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
// sets the step value as '2' to increase/decrease the value by '2'
// sets the showSpinButton value as `false` to hide the spin buttons
ReactDOM.render(<NumericTextBoxComponent step={2} showSpinButton={false}
min={10} max={100} value={16} />
,document.getElementById('numericContainer'));
```

Maintain trailing zeros in numerictextbox in React Numerictextbox component

By default, trailing zeros disappear when the NumericTextBox gets focus. However, you can use the following sample to maintain the trailing zeros while focusing the NumericTextBox.

INDEX.JSX

```
{% raw %}
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  numericInstance;
  constructor(props) {
    super(props);
    this.onNumericFocus = this.onNumericFocus.bind(this);
  }
  onNumericFocus() {
    this.numericInstance.element.value =
this.numericInstance.formattedValue(this.numericInstance.decimals,
parseFloat(this.numericInstance.element.value));
  }
  render() {
    return (<NumericTextBoxComponent id="numeric" decimals={2}
format={"n2"} ref={(numeric) => { this.numericInstance = numeric; }}
value={10} change={this.onNumericFocus} onFocus={this.onNumericFocus}/>);
  }
}
ReactDOM.render(<App />, document.getElementById('numeric1'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  public numericInstance: any;
```

```

    constructor(props:any) {
        super(props);
        this.onNumericFocus = this.onNumericFocus.bind(this);
    }
    public onNumericFocus() : void {
        this.numericInstance.element.value =
this.numericInstance.formattedValue(this.numericInstance.decimals,
parseFloat(this.numericInstance.element.value));
    }
    public render() {
        return (
            <NumericTextBoxComponent id="numeric" decimals={2} format={"n2"}
ref={ (numeric) => { this.numericInstance = numeric as
NumericTextBoxComponent; }} value={10} change={this.onNumericFocus}
onFocus={this.onNumericFocus}/>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('numeric1'));

{% enddraw %}

```

Perform custom validation using form validator in React Numerictextbox component

This section explains how to perform custom validation on the NumericTextBox using FormValidator.

The NumericTextBox will be validated when the value changes or the user clicks the submit button.

Validation can be performed by adding custom validation in the rules collection of the FormValidator.

INDEX.JSX

```

{% raw %}
import { FormValidator } from '@syncfusion/ej2-inputs';
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
export default class App extends React.Component {
    numericInstance;
    formObject;
    constructor(props) {
        super(props);
        this.onChange = this.onChange.bind(this);
        this.onCreate = this.onCreate.bind(this);
    }
    onChange() {
        this.formObject.validate("numeric");
    }
    onCreate() {
        // checks the value of NumericTextbox and returns the corresponding
        boolean value
        const customFn = (args) => {
            if (this.numericInstance.value >= 10 &&
this.numericInstance.value <= 100) {
                return true;
            }
            else {

```

```

        return false;
    }
};
const options = {
    rules: {
        'numeric': { required: [true, "Number is required"] },
    }
};
// defines FormValidator to validate the NumericTextBox
this.formObject = new FormValidator('#form-element', options);
this.formObject.addRules('numeric', { range: [customFn, "Please enter
a number between 10 to 100"] });
const proxy = this;
document.getElementById('submit_btn').addEventListener('click', () =>
{
    proxy.formObject.validate('numeric');
    const ele = document.getElementById('numeric');
    // checks for incomplete value and alerts the formt submit
    if (ele.value !== "" && parseInt(ele.value, 10) >= 10 &&
parseInt(ele.value, 10) <= 100) {
        alert("Submitted");
    }
});
}
render() {
    return (<div>
        <NumericTextBoxComponent id="numeric" name='numeric'
placeholder="NumericTextbox" min={10} max={100} strictMode={false}
ref={(numeric) => { this.numericInstance = numeric; }} change={this.onChange}
created={this.onCreate} floatLabelType='Always'/>
        <label className='e-error' htmlFor='numeric'/>
        <button id="submit_btn" style={{ marginTop: '10px'
}}>Submit</button>
    </div>);
}
}
;
ReactDOM.render(<App />, document.getElementById('numericContainer'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
// initializes NumericTextBox component
export default class App extends React.Component<{}, {}> {
    public numericInstance: any;
    private formObject: FormValidator;
    constructor(props: any) {
        super(props);
        this.onChange = this.onChange.bind(this);
        this.onCreate = this.onCreate.bind(this);
    }
}

```

```

    public onChange() {
        this.formObject.validate("numeric");
    }
    public onCreate(): void {
        // checks the value of NumericTextbox and returns the corresponding
        boolean value
        const customFn: (args: { [key: string]: string }) => boolean = (args:
        { [key: string]: string }) => {
            if(this.numericInstance.value>=10 &&
            this.numericInstance.value<=100) {
                return true;
            }
            else {
                return false;
            }
        };
        const options: FormValidatorModel = {
            rules: {
                'numeric': { required: [true, "Number is required"] },
            }
        };
        // defines FormValidator to validate the NumericTextBox
        this.formObject = new FormValidator('#form-element', options);
        this.formObject.addRules('numeric', { range: [customFn, "Please enter
        a number between 10 to 100"] });
        const proxy = this;
        (document.getElementById('submit_btn') as
        HTMLElement).addEventListener('click', () => {
            proxy.formObject.validate('numeric');
            const ele: HTMLInputElement =
            document.getElementById('numeric') as HTMLInputElement;
            // checks for incomplete value and alerts the formt submit
            if (ele.value !== "" && parseInt(ele.value, 10) >=10 &&
            parseInt(ele.value, 10)<=100) {
                alert("Submitted");
            }
        });
    }
    public render() {
        return (
            <div>
                <NumericTextBoxComponent id="numeric" name='numeric'
                placeholder="NumericTextbox" min= {10} max={100} strictMode={false}
                ref={(numeric) => { this.numericInstance = numeric as
                NumericTextBoxComponent; }} change={this.onChange} created={this.onCreate}
                floatLabelType='Always' />
                <label className='e-error' htmlFor={'numeric'}/>
                <button id="submit_btn" style={{ marginTop: '10px'
            }}>Submit</button>
            </div>
        );
    }
};
ReactDOM.render(<App />, document.getElementById('numericContainer'));
{% endraw %}

```

Prevent nullable input in numerictextbox in React Numerictextbox component

By default, the value of the NumericTextBox sets to null. In some applications, the value of the NumericTextBox should not be null at any instance. In such cases, following sample can be used to prevent nullable input in NumericTextBox.

INDEX.JSX

```
{% raw %}
import { NumericTextBoxComponent } from '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  numericInstance;
  numericValue = 0;
  constructor(props) {
    super(props);
    this.onCreate = this.onCreate.bind(this);
    this.onBlur = this.onBlur.bind(this);
  }
  // prevents nullable value during initialization
  onCreate() {
    if (this.numericInstance.value == null) {
      this.numericInstance.value = 0;
      this.numericInstance.dataBind();
    }
  }
  onBlur(args) {
    if (args.value == null) {
      this.numericInstance.value = 0;
      this.numericInstance.dataBind();
    }
  }
  render() {
    return (<NumericTextBoxComponent id="numeric"
value={this.numericValue} ref={(numeric) => { this.numericInstance = numeric;
}} created={this.onCreate} blur={this.onBlur}/>);
  }
}
ReactDOM.render(<App />, document.getElementById('numeric1'));
```

INDEX.TSX

```
{% raw %}
import { NumericBlurEventArgs, NumericTextBoxComponent } from
 '@syncfusion/ej2-react-inputs';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  public numericInstance: any;
  public numericValue: number = 0;
  constructor(props: any) {
    super(props);
    this.onCreate = this.onCreate.bind(this);
    this.onBlur = this.onBlur.bind(this);
  }
}
```

```
// prevents nullable value during initialization
public onCreate() {
    if (this.numericInstance.value == null) {
        this.numericInstance.value = 0;
        this.numericInstance.dataBind()
    }
}

public onBlur(args: NumericBlurEventArgs) {
    if (args.value == null) {
        this.numericInstance.value = 0;
        this.numericInstance.dataBind()
    }
}

public render() {
    return (
        <NumericTextBoxComponent id="numeric" value={this.numericValue}
        ref={ (numeric) => { this.numericInstance = numeric as
        NumericTextBoxComponent; }} created={this.onCreate} blur={this.onBlur} />
    );
}

ReactDOM.render(<App />, document.getElementById('numeric1'));
```

{% endraw %}

Ej1 api migration in React Numerictextbox component

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

Common

{% raw %}

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

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

| Triggers on creation | **Event** *create*
 create={this.onCreate}</EJ.NumericTextbox >
<script>onCreate: function() {} | **Event:** *created*
 created={this.onCreate.bind(this)}</NumericTextBoxComponent>
<public
 onCreate() {} |

| Adding custom classes | **Property** *cssClass*
 value={120} cssClass="custom"></EJ.NumericTextbox > | **Property:** *cssClass*
 /><NumericTextBoxComponent id="numeric" value='120'
 cssClass="custom"></NumericTextBoxComponent> |

| Triggers when editor is destroyed | **Event** *destroy*
 value={120} destroy={this.onDestroy}></EJ.NumericTextbox >
<script>onDestroy:
 function() {} | **Event:** *destroyed*
 value='120' destroyed={this.onDestroy.bind(this)}></NumericTextBoxComponent>
<public
 onDestroyed() {} |

| Destroys textbox | **Method** *destroy*

<EJ.NumericTextbox id="numeric" value={100}></EJ.NumericTextbox >
var numericObj = \$("#numeric").data("ejNumericTextbox");
numericObj.destroy(); | **Method:** *destroy*

<NumericTextBoxComponent id="numeric" value='100'></NumericTextBoxComponent>
var numeric = document.getElementById('numeric').ej2_instances[0];
numeric.destroy(); |

| Control state | **Property** *enabled*

<EJ.NumericTextbox id="numeric" value={100} enabled={false}></EJ.NumericTextbox > | **Property:** *enabled*

<NumericTextBoxComponent id="numeric" value='100' enabled=false></NumericTextBoxComponent> |

| Persistence | **Property** *enablePersistence*

<EJ.NumericTextbox id="numeric" value={100} enablePersistence={true}></EJ.NumericTextbox > | **Property:** *enablePersistence*

<NumericTextBoxComponent id="numeric" value='100' enablePersistence=true></NumericTextBoxComponent> |

| Right To Left | **Property** *enableRTL*

<EJ.NumericTextbox id="numeric" value={100} enableRTL={true}></EJ.NumericTextbox > | **Property:** *enableRtl*

<NumericTextBoxComponent id="numeric" value='100' enableRtl=true></NumericTextBoxComponent> |

| Triggers when editor is focused in | **Event** *focusIn*

<EJ.NumericTextbox id="numeric" value={20} focusIn={this.focusIn}></EJ.NumericTextbox >

<script>
focusIn: function() {} | **Event:** *focus*

<NumericTextBoxComponent id="numeric" value={100} focus={this.onFocus.bind(this)}></NumericTextBoxComponent>

<script>
public onFocus() {} |

| Triggers when editor is focused out | **Event** *focusOut*

<EJ.NumericTextbox id="numeric" value={100} focusOut={this.focusOut}></EJ.NumericTextbox >

<script>
focusOut: function() {} | **Event:** *blur*

<NumericTextBoxComponent id="numeric" value={100} blur={this.onBlur.bind(this)}></NumericTextBoxComponent>

<script>
public onBlur() {} |

| Sets Height | **Property** *height*

<EJ.NumericTextbox id="numeric" value={100} height="100px"></EJ.NumericTextbox > | **Can be achieved using**,

<NumericTextBoxComponent id="numeric" value='100' cssClass="custom"></NumericTextBoxComponent>
<css>
.e-numerictextbox.custom{
height: 40px;
} |

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

<EJ.NumericTextbox id="numeric" value={100} htmlAttributes= {htmlAttr}></EJ.NumericTextbox>

<script>
var htmlAttr = {name: "numerictextbox"}; | **Can be achieved using**

<NumericTextBoxComponent value= '32' id="numeric" placeholder="Numeric TextBox" created={this.onCreated.bind(this)} floatLabelType="Always"></NumericTextBoxComponent>
public onCreated(){
document.getElementById("numeric").setAttribute("name","Numeric")
} |

| Name of editor | **Property** *name*
 <EJ.NumericTextbox id="numeric" value={100}
 htmlAttributes= {name="textbox"}></EJ.NumericTextbox> | **Can be achieved**
using
 <NumericTextBoxComponent value= '32' id="numeric" placeholder="Numeric
 TextBox" created={this.onCreated.bind(this)}
 floatLabelType="Always"></NumericTextBoxComponent>
public
 onCreated(){
document.getElementById("numeric").setAttribute("name","Numeric")
}

| Read only | **Property** *readOnly*
 <EJ.NumericTextbox id="numeric" value={80}
 readOnly={true}></EJ.NumericTextbox> | **Property:** *readOnly*
 <NumericTextBoxComponent id="numeric" value='80'
 readOnly=true></NumericTextBoxComponent> |

| Rounded corners | **Property** *showRoundedCorner*
 <EJ.NumericTextbox id="numeric"
 value={80} showRoundedCorner={true}></EJ.NumericTextbox> | **Can be achieved**
using
 <NumericTextBoxComponent id="numeric" value='100' cssClass='e-
 style'></NumericTextBoxComponent>
Css
 .e-control-wrapper.e-numeric.e-input-
 group.e-style {
border: 2.5px solid;
border-radius: 1rem;
 padding-left: 12px;

}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::before, .e-control-
 wrapper.e-numeric.e-float-input.e-style .e-float-line::after{
background: none
 ;
}
.e-control-wrapper.e-numeric.e-input-group.e-style.e-input-focus{
 border:
 solid grey !important;
} |

| Spin Button | **Property** *showSpinButton*
 <EJ.NumericTextbox id="numeric" value={20}
 showSpinButton={false}></EJ.NumericTextbox> | **Property:** *showSpinButton*
 <NumericTextBoxComponent id="numeric" value='20'
 showSpinButton=false></NumericTextBoxComponent> |

| Width | **Property** *width*
 <EJ.NumericTextbox id="numeric" value={20}
 width="220px"></EJ.NumericTextbox> | **Property:** *width*
 <NumericTextBoxComponent
 id="numeric" value='20' width="220px"></NumericTextBoxComponent> |

| Clear Button | Not Applicable | **Property:** *showClearButton*
 <NumericTextBoxComponent id="numeric" value='100'
 showClearButton=true></NumericTextBoxComponent> |

Globalization

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

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

| Localization culture | **Property** *locale*
 <EJ.NumericTextbox id="numeric" value={80}
 locale="de-DE"></EJ.NumericTextbox> | **Property:** *locale*
 <NumericTextBoxComponent id="numeric" value='80' locale="de-
 DE"></NumericTextBoxComponent> |

Group

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

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

| Group digits in editor | **Property** *groupSize*
value={100} groupSize={2}></EJ.NumericTextbox> | Not Applicable |

| Group Separator | **Property** *groupSeparator*
value={100} groupSeparator="-"></EJ.NumericTextbox> | Not Applicable |

Numeric configuration

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

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

| Triggers on value change | **Event** *change*
value={120} change={this.onChange}></EJ.NumericTextbox>
script
onChange: function() {} | **Event:** *change*
change={this.onChange.bind(this)}></NumericTextBoxComponent>
public
onChange() {} |

| Sets digits allowed after decimal point | **Property** *decimalPlaces*
id="numeric" value={100} decimalPlaces={2}></EJ.NumericTextbox> | **Property:** *decimals*
decimals="2"></NumericTextBoxComponent> |

| Decrement value | Not Applicable | **Method:** *decrement*
id="numeric" value={100}></NumericTextBoxComponent>
var numeric = document.getElementById('numeric').ej2_instances[0];
numeric.decrement(); |

| Disable the textbox | **Method** *disable*
></EJ.NumericTextbox>
var numericObj = \$("#numeric").data("ejNumericTextbox");
numericObj.disable(); | **Can be achieved by**
value='100' showClearButton=true></NumericTextBoxComponent>
var numeric = document.getElementById('numeric').ej2_instances[0];
numeric.enabled = false; |

| Enable the textbox | **Method** *enable*
></EJ.NumericTextbox>
var numericObj = \$("#numeric").data("ejNumericTextbox");
numericObj.enable(); | **Can be achieved by**
value='100' showClearButton=true></NumericTextBoxComponent>
var numeric = document.getElementById('numeric').ej2_instances[0];
numeric.enabled = true; |

| Gets value of editor | **Method** *getValue*
></EJ.NumericTextbox>
var numericObj = \$("#numeric").data("ejNumericTextbox");
numericObj.getValue(); | **Method:** *getText*
id="numeric" value='100'></NumericTextBoxComponent>
var numeric = document.getElementById('numeric').ej2_instances[0];
numeric.getText(); |

| Increment value | Not Applicable | **Method:** *increment*
id="numeric" value='100'></NumericTextBoxComponent>
var numeric = document.getElementById('numeric').ej2_instances[0];
numeric.increment(); |

| Step value | **Property** *incrementStep*
id="numeric" value={100} incrementStep={2}></EJ.NumericTextbox> | **Property:** *step*

```

/><NumericTextBoxComponent id="numeric" value='100'
step="2"></NumericTextBoxComponent> |

```

```

| Sets Maximum value | Property maxValue<br /><br /><EJ.NumericTextbox id="numeric"
value={100} maxValue={200}></EJ.NumericTextbox> | Property: max<br /><br />
/><NumericTextBoxComponent id="numeric" value='100'
max="200"></NumericTextBoxComponent> |

```

```

| Sets Minimum value | Property minValue<br /><br /><EJ.NumericTextbox id="numeric"
value={100} minValue={20}></EJ.NumericTextbox> | Property: min<br /><br />
/><NumericTextBoxComponent id="numeric" value='100'
min="20"></NumericTextBoxComponent> |

```

```

| Negative pattern for formatting values | Property negativePattern<br /><br /><EJ.NumericTextbox
id="numeric" value={-20} negativePattern="( n)"></EJ.NumericTextbox> | Not Applicable |

```

```

| Positive pattern for formatting values | Property positivePattern<br /><br /><EJ.NumericTextbox
id="numeric" value={20} positivePattern="n kg"></EJ.NumericTextbox> | Not Applicable |

```

```

| Specifies value | Property value<br /><br /><EJ.NumericTextbox id="numeric"
value={100}></EJ.NumericTextbox> | Property: value<br /><br /><NumericTextBoxComponent
id="numeric" value='100'></NumericTextBoxComponent> |

```

```

| Displays hint on editor | Property watermarkText<br /><br /><EJ.NumericTextbox id="numeric"
value={80} watermarkText="Enter value"></EJ.NumericTextbox> | Property: placeholder<br /><br />
/><NumericTextBoxComponent id="numeric" value='100' placeholder="Enter
value"></NumericTextBoxComponent> |

```

```

| Placeholder float type | Not Applicable | Property: floatLabelType<br /><br />
/><NumericTextBoxComponent id="numeric" value='200' placeholder="Enter value"
floatLabelType="Auto"></NumericTextBoxComponent> |

```

Number Formats

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

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

```

| Set Currency symbol | Property currencySymbol<br /><br /><EJ.CurrencyTextbox id="currency"
value={100} currencySymbol="EUR"></EJ.CurrencyTextbox> | Property: currency<br /><br />
/><NumericTextBoxComponent id="currency" value='100' format="c2"
currency="EUR"></NumericTextBoxComponent> |

```

```

| Number Format | Not Applicable | Property: format<br /><br /><NumericTextBoxComponent
id="numeric" value='200' format="n2"></NumericTextBoxComponent> |

```

Validation

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

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

```

| Strict Mode | Property enableStrictMode<br /><br /><EJ.NumericTextbox id="numeric"
value={80} enableStrictMode={true}></EJ.NumericTextbox> | Property: strictMode<br /><br />

```

```

/><NumericTextBoxComponent id="numeric" value='80'
strictMode=true></NumericTextBoxComponent> |

| Validation on typing | Property validateOnType<br /><br /><EJ.NumericTextbox id="numeric"
value={80} validateOnType={true}></EJ.NumericTextbox> | Property: validateDecimalOnType<br
/><br /><NumericTextBoxComponent id="numeric" value='100'
validateDecimalOnType=true></NumericTextBoxComponent> |

| Validation message | Property: validationMessage<br /><br /><EJ.NumericTextbox id="numeric"
value={100} validationRules= {validationRules} validationMessage=
{validationMessage}></EJ.NumericTextbox><br /><br />var validationRules = {required:
{true}};<br />var validationMessage = {required: "Required value"}; | Can be achieved
by<br/><NumericTextBoxComponent id="numeric" change={this.onChange.bind(this)}
created={this.onCreate.bind(this)} floatLabelType='Always' /><br/>let options:
FormValidatorModel = {<br/>rules: {<br/>'numeric': { required: [true, "Number is required"]
},<br/>,<br/>customPlacement: (inputElement: HTMLElement, errorElement: HTMLElement) =>
{<br/>inputElement.parentNode.parentNode.parentNode.appendChild(errorElement);<br/>}<br
/>,<br/>this.formObject = new FormValidator('#form-element', options);|

| Validation Rules | Property: validationRules<br /><br /><EJ.NumericTextbox id="numeric"
value={100} validationRules= {validationRules}></EJ.NumericTextbox><br /><br />var
validationRules = {required: {true}}; | Can be achieved by<br/><NumericTextBoxComponent
id="numeric" change={this.onChange.bind(this)} created={this.onCreate.bind(this)}
floatLabelType='Always' /><br/>let options: FormValidatorModel = {<br/>rules: {<br/>'numeric':
{ required: [true] },<br/>,<br/>customPlacement: (inputElement: HTMLElement, errorElement:
HTMLElement) =>
{<br/>inputElement.parentNode.parentNode.parentNode.appendChild(errorElement);<br/>}<br
/>,<br/>this.formObject = new FormValidator('#form-element', options); |

{% endraw %}

```

Pager

Getting started

This section explains you the steps required to create a simple Pager and demonstrate the basic usage of the Pager component in React environment.

Dependencies

Below is the list of minimum dependencies required to use the Pager.

```

`javascript
|-- @syncfusion/ej2-react-grids
|-- @syncfusion/ej2-grids
`

```

Setup for Local Development

You can use [create-react-app](#) to setup the applications.

To install [create-react-app](#) run the following command.

```
`  
npm install -g create-react-app  
`
```

- To setup basic `React` sample use following commands.

```
`  
create-react-app quickstart --scripts-version=react-scripts-ts  
cd quickstart  
npm install  
`
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry.

To install Pager component, use the following command

```
`  
npm install @syncfusion/ej2-react-grids --save  
`
```

Adding CSS reference

Add Pager component's styles as given below in `src/App.css`.

```
`css  
@import "../node_modules/@syncfusion/ej2-react-grids/styles/material.css";  
`
```

To refer `App.css` in the application then import it in the `src/App.tsx` file.

Adding Pager component

Now, you can start adding Pager component in the application. For getting started, add the Pager component in `src/App.tsx` file using following code.

Now place the below Pager code in the `src/App.tsx`.

Here the Pager is rendered with `totalRecordsCount` which is used to render numeric container.

```
`ts  
import * as React from "react";  
import * as ReactDOM from "react-dom";  
import { PagerComponent } from '@syncfusion/ej2-react-grids';  
import { data } from './datasource';  
class App extends React.Component<{}, {}>{  
  render() {
```

```

return <PagerComponent totalRecordsCount = {20}>
</PagerComponent>
}
};

ReactDOM.render(<App />, document.getElementById('pager'));

```

Page Size

`pageSize` value defines the number of records to be displayed per page. The default value for the `pageSize` is 12.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { PagerComponent } from '@syncfusion/ej2-react-grids';
class App extends React.Component {
  render() {
    return <PagerComponent totalRecordsCount={20} pageSize={1}>
      </PagerComponent>;
  }
}
;
ReactDOM.render(<App />, document.getElementById('pager'));

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { PagerComponent } from '@syncfusion/ej2-react-grids';
class App extends React.Component<{}, {}>{
  render() {
    return <PagerComponent totalRecordsCount = {20} pageSize = {1}>
      </PagerComponent>
  }
};
ReactDOM.render(<App />, document.getElementById('pager'));

```

Page Count

`pageCount` value defines the number of pages to be displayed in the pager component for navigation.

The default value for `pageCount` is 10 and value will be updated based on [totalRecordsCount](#) and [pageSize](#) values.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { PagerComponent } from '@syncfusion/ej2-react-grids';
class App extends React.Component {
  render() {

```

```

        return <PagerComponent totalRecordsCount={20} pageSize={1}
        pageCount={3}>
            </PagerComponent>;
        }
    }
    ;
    ReactDOM.render(<App />, document.getElementById('pager'));

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { PagerComponent } from '@syncfusion/ej2-react-grids';
class App extends React.Component<{}, {}>{
    render() {
        return <PagerComponent totalRecordsCount = {20} pageSize = {1}
        pageCount = {3}>
            </PagerComponent>
        }
    };
    ReactDOM.render(<App />, document.getElementById('pager'));

```

Run the application

The [create-react-app](#) will pre-configure the project to compile and run the application in browser. Use the following command to run the application.

`

npm start

`

Output will be appears as follows.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { PagerComponent } from '@syncfusion/ej2-react-grids';
class App extends React.Component {
    render() {
        return <PagerComponent pageSize={8} totalRecordsCount={20}
        pageCount={3}>
            </PagerComponent>;
        }
    }
    ;
    ReactDOM.render(<App />, document.getElementById('pager'));

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { PagerComponent } from '@syncfusion/ej2-react-grids';
class App extends React.Component<{}, {}>{
    render() {

```

```
    return <PagerComponent pageSize = {8} totalRecordsCount = {20}  
    pageCount = {3}>  
      </PagerComponent>  
    }  
  };  
  ReactDOM.render(<App />, document.getElementById('pager'));
```

Style and appearance in React Pager component

To modify the Pager appearance, you need to override the default CSS of Pager. Please find the CSS structure that can be used to modify the Pager appearance. Also, you have an option to create your own custom theme for all the JavaScript controls using our [Theme Studio](#).

Customizing the Pager

Use the below CSS to customize the Pager root element.

```
`css  
  
.e-pager {  
  background-color: #deecf9;  
}
```

Customizing the Pager container element

Use the below CSS to customize the pager container element.

```
`css  
  
.e-pager .e-pagercontainer {  
  background-color: #deecf9;  
}
```

Customizing the Pager navigation elements

Customize the pager navigation elements, using the below selector.

```
`css  
  
.e-pager .e-prevpagedisabled,  
.e-pager .e-prevpage,  
.e-pager .e-nextpage,  
.e-pager .e-nextpagedisabled,  
.e-pager .e-lastpagedisabled,  
.e-pager .e-lastpage,  
.e-pager .e-firstpage,  
.e-pager .e-firstpagedisabled {  
  background-color: #deecf9;
```

```
}
,
```

Customizing the Pager page numeric link elements

Use the below CSS to customize the pager current page numeric link elements.

```
`css
.e-pager .e-numericitem {
border-radius: initial;
}
,
```

Customizing the Pager current page numeric element

Using this CSS, you can customize the pager current page numeric item.

```
`css
.e-pager .e-currentitem {
background-color: #0078d7;
}
,
```

Accessibility in React Pager component

Accessibility is achieved in the Pager component through WAI-ARIA standard and keyboard navigations. The Pager features can be effectively accessed through assistive technologies such as screen readers.

WAI-ARIA

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components.

The following ARIA attributes are used in the Pager:

- pager (data-role)
- aria-selected (attribute)
- aria-owns (attribute)
- aria-label (attribute)

Keyboard navigation

Pager functionalities can be interactive with keyboard shortcuts.

The following keyboard shortcuts are supported by the Pager.

Interaction Keys | Description

Pager | |

Alt + J | Focus on the first pager item.

Tab / Shift + Tab | Focus on the next/previous pager item.

Enter / Space | Select the currently focused page.

Right Arrow / PageDown | Navigate to next page.

Left Arrow / PageUp | Navigate to previous page.

Home / End | Navigate to first and last page.

PDF Viewer

Getting Started

Getting Started with Standalone PDF Viewer component

This section explains the steps required to create a simple Standalone React PDF Viewer and demonstrates the basic usage of the PDF Viewer control in a React application.

Prerequisites

To get started with Syncfusion React UI components, ensure the compatible version of React.

- React supported version $\geq 15.5.4+$.
- Required node version $\geq 14.0.0+$ (NPM Package Manager).

Setup for Local Development

1. Create a new React app [create-react-app](#) and install it using the following command.

```
`bash
```

```
npm install -g create-react-app
```

```
`
```

2. To setup basic **React** sample use following commands.

JSX

```
create-react-app quickstart
cd quickstart
npm install
```

TSX

```
npx create-react-app quickstart --template cra-template-typescript
cd quickstart
npm install
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](#) public registry.

- To install PDF Viewer component, use the following command

```
`
```

```
npm install @syncfusion/ej2-react-pdfviewer --save
```

- Copy the contents of the ej2-pdfviewer-lib folder from `./node_modules/@syncfusion/ej2-pdfviewer/dist` to the public directory using the command:

```
`bash
```

```
cp -R ./node_modules/@syncfusion/ej2-pdfviewer/dist/ej2-pdfviewer-lib public/ej2-pdfviewer-lib
```

- Confirm that there is an 'ej2-pdfviewer-lib' directory within your public directory.
- Validate that your server has been configured to utilize the Content-Type: application/wasm MIME type. Additional information can be found in the [Troubleshooting](#) section.

Adding PDF Viewer component and the CSS reference

- Add an HTML div element to act as the PDF Viewer element `public/index.html` using the following code.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Syncfusion React PDF Viewer</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Essential JS 2 for React Components" />
<meta name="author" content="Syncfusion" />
</head>
<body>
<div id='sample'>
<div id='loader'>Loading....</div>
</div>
</body>
</html>
```

- Add the React PDF Viewer component's CSS reference as given below in `src/index.css` file.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-pdfviewer/styles/material.css";
`
```

- Add the React PDF Viewer as below in `src/index.js` file to render the PDF Viewer component.

JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
  return (
    <div>
      <div className='control-section'>
        { /* Render the PDF Viewer */ }
        <PdfViewerComponent
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          resourceUrl="https://cdn.syncfusion.com/ej2/23.1.40/dist/ej2-pdfviewer-lib"
          style={{ 'height': '640px' }}>
          <Inject services={[ Toolbar, Magnification, Navigation, Annotation,
            LinkAnnotation, BookmarkView,
            ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
          ]}>
          </PdfViewerComponent>
        </div>
      </div>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('sample'));
  root.render(<App />);
}{% endraw %}
```

TSX

```
{% raw %}
import React from 'react';
import ReactDOM from 'react-dom/client';
```

```
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject} from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  return (<div>
    <div className='control-section'>
    <PdfViewerComponent
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      resourceUrl="https://cdn.syncfusion.com/ej2/23.1.40/dist/ej2-pdfviewer-lib"
      style={{ 'height': '640px' }}>
    <Inject services=[ Toolbar, Magnification, Navigation, Annotation,
      LinkAnnotation, BookmarkView,
      ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
    ]/>
    </PdfViewerComponent>
    </div>
  </div>);
}
const rootElement = document.getElementById('sample')!;
const root = ReactDOM.createRoot(rootElement);
root.render(<App />);
{% endraw %}
```

Run the application

Use the following command to run the application in browser with the port number **localhost:8080**.

npm start

Output will be appears as follows.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject} from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  return (<div>
    <div className='control-section'>
    <PdfViewerComponent
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
      style={{ 'height': '640px' }}>
    {/ * Inject the required services */}
    <Inject services=[ Toolbar, Magnification, Navigation, Annotation,
      LinkAnnotation, BookmarkView, ThumbnailView,
```

```
Print, TextSelection, TextSearch, FormFields, FormDesigner]] />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  return <div>
    <div className='control-section'>
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        style={{ 'height': '640px' }}>
        /* Inject the required services */
        <Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Syncfusion React PDF Viewer</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Essential JS 2 for React Components" />
<meta name="author" content="Syncfusion" />
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/material.css" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
```

```
<body>
<div id='sample'>
<div id='loader'>Loading....</div>
</div>
</body>
</html>
{% endraw %}
```

You can refer to our [React PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [React PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Limitation over Server-Backed PDF Viewer to Standalone PDF Viewer control

When comparing a Standalone PDF Viewer to a Server-Backed PDF Viewer control, it's crucial to understand the limitations that the Standalone PDF Viewer may have in comparison. These limitations are important to consider

PNG Image Support

The Standalone PDF Viewer does not have the capability to utilize PNG format for adding images to handwritten annotations, custom stamp, signature and initial form fields. It's important to be aware that only certain image formats, such as JPEG, are compatible for these purposes.

Local File Access

- The Standalone PDF Viewer control does not have the capability to directly access and load local physical files from a user's device. As a result, it is not possible to use a documentPath to load a PDF file directly from a local server within the viewer.
- The Standalone PDF Viewer allows users to export annotations and form fields from the viewer, it's important to be aware that the viewer does not support the direct import of annotations and form fields from a locally specified file path. In other words, you can extract annotations and form fields from the viewer, but you cannot reintroduce them into the viewer from external sources by specifying a file path located on your local device.

Note: These limitations are temporary and are expected to be addressed in the near future.

Getting Started

This section explains the steps required to create a simple React PDF Viewer and demonstrates the basic usage of the PDF Viewer control in a React application.

Prerequisites

To get started with Syncfusion React UI components, ensure the compatible version of React.

- React supported version >= 15.5.4+.
- Required node version >= 14.0.0+(NPM Package Manager).

Setup for Local Development

1. Create a new React app [create-react-app](#) and install it using the following command.

```
`bash
```

```
npm install -g create-react-app
```

,

2. To setup basic **React** sample use following commands.

JSX

```
create-react-app quickstart
cd quickstart
npm install
```

TSX

```
npx create-react-app quickstart --template cra-template-typescript
cd quickstart
npm install
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry.

To install PDF Viewer component, use the following command

,

```
npm install @syncfusion/ej2-react-pdfviewer --save
```

,

Adding PDF Viewer component and the CSS reference

- Add an HTML div element to act as the PDF Viewer element **public/index.html** using the following code.

,

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Syncfusion React PDF Viewer</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Essential JS 2 for React Components" />
<meta name="author" content="Syncfusion" />
</head>
<body>
<div id='sample'>
```

```
<div id='loader'>Loading....</div>
</div>
</body>
</html>
```

- Add the React PDF Viewer component's CSS reference as given below in `src/index.css` file.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-pdfviewer/styles/material.css";
```

- Add the React PDF Viewer as below in `src/index.js` file to render the PDF Viewer component.

JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
  return (
    <div>
      <div className='control-section'>
        { /* Render the PDF Viewer */ }
        <PdfViewerComponent
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
          style={{ 'height': '640px' }}>
          <Inject services=[ Toolbar, Magnification, Navigation, Annotation,
            LinkAnnotation, BookmarkView,
            ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
          ]/>
        </PdfViewerComponent>
      </div>
    </div>
```



```

</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

TSX

```

{% raw %}
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject} from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  return (
    <div>
      <div className='control-section'>
        <PdfViewerComponent
          id='container'
          documentPath='https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf'
          serviceUrl='https://services.syncfusion.com/react/production/api/pdfviewer'
          style={{ 'height': '640px' }}>
          <Inject services=[ Toolbar, Magnification, Navigation, Annotation,
            LinkAnnotation, BookmarkView,
            ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
          ] />
        </PdfViewerComponent>
      </div>
    </div>);
}
const rootElement = document.getElementById('sample');
const root = ReactDOM.createRoot(rootElement);
root.render(<App />);
{% endraw %}

```

Run the application

Use the following command to run the application in browser with the port number `localhost:8080`.

```
npm start
```

Output will be appears as follows.

INDEX.JSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView,
  ThumbnailView, Print, TextSelection, Annotation, TextSearch,
  FormFields, FormDesigner, Inject} from '@syncfusion/ej2-react-pdfviewer';
export function App() {

```

```

return (<div>
  <div className='control-section'>
    <PdfViewerComponent
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf"

serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
      height= '640px'>
      {/* Inject the required services */}
      <Inject services={[ Toolbar, Magnification, Navigation,
Annotation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch,
FormFields, FormDesigner]} />
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

INDEX.TSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch,
FormFields, FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
return (<div>
  <div className='control-section'>
    <PdfViewerComponent
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf"

serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
      height= '640px'>
      {/* Inject the required services */}
      <Inject services={[ Toolbar, Magnification, Navigation,
Annotation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch,
FormFields, FormDesigner]} />
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

INDEX.HTML

```

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

```

```

<title>Syncfusion React PDF Viewer</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Essential JS 2 for React Components" />
<meta name="author" content="Syncfusion" />
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/material.css"
/>
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

For PDF Viewer serviceUrl creation, follow the steps provided in the [link](#)

How to run the PDF Viewer web service

1.Download the sample from the [Web service sample in GitHub](#) link. 2.Navigate to the **ASP.NET Core** folder and open it in the command prompt. 3.Use the below command to restore the required packages.

,

dotnet restore

,

4.Use the below command to run the web service.

,

dotnet run

,

5.You can see that the PDF Viewer server instance runs in the local host with the port number **localhost:5001** and navigate to the PDF Viewer Web control **localhost:5001/pdfviewer** which returns the default get response method. We can bind the link to the **serviceUrl** property of PDF Viewer as below.

```
{% raw %}
```

```
`js
```

```
<PdfViewerComponent
```

```
id="container"
```

```
documentPath="PDF_Succinctly.pdf"
```

```

serviceUrl="https://localhost:5001/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]}/>
</PdfViewerComponent>
`

{% endraw %}

```

Note: When configuring the server-backed PDF viewer, it's essential to understand that there is no need to include the pdfium.js and pdfium.wasm files. Unlike the standalone PDF viewer, which relies on these files for local rendering, the server-backed PDF viewer fetches and renders PDFs directly from the server. Consequently, you can exclude the copy command for deployment process, as they are not required to load and display PDFs in this context.

[View sample in GitHub.](#)

You can refer to our [React PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [React PDF Viewer example](#) to understand how to explain core features of PDF Viewer.

Getting Started with the React PDF Viewer Component in the Preact Framework

This article provides a step-by-step guide for setting up a [Preact](#) project and integrating the Syncfusion React PDF Viewer component.

Preact is a fast and lightweight JavaScript library for building user interfaces. It's often used as an alternative to larger frameworks like React. The key difference is that Preact is designed to be smaller in size and faster in performance, making it a good choice for projects where file size and load times are critical factors.

Prerequisites

[System requirements for Syncfusion React UI components](#)

[Set up the Preact project](#)

To create a new **Preact** project, use one of the commands that are specific to either NPM or Yarn.

```

`bash
npm init preact
`

```

or

```

`bash
yarn init preact
`

```

Using one of the above commands will lead you to set up additional configurations for the project, as below:

1\ Define the project name: We can specify the name of the project directly. Let's specify the name of the project as `my-project` for this article.

```
`bash
```

```
T Preact - Fast 3kB alternative to React with the same modern API
```

```
|
```

- Project directory:

```
| my-project
```

```
—
```

```
`
```

2\ Choose `JavaScript` as the framework variant to build this Preact project using JavaScript and React.

```
`bash
```

```
T Preact - Fast 3kB alternative to React with the same modern API
```

```
|
```

- Project language:

```
| > JavaScript
```

```
| TypeScript
```

```
—
```

```
`
```

3\ Then configure the project as below for this article.

```
`bash
```

```
T Preact - Fast 3kB alternative to React with the same modern API
```

```
|
```

- Use router?

```
| Yes / > No
```

```
—
```

```
|
```

- Prerender app (SSG)?

```
| Yes / > No
```

```
—
```

```
|
```

- Use ESLint?

| Yes / > No

—

,

5\). Upon completing the aforementioned steps to create `my-project`, run the following command to jump into the project directory:

```
`bash
```

```
cd my-project
```

,

Now that `my-project` is ready to run with default settings, let's add Syncfusion components to the project.

Add the Syncfusion React packages

Syncfusion React component packages are available at [npmjs.com](https://www.npmjs.com). To use Syncfusion React components in the project, install the corresponding npm package.

This article uses the [React PDF Viewer component](#) as an example. To use the React PDF Viewer component in the project, the `@syncfusion/ej2-react-pdfviewer` package needs to be installed using the following command:

```
`bash
```

```
npm install @syncfusion/ej2-react-pdfviewer --save
```

,

or

```
`bash
```

```
yarn add @syncfusion/ej2-react-pdfviewer
```

,

Import Syncfusion CSS styles

You can import themes for the Syncfusion React component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and [Theme Studio](#). Refer to [themes topic](#) to know more about built-in themes and different ways to refer to theme's in a React project.

In this article, the `Material 3` theme is applied using CSS styles, which are available in installed packages. The necessary `Material 3` CSS styles for the PDF Viewer component and its dependents were imported into the `src/style.css` file.

~/SRC/STYLE.CSS

```
@import '../node_modules/@syncfusion/ej2-base/styles/material3.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material3.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material3.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material3.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material3.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material3.css';
```

```
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material3.css';
@import "../node_modules/@syncfusion/ej2-react-pdfviewer/styles/material3.css";
```

The order of importing CSS styles should be in line with its dependency graph.

Add the Syncfusion React component

Follow the below steps to add the React PDF Viewer component to the Vite project:

1\. Before adding the PDF Viewer component to your markup, import the PDF Viewer component in the **src/index.jsx** file.

~/SRC/INDEX.JSX

```
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject} from '@syncfusion/ej2-react-pdfviewer';
```

2\. Then, define the PDF Viewer component in the **src/index.jsx** file, as shown below:

~/SRC/INDEX.JSX

```
import { render } from 'preact';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject} from '@syncfusion/ej2-react-pdfviewer';
import './style.css';
export function App() {
  return (
    <div>
      <div className='control-section'>
        { /* Render the PDF Viewer */ }
        <PdfViewerComponent
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
          style={{ 'height': '640px' }}>
          <Inject services={[ Toolbar, Magnification, Navigation, Annotation,
            LinkAnnotation, BookmarkView,
            ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
          ]}/>
        </PdfViewerComponent>
      </div>
    </div>
  );
}
render(<App />, document.getElementById('app'));
```

Run the project

To run the project, use the following command:

```
`bash
```

```
npm run dev
```

```
`  
or  
`bash  
yarn run dev  
`
```

The output will appear as follows:



See also

[Getting Started with the Syncfusion React UI Component](#)

Feature module in React Pdfviewer component

The PDF Viewer features are segregated into individual feature-wise modules to enable selectively referencing in the application. The required modules should be injected to extend its functionality. The following are the selective modules of PDF Viewer that can be included as required:

The available PdfViewer modules are:

- **Toolbar**:- Built-in toolbar for better user interaction.
- **Magnification**:- Perform zooming operation for better viewing experience.
- **Navigation**:- Easy navigation across the PDF pages.
- **LinkAnnotation**:- Easy navigation within and outside of the PDF document.
- **ThumbnailView**:- Easy navigation with in the PDF document.
- **BookmarkView**:- Easy navigation based on the bookmark content of the PDF document.
- **TextSelection**:- Select and copy text from a PDF file.
- **TextSearch**:- Search a text easily across the PDF document.
- **Print**:- Print the entire document or a specific page directly from the browser.
- **Annotation**:- Annotations can be added or edited in the PDF document.
- **FormFields**:- Preserve the form fields in the PDF document.
- **FormDesigner**:- Form fields can be added or edited in the PDF document.

In addition to injecting the required modules in your application, enable corresponding properties to extend the functionality for a PDF Viewer instance.

Refer to the following table.

Module	Property to enable the functionality for a PDF Viewer instance
---	---
Toolbar	<code><PdfViewerComponent enableToolbar= {true} ></PdfViewerComponent></code>
Magnification	<code><PdfViewerComponent enableMagnification= {true} ></PdfViewerComponent></code>
Navigation	<code><PdfViewerComponent enableNavigation= {true} ></PdfViewerComponent></code>
LinkAnnotation	<code><PdfViewerComponent enableHyperlink= {true} ></PdfViewerComponent></code>
ThumbnailView	<code><PdfViewerComponent enableThumbnail= {true} ></PdfViewerComponent></code>
BookmarkView	<code><PdfViewerComponent enableBookmark= {true} ></PdfViewerComponent></code>
TextSelection	<code><PdfViewerComponent enableTextSelection= {true} ></PdfViewerComponent></code>
TextSearch	<code><PdfViewerComponent enableTextSearch= {true} ></PdfViewerComponent></code>
Print	<code><PdfViewerComponent enablePrint= {true} ></PdfViewerComponent></code>
Annotation	<code><PdfViewerComponent enableAnnotation= {true} ></PdfViewerComponent></code>
FormFields	<code><PdfViewerComponent enableFormFields= {true} ></PdfViewerComponent></code>
FormDesigner	<code><PdfViewerComponent enableFormDesigner= {true} ></PdfViewerComponent></code>

See also

- [Toolbar items](#)
- [Toolbar customization](#)

Open PDF files

You might need to open and view the PDF files from various location. In this section, you can find the information about how to open PDF files from URL, database and as base64 string.

Opening a PDF from URL

If you have your PDF files in the web, you can open it in the viewer using URL.

Step 1: Create a Simple PDF Viewer Sample in React

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in React. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the PdfViewerController.cs File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this [link](#) for instructions on how to create a web service project.
2. Open the PdfViewerController.cs file in your web service project.
3. Modify the Load() method to open it in the viewer using URL

```
`csharp
public IActionResult Load([FromBody] Dictionary<string, string> jsonData)
{
    // Initialize the PDF viewer object with memory cache object
    PdfRenderer pdfviewer = new PdfRenderer(_cache);
    MemoryStream stream = new MemoryStream();
    object jsonResult = new object();
    if (jsonObject != null && jsonObject.ContainsKey("document"))
    {
        if (bool.Parse(jsonObject["isFileName"]))
        {
            string documentPath = GetDocumentPath(jsonData["document"]);
            if (!string.IsNullOrEmpty(documentPath))
            {
                byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
                stream = new MemoryStream(bytes);
            }
            else
            {
                string fileName = jsonData["document"].Split(new string[] { "://" }, StringSplitOptions.None)[0];
                if (fileName == "http" || fileName == "https")
                {
                    WebClient WebClient = new WebClient();
                    byte[] pdfDoc = WebClient.DownloadData(jsonData["document"]);
                    stream = new MemoryStream(pdfDoc);
                }
                else
                {
                    return this.Content(jsonData["document"] + " is not found");
                }
            }
        }
        else
        {
            return this.Content(jsonData["document"] + " is not found");
        }
    }
}
```

```

{
byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
stream = new MemoryStream(bytes);
}
}

jsonResult = pdfviewer.Load(stream, jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}
`

```

Step 3: Set the PDF Viewer Properties in React PDF viewer component

Modify the `serviceUrl` property of the PDF viewer component with the accurate URL of your web service project, replacing `https://localhost:44396/pdfviewer` with the actual URL of your server. Modify the `documentPath` with the correct PDF Document URL want to load.

```

{% raw %}
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { PdfViewerComponent, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner, FormFields} from '@syncfusion/ej2-
react-pdfviewer';

function App() {
return (<div>
<div className='control-section'>
{/ Render the PDF Viewer /}
<PdfViewerComponent
id="container"
// Replace correct PDF Document URL want to load
documentPath="https://cdn.syncfusion.com/content/PDFViewer/flutter-succinctly.pdf"
// Replace the "localhost:44396" with the actual URL of your server
serviceUrl="https://localhost:44396/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services=[{ Toolbar, Magnification, Navigation, Annotation, LinkAnnotation, BookmarkView,
ThumbnailView,
Print, TextSelection, TextSearch, FormDesigner, FormFields }]/>

```

```

</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
`

{% endraw %}

```

[View sample in GitHub](#)

Opening a PDF from base64 data

The following code steps how the PDF file can be loaded in PDF Viewer as base64 string.

Step 1: Create a Simple PDF Viewer Sample in Angular

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in Angular. This will give you a basic setup of the PDF viewer component.

Step 2: Use the following code snippet to load the PDF document using a base64 string.

```

`
<button onClick={load}>LoadDocumentFromBase64</button>
<script>
// Load PDF document from Base64 string
function load(){
var viewer = document.getElementById('container').ej2_instances[0];
viewer.load('data:application/pdf;base64,'+ AddBase64String, null);
}
</script>
`

```

[View sample in GitHub](#)

Opening a PDF from database

To load a PDF file from SQL Server database in a PDF Viewer, you can follow the steps below

Step 1: Create a Simple PDF Viewer Sample in React

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in React. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the PdfViewerController.cs File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this [link](#) for instructions on how to create a web service project.

2. Open the PdfViewerController.cs file in your web service project.
3. Import the required namespaces at the top of the file:

```
`csharp
using System.IO;
using System.Data.SqlClient;
`
```

4. Add the following private fields and constructor parameters to the PdfViewerController class, In the constructor, assign the values from the configuration to the corresponding fields

```
`csharp
private IConfiguration _configuration;
public readonly string _connectionString;
public PdfViewerController(IWebHostEnvironment hostingEnvironment, IMemoryCache cache,
IConfiguration configuration)
{
    _hostingEnvironment = hostingEnvironment;
    _cache = cache;
    _configuration = configuration;
    connectionString = configuration.GetValue<string>("ConnectionString");
}
`
```

5. Modify the Load() method to open it in the viewer using URL

```
`csharp
public IActionResult Load([FromBody] Dictionary<string, string> jsonData)
{
    // Initialize the PDF viewer object with memory cache object
    PdfRenderer pdfviewer = new PdfRenderer(_cache);
    MemoryStream stream = new MemoryStream();
    object jsonResult = new object();
    if (jsonObject != null && jsonObject.ContainsKey("document"))
    {
        if (bool.Parse(jsonObject["isFileName"]))
        {

```

```
string documentPath = GetDocumentPath(jsonData["document"]);
if (!string.IsNullOrEmpty(documentPath))
{
    byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
    stream = new MemoryStream(bytes);
}

string documentName = jsonObject["document"];
string connectionString = _connectionString;
System.Data.SqlClient.SqlConnection connection = new
System.Data.SqlClient.SqlConnection(connectionString);
//Searches for the PDF document from the database
string query = "SELECT FileData FROM Table WHERE FileName = '" + documentName + "'";
System.Data.SqlClient.SqlCommand command = new System.Data.SqlClient.SqlCommand(query,
connection);
connection.Open();
using (SqlDataReader reader = command.ExecuteReader())
{
    if (reader.Read())
    {
        byte[] byteArray = (byte[])reader["FileData"];
        stream = new MemoryStream(byteArray);
    }
}
else
{
    byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
    stream = new MemoryStream(bytes);
}

jsonResult = pdfviewer.Load(stream, jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}
```

6. Open the `appsettings.json` file in your web service project, Add the following lines below the existing `"AllowedHosts"` configuration

```
`json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionString": "Your connection string for SQL server"
}
```

Note: Replace **Your Connection string from SQL server** with the actual connection string for your SQL Server database

Note: The **System.Data.SqlClient** package must be installed in your application to use the previous code example. You need to modify the `ConnectionString` variable in the previous code example as per the connection string of your database.

[View sample in GitHub](#)

Saving PDF file

After editing the PDF file with various annotation tools, you will need to save the updated PDF to the server, database, or local file system.

Save PDF file to Server

Need to save the modified PDF back to a server. To achieve this, proceed with the following steps

Step 1: Create a Simple PDF Viewer Sample in React

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in React. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the `PdfViewerController.cs` File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this [link](#) for instructions on how to create a web service project.
2. Open the `PdfViewerController.cs` file in your web service project.
3. Modify the `Download()` method to open it in the viewer using URL

```
`csharp
public IActionResult Download([FromBody] Dictionary<string, string> jsonObject)
```

```

{
//Initialize the PDF Viewer object with memory cache object
PdfRenderer pdfviewer = new PdfRenderer(_cache);
string documentBase = pdfviewer.GetDocumentAsBase64(jsonObject);
MemoryStream stream = new MemoryStream();
string documentName = jsonObject["document"];
string result = Path.GetFileNameWithoutExtension(documentName);
string fileName = result + "_downloaded.pdf";
// Save the file on the server
string serverFilePath = @"Path to where you need to save your file in the server";
string filePath = Path.Combine(serverFilePath, fileName);
using (FileStream fileStream = new FileStream(filePath, FileMode.Create))
{
//Saving the new file in root path of application
stream.CopyTo(fileStream);
fileStream.Close();
}
return Content(documentBase);
}
,

```

Step 3: Set the PDF Viewer Properties in React PDF viewer component

Modify the `serviceUrl` property of the PDF viewer component with the accurate URL of your web service project, replacing `https://localhost:44396/pdfviewer` with the actual URL of your server. Modify the `documentPath` with the correct PDF Document URL want to load.

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { PdfViewerComponent, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner, FormFields} from '@syncfusion/ej2-react-pdfviewer';

function App() {
return (<div>
<div className='control-section'>

```



```

{/ Render the PDF Viewer /}
<PdfViewerComponent
id="container"
// Replace correct PDF Document URL want to load
documentPath="https://cdn.syncfusion.com/content/PDFViewer/flutter-succinctly.pdf"
serviceUrl="https://localhost:44396/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation, LinkAnnotation, BookmarkView,
ThumbnailView,
Print, TextSelection, TextSearch, FormDesigner, FormFields ]} />
</PdfViewerComponent>
</div>
</div>;
}

const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

[View sample in GitHub](#)

Download PDF file as a copy

In the built-in toolbar, you have an option to download the updated PDF to the local file system, you can use it to download the PDF file.

```

{% raw %}
import * as ReactDOM from 'react-dom/client';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner, Inject } from
'@syncfusion/ej2-react-pdfviewer';
export function App() {
function downloadClicked() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.download();
}
return (<div>

```

```

<div className='control-section'>
  <button onClick={downloadClicked}>Download</button>
  <PdfViewerComponent
    id="container"
    // Replace PDF_Succinctly.pdf with the actual document name that you want to load
    documentPath:"PDF_Succinctly.pdf"
    // Replace the "localhost:44396" with the actual URL of your server
    serviceUrl="https://localhost:44396/pdfviewer"
    style={{ 'height': '640px' }}>
    {/ Inject the required services /}
    <Inject services=[ Toolbar, Magnification, Navigation, Annotation, LinkAnnotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner] />
  </PdfViewerComponent>
</div>
</div>;
}

const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Save PDF file to Database

If you have plenty of PDF files stored in database and you want to save the updated PDF file back to the database, use the following code example.

Step 1: Create a Simple PDF Viewer Sample in React

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in React. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the PdfViewerController.cs File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this [link](#) for instructions on how to create a web service project.
2. Open the PdfViewerController.cs file in your web service project.
3. Import the required namespaces at the top of the file:

```

`csharp
using System.IO;
using System.Data.SqlClient;
`

```

4. Add the following private fields and constructor parameters to the PdfViewerController class, In the constructor, assign the values from the configuration to the corresponding fields

```
`csharp
private IConfiguration _configuration;
public readonly string _connectionString;
public PdfViewerController(IWebHostEnvironment hostingEnvironment, IMemoryCache cache,
IConfiguration configuration)
{
    _hostingEnvironment = hostingEnvironment;
    _cache = cache;
    _configuration = configuration;
    connectionString = configuration.GetValue<string>("ConnectionString");
}
`
```

5. Modify the Download() method to open it in the viewer using URL

```
`csharp
[HttpPost("Download")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/Download")]
//Post action for downloading the PDF documents
public async Task<ActionResult> Download([FromBody] Dictionary<string, string> jsonObject)
{
    //Initialize the PDF Viewer object with memory cache object
    PdfRenderer pdfviewer = new PdfRenderer(_cache);
    string documentBase = pdfviewer.GetDocumentAsBase64(jsonObject);
    byte[] documentBytes = Convert.FromBase64String(documentBase.Split(",")[1]);
    string documentId = jsonObject["documentId"];
    string result = Path.GetFileNameWithoutExtension(documentId);
    string fileName = result + "_downloaded.pdf";
    string connectionString = _connectionString;
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
```

```

connection.Open();

using (SqlCommand cmd = new SqlCommand("INSERT INTO Table (FileName, fileData) VALUES
(@FileName, @fileData)", connection))
{
    cmd.Parameters.AddWithValue("@FileName", fileName);
    cmd.Parameters.AddWithValue("@fileData", documentBytes);
    cmd.ExecuteNonQuery();
}
connection.Close();
}
return Content(documentBase);
}
`

```

6. Open the `appsettings.json` file in your web service project, Add the following lines below the existing `"AllowedHosts"` configuration

```

`json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionString": "Your connection string for SQL server"
}
`

```

Note: Replace **Your Connection string from SQL server** with the actual connection string for your SQL Server database

Note: The **System.Data.SqlClient** package must be installed in your application to use the previous code example. You need to modify the `connectionString` variable in the previous code example as per the connection string of your database.

[View sample in GitHub](#)

Toolbar in React Pdfviewer component

The PDF Viewer comes with a powerful built-in toolbar to execute important actions such as page navigation, text search, view mode, download, print, bookmark, and thumbnails.

The following table shows built-in toolbar items and its actions:-

Option	Description
OpenOption	This option provides an action to load the PDF documents to the PDF Viewer.
PageNavigationTool	This option provides an action to navigate the pages in the PDF Viewer. It contains GoToFirstPage, GoToLastPage, GotoPage, GoToNext, and GoToLast tools.
MagnificationTool	This option provides an action to magnify the pages either with predefined or user defined zoom factors in the PDF Viewer. Contains ZoomIn, ZoomOut, Zoom, FitPage and FitWidth tools.
PanTool	This option provides an action for panning the pages in the PDF Viewer.
SelectionTool	This option provides an action to enable/disable the text selection in the PDF Viewer.
SearchOption	This option provides an action to search a word in the PDF documents.
PrintOption	This option provides an action to print the PDF document being loaded in the PDF Viewer.
DownloadOption	This Download option provides an action to download the PDF document that has been loaded in the PDF Viewer.
UndoRedoTool	This tool provides options to undo and redo the annotation actions performed in the PDF Viewer.
AnnotationEditTool	This tool provides options to enable or disable the edit mode of annotation in the PDF Viewer.
FormDesignerEditTool	This tool provides options to enable or disable the edit mode of form fields in the PDF Viewer.

Show/Hide the default toolbar

The PDF Viewer has an option to show or hide the complete default toolbar. You can achieve this by using following two ways.,

- **Show/Hide toolbar using enableToolbar API as in the following code snippet**

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  return (
    <div>
    <div className='control-section'>
```

```

{/* Render the PDF Viewer */}
<PdfViewerComponent
  id="container"
  documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
  resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
  enableToolbar={false}
  style={{ 'height': '640px' }}>
  <Inject services={[ Toolbar, Magnification, Navigation, Annotation,
    LinkAnnotation, BookmarkView, ThumbnailView,
    Print, TextSelection, TextSearch, FormFields, FormDesigner]} />
</PdfViewerComponent>
</div>
</div>;
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView,
  ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
  FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  return (
    <div>
      <div className='control-section'>
        {/* Render the PDF Viewer */}
        <PdfViewerComponent
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
          enableToolbar={false}
          style={{ 'height': '640px' }}>
          <Inject services={[ Toolbar, Magnification, Navigation, Annotation,
            LinkAnnotation, BookmarkView, ThumbnailView,
            Print, TextSelection, TextSearch, FormFields, FormDesigner]} />
          </PdfViewerComponent>
        </div>
      </div>;
    )
  }
  const root = ReactDOM.createRoot(document.getElementById('sample'));
  root.render(<App />);
  {% endraw %}

```

Note: To set up the **server-backed PDF Viewer**, add the following `serviceUrl` within the `<div>` element in either the `index.tsx` or `index.jsx` file:

`serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"`.

[View sample in GitHub](#)

- Show/Hide toolbar using showToolbar as in the following code snippet

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  function showToolbarClicked() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.toolbar.showToolbar(false);
  }
  return (
    <div>
    <div className='control-section'>
    <button onClick={showToolbarClicked}>ShowToolbarItem</button>
    {/ * Render the PDF Viewer */}
    <PdfViewerComponent
    id="container"
    documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
    resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
    style={{ 'height': '640px' }}>
    <Inject services=[ Toolbar, Magnification, Navigation, Annotation,
    LinkAnnotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner] />
    </PdfViewerComponent>
    </div>
    </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  function showToolbarClicked() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.toolbar.showToolbar(false);
  }
  return (
    <div>
    <div className='control-section'>
    <button onClick={showToolbarClicked}>ShowToolbarItem</button>
```

```

{/* Render the PDF Viewer */}
<PdfViewerComponent
  id="container"
  documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
  resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
  style={{ 'height': '640px' }}>
  <Inject services={[ Toolbar, Magnification, Navigation, Annotation,
    LinkAnnotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch , FormFields, FormDesigner]}
  />
</PdfViewerComponent>
</div>
</div>;
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Note: To set up the **server-backed PDF Viewer**, add the following **serviceUrl** within the **<div>** element in either the **index.tsx** or **index.jsx** file:

serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer".

Show/Hide the default toolbaritem

The PDF Viewer has an option to show or hide these grouped items in the default toolbar.

- **Show/Hide toolbaritem using toolbarSettings as in the following code snippet.**

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Annotation, Navigation,
  LinkAnnotation, BookmarkView,
  ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner,
  Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  return (
    <div>
      <div className='control-section'>
        {/* Render the PDF Viewer */}
        <PdfViewerComponent
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
          toolbarSettings={{ showTooltip : true, toolbarItems: ['OpenOption',
            'UndoRedoTool', 'PageNavigationTool', 'MagnificationTool', 'PanTool',
            'SelectionTool', 'CommentTool', 'SubmitForm', 'AnnotationEditTool',
            'FormDesignerEditTool', 'FreeTextAnnotationOption', 'InkAnnotationOption',
            'ShapeAnnotationOption', 'StampAnnotation', 'SignatureOption',
            'SearchOption', 'PrintOption', 'DownloadOption'], annotationToolbarItems:
            ['HighlightTool', 'UnderlineTool', 'StrikethroughTool', 'ColorEditTool',
            'OpacityEditTool', 'AnnotationDeleteTool', 'StampAnnotationTool',
            'HandWrittenSignatureTool', 'InkAnnotationTool', 'ShapeTool',

```



```

'CalibrateTool', 'StrokeColorEditTool', 'ThicknessEditTool',
'FreeTextAnnotationTool', 'FontFamilyAnnotationTool',
'FontSizeAnnotationTool', 'FontStylesAnnotationTool',
'FontAlignAnnotationTool', 'FontColorAnnotationTool', 'CommentPanelTool'],
formDesignerToolbarItems: ['TextboxTool', 'PasswordTool', 'CheckBoxTool',
'RadioButtonTool', 'DropdownTool', 'ListboxTool', 'DrawSignatureTool',
'DeleteTool']}]})
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Annotation, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
return (
<div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
toolbarSettings={{ showTooltip : true, toolbarItems: ['OpenOption',
'UndoRedoTool', 'PageNavigationTool', 'MagnificationTool', 'PanTool',
'SelectionTool', 'CommentTool', 'SubmitForm', 'AnnotationEditTool',
'FormDesignerEditTool', 'FreeTextAnnotationOption', 'InkAnnotationOption',
'ShapeAnnotationOption', 'StampAnnotation', 'SignatureOption',
'SearchOption', 'PrintOption', 'DownloadOption'], annotationToolbarItems:
['HighlightTool', 'UnderlineTool', 'StrikethroughTool', 'ColorEditTool',
'OpacityEditTool', 'AnnotationDeleteTool', 'StampAnnotationTool',
'HandWrittenSignatureTool', 'InkAnnotationTool', 'ShapeTool',
'CalibrateTool', 'StrokeColorEditTool', 'ThicknessEditTool',
'FreeTextAnnotationTool', 'FontFamilyAnnotationTool',
'FontSizeAnnotationTool', 'FontStylesAnnotationTool',
'FontAlignAnnotationTool', 'FontColorAnnotationTool', 'CommentPanelTool'],
formDesignerToolbarItems: ['TextboxTool', 'PasswordTool', 'CheckBoxTool',
'RadioButtonTool', 'DropdownTool', 'ListboxTool', 'DrawSignatureTool',
'DeleteTool']}]}}
style={{ 'height': '640px' }}>

```

```

<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Note: To set up the **server-backed PDF Viewer**, add the following **serviceUrl** within the <div> element in either the **index.tsx** or **index.jsx** file:

serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer".

- **Show/Hide toolbaritem using showToolbaritem as in the following code snippet**

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Annotation, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
function showToolbarClicked() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.toolbar.showToolbarItem(["OpenOption"], false);
}
return (
<div>
<div className='control-section'>
<button onClick={showToolbarClicked}>ShowToolbarItem</button>
</* Render the PDF Viewer */>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Annotation, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
function showToolbarClicked() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.toolbar.showToolbarItem(["OpenOption"], false);
}
return (
<div>
<div className='control-section'>
<button onClick={showToolbarClicked}>ShowToolbarItem</button>
{/* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Note: To set up the **server-backed PDF Viewer**, add the following **serviceUrl** within the **<div>** element in either the **index.tsx** or **index.jsx** file:

serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer".

See also

- [Toolbar customization](#)
- [Feature Modules](#)

Navigation in React Pdfviewer component

The PDF Viewer supports different internal and external navigations.

Toolbar page navigation option

The default toolbar of PDF Viewer contains the following navigation options

- [Go to page](#):- Navigates to the specific page of a PDF document.
- [Show next page](#):- Navigates to the next page of PDF a document.

- [Show previous page](#):- Navigates to the previous page of a PDF document.
- [Show first page](#):- Navigates to the first page of a PDF document.
- [Show last page](#):- Navigates to the last page of a PDF document.

You can enable/disable page navigation option in PDF Viewer using the following code snippet.,

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        enableNavigation={true}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation,
        BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

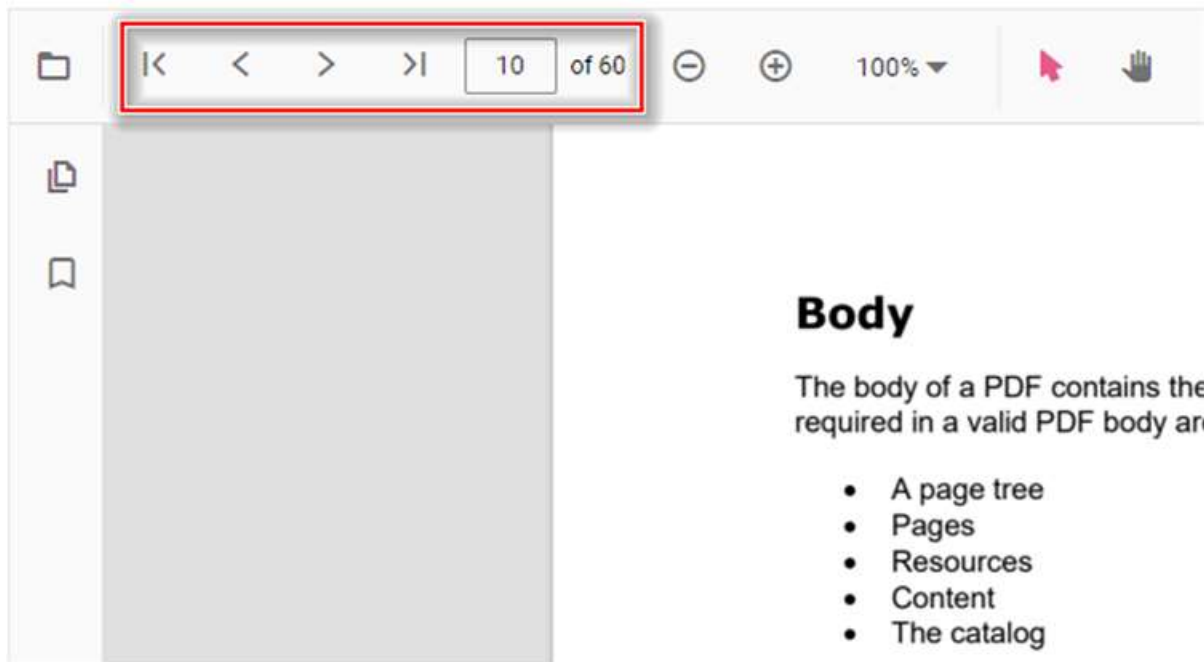
SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        enableNavigation={true}
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
      >
```

```

style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



Bookmark navigation

The Bookmarks saved in PDF files are loaded and made ready for easy navigation. You can enable/disable bookmark navigation by using the following code snippet.,

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
return <div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"

```

```

resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
enableBookmark={true}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

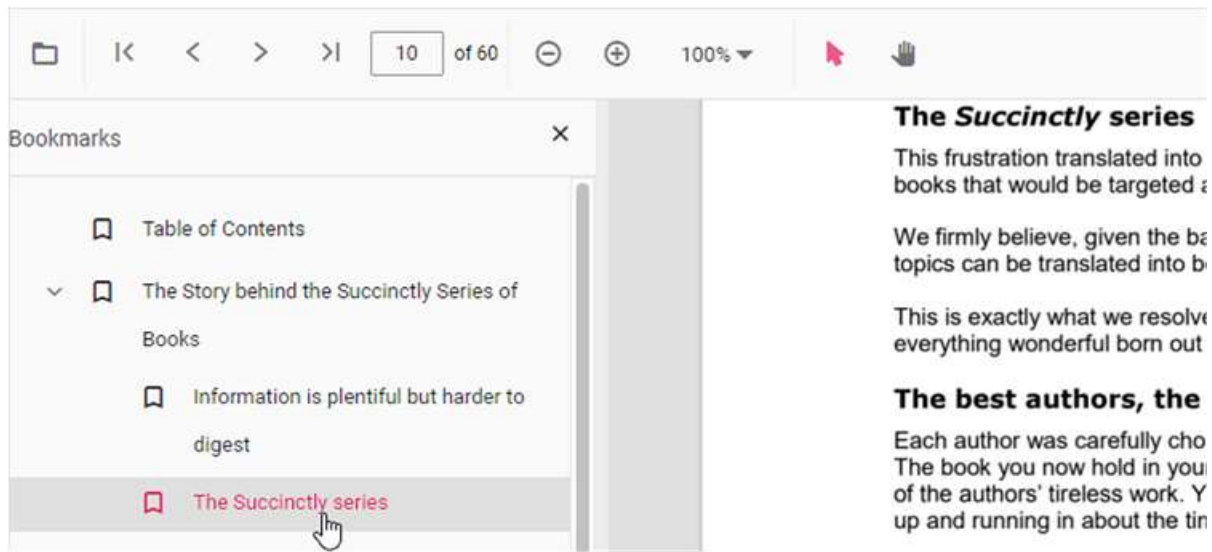
```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
return <div>
<div className='control-section'>
/* Render the PDF Viewer */
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
enableBookmark={true}
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



Thumbnail navigation

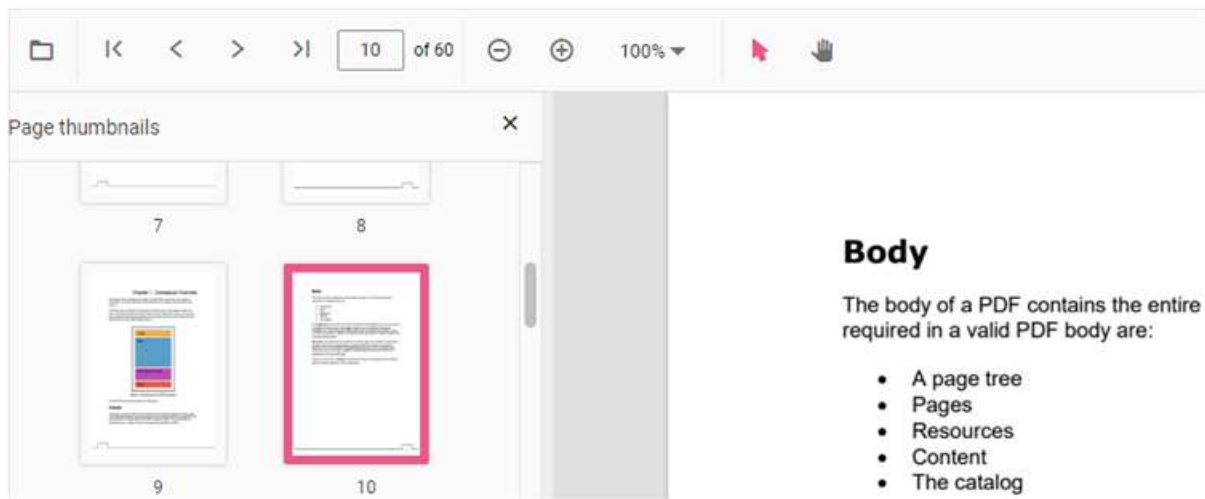
Thumbnails is the miniature representation of actual pages in PDF files. This feature displays thumbnails of the pages and allows navigation. You can enable/disable thumbnail navigation by using the following code snippet.,

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
return (<div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
enableThumbnail={true}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        enableThumbnail={true}
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation,
          BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```



Hyperlink navigation

Hyperlink navigation features enables navigation to the URLs (website links) in a PDF file.

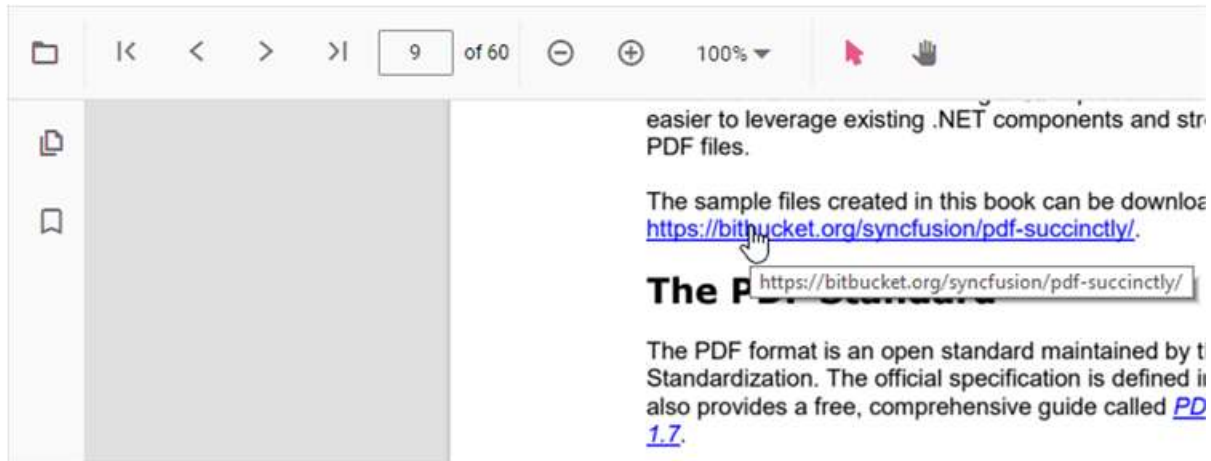


Table of content navigation

Table of contents navigation allows users to navigate to different parts of a PDF file that are listed in the table of contents section.

You can enable/disable link navigation by using the following code snippet.,

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        enableHyperlink={true}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation,
        BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
enableHyperlink={true}
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

You can change the open state of the hyperlink in the PDF Viewer by using the following code snippet,

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
enableHyperlink="true"
hyperlinkOpenState="NewTab"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
{% endraw %}
```

```

}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

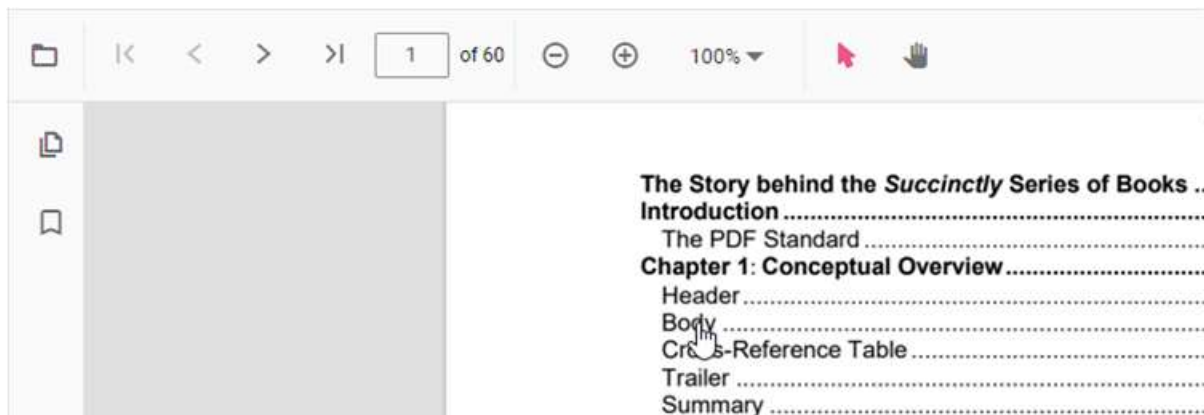
```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
return <div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
enableHyperlink="true"
hyperlinkOpenState="NewTab"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



See also

- [Toolbar items](#)

- [Feature Modules](#)

Magnification in React Pdfviewer component

The magnification tools of the PDF Viewer contains ZoomIn, ZoomOut, Zoom, FitPage, and FitWidth tools in the

default toolbar. The PDF Viewer also has an option to show or hide the magnification tools in the default toolbar.

The following code snippet describes how to enable the magnification in PDF Viewer.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      /* Render the PDF Viewer */
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        enableMagnification={true}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation, BookmarkView,
          ThumbnailView, Print, TextSelection, TextSearch]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      /* Render the PDF Viewer */
```

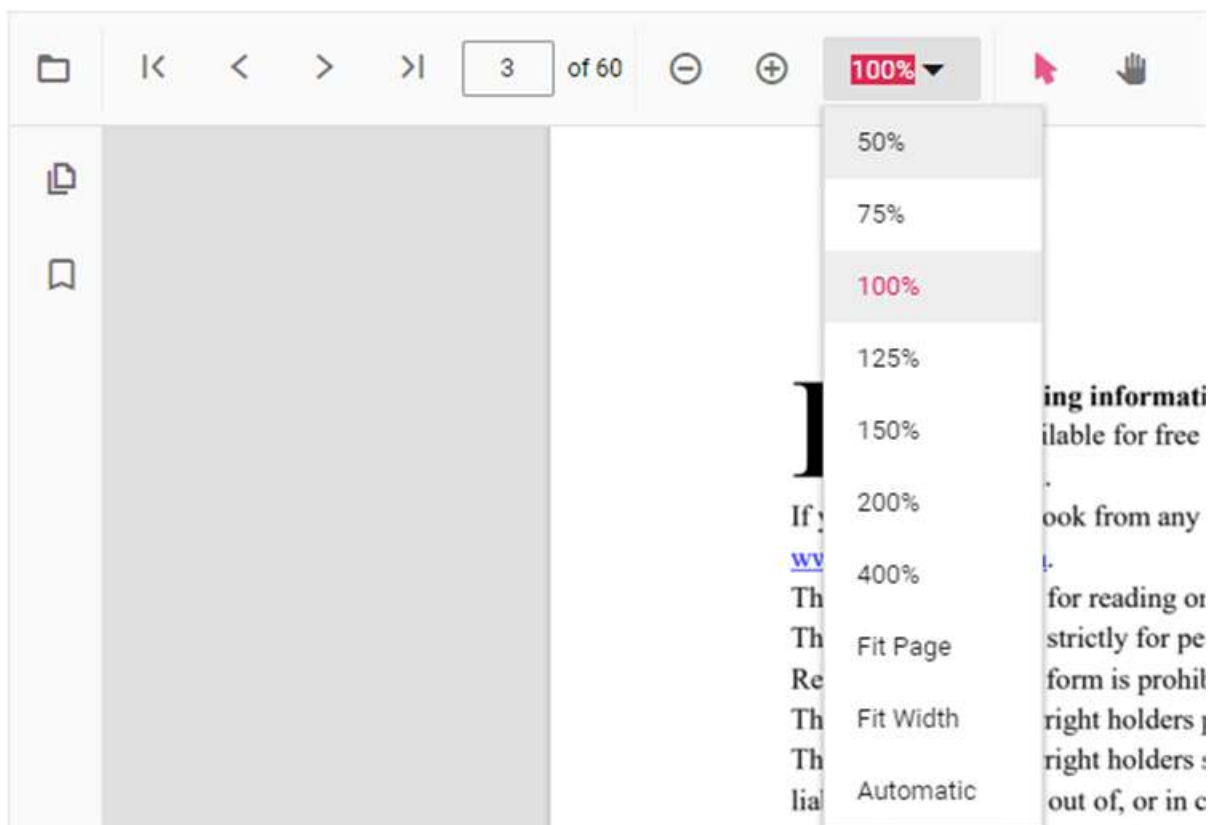
```

<PdfViewerComponent
  id="container"
  documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
  enableMagnification={true}
  serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
  style={{ 'height': '640px' }}>
  <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
    LinkAnnotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

The following magnification options are available in the default toolbar of PDF Viewer,

- [ZoomIn](#):- Zoom in from the current zoom value of PDF pages.
- [ZoomOut](#):- Zoom out from the current zoom value of PDF pages.
- [Zoom](#):- Zoom to specific zoom value of PDF pages.
- [FitPage](#):- Fits the page width with in the available view port size.
- [FitWidth](#):- Fits the view port width based on the page content size.



PDF Viewer can support the zoom value ranges from 50 to 400.

See also

- [Toolbar items](#)
- [Feature Modules](#)

Accessibility in Syncfusion React PDF Viewer components

The PDF Viewer 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 PDF Viewer 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

[WAI-ARIA](#) (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components. The following ARIA attributes are used in the PDF Viewer component

Attributes	Purpose
---	---
aria-disabled	Indicates whether the PDF Viewer component is in a disabled state or not.
aria-expanded	Indicates whether the suggestion list has expanded or not.
aria-readonly	Indicates the readonly state of the PDF Viewer element.
aria-haspopup	Indicates whether the PDF Viewer input element has a suggestion list or not.
aria-label	Indicates the breadcrumb item text.
aria-labelledby	Provides a label for the PDF Viewer. Typically, the "aria-labelledby" attribute will contain the id of the element used as the PDF Viewer's title.
aria-describedby	This attribute points to the PDF Viewer element describing the one it's set on.
aria-orientation	Indicates whether the PDF Viewer element is oriented horizontally or vertically.
aria-valuetext	Returns the current text of the PDF Viewer.
aria-valuemax	Indicates the Maximum value of the PDF Viewer.
aria-valuemin	Indicates the Minimum value of the PDF Viewer.
aria-valuenow	Indicates the current value of the PDF Viewer.
aria-controls	Attribute is set to the button and it points to the corresponding content.

Keyboard interaction

The PDF Viewer 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 Message component.

Press (Windows)	Press (Macintosh)	To do this
---	---	---
Shortcuts for page navigation		
CONTROL + Left Arrow (or) CONTROL + Up Arrow COMMAND + Left Arrow (or) COMMAND + Up Arrow Navigate to the first page		
CONTROL + Right Arrow (or) CONTROL + Down Arrow COMMAND + Right Arrow (or) COMMAND + Down Arrow Navigate to the last page		

| Left Arrow | Left Arrow (or) Shift + Space | Navigate to the previous page |

| Right Arrow | Right Arrow (or) Space | Navigate to the next page |

| CONTROL + G | COMMAND + G | Go To The Page |

| Up Arrow | Up Arrow | Scroll up |

| Down Arrow | Down Arrow | Scroll down |

|| Shortcuts for Zooming |

| CONTROL + = | COMMAND + = | Perform zoom-in operation |

| CONTROL + - | COMMAND + - | Perform zoom-out operation |

| CONTROL + 0 | COMMAND + 0 | Retain the zoom level to 1 |

|| Shortcut for Text Search |

| CONTROL + F | COMMAND + F | Open the search toolbar |

|| Shortcut for Text Selection |

| CONTROL + C | CONTROL + C | Copy the selected text or annotation or form field |

| CONTROL + X | CONTROL + X | Cut the selected text or annotation of the form field |

| CONTROL + Y | CONTROL + Y | Paste the selected text or annotation or form field |

|| Shortcuts for the general operation |

| CONTROL + Z | CONTROL + Z | Undo the action |

| CONTROL + Y | CONTROL + Y | Redo the action |

| CONTROL + P | COMMAND + P | Print the document |

| Delete | Delete | Delete the annotations and form fields |

| CONTROL + Shift + A | COMMAND + Shift + A | Toggle Annotation Toolbar |

| CONTROL + Alt + 0 | COMMAND + Option + 0 | Show Command panel |

| CONTROL + Alt + 2 | COMMAND + Option + 2 | Show Bookmarks |

| CONTROL + Alt + 1 | COMMAND + Option + 1 | Show Thumbnails |

| CONTROL + S | COMMAND + S | Download |

| Shift + H | Shift + H | Enable pan mode |

| Shift + V | Shift + V | Enable text selection mode |

The current implementation of our PDF Viewer includes keyboard shortcuts for various functions like scrolling, zooming, text search, printing, and annotation deletion.

To enhance user experience, we're adding additional keyboard shortcuts for actions such as navigating between pages, accessing specific pages, toggling annotation tools, and displaying PDF elements like outlines, annotations, bookmarks, and thumbnails.

To support this, we're introducing a new class called **commandManager**, which handles custom commands triggered by specific key gestures. These custom commands will be defined by users and executed accordingly.

The **commandManager** will have a parameter called **Commands**, which will hold the collection of custom keyboard commands specified by users. Each custom command will be represented by a **KeyboardCommand** class, containing the **command name** and associated **keyboard combination**.

Additionally, we're introducing the **keyboardCustomCommands** parameter for the **CommandManager**, which will utilize the **EventCallback** to handle keyboard events and trigger appropriate methods when specific key combinations are pressed.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import {PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function commandManager() {
keyboardCommand: [{
name: 'customCopy',
gesture: {
pdfKeys: PdfKeys.G,
modifierKeys: ModifierKeys.Shift | ModifierKeys.Alt
}
},
{
name: 'customPaste',
gesture: {
pdfKeys: PdfKeys.H,
modifierKeys: ModifierKeys.Shift | ModifierKeys.Alt
}
},
{
name: 'customCut',
gesture: {
pdfKeys: PdfKeys.Z,
modifierKeys: ModifierKeys.Control
}
},
{
name: 'customSelectAll',
gesture: {
pdfKeys: PdfKeys.E,
modifierKeys: ModifierKeys.Control
}
},
]
}
return (<div>
<div className='control-section'>
```

```

{/* Render the PDF Viewer */}
<PdfViewerComponent
  ref={(scope) => { pdfviewer = scope; }}
  id="container"
  documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
  resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
  commandManager = {commandManager}
  style={{ 'height': '640px' }}>
  <Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
    Annotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import {PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView,
  ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
  '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function commandManager() {
    keyboardCommand: [{
      name: 'customCopy',
      gesture: {
        pdfKeys: PdfKeys.G,
        modifierKeys: ModifierKeys.Shift | ModifierKeys.Alt
      }
    },
    {
      name: 'customPaste',
      gesture: {
        pdfKeys: PdfKeys.H,
        modifierKeys: ModifierKeys.Shift | ModifierKeys.Alt
      }
    },
    {
      name: 'customCut',
      gesture: {
        pdfKeys: PdfKeys.Z,
        modifierKeys: ModifierKeys.Control
      }
    },
    {
      name: 'customSelectAll',
      gesture: {

```

```

pdfKeys: PdfKeys.E,
modifierKeys: ModifierKeys.Control
},
],
}
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent
ref={ (scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
commandManager = {commandManager}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
Annotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Each `keyboardCommand` object consists of a `name` property, specifying the name of the custom command, and a `gesture` property, defining the key gesture associated with the command.

For example, the first command named `customCopy` is associated with the **G** key and requires both the **Shift** and **Alt** modifier keys to be pressed simultaneously.

Additionally, there's an explanation of the key modifiers used in the gestures:

- Ctrl corresponds to the Control key, represented by the value **1**.
- Alt corresponds to the Alt key, represented by the value **2**.
- Shift corresponds to the Shift key, represented by the value **4**.
- Meta corresponds to the Command key on macOS or the Windows key on Windows, represented by the value **8**.

This setup allows users to perform custom actions within the PDF viewer by pressing specific key combinations, enhancing the user experience and providing more efficient navigation and interaction options.

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Text search in React Pdfviewer component

The Text Search option in PDF Viewer is used to find and highlight the text content from the document. You can enable/disable the text search using the following code snippet.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        enableTextSearch={true}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation, BookmarkView,
          ThumbnailView, Print, TextSelection, TextSearch]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

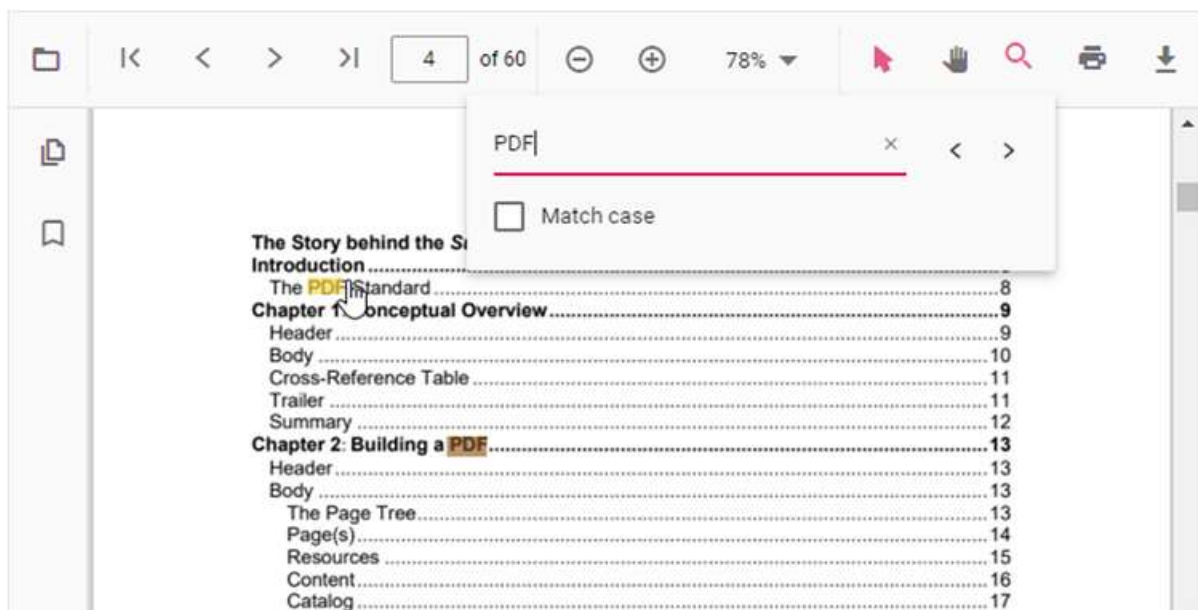
SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
```

```
enableTextSearch={true}
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

The following text search methods are available in the PDF Viewer,

- [Search text](#):- Searches the target text in the PDF document and highlights the occurrences in the pages.
- [Search next](#):- Searches the next occurrence of the searched text from the current occurrence of the PdfViewer.
- [Search previous](#):- Searches the previous occurrence of the searched text from the current occurrence of the PdfViewer.
- [Cancel text search](#):- The text search can be cancelled and the highlighted occurrences from the PDF Viewer can be removed .



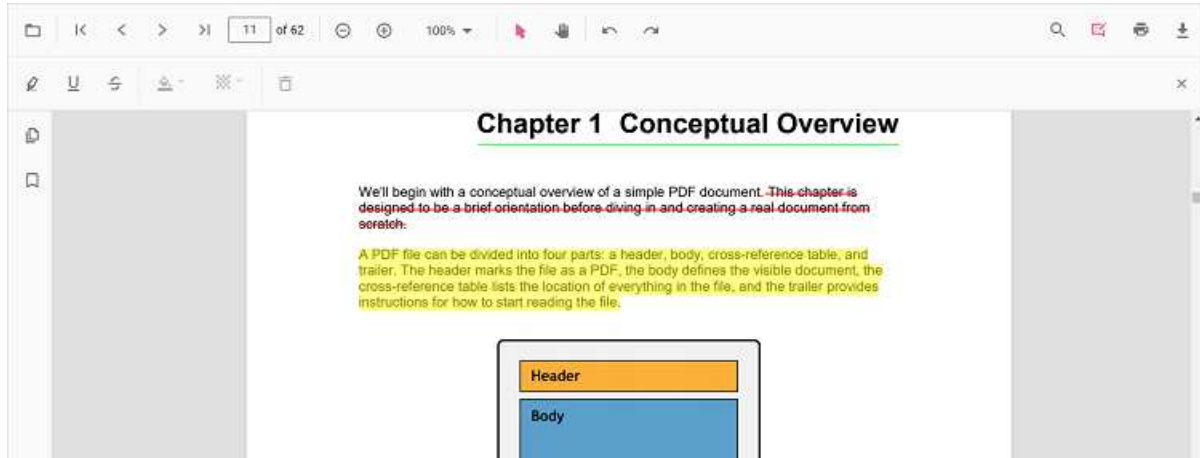
See also

- [Toolbar items](#)
- [Feature Modules](#)

Annotation

Text markup annotation in React Pdfviewer component

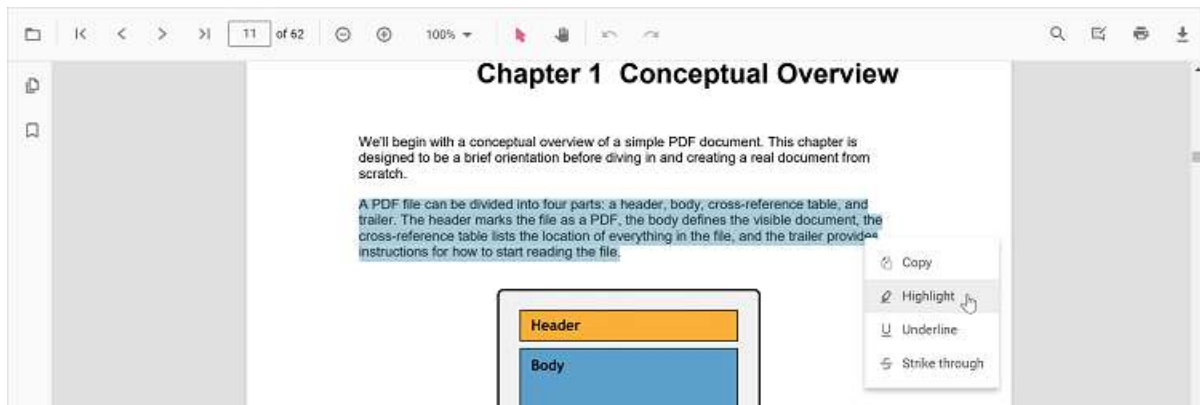
The PDF Viewer control provides the options to add, edit, and delete text markup annotations such as highlight, underline, and strikethrough annotations in the PDF document.



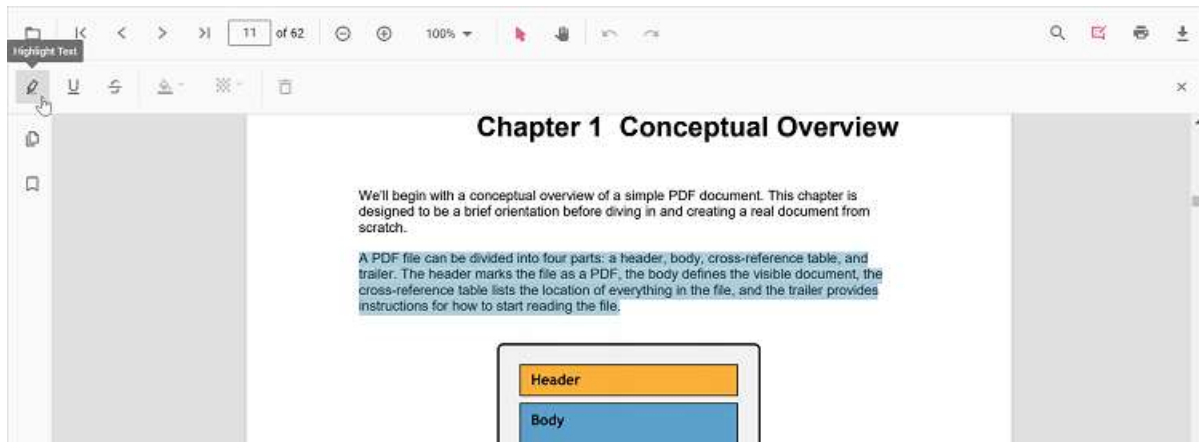
Highlight a text

There are two ways to highlight a text in the PDF document:

1. Using the context menu *Select a text in the PDF document and right-click it.* Select **Highlight** option in the context menu that appears.



2. Using the annotation toolbar *Click the Edit Annotation button in the PDF Viewer toolbar. A toolbar appears below it.* Select the **Highlight** button in the annotation toolbar. It enables the highlight mode. *Select the text and the highlight annotation will be added.* You can also select the text and apply the highlight annotation using the **Highlight** button.



In the pan mode, if the highlight mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for highlighting the text.

Refer to the following code snippet to switch to highlight mode.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function highlightMode() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Highlight');
}
return (
<div>
<button onClick={highlightMode}>Highlight</button>
<div className='control-section'>
{ /* Render the PDF Viewer */ }
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
 '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function highlightMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Highlight');
  }
  return (
    <div>
      <button onClick={highlightMode}>Highlight</button>
      <div className='control-section'>
        {/ * Render the PDF Viewer */}
        <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; }}
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
          style={{ 'height': '640px' }}>
          <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
            LinkAnnotation, BookmarkView, ThumbnailView,
            Print, TextSelection, TextSearch]} />
        </PdfViewerComponent>
      </div>
    </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Refer to the following code snippet to switch back to normal mode from highlight mode.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
 '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function highlightMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Highlight');
  }
  function normalMode () {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('None');
  }
}
```



```

return (<div>
  <button onClick={highlightMode}>Highlight</button>
  <button onClick={normalMode}>Normal Mode</button>
  <div className='control-section'>
    { /* Render the PDF Viewer */ }
    <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
    id="container"
    documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
    resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
    style={{ 'height': '640px' }}>
    <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
    LinkAnnotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch]} />
  </PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function highlightMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Highlight');
  }
  function normalMode () {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('None');
  }
  return (<div>
    <button onClick={highlightMode}>Highlight</button>
    <button onClick={normalMode}>Normal Mode</button>
    <div className='control-section'>
      { /* Render the PDF Viewer */ }
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
      style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
      LinkAnnotation, BookmarkView,
      ThumbnailView, Print, TextSelection, TextSearch]} />
    </PdfViewerComponent>
  </div>
</div>);

```

```

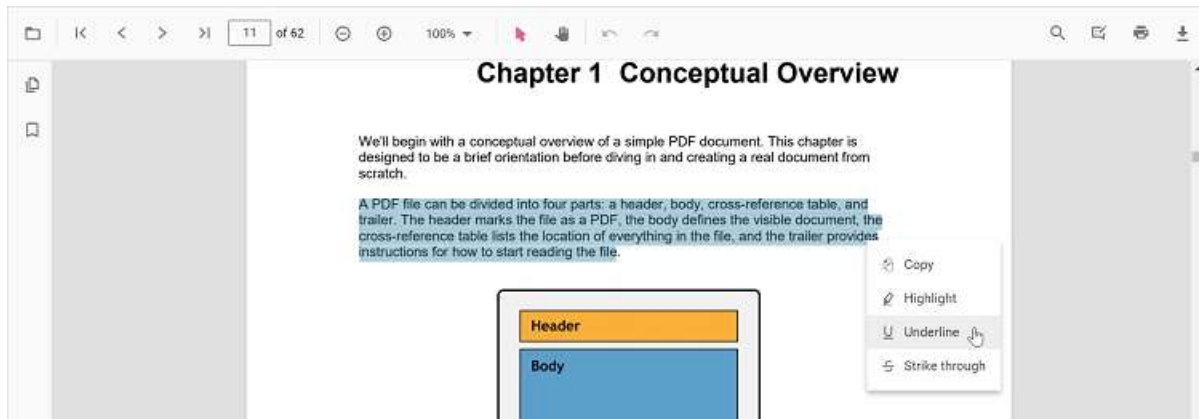
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

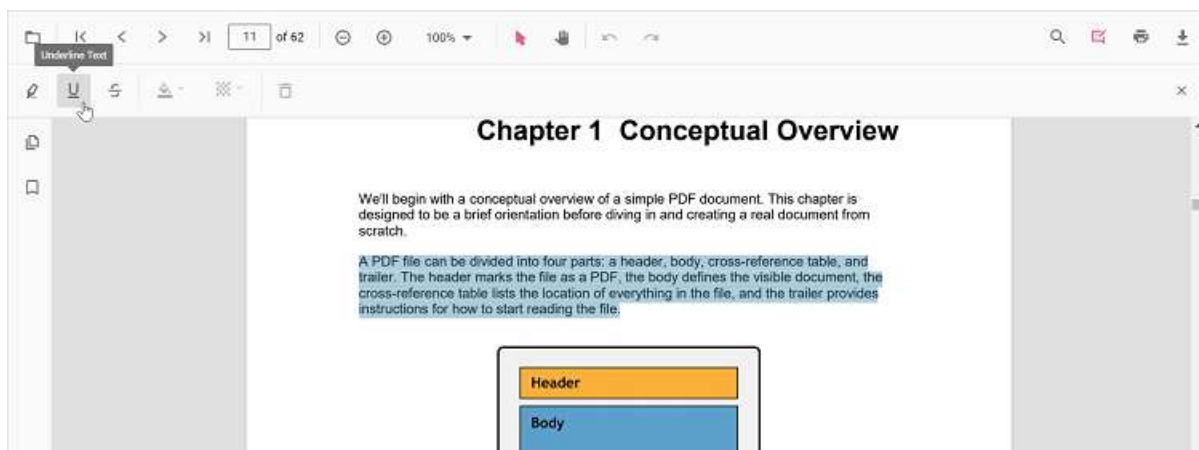
Underline a text

There are two ways to underline a text in the PDF document:

1. Using the context menu *Select a text in the PDF document and right-click it.* Select **Underline** option in the context menu that appears.



2. Using the annotation toolbar *Click the Edit Annotation button in the PDF Viewer toolbar.* A toolbar appears below it. Select the **Underline** button in the annotation toolbar. It enables the underline mode. *Select the text and the underline annotation will be added.* You can also select the text and apply the underline annotation using the **Underline** button.



In the pan mode, if the underline mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for underlining the text.

Refer to the following code snippet to switch to underline mode.

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';

```

```

import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
 '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function underlineMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Underline');
  }
  return ( <div>
    <button onClick={underlineMode}>Underline</button>
    <div className='control-section'>
      { /* Render the PDF Viewer */ }
      <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; } }
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
      style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
      LinkAnnotation, BookmarkView,
      ThumbnailView, Print, TextSelection, TextSearch]} />
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
 '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function underlineMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Underline');
  }
  return ( <div>
    <button onClick={underlineMode}>Underline</button>
    <div className='control-section'>
      { /* Render the PDF Viewer */ }
      <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; } }
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
      style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
      LinkAnnotation, BookmarkView,

```

```
ThumbnailView, Print, TextSelection, TextSearch]] />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Refer to the following code snippet to switch back to normal mode from underline mode.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
 '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function underlineMode() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Underline');
}
function normalMode () {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('None');
}
return ( <div>
<button onClick={underlineMode}>Underline</button>
<button onClick={normalMode}>Normal Mode</button>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]] />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
```

```

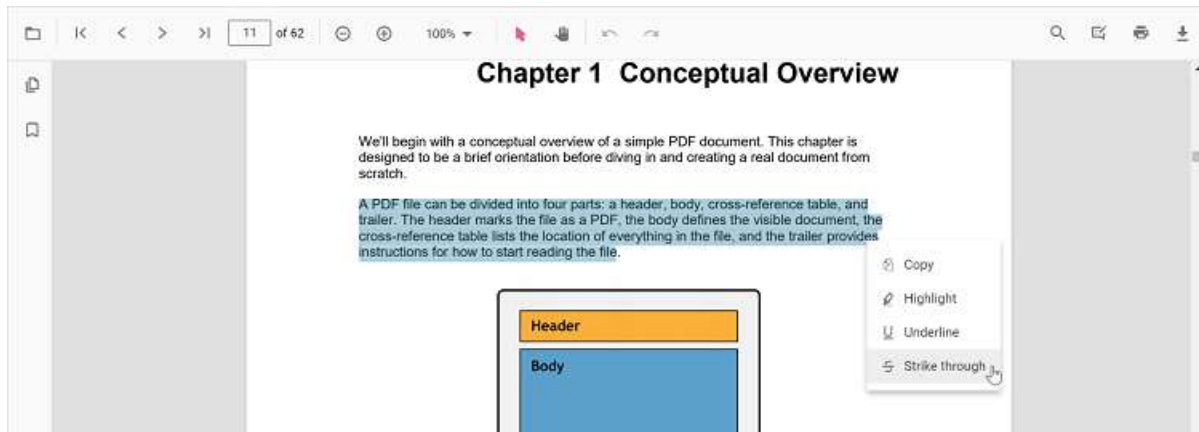
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function underlineMode() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Underline');
}
function normalMode () {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('None');
}
return ( <div>
<button onClick={underlineMode}>Underline</button>
<button onClick={normalMode}>Normal Mode</button>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

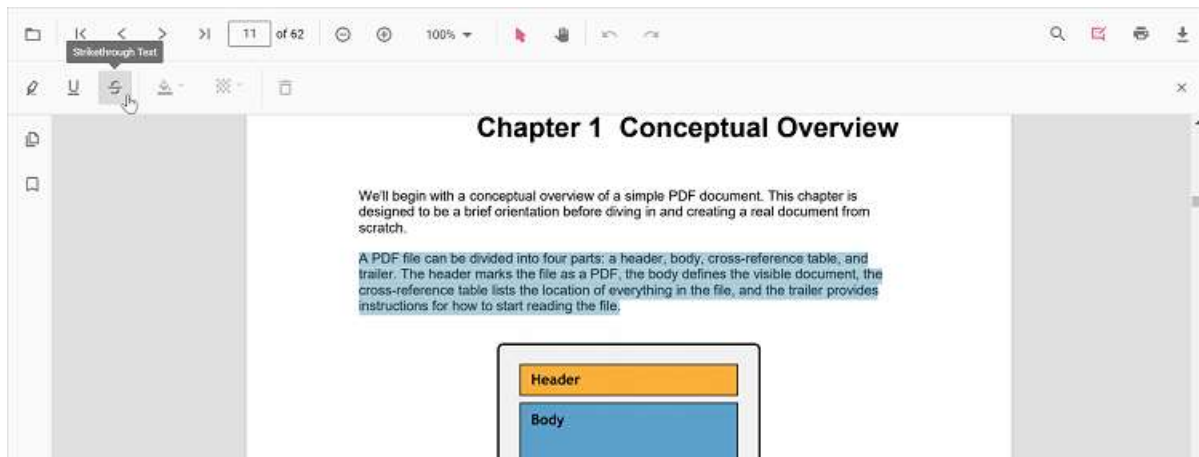
Strikethrough a text

There are two ways to strikethrough a text in the PDF document:

1. Using the context menu *Select a text in the PDF document and right-click it.* Select **Strikethrough** option in the context menu that appears.



2. Using the annotation toolbar Click the *Edit Annotation* button in the PDF Viewer toolbar. A toolbar appears below it. Select the **Strikethrough** button in the annotation toolbar. It enables the strikethrough mode. Select the text and the strikethrough annotation will be added. You can also select the text and apply the strikethrough annotation using the **Strikethrough** button.



In the pan mode, if the strikethrough mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for striking through the text.

Refer to the following code snippet to switch to strikethrough mode.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function strikethroughMode() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Strikethrough');
}
return ( <div>
<button onClick={strikethroughMode}>Strikethrough</button>
```

```

<div className='control-section'>
  {/* Render the PDF Viewer */}
  <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; }}
    id="container"
    documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
    resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
    style={{ 'height': '640px' }}>
    <Inject services=[ Toolbar, Annotation, Magnification, Navigation,
      LinkAnnotation, BookmarkView,
      ThumbnailView, Print, TextSelection, TextSearch ] />
  </PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView,
  ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
  '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function strikethroughMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Strikethrough');
  }
  return ( <div>
    <button onClick={strikethroughMode}>Strikethrough</button>
    <div className='control-section'>
      {/* Render the PDF Viewer */}
      <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        style={{ 'height': '640px' }}>
      <Inject services=[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch ] />
    </PdfViewerComponent>
    </div>
  </div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Refer to the following code snippet to switch back to normal mode from underline mode.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
 '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function strikethroughMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Strikethrough');
  }
  function normalMode () {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('None');
  }
  return (
    <div>
      <button onClick={strikethroughMode}>Strikethrough</button>
      <button onClick={normalMode}>Normal Mode</button>
      <div className='control-section'>
        {/ * Render the PDF Viewer */}
        <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
          style={{ 'height': '640px' }}>
          <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
            LinkAnnotation, BookmarkView,
            ThumbnailView, Print, TextSelection, TextSearch]} />
        </PdfViewerComponent>
      </div>
    </div>
  );
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
 '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function strikethroughMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
```

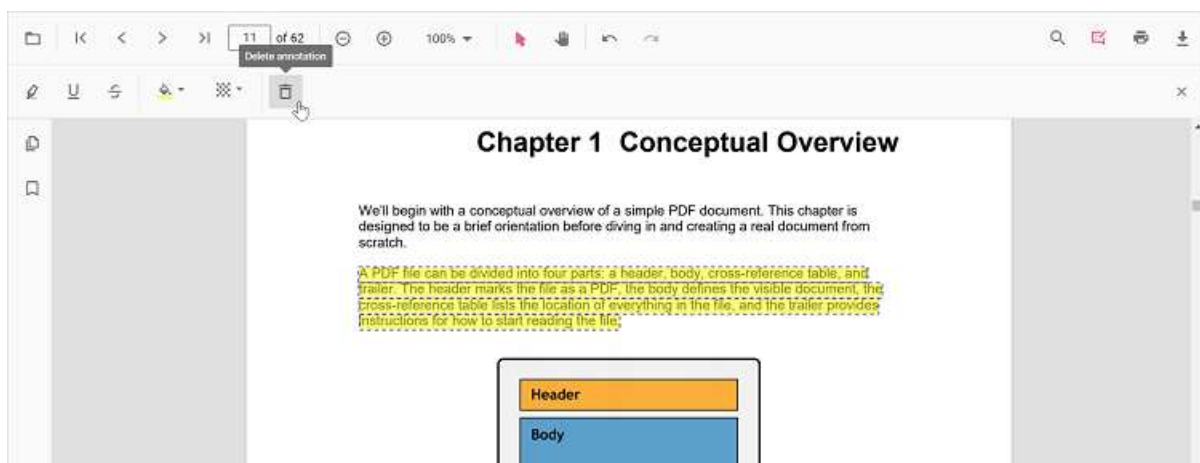


```
viewer.annotation.setAnnotationMode('Strikethrough');
}
function normalMode () {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('None');
}
return (
<div>
<button onClick={strikethroughMode}>Strikethrough</button>
<button onClick={normalMode}>Normal Mode</button>
<div className='control-section'>
  { /* Render the PDF Viewer */ }
  <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; } }
  id="container"
  documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
  serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
  style={{ 'height': '640px' }}>
    <Inject services={ [ Toolbar, Annotation, Magnification, Navigation,
    LinkAnnotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch ] } />
  </PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}
```

Deleting a text markup annotation

The selected annotation can be deleted by the following ways:

1. Using Delete key *Select the annotation to be deleted.* Click the Delete key in the keyboard. The selected annotation will be deleted.
2. Using the annotation toolbar *Select the annotation to be deleted.* Click the **Delete Annotation** button in the annotation toolbar. The selected annotation will be deleted.

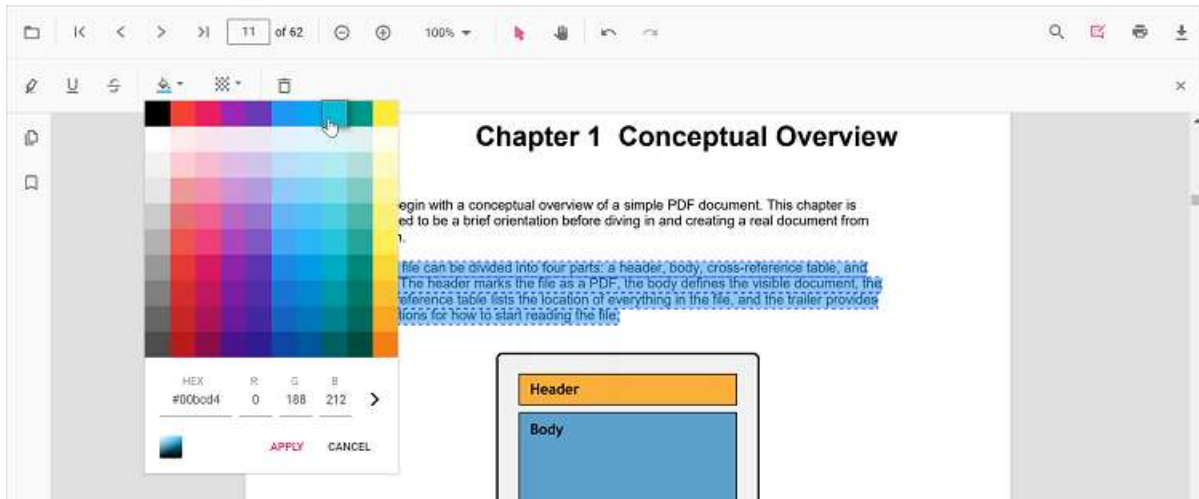


Editing the properties of the text markup annotation

The color and the opacity of the text markup annotation can be edited using the Edit Color tool and the Edit Opacity tool in the annotation toolbar.

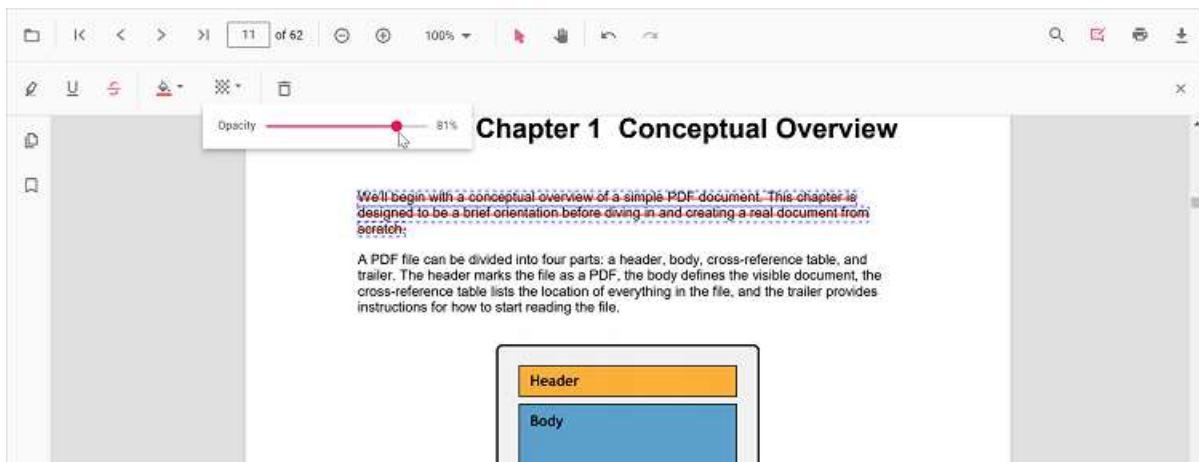
Editing color

The color of the annotation can be edited using the color palette provided in the Edit Color tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Setting default properties during control initialization

The properties of the text markup annotation can be set before creating the control using `highlightSettings`, `underlineSettings`, and `strikethroughSettings`.

After editing the default color and opacity using the Edit Color tool and Edit Opacity tool, they will be changed to the selected values.

Refer to the following code snippet to set the default annotation settings.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
```

```
function App() {
  return (
    <div>
      <div className='control-section'>
        {/* Render the PDF Viewer */}
        <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
          style={{ 'height': '640px' }}
          highlightSettings = {{author: 'Guest User', subject: 'Important', color:
            '#ffff00', opacity: 0.9, modifiedDate: ''}}
          underlineSettings = {{author: 'Guest User', subject: 'Points to be
            remembered', color: '#00ffff', opacity: 0.9, modifiedDate: ''}}
          strikethroughSettings = {{author: 'Guest User', subject: 'Not Important',
            color: '#ff00ff', opacity: 0.9, modifiedDate: ''}}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation, BookmarkView,
          ThumbnailView, Print, TextSelection, TextSearch]} />
        </PdfViewerComponent>
      </div>
    </div>
  );
}

const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView,
  ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
  '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (
    <div>
      <div className='control-section'>
        {/* Render the PDF Viewer */}
        <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
          style={{ 'height': '640px' }}
          highlightSettings = {{author: 'Guest User', subject: 'Important', color:
            '#ffff00', opacity: 0.9, modifiedDate: ''}}
          underlineSettings = {{author: 'Guest User', subject: 'Points to be
            remembered', color: '#00ffff', opacity: 0.9, modifiedDate: ''}}
          strikethroughSettings = {{author: 'Guest User', subject: 'Not Important',
            color: '#ff00ff', opacity: 0.9, modifiedDate: ''}}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation, BookmarkView,
```

```
ThumbnailView, Print, TextSelection, TextSearch]] />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Performing undo and redo

The PDF Viewer performs undo and redo for the changes made in the PDF document. In text markup annotation, undo and redo actions are provided for:

- Inclusion of the text markup annotations.
- Deletion of the text markup annotations.
- Change of either color or opacity of the text markup annotations.

Undo and redo actions can be done by the following ways:

1.Using keyboard shortcuts:

After performing a text markup annotation action, you can undo it by using Ctrl + Z shortcut and redo by using Ctrl + Y shortcut.

2.Using toolbar:

Undo and redo can be done using the **Undo** tool and **Redo** tool provided in the toolbar.

Refer to the following code snippet for calling undo and redo actions from the client-side.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function undo() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.undo();
}
function redo () {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.redo();
}
return (
<div>
<button onClick={undo}>Undo</button>
<button onClick={redo}>Redo</button>
<div className='control-section'>
{ /* Render the PDF Viewer */}
```

```

<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function undo() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.undo();
}
function redo () {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.redo();
}
return (
<div>
<button onClick={undo}>Undo</button>
<button onClick={redo}>Redo</button>
<div className='control-section'>
/* Render the PDF Viewer */
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));

```

```
root.render(<App />);
{% endraw %}
```

Saving the text markup annotation

When you click the download tool in the toolbar, the text markup annotations will be saved in the PDF document. This action will not affect the original document.

Printing the text markup annotation

When the print tool is selected in the toolbar, the PDF document will be printed along with the text markup annotations added to the pages. This action will not affect the original document.

Disabling text markup annotation

The PDF Viewer control provides an option to disable the text markup annotation feature. The code snippet for disabling the feature is as follows.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (
    <div>
      <div className='control-section'>
        {/ * Render the PDF Viewer */}
        <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
          enableTextMarkupAnnotation = {false}
          style={{ 'height': '640px' }}>
          <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
            LinkAnnotation, BookmarkView,
            ThumbnailView, Print, TextSelection, TextSearch]} />
        </PdfViewerComponent>
      </div>
    </div>
  );
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
```

```
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (
    <div>
      <div className='control-section'>
        { /* Render the PDF Viewer */ }
        <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; } }
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          enableTextMarkupAnnotation = { false }
          serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
          style={{ 'height': '640px' }}>
          <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
            LinkAnnotation, BookmarkView,
            ThumbnailView, Print, TextSelection, TextSearch]} />
        </PdfViewerComponent>
      </div>
    </div>
  );
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

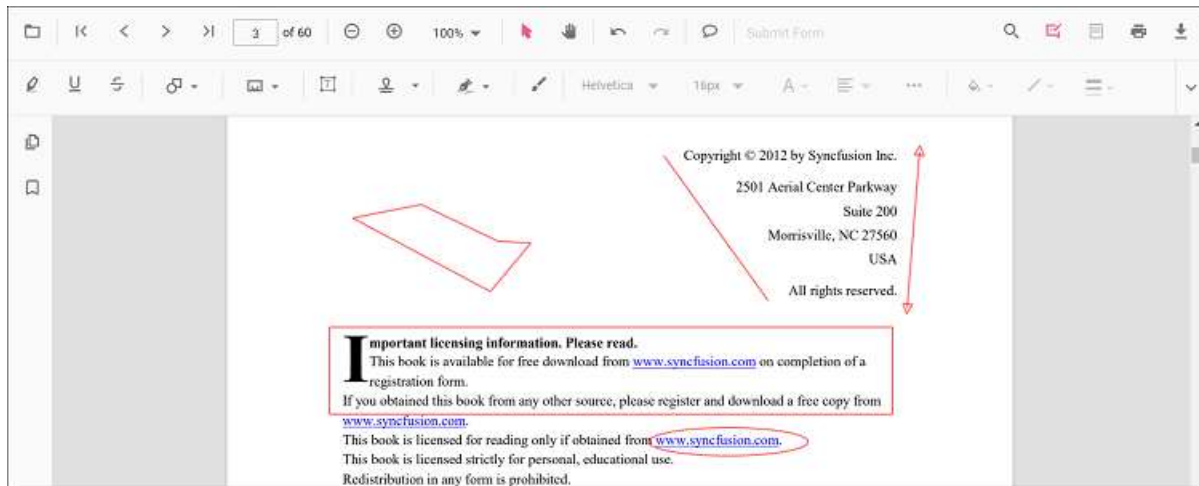
See also

- [Toolbar items](#)
- [Feature Modules](#)

Shape annotation in React Pdfviewer component

The PDF Viewer control provides the options to add, edit, and delete the shape annotations. The shape annotation types supported in the PDF Viewer control are:

- Line
- Arrow
- Rectangle
- Circle
- Polygon

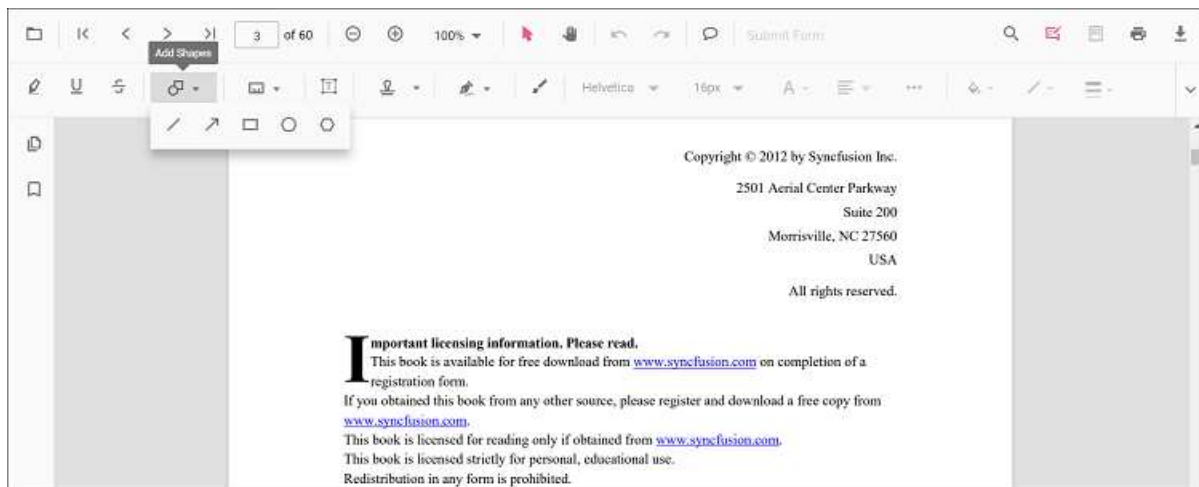


Adding a shape annotation to the PDF document

Shape annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Shape Annotation** drop-down button. A drop-down pop-up will appear and shows the shape annotations to be added.
- Select the shape types to be added to the page in the drop-down pop-up. It enables the selected shape annotation mode.
- You can add the shapes over the pages of the PDF document.

In the pan mode, if the shape annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code sample to switch to the circle annotation mode.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
```



```

Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
  function circleMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Circle');
  }
  return (<div>
    <button onClick={circleMode}>Circle</button>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        style={{ 'height': '640px' }}>
      <Inject services=[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch ]/>
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
  function circleMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Circle');
  }
  return (<div>
    <button onClick={circleMode}>Circle</button>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        style={{ 'height': '640px' }}>
      <Inject services=[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch ]/>
    </PdfViewerComponent>
  </div>
</div>);

```

```

}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}

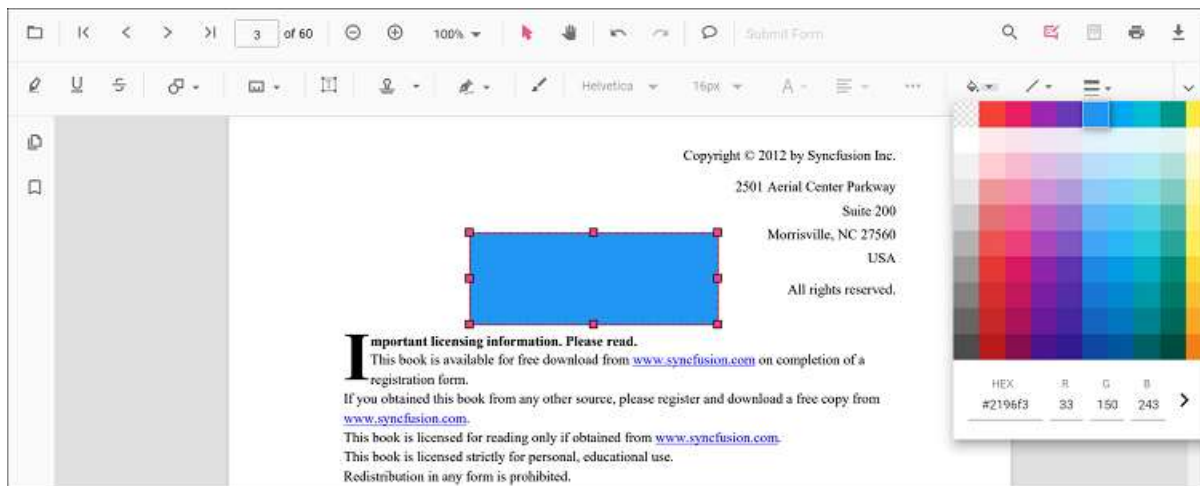
```

Editing the properties of the shape annotation

The fill color, stroke color, thickness, and opacity of the shape annotation can be edited using the Edit color tool, Edit stroke color tool, Edit thickness tool, and Edit opacity tool in the annotation toolbar.

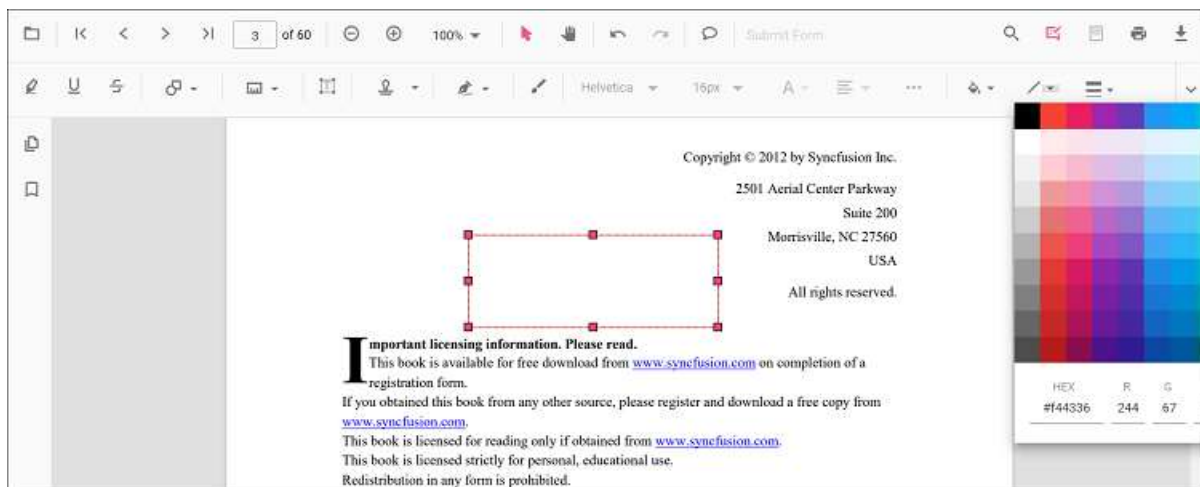
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



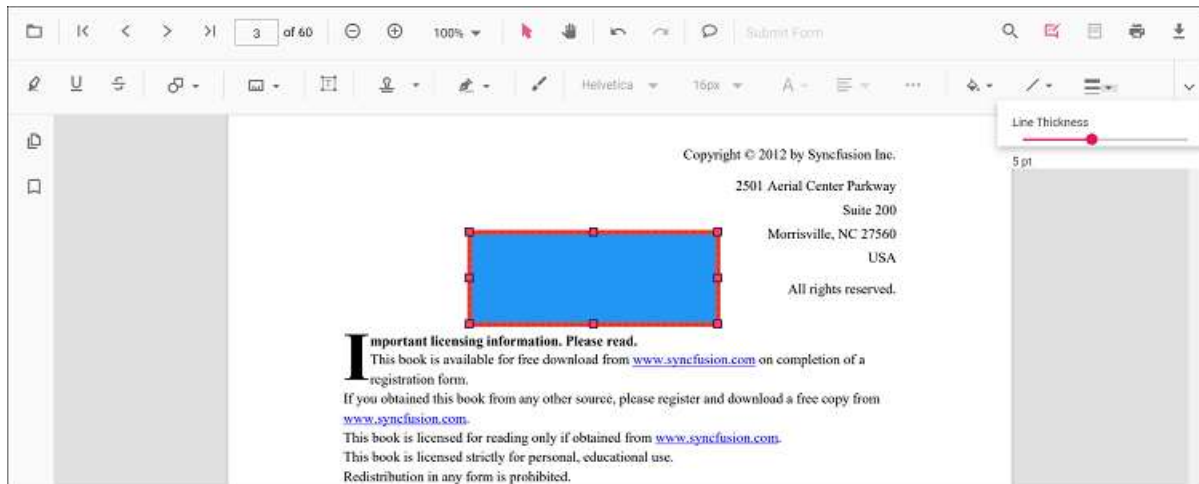
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



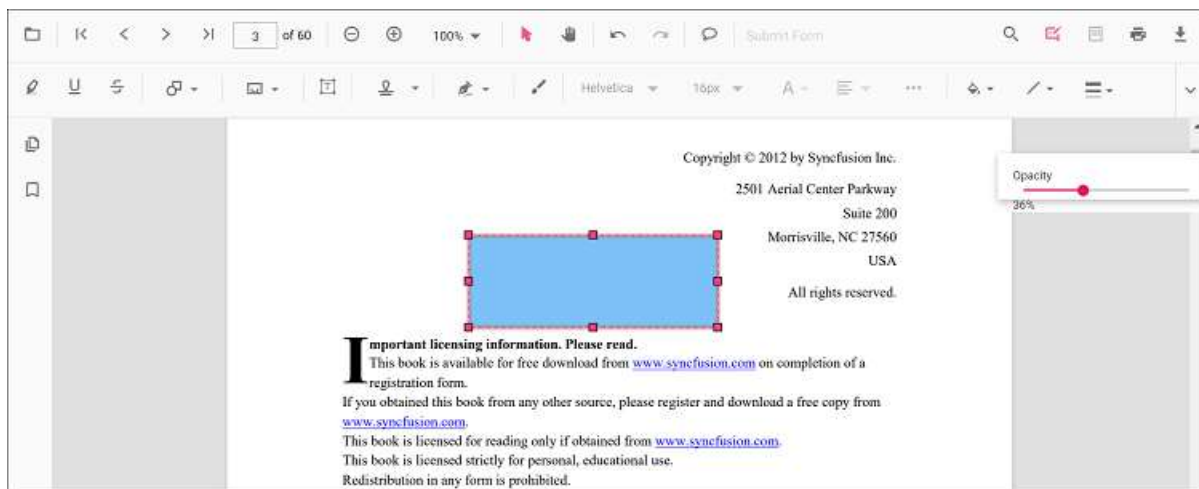
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

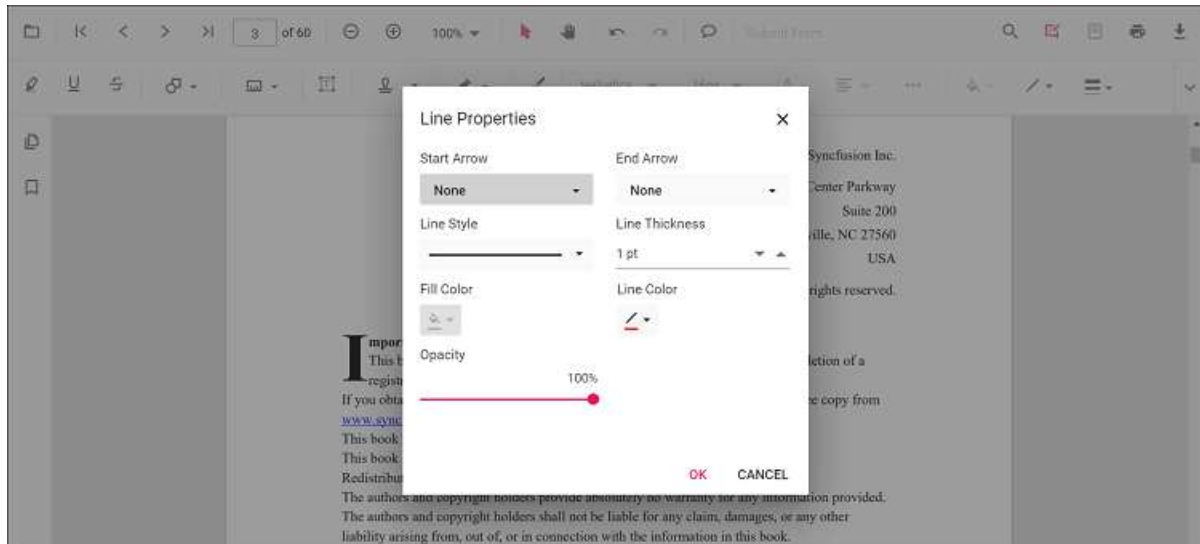
The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Editing the line properties

The properties of the line shapes such as line and arrow annotations can be edited using the Line Properties window. It can be opened by selecting the Properties option in the context menu that appears on right-clicking the line and arrow annotations.

Refer to the following code sample to set the default annotation settings.



Edit annotation programmatically

We can edit the annotations programmatically using the **editAnnotation()** method.

Here is an example of how you can use this method to modify an annotation:

```
<button onclick="editAnnotation()">Edit Annotation</button>
<script>
//Edit Annotation
function editAnnotation(){
var pdfviewer = document.getElementById('container').ej2_instances[0];
pdfviewer.annotationModule.selectAnnotation(pdfviewer.annotationCollection[0].annotationId);
pdfviewer.annotationCollection[0].opacity="0.5";
pdfviewer.annotation.editAnnotation(pdfviewer.annotationCollection[0]);
}
</script>
```

Delete annotation programmatically

We can delete a specific annotation using the **deleteAnnotationById()** method. This method is used to delete a specific annotation using its id.

Here is an example of how you can use this method to delete an annotation:

```
<button onclick="deleteAnnotationById()">Delete Annotation by ID</button>
<script>
//Delete Annotation by id.
```

```
function deleteAnnotationById() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotationModule.deleteAnnotationById(viewer.annotationCollection[0].annotationId);
}
</script>
`
```

Setting default properties during the control initialization

The properties of the shape annotations can be set before creating the control using LineSettings, ArrowSettings, RectangleSettings, CircleSettings, and PolygonSettings.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
return (<div>
<div className='control-section'>
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
lineSettings={{fillColor: 'blue', opacity: 0.6, strokeColor: 'green'}}
arrowSettings={{fillColor: 'green', opacity: 0.6, strokeColor: 'blue'}}
rectangleSettings={{fillColor: 'yellow', opacity: 0.6, strokeColor:
'orange'}}
circleSettings={{fillColor: 'orange', opacity: 0.6, strokeColor: 'pink'}}
polygonSettings={{fillColor: 'pink', opacity: 0.6, strokeColor: 'yellow'}}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]}/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

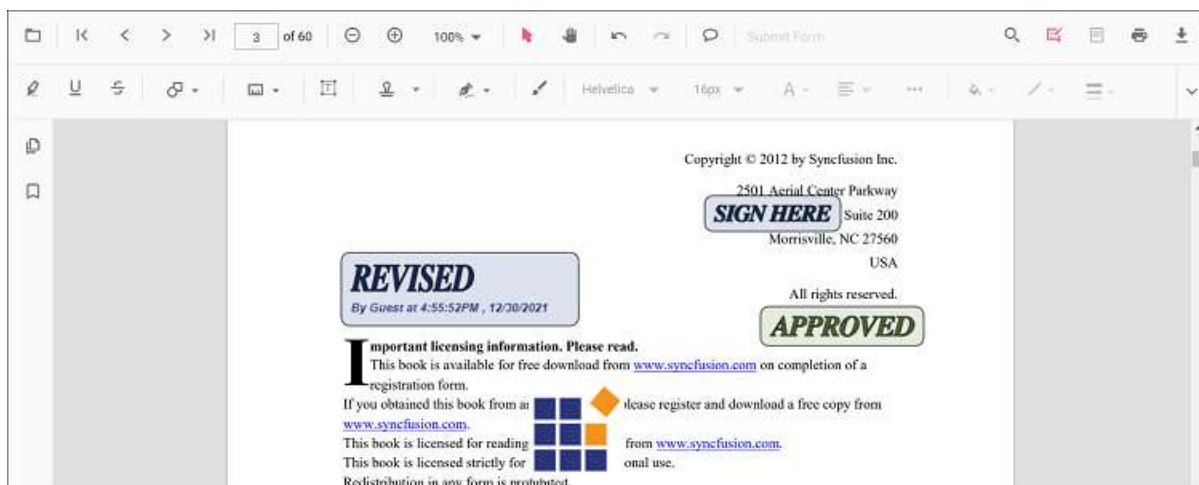
```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
```

```
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        lineSettings={{fillColor: 'blue', opacity: 0.6, strokeColor: 'green'}}
        arrowSettings={{fillColor: 'green', opacity: 0.6, strokeColor: 'blue'}}
        rectangleSettings={{fillColor: 'yellow', opacity: 0.6, strokeColor:
          'orange'}}
        circleSettings={{fillColor: 'orange', opacity: 0.6, strokeColor: 'pink'}}
        polygonSettings={{fillColor: 'pink', opacity: 0.6, strokeColor: 'yellow'}}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation, BookmarkView,
          ThumbnailView, Print, TextSelection, TextSearch]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Stamp annotation in React Pdfviewer component

The PDF Viewer control provides options to add, edit, delete, and rotate the following stamp annotation in the PDF documents:

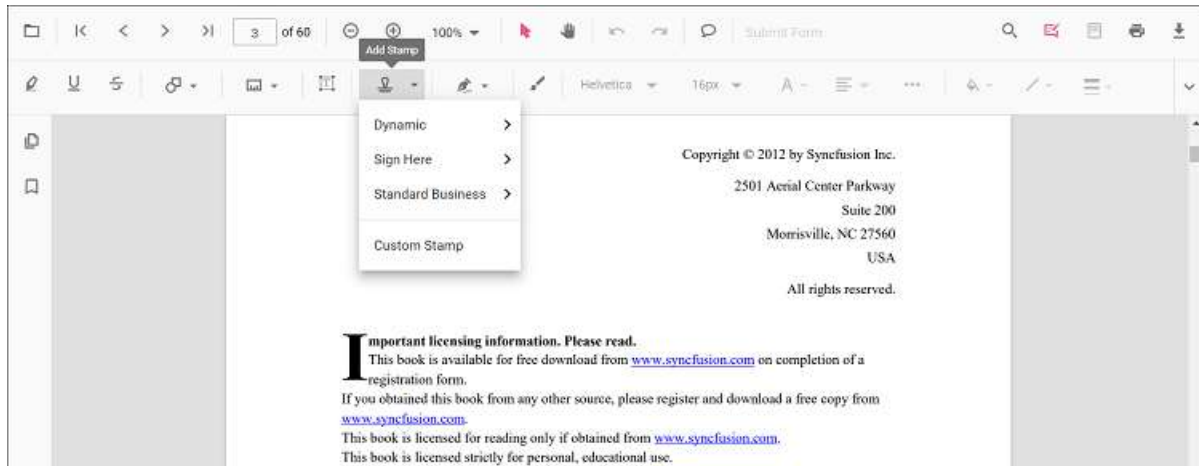
- Dynamic
- Sign Here
- Standard Business
- Custom Stamp



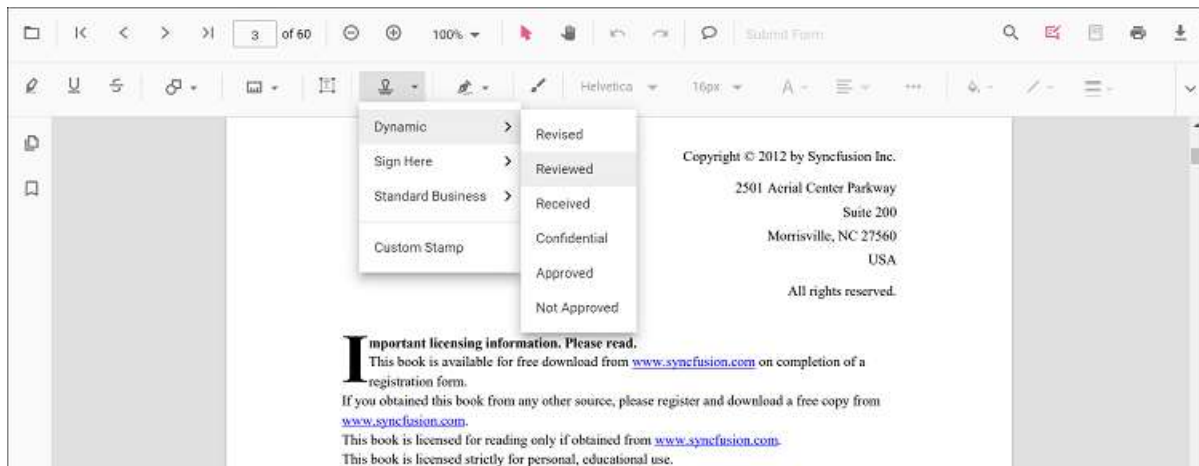
Adding stamp annotations to the PDF document

The stamp annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Stamp Annotation** drop-down button. A drop-down pop-up will appear and shows the stamp annotations to be added.



- Select the annotation type to be added to the page in the pop-up.



- You can add the annotation over the pages of the PDF document.

In the pan mode, if the stamp annotation mode is entered, the PDF Viewer control will switch to text select mode.

Refer to the following code sample to switch to the stamp annotation mode.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
```

```

import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection,
TextSearch, Annotation, SignStampItem, StandardBusinessStampItem,
DynamicStampItem, Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function dynamicStamp() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Stamp', DynamicStampItem.NotApproved);
}
function signStamp() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Stamp', null, SignStampItem.Witness);
}
function standardBusinessStamp() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Stamp', null, null,
StandardBusinessStampItem.Approved,);
}
return (<div>
<button onClick={dynamicStamp}>Dynamic Stamp</button>
<button onClick={signStamp}>Sign Stamp</button>
<button onClick={standardBusinessStamp}>Standard Business Stamp</button>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch ]/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection,
TextSearch, Annotation, SignStampItem, StandardBusinessStampItem,
DynamicStampItem, Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function dynamicStamp() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Stamp', DynamicStampItem.NotApproved);
}
function signStamp() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Stamp', null, SignStampItem.Witness);
}

```



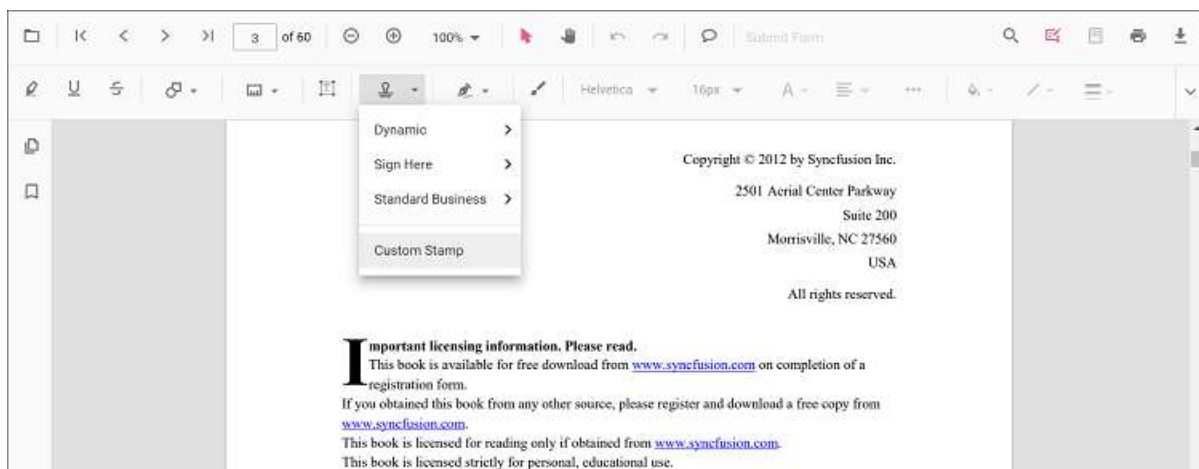
```

}
function standardBusinessStamp() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Stamp', null, null,
StandardBusinessStampItem.Approved,);
}
return (<div>
<button onClick={dynamicStamp}>Dynamic Stamp</button>
<button onClick={signStamp}>Sign Stamp</button>
<button onClick={standardBusinessStamp}>Standard Business Stamp</button>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch ]/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}

```

Adding custom stamp to the PDF document

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Stamp Annotation** drop-down button. A drop-down pop-up will appear and shows the stamp annotations to be added.
- Click the Custom Stamp button.



- The file explorer dialog will appear, choose the image and then add the image to the PDF page.

The JPG and JPEG image format is only supported in the custom stamp annotations.

Setting default properties during control initialization

The properties of the stamp annotation can be set before creating the control using the StampSettings.

After editing the default opacity using the Edit Opacity tool, they will be changed to the selected values.

Refer to the following code sample to set the default sticky note annotation settings.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        stampSettings={{opacity: 0.3, author: 'Guest User'}}
        style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch]}/>
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        stampSettings={{opacity: 0.3, author: 'Guest User'}}
        style={{ 'height': '640px' }}>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

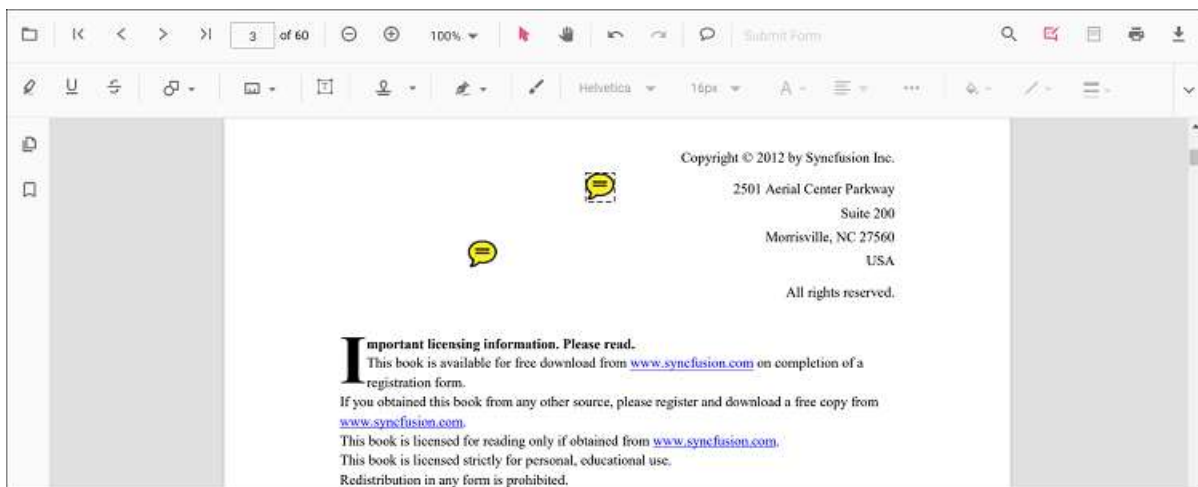
```

<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]}/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Sticky notes annotation in React Pdfviewer component

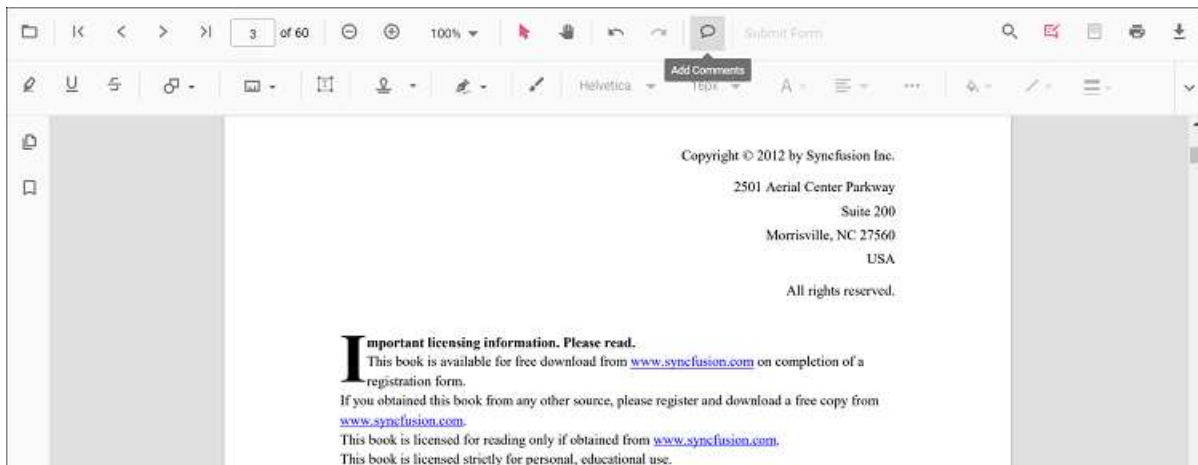
The PDF Viewer control provides the options to add, edit, and delete the sticky note annotations in the PDF document.



Adding a sticky note annotation to the PDF document

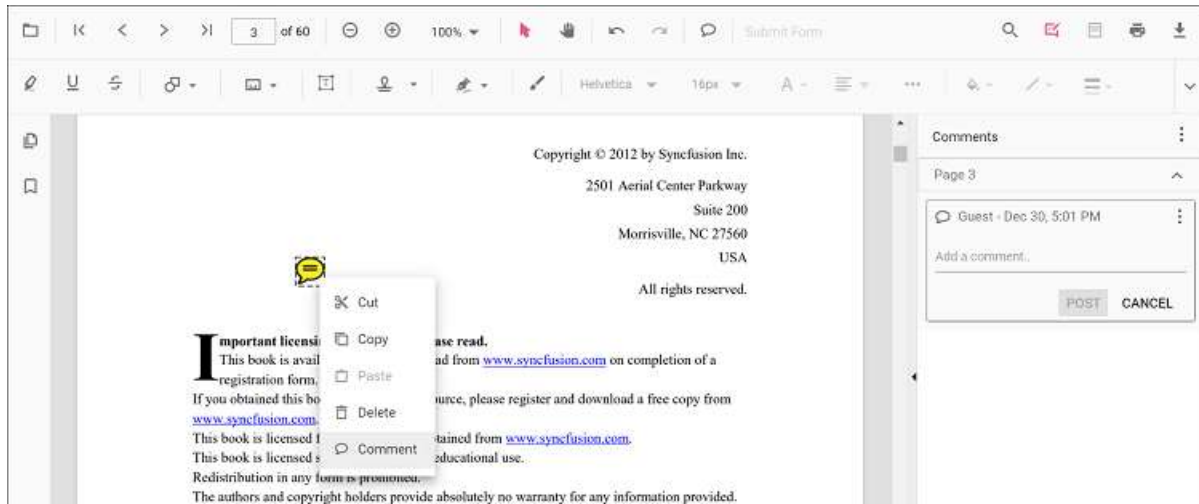
Sticky note annotations can be added to the PDF document using the annotation toolbar.

- Click the **Comments** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the position where you want to add sticky note annotation in the PDF document.
- Sticky note annotation will be added in the clicked positions.



Annotation comments can be added to the PDF document using the comment panel.

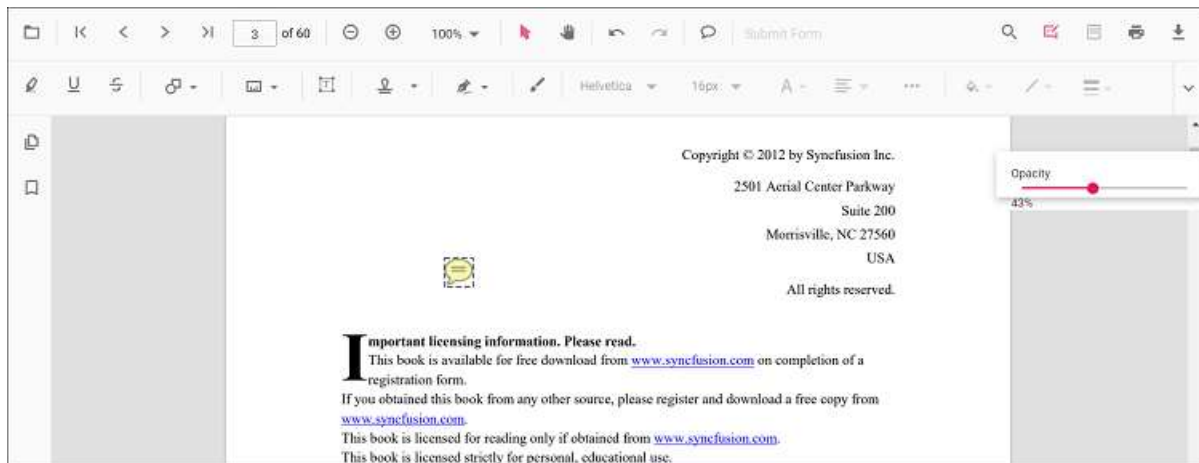
- Select a Sticky note annotation in the PDF document and right-click it.
- Select the Comment option in the context menu that appears.
- Now, you can add Comments, Reply, and Status using the Comment Panel.
- Now, you can add Comments, Reply, and Status using the Comment Panel.



Editing the properties of the sticky note annotation

Editing opacity

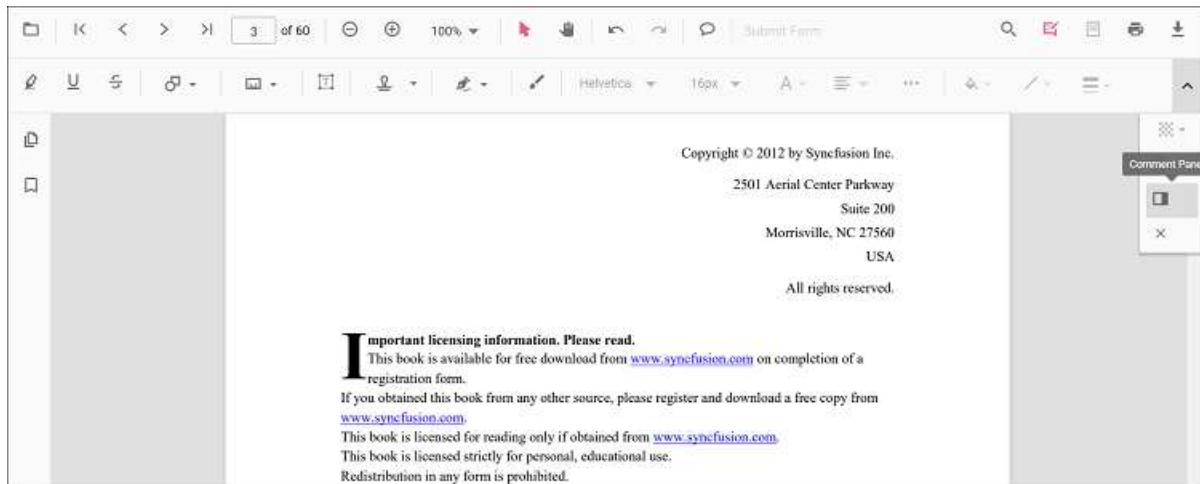
The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



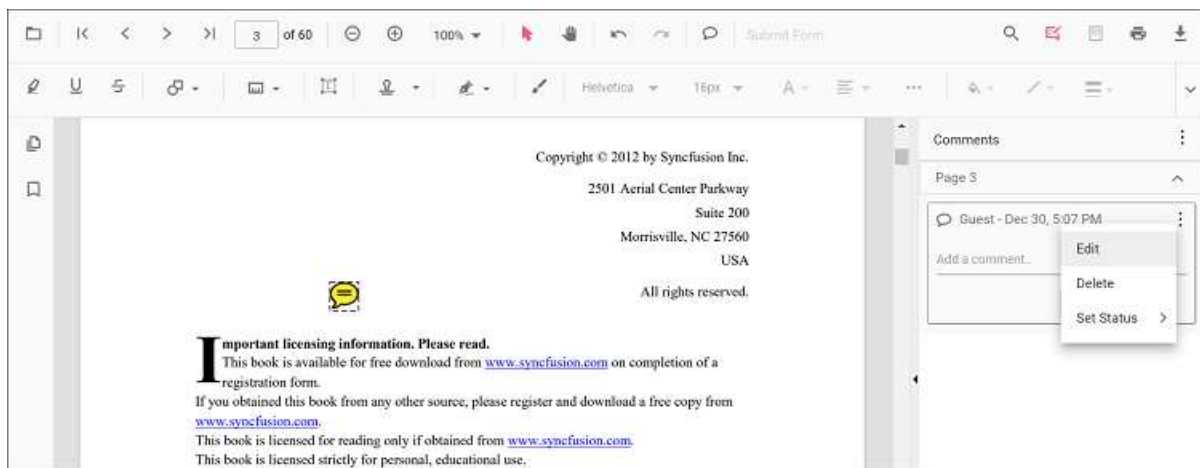
Editing comments

The comment, comment reply, and comment status of the annotation can be edited using the Comment Panel.

- Open the comment panel using the Comment Panel button showing in the annotation toolbar.



You can modify or delete the comments or comments replay and it's status using the menu option provided in the comment panel.



Setting default properties during the control initialization

The properties of the sticky note annotation can be set before creating the control using the `StickyNoteSettings`.

After editing the default opacity using the Edit Opacity tool, they will be changed to the selected values. Refer to the following code sample to set the default sticky note annotation settings.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
return (<div>
<div className='control-section'>
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
```

```

documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
stickyNotesSettings={{author: 'Syncfusion'}}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]}/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
return <div>
<div className='control-section'>
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
stickyNotesSettings={{author: 'Syncfusion'}}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]}/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Disabling sticky note annotations

The PDF Viewer control provides an option to disable the sticky note annotations feature. The code sample for disabling the feature is as follows.

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';

```

```

import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        enableStickyNotesAnnotation={false}
        style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch]}/>
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

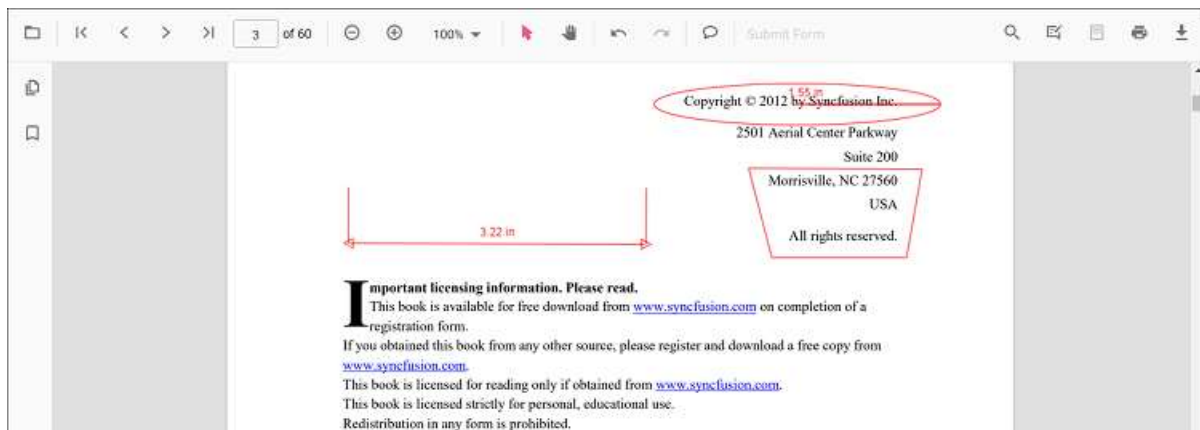
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        enableStickyNotesAnnotation={false}
        style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch]}/>
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Measurement annotation in React Pdfviewer component

The PDF Viewer provides the options to add measurement annotations. You can measure the page annotations with the help of measurement annotation. The supported measurement annotations in the PDF Viewer control are:

- Distance
- Perimeter
- Area
- Radius
- Volume

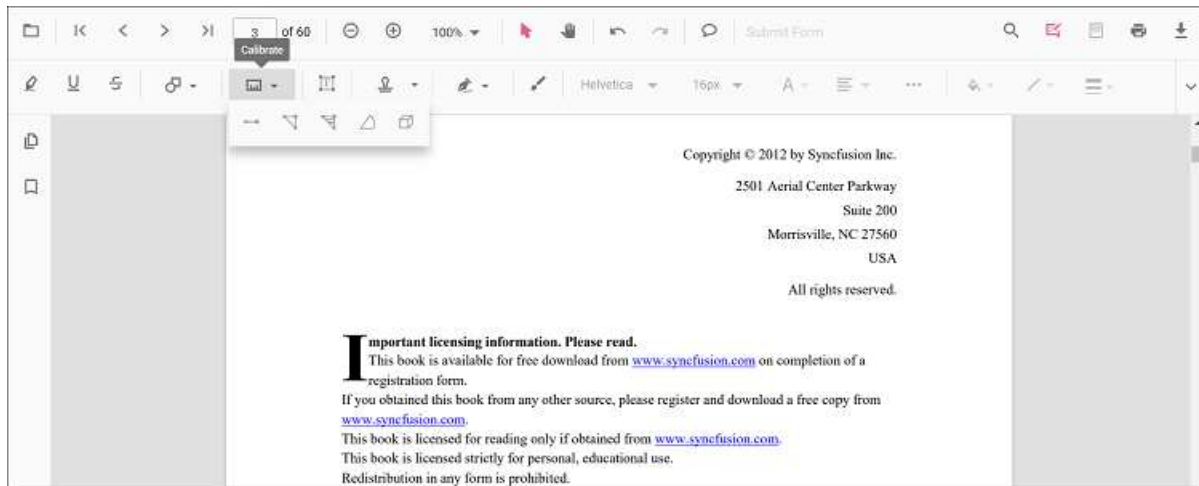


Adding measurement annotations to the PDF document

The measurement annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Measurement Annotation** dropdown button. A dropdown pop-up will appear and shows the measurement annotations to be added.
- Select the measurement type to be added to the page in the dropdown pop-up. It enables the selected measurement annotation mode.
- You can measure and add the annotation over the pages of the PDF document.

In the pan mode, if the measurement annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code snippet to switch to distance annotation mode.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function distanceMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('Distance');
  }
  return (<div>
    <button onClick={distanceMode}>Distance</button>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch ]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
```

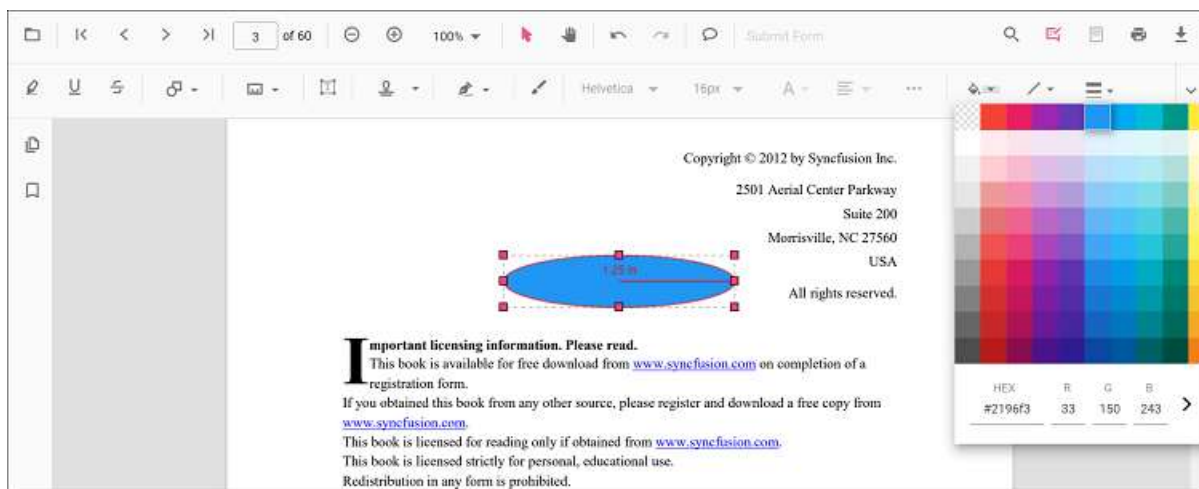
```
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function distanceMode() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Distance');
}
return (<div>
<button onClick={distanceMode}>Distance</button>
<div className='control-section'>
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch ]}/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Editing the properties of measurement annotation

The fill color, stroke color, thickness, and opacity of the measurement annotation can be edited using the Edit Color tool, Edit Stroke Color tool, Edit Thickness tool, and Edit Opacity tool in the annotation toolbar.

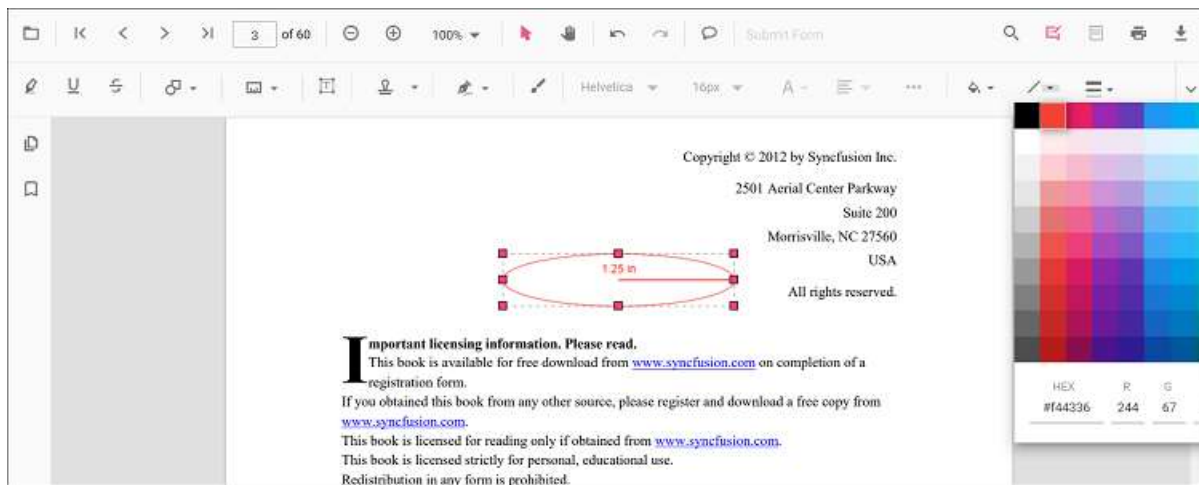
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



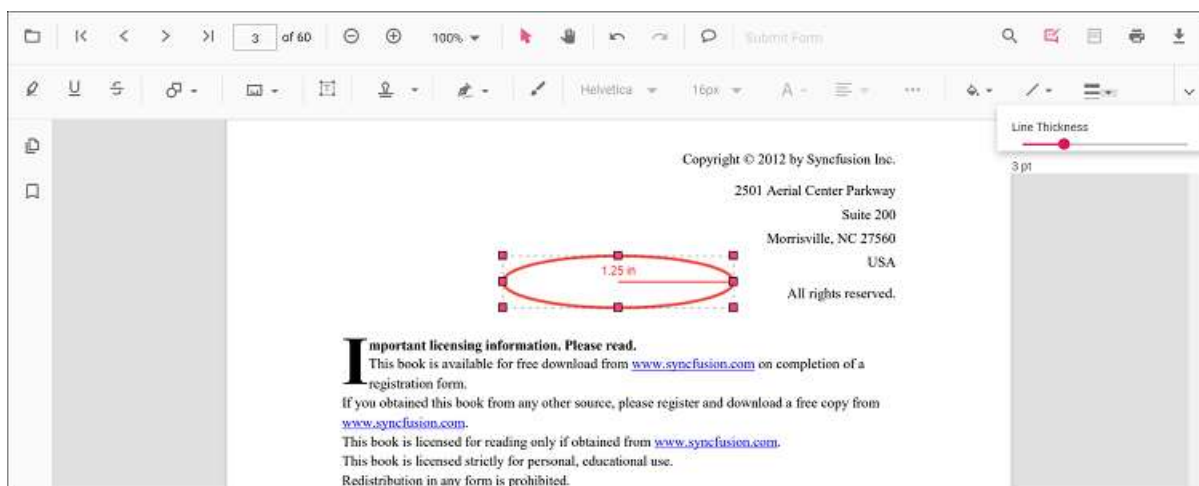
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



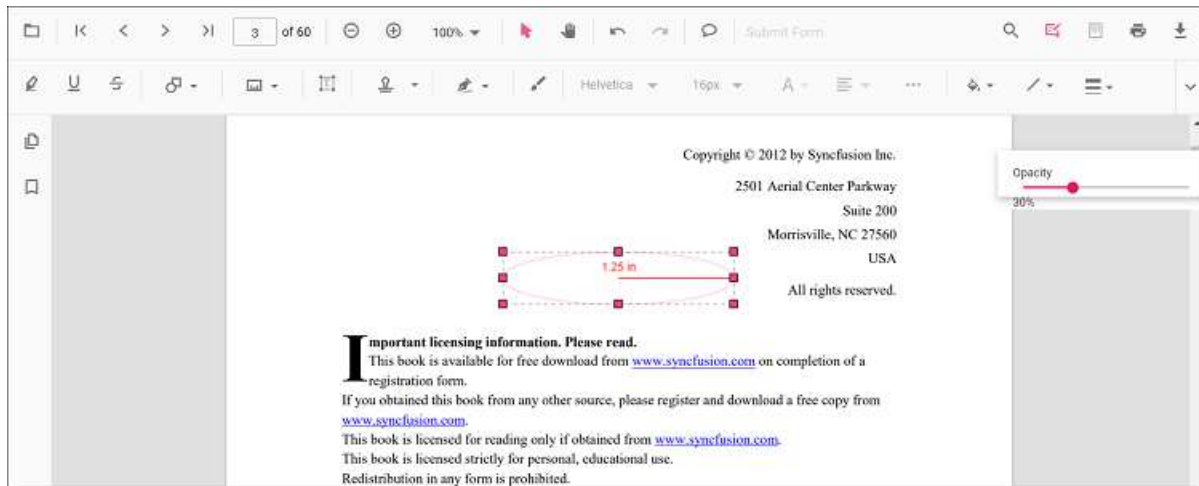
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit thickness tool.



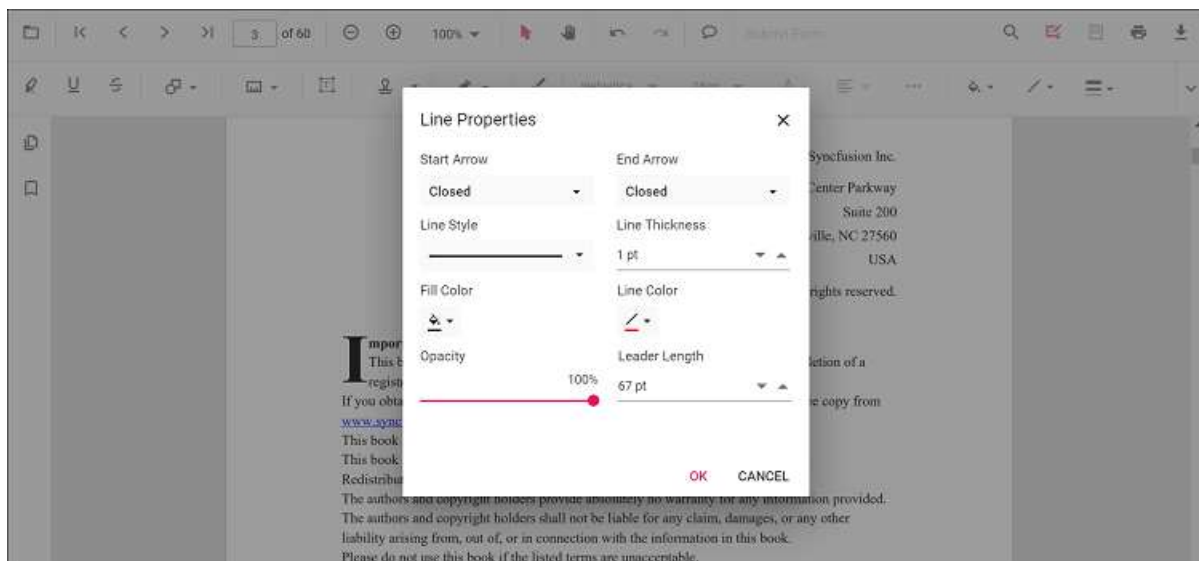
Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Editing the line properties

The properties of the line shapes such as distance and perimeter annotations can be edited using the Line properties window. It can be opened by selecting the Properties option in the context menu that appears on right-clicking the distance and perimeter annotations.



Setting default properties during control initialization

The properties of the shape annotations can be set before creating the control using `distanceSettings`, `perimeterSettings`, `areaSettings`, `radiusSettings` and `volumeSettings`.

Refer to the following code snippet to set the default annotation settings.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
```

```

export function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        distanceSettings={{fillColor: 'blue', opacity: 0.6, strokeColor: 'green'}}
        perimeterSettings={{fillColor: 'green', opacity: 0.6, strokeColor: 'blue'}}
        areaSettings={{fillColor: 'yellow', opacity: 0.6, strokeColor: 'orange'}}
        radiusSettings={{fillColor: 'orange', opacity: 0.6, strokeColor: 'pink'}}
        volumeSettings={{fillColor: 'pink', opacity: 0.6, strokeColor: 'yellow'}}
        style={{ 'height': '640px' }}>
      <Inject services=[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch ]/>
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

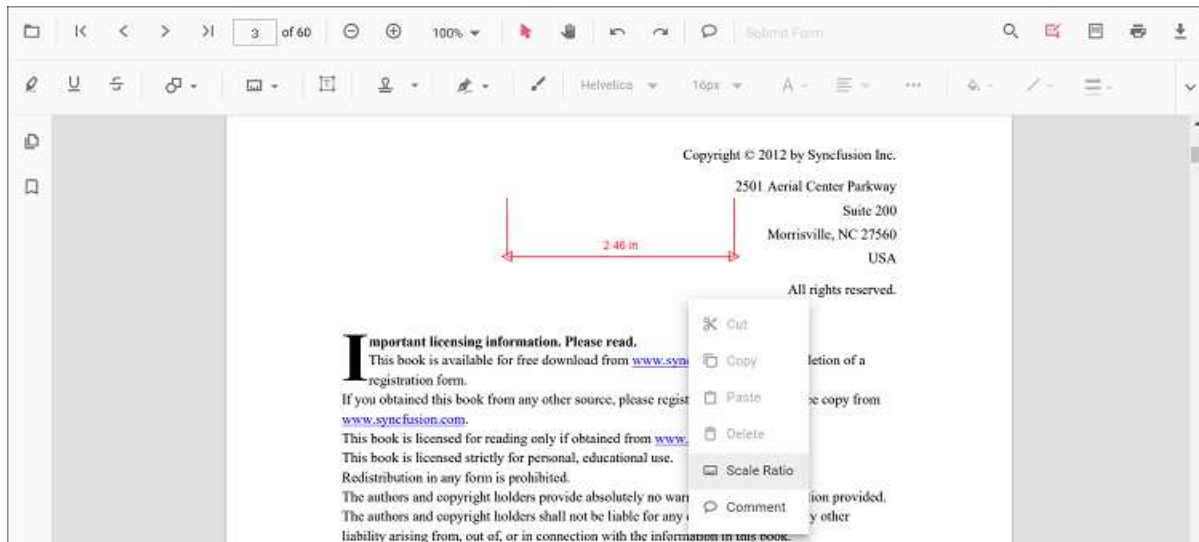
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView, ThumbnailView,
  Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
  react-pdfviewer';
let pdfviewer;
export function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        distanceSettings={{fillColor: 'blue', opacity: 0.6, strokeColor: 'green'}}
        perimeterSettings={{fillColor: 'green', opacity: 0.6, strokeColor: 'blue'}}
        areaSettings={{fillColor: 'yellow', opacity: 0.6, strokeColor: 'orange'}}
        radiusSettings={{fillColor: 'orange', opacity: 0.6, strokeColor: 'pink'}}
        volumeSettings={{fillColor: 'pink', opacity: 0.6, strokeColor: 'yellow'}}
        style={{ 'height': '640px' }}>
      <Inject services=[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch ]/>
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

```
{% enddraw %}
```

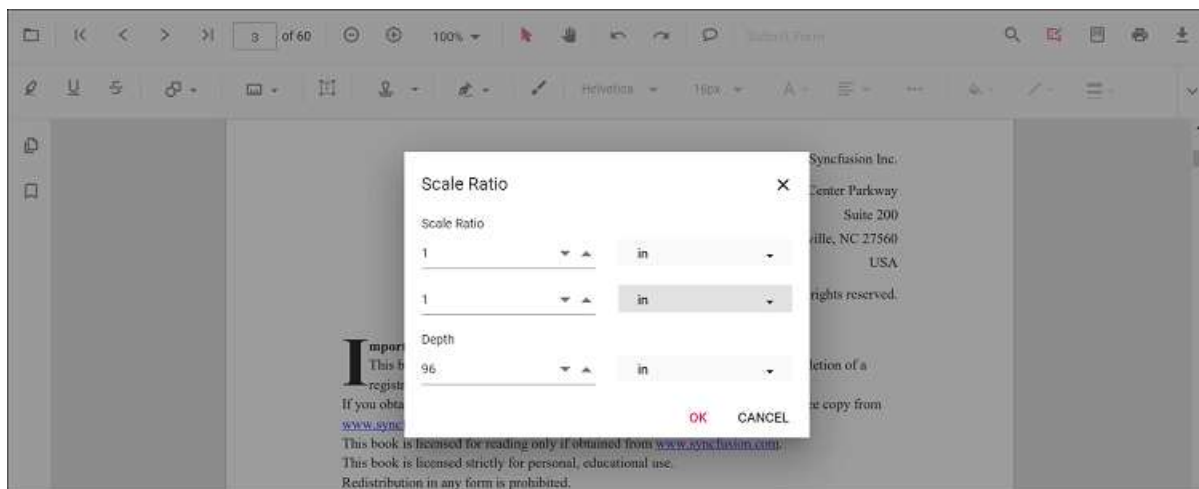
Editing scale ratio and unit of the measurement annotation

The scale ratio and unit of measurement can be modified using the scale ratio option provided in the context menu of the PDF Viewer control.



The Units of measurements support for the measurement annotations in the PDF Viewer are

- Inch
- Millimeter
- Centimeter
- Point
- Pica
- Feet



Setting default scale ratio settings during control initialization

The properties of scale ratio for measurement annotation can be set before creating the control using `ScaleRatioSettings` as shown in the following code snippet,

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        measurementSettings={{scaleRatio: 2, conversionUnit: 'cm', displayUnit:
          'cm'}}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation, BookmarkView, ThumbnailView,
          Print, TextSelection, TextSearch]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        measurementSettings={{scaleRatio: 2, conversionUnit: 'cm', displayUnit:
          'cm'}}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation, BookmarkView, ThumbnailView,
          Print, TextSelection, TextSearch]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}
```

```

</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Free text annotation in React Pdfviewer component

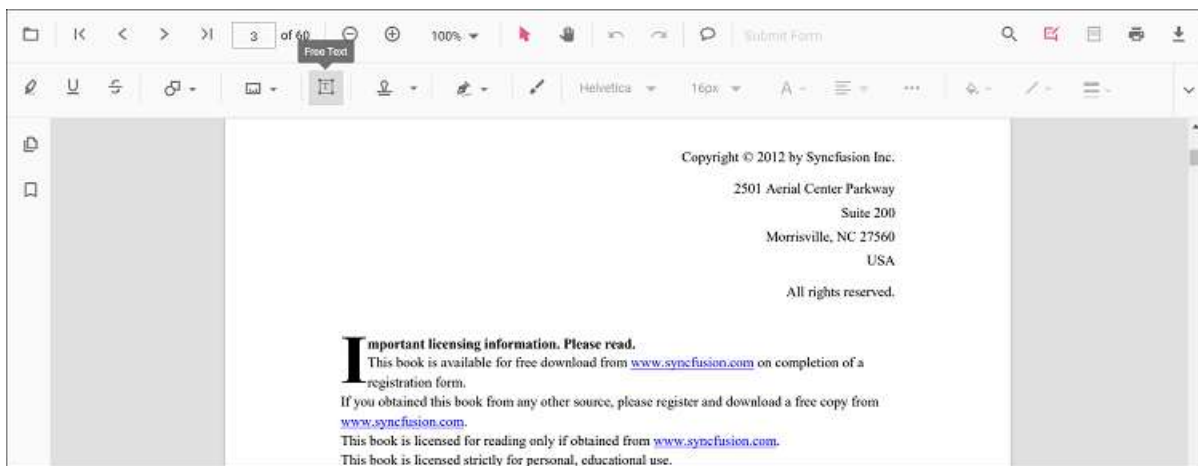
The PDF Viewer control provides the options to add, edit, and delete the free text annotations.

Adding a free text annotation to the PDF document

The Free text annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Free Text Annotation** button in the annotation toolbar. It enables the Free Text annotation mode.
- You can add the text over the pages of the PDF document.

In the pan mode, if the free text annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code sample to switch to the Free Text annotation mode.

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
function freetextMode() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('FreeText');
}
return (<div>
<button onClick={freetextMode}>FreeText</button>

```



```

<div className='control-section'>
  <PdfViewerComponent
    ref={(scope) => { pdfviewer = scope; }}
    id="container"
    documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
    resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
    style={{ 'height': '640px' }}>
    <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
      LinkAnnotation, BookmarkView, ThumbnailView,
      Print, TextSelection, TextSearch]} />
  </PdfViewerComponent>
</div>
</div>;
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView, ThumbnailView,
  Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
  react-pdfviewer';
let pdfviewer;
function App() {
  function freetextMode() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.annotation.setAnnotationMode('FreeText');
  }
  return (<div>
    <button onClick={freetextMode}>FreeText</button>
    <div className='control-section'>
      <PdfViewerComponent
        ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation, BookmarkView, ThumbnailView,
          Print, TextSelection, TextSearch]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

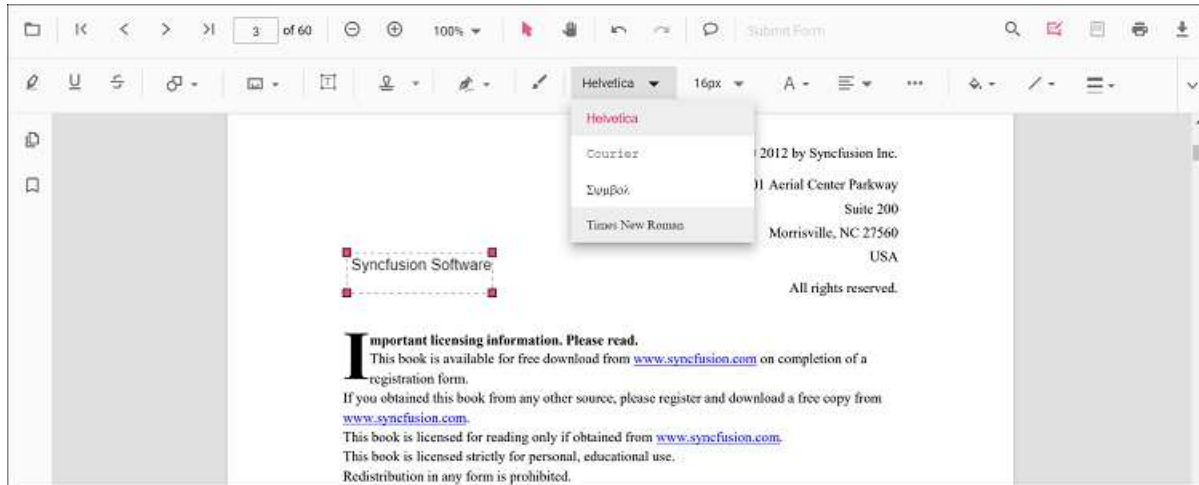
```

Editing the properties of free text annotation

The font family, font size, font styles, font color, text alignment, fill color, the border stroke color, border thickness, and opacity of the free text annotation can be edited using the Font Family tool, Font Size tool, Font Color tool, Text Align tool, Font Style tool, Edit Color tool, Edit Stroke Color tool, Edit Thickness tool, and Edit Opacity tool in the annotation toolbar.

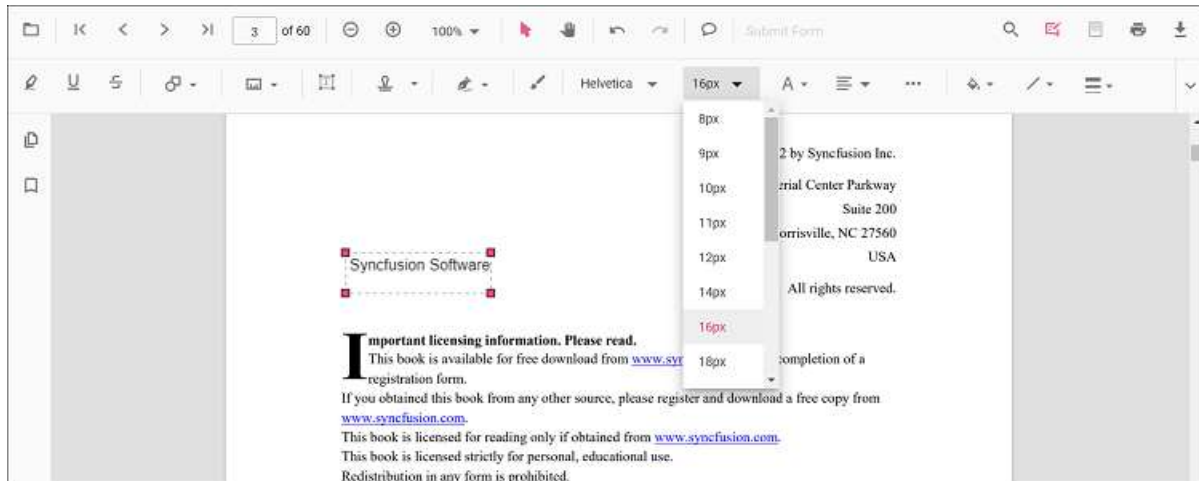
Editing font family

The font family of the annotation can be edited by selecting the desired font in the Font Family tool.



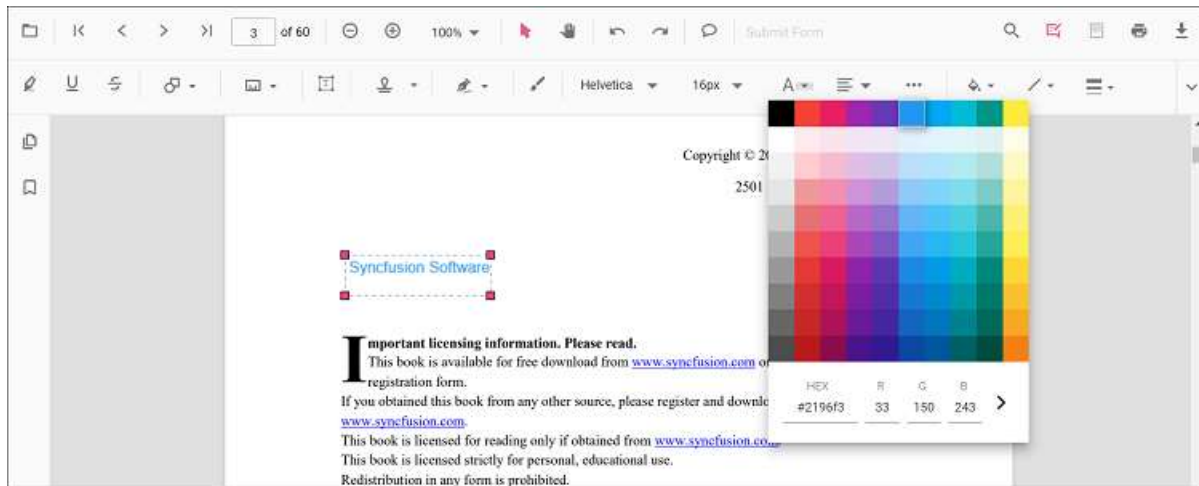
Editing font size

The font size of the annotation can be edited by selecting the desired size in the Font Size tool.



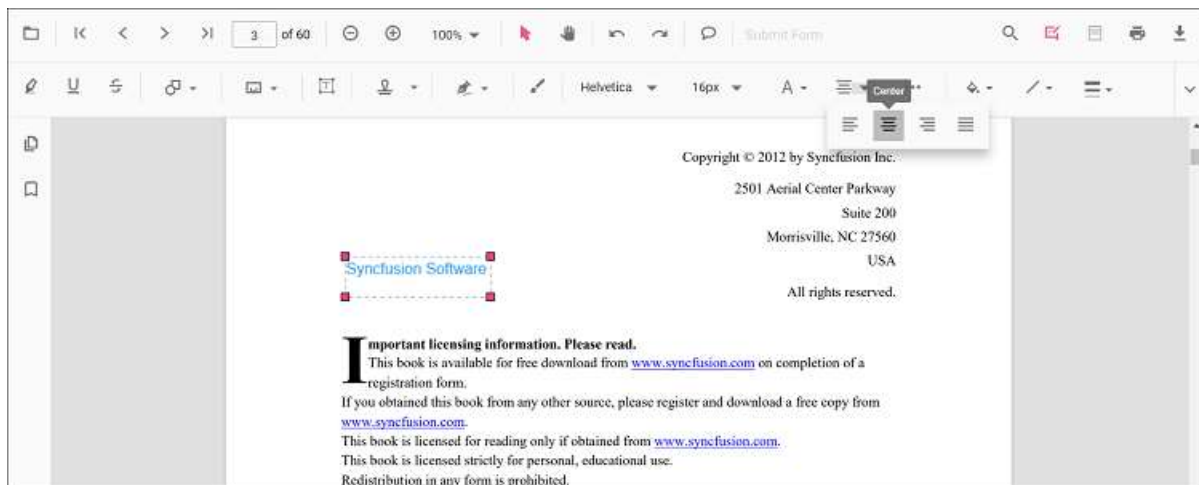
Editing font color

The font color of the annotation can be edited using the color palette provided in the Font Color tool.



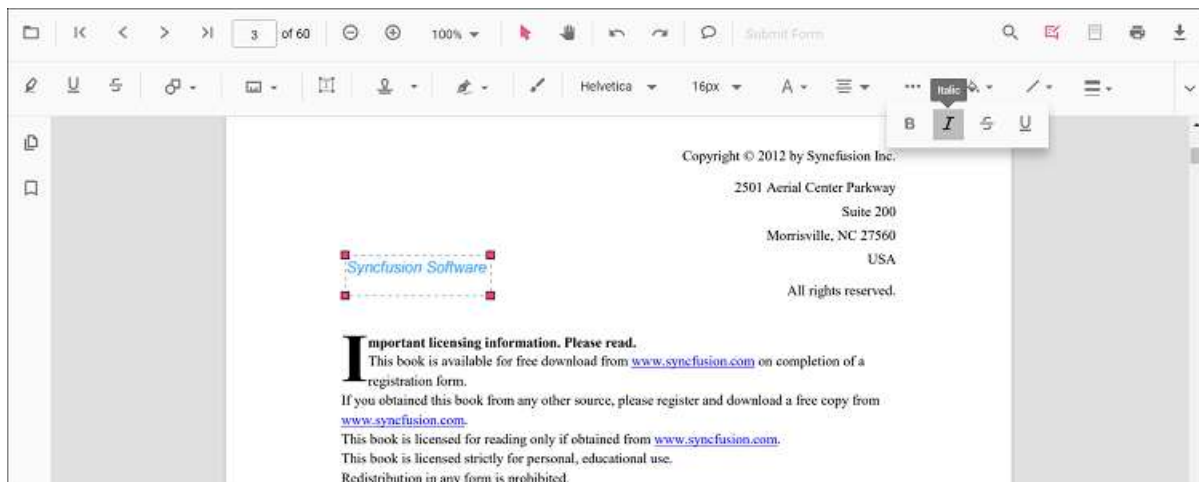
Editing the text alignment

The text in the annotation can be aligned by selecting the desired styles in the drop-down pop-up in the Text Align tool.



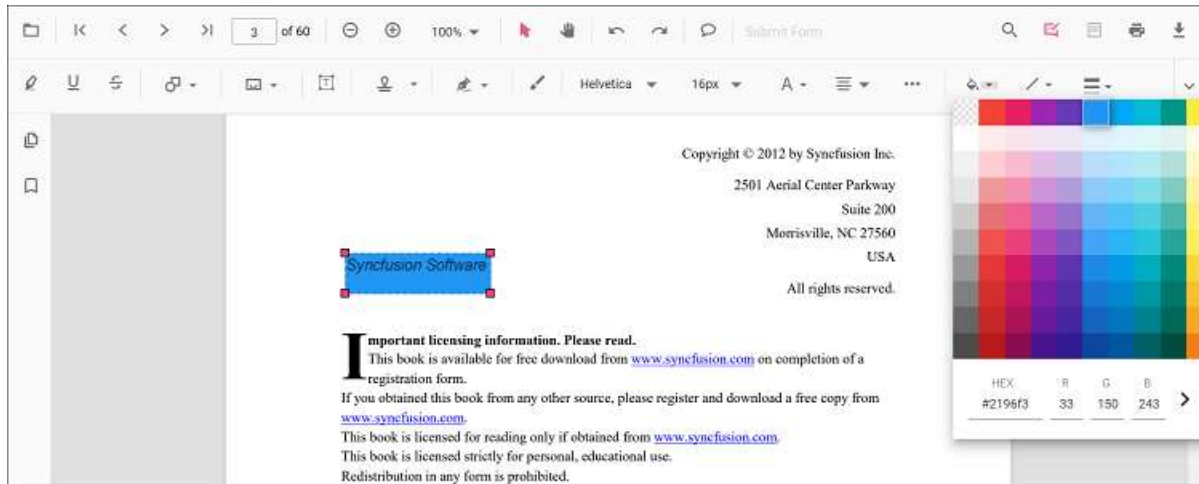
Editing text styles

The style of the text in the annotation can be edited by selecting the desired styles in the drop-down pop-up in the Font Style tool.



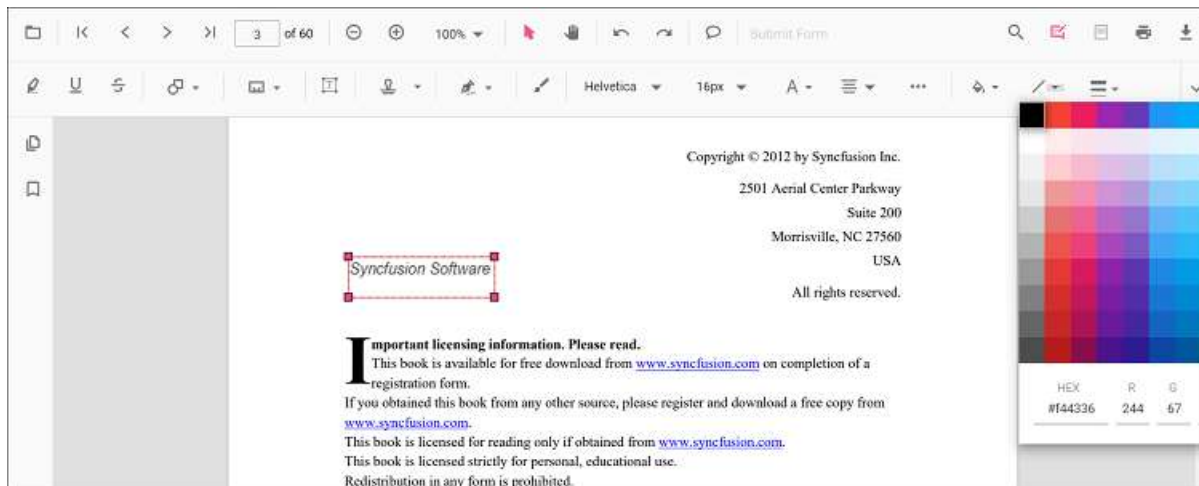
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



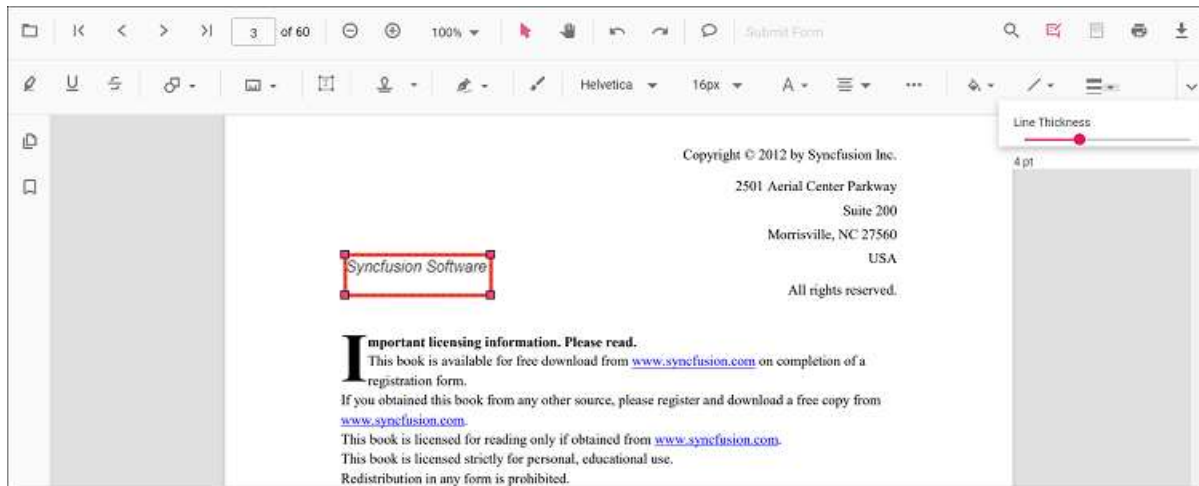
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



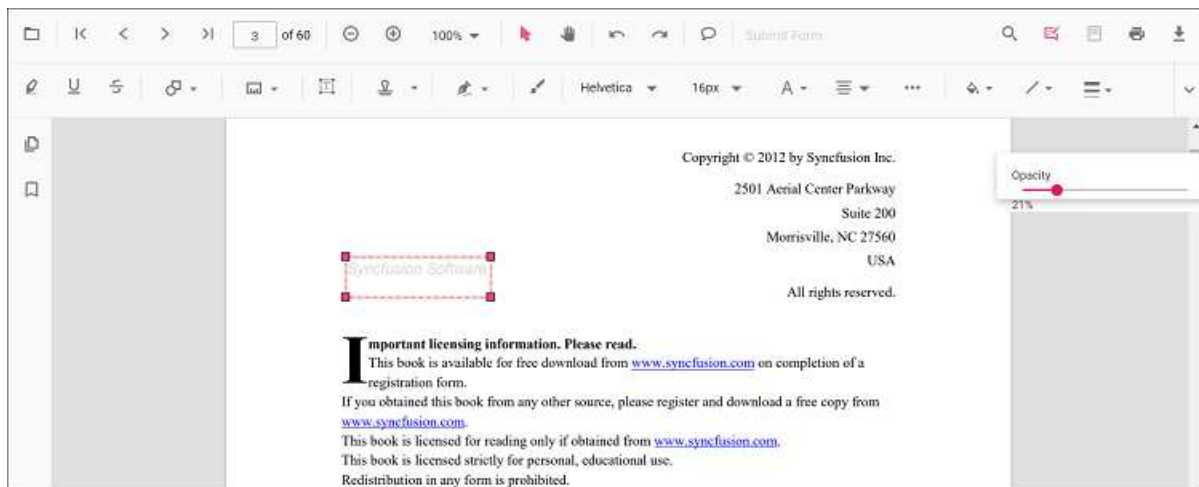
Editing thickness

The border thickness of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Setting default properties during control initialization

The properties of the free text annotation can be set before creating the control using the `FreeTextSettings`.

After editing the default values, they will be changed to the selected values. Refer to the following code sample to set the default free text annotation settings.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@synCFusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent
```

```

ref={ (scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
freeTextSettings={{fillColor: 'green', borderColor: 'blue', fontColor:
'yellow'}}
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch] />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
return (<div>
<div className='control-section'>
<PdfViewerComponent
ref={ (scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
freeTextSettings={{fillColor: 'green', borderColor: 'blue', fontColor:
'yellow'}}
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch] />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

You can also enable the autofit support for free text annotation by using the enableAutoFit boolean property in freeTextSettings as below. The width of the free text rectangle box will be increased based on the text added to it.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        freeTextSettings={{enableAutoFit: true}}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        freeTextSettings={{enableAutoFit: true}}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}

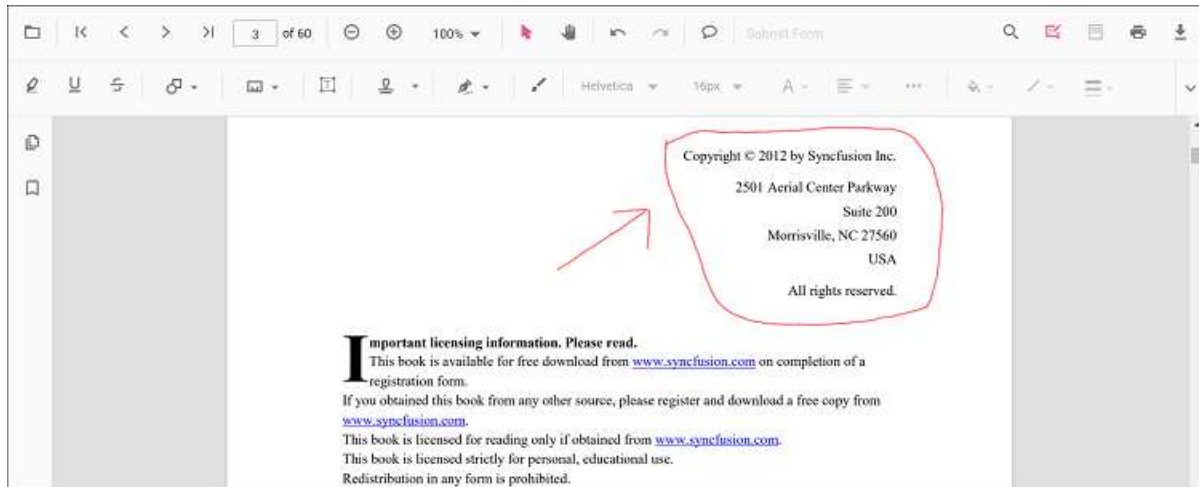
```



```
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Ink annotation in React Pdfviewer component

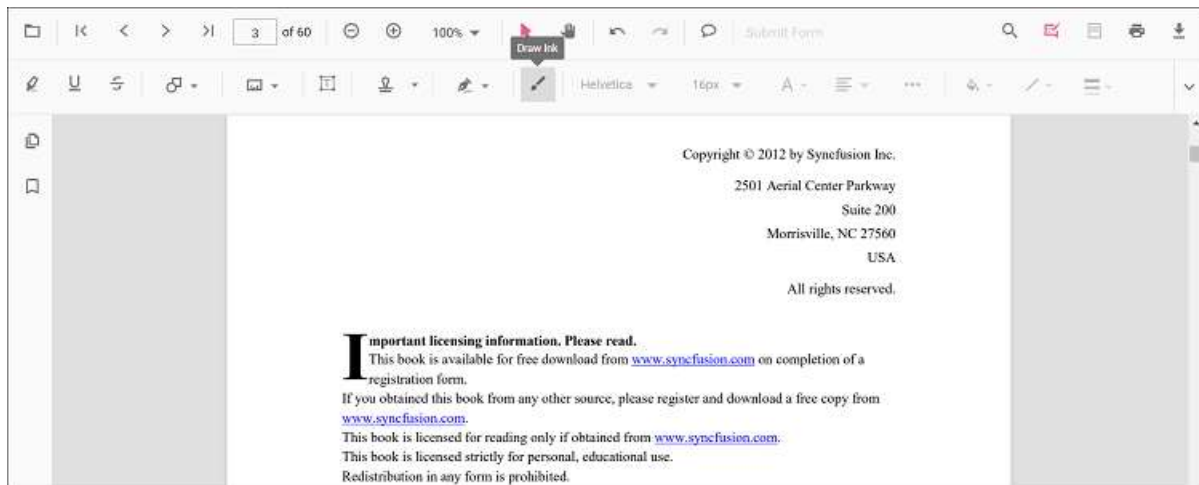
The PDF Viewer control provides the options to add, edit, and delete the ink annotations.



Adding an ink annotation to the PDF document

The ink annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Draw Ink** button in the annotation toolbar. It enables the ink annotation mode.
- You can draw anything over the pages of the PDF document.



Refer to the following code sample to switch to the ink annotation mode.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
```



```
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function inkMode() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Ink');
}
return (<div>
<button onClick={inkMode}>Draw Ink</button>
<div className='control-section'>
<PdfViewerComponent
ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]}/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function inkMode() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('Ink');
}
return (<div>
<button onClick={inkMode}>Draw Ink</button>
<div className='control-section'>
<PdfViewerComponent
ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
```

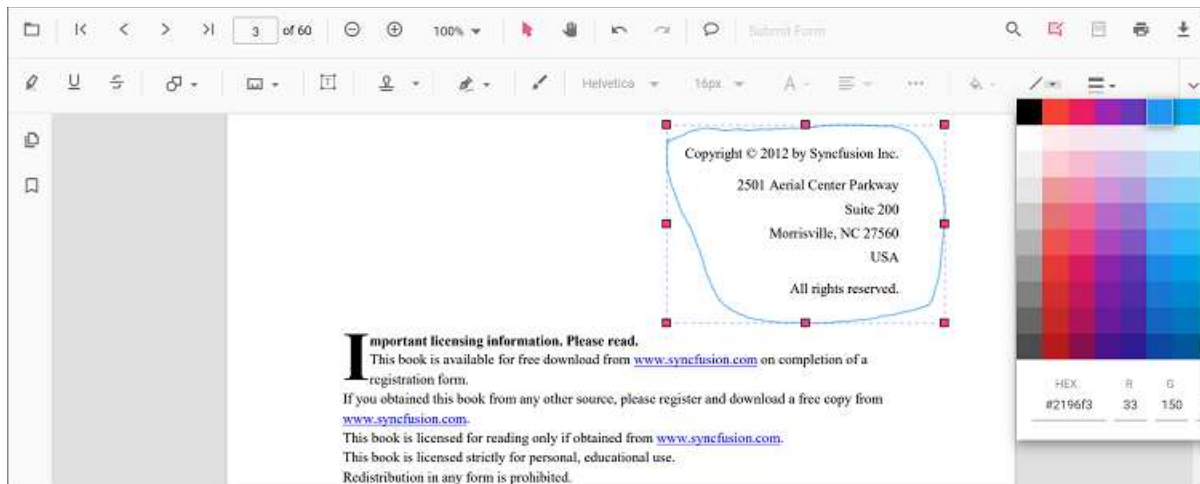
```
ThumbnailView, Print, TextSelection, TextSearch]]/>
</PdfViewerComponent>
</div>
</div>;
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Editing the properties of the ink annotation

The stroke color, thickness, and opacity of the ink annotation can be edited using the Edit stroke color tool, Edit thickness tool, and Edit opacity tool in the annotation toolbar.

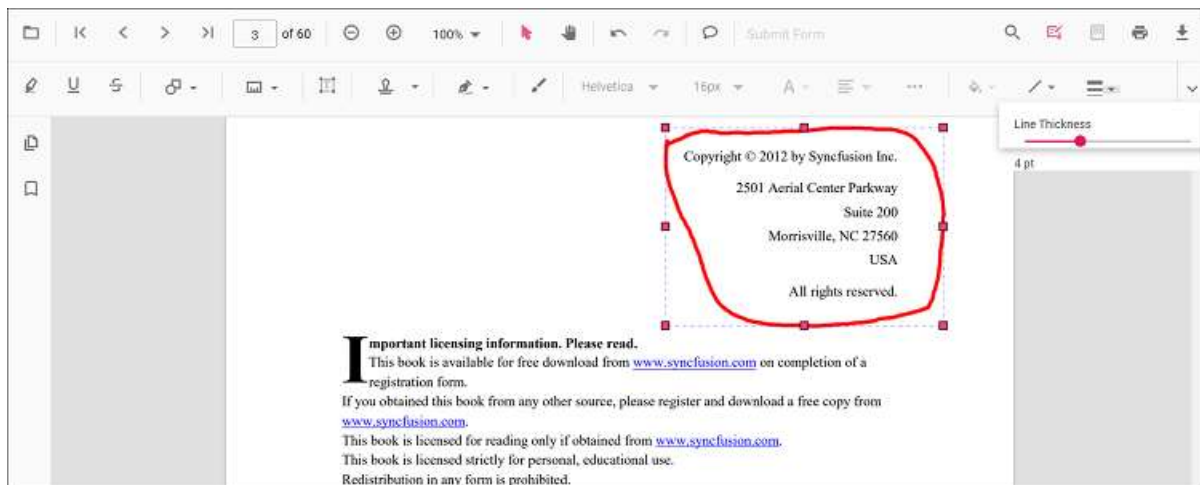
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



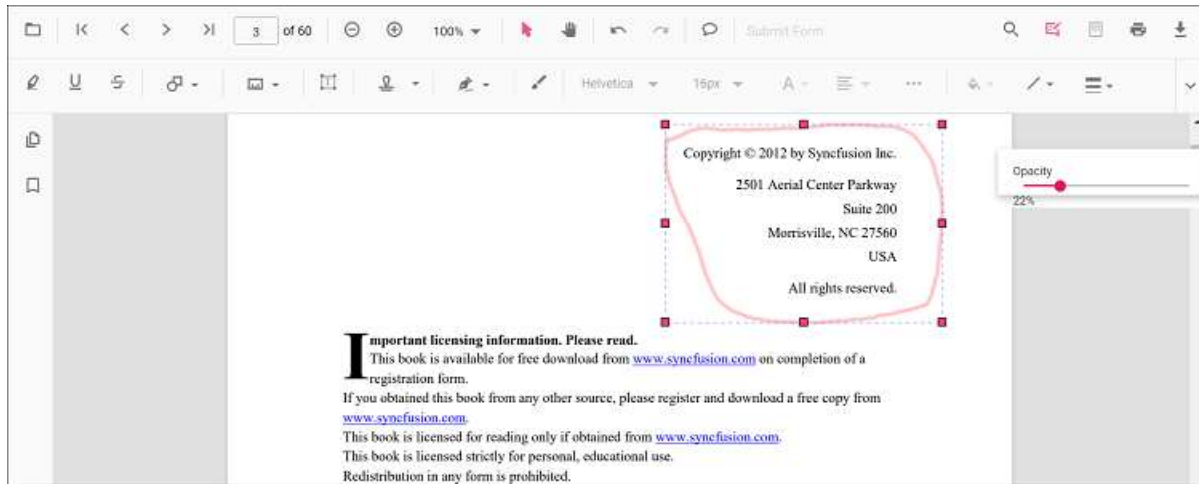
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Setting default properties during the control initialization

The properties of the ink annotation can be set before creating the control using the InkAnnotationSettings.

After editing the default values, they will be changed to the selected values. Refer to the following code sample to set the default ink annotation settings.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        inkAnnotationSettings={{author: 'Syncfusion', strokeColor: 'green',
          thickness: 3, opacity: 0.6}}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
          LinkAnnotation, BookmarkView,
          ThumbnailView, Print, TextSelection, TextSearch]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

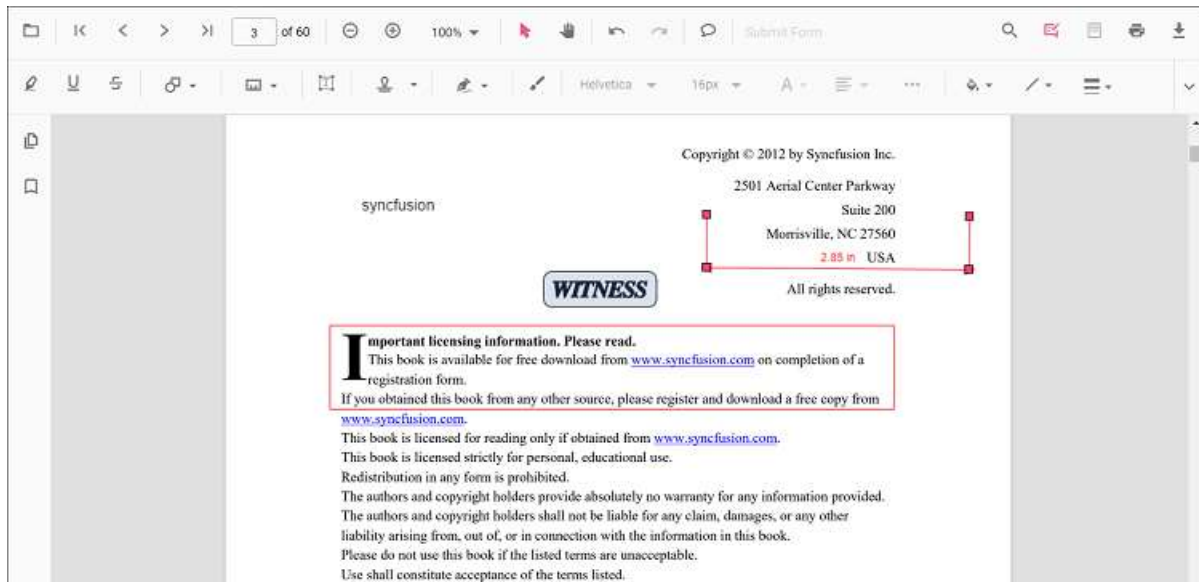
SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
 '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  return (<div>
    <div className='control-section'>
      <PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        inkAnnotationSettings={{author: 'Syncfusion', strokeColor: 'green',
        thickness: 3, opacity: 0.6}}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Comments in React Pdfviewer component

The PDF Viewer control provides options to add, edit, and delete the comments to the following annotation in the PDF documents:

- Shape annotation
- Stamp annotation
- Sticky note annotation
- Measurement annotation
- Text markup annotation
- Free text annotation
- Ink annotation



Adding a comment to the annotation

Annotation comment, comment replies, and status can be added to the PDF document using the comment panel.

Comment panel

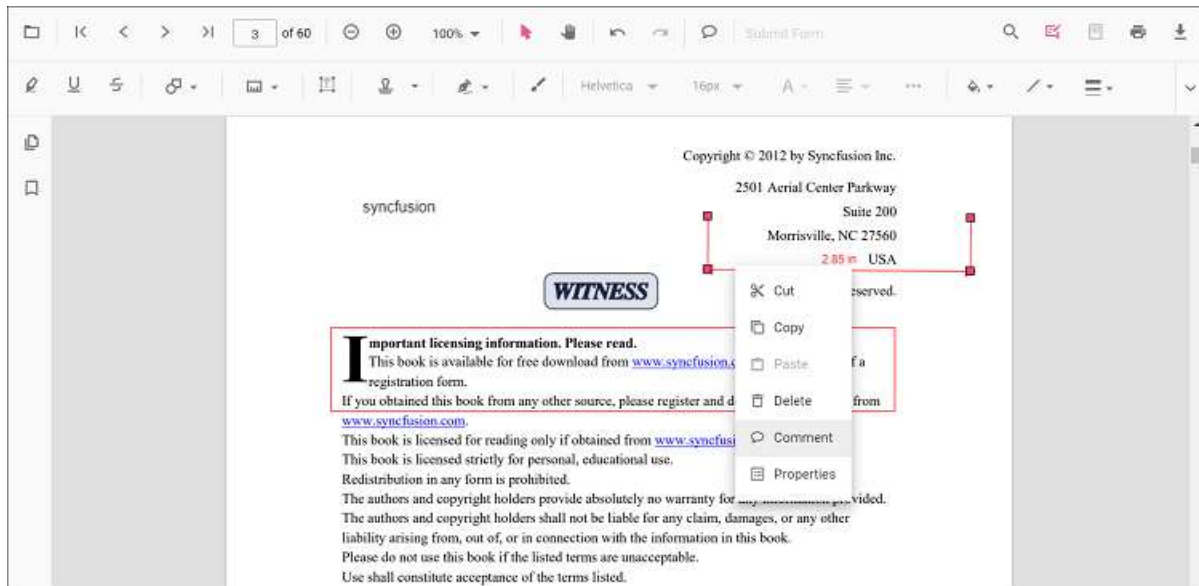
Annotation comments can be added to the PDF using the comment panel. The comment panel can be opened in the following ways:

1. Using the annotation menu
 - Click the Edit Annotation button in the PDF Viewer toolbar. A toolbar appears below it.
 - Click the Comment Panel button. A comment panel will appear.
2. Using Context menu
 - Select annotation in the PDF document and right-click it.
 - Select the comment option in the context menu that appears.
3. Using the Mouse click
 - Select annotation in the PDF document and double click it, a comment panel will appear.

If the comment panel is already in the open state, you can select the annotations and add annotation comments using the comment panel.

Adding comments

- Select annotation in the PDF document and click it.
- The selected annotation comment container is highlighted in the comment panel.
- Now, you can add comment and comment replies using the comment panel.

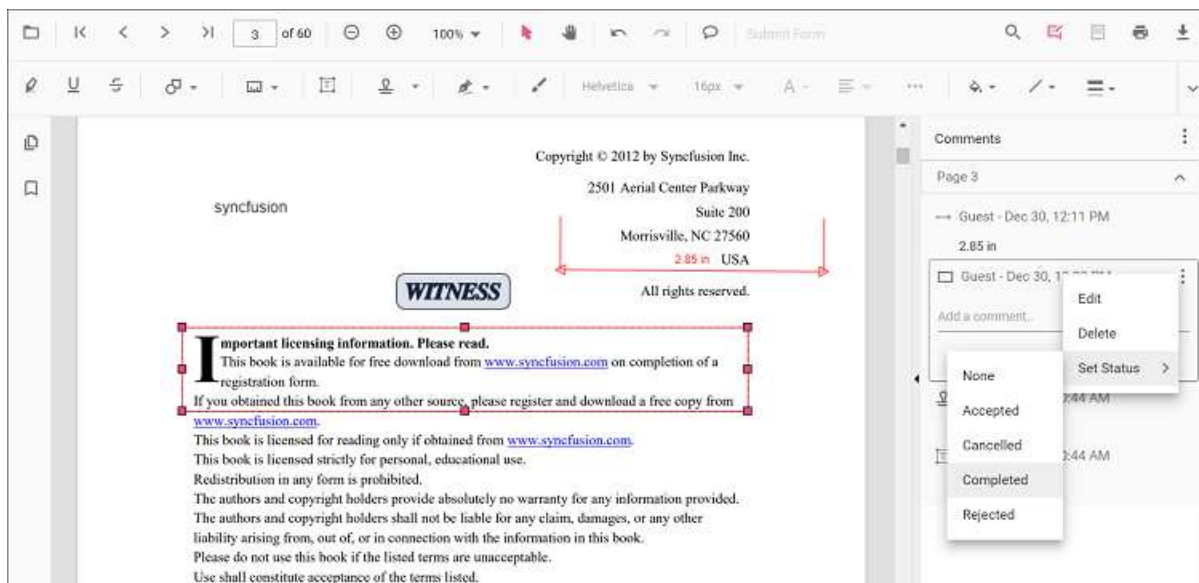


Adding Comment Replies

- The PDF Viewer control provides an option to add multiple replies to the comment.
- After adding the annotation comment, you can add a reply to the comment.

Adding Comment or Reply Status

- Select the Annotation Comments in the comment panel.
- Click the more options button showing in the Comments or reply container.
- Select the Set Status option in the context menu that appears.
- Select the status of the annotation comment in the context menu that appears.



Editing the comments and comments replies of the annotations

The comment, comment replies, and status of the annotation can be edited using the comment panel.

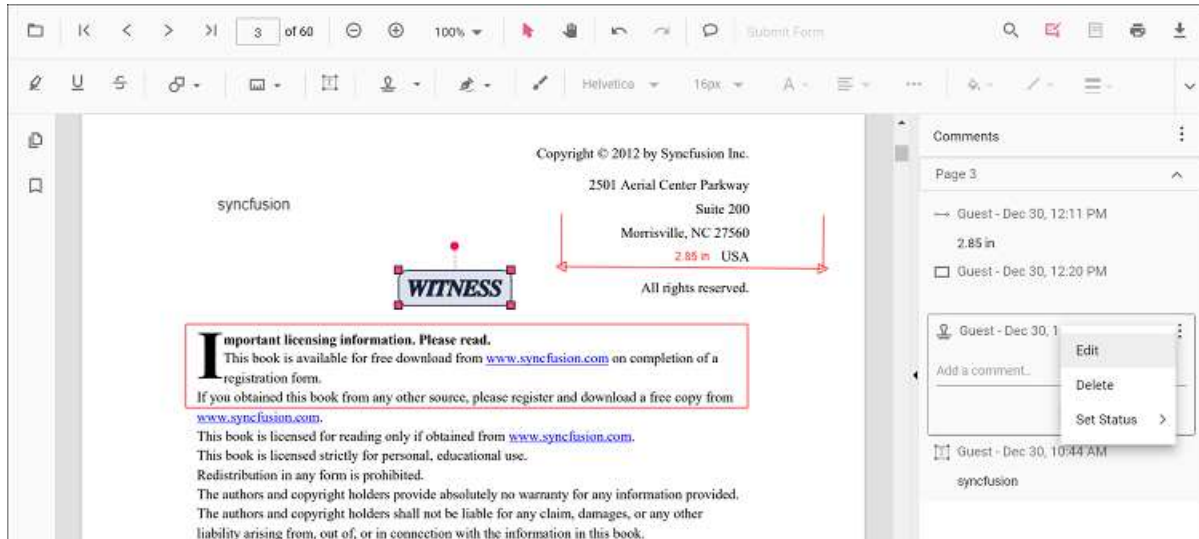
Editing the Comment or Comment Replies

The annotation comment and comment replies can be edited in the following ways:

1. Using the Context menu
 - Select the Annotation Comments in the comment panel.
 - Click the More options button showing in the Comments or reply container.
 - Select the Edit option in the context menu that appears.
 - Now, an editable text box appears. You can change the content of the annotation comment or comment reply.
2. Using the Mouse Click
 - Select the annotation comments in the comment panel.
 - Double click the comment or comment reply content.
 - Now, an editable text box appears. You can change the content of the annotation comment or comment reply.

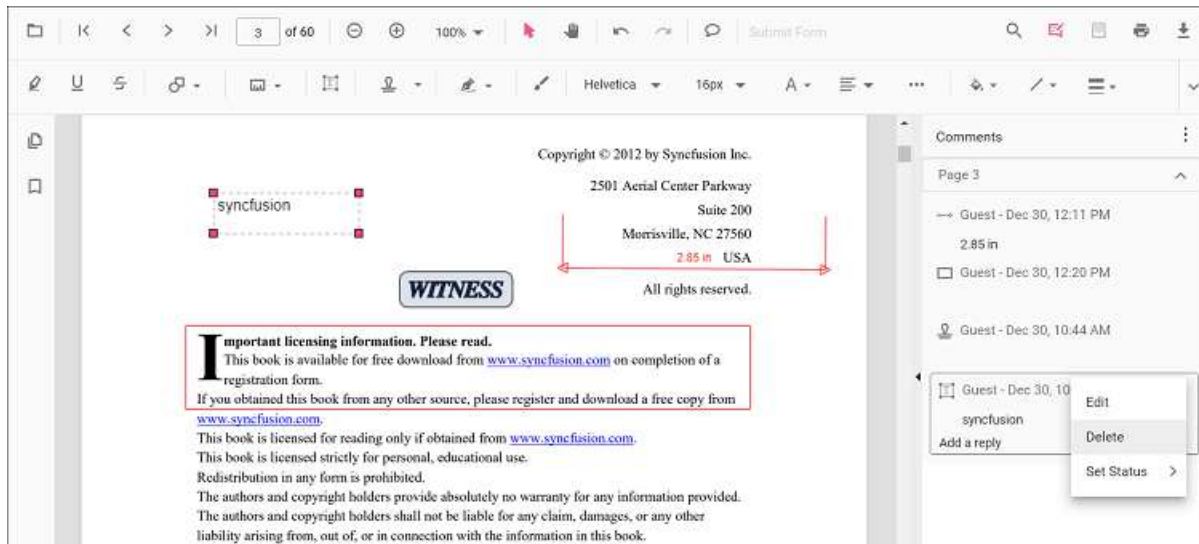
Editing Comment or Reply Status

- Select the Annotation Comments in the comment panel.
- Click the more options button showing in the Comments or reply container.
- Select the Set Status option in the context menu that appears.
- Select the status of the annotation comment in the context menu that appears.
- Status 'None' is the default state. If the status is set to 'None,' the comments or reply does not appear.



Delete Comment or Comment Replies

- Select the Annotation Comments in the comment panel.
- Click the more options button shown in the Comments or reply container.
- Select the Delete option in the context menu that appears.



The annotation will be deleted on deleting the comment using comment panel.

Handwritten signature in React PDF Viewer component

The PDF Viewer control supports adding handwritten signatures to a PDF document. The handwritten signature reduces the paper work of reviewing the content and verifies it digitally.

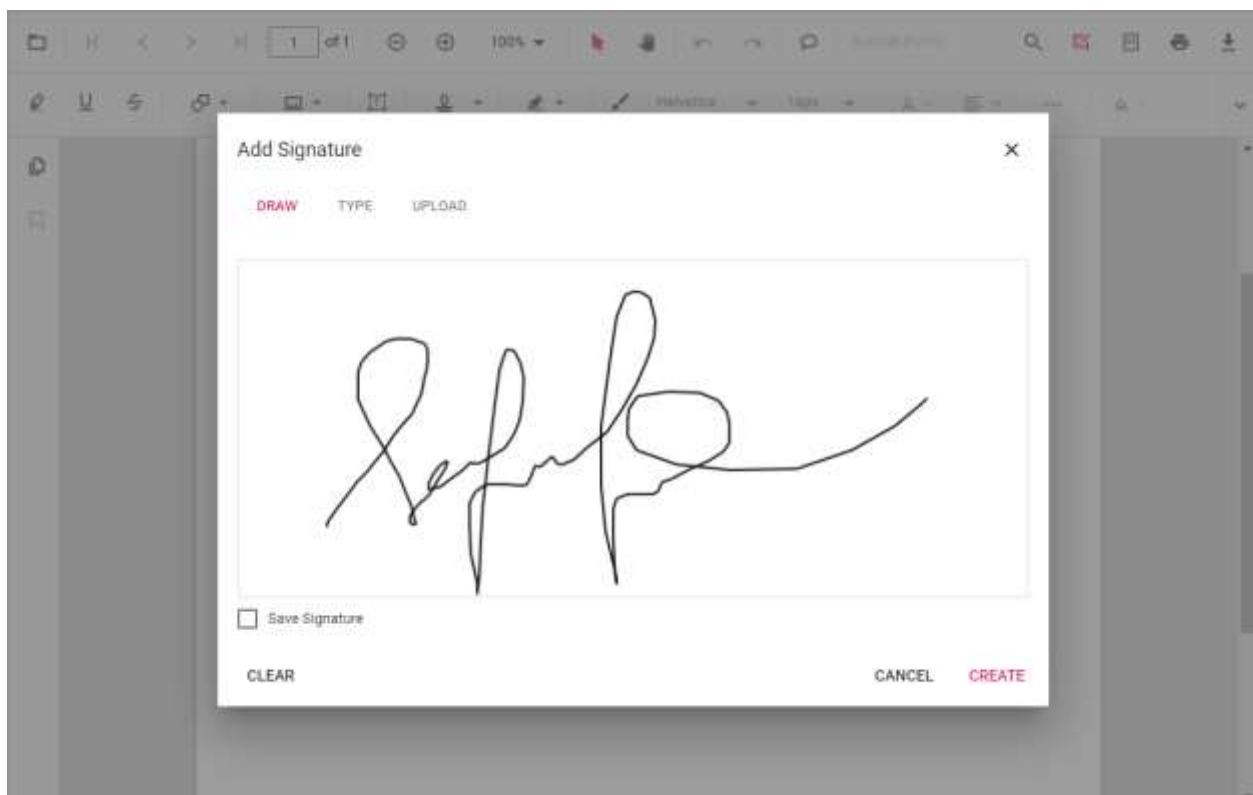
Adding a handwritten signature to the PDF document

The handwritten signature can be added to the PDF document using the annotation toolbar.

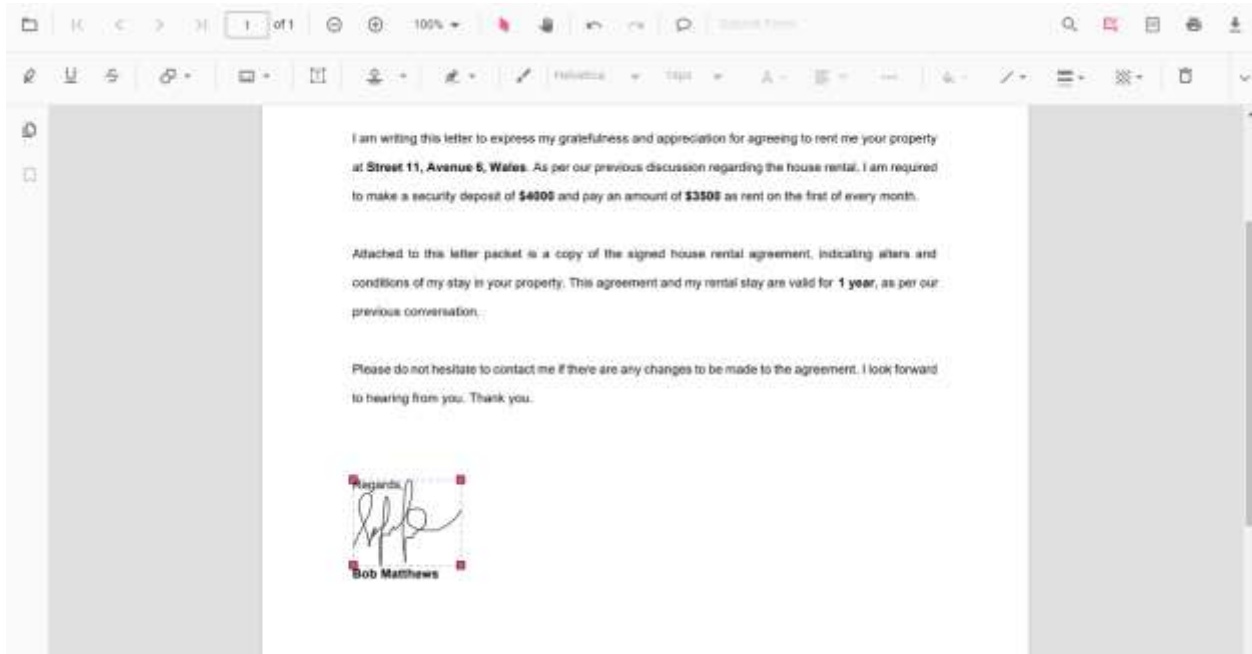
- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **HandWritten Signature** button in the annotation toolbar. The signature panel will appear.



- Draw the signature in the signature panel.



- Then click **Create** button and move the signature using the mouse and place them in the desired location.



Refer to the following code sample to switch to the handwritten signature mode programmatically.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function handwrittenSignature() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('HandWrittenSignature');
}
return (<div>
<button onClick={handwrittenSignature}>HandWritten Signature mode</button>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields
]]/>
</PdfViewerComponent>
</div>
```

```

</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function handwrittenSignature() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotation.setAnnotationMode('HandWrittenSignature');
}
return (<div>
<button onClick={handwrittenSignature}>HandWritten Signature mode</button>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields
]/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

How to enable the handwritten signature

The following code snippet describes how to enable the handwritten signature in PDF Viewer. When the value is set to `false` it disables the handwritten signature.

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';

```

```
function App() {
  return (<div>
    <div className='control-section'>
      {/* Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        // Enable or disable handwritten signature.
        enableHandwrittenSignature = {true}
        style={{ 'height': '640px' }}>
        <Inject services={[Toolbar, Magnification, Navigation, Annotation,
          LinkAnnotation, BookmarkView, ThumbnailView,
          Print, TextSelection, TextSearch, FormFields, FormDesigner]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView, ThumbnailView,
  Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
  Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/* Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        // Enable or disable handwritten signature.
        enableHandwrittenSignature = {true}
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        style={{ 'height': '640px' }}>
        <Inject services={[Toolbar, Magnification, Navigation, Annotation,
          LinkAnnotation, BookmarkView, ThumbnailView,
          Print, TextSelection, TextSearch, FormFields, FormDesigner]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Editing the properties of handwritten signature

The stroke color, border thickness, and opacity of the handwritten signature can be edited using the edit stroke color tool, edit thickness tool, and edit opacity tool in the annotation toolbar.

Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Interaction mode in React Pdfviewer component

The PDF Viewer provides interaction mode for easy interaction with the loaded PDF document. Selection mode and panning mode are the two interactions modes.

Selection mode

In this mode, the text selection can be performed in the PDF document loaded in PDF Viewer. The panning and scrolling of the pages by touch cannot be performed in this mode. It allows users to select and copy text from the PDF files. This is helpful for copying and sharing text content. You can enable/disable the text selection using the following code snippet.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        enableTextSelection={true}
        style={{ 'height': '640px' }}>
```

```

<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

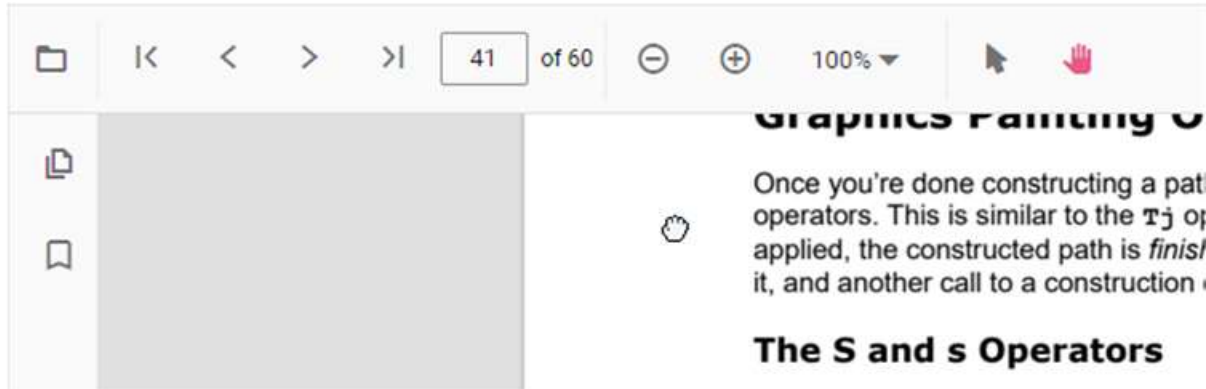
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
return <div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
enableTextSelection={true}
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



Panning Mode

In this mode, the panning and scrolling of the pages by touch can be performed in the PDF document loaded in the PDF Viewer, but the text selection cannot be performed.



You can switch the interaction mode of PDF Viewer by using the following code snippet.,

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (
    <div>
      <div className='control-section'>
        {/ * Render the PDF Viewer */}
        <PdfViewerComponent
          id="container"
          documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
          resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
          enableTextSelection={false}
          interactionMode="interactionMode"
          style={{ 'height': '640px' }}>
          <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
            LinkAnnotation,
            BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
        </PdfViewerComponent>
      </div>
    </div>
  );
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
```



```
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      { /* Render the PDF Viewer */ }
    <PdfViewerComponent
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      enableTextSelection={false}
      interactionMode="interactionMode"
      serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
      style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation,
        BookmarkView, ThumbnailView, Print, TextSelection, TextSearch ]} />
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

See also

- [Toolbar items](#)
- [Feature Modules](#)

Form Designer

Create programmatically in React Pdfviewer component

The PDF Viewer control provides the option to add, edit and delete the Form Fields. The Form Fields type supported by the PDF Viewer Control are:

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Add a form field to PDF document programmatically

Using addFormField method, the form fields can be added to the PDF document programmatically. We need to pass two parameters in this method. They are Form Field Type and Properties of Form Field Type. To add form field programmatically, Use the following code.

INDEX.JSX

```
{% raw %}
```

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner,
FormFields} from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoaded(){
var viewer = document.getElementById('container').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
bounds: { X: 338, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField('Textbox', {name: 'Last Name',bounds:
{ X: 530, Y: 229, Width: 150, Height: 24 },});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Month',bounds:
{ X: 146, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Date',bounds: {
X: 193, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Year',bounds: {
X: 242, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Agree',bounds:
{ X: 148, Y: 408, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds: {
X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Email',bounds: {
X: 146, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Phone',bounds: {
X: 482, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Month',bounds:
{ X: 386, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Date',bounds: {
X: 434, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Year',bounds: {
X: 482, Y: 923, Width: 35, Height: 24 },});
}
return (<div>

```

```

<div className='control-section'>
  {/* Render the PDF Viewer */}
  <PdfViewerComponent
    id="container"
    documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
    resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
    documentLoad={documentLoaded}
    style={{ 'height': '640px' }}>
    <Inject services={[ Toolbar, Magnification, Navigation, Annotation,
      LinkAnnotation, BookmarkView, ThumbnailView,
      Print, TextSelection, TextSearch, FormDesigner, FormFields ]} />
  </PdfViewerComponent>
</div>
</div>;
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView, ThumbnailView,
  Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner,
  FormFields} from '@syncfusion/ej2-react-pdfviewer';
export function App() {
  function documentLoaded() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
      { X: 146, Y: 229, Width: 150, Height: 24 } });
    viewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
      bounds: { X: 146, Y: 229, Width: 150, Height: 24 } });
    viewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
      bounds: { X: 338, Y: 229, Width: 150, Height: 24 } });
    viewer.formDesignerModule.addFormField("Textbox", {name: 'Last Name',bounds:
      { X: 530, Y: 229, Width: 150, Height: 24 },});
    viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148, Y:
      289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
    viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292, Y:
      289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
    viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Month',bounds:
      { X: 146, Y: 320, Width: 35, Height: 24 },});
    viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Date',bounds: {
      X: 193, Y: 320, Width: 35, Height: 24 },});
    viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Year',bounds: {
      X: 242, Y: 320, Width: 35, Height: 24 },});
    viewer.formDesignerModule.addFormField('InitialField', {name: 'Agree',bounds:
      { X: 148, Y: 408, Width: 200, Height: 43 },});
    viewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
      Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
    viewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
      Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
      false,});
  }
}

```

```

viewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds: {
X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Email',bounds: {
X: 146, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Phone',bounds: {
X: 482, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Month',bounds:
{ X: 386, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Date',bounds: {
X: 434, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Year',bounds: {
X: 482, Y: 923, Width: 35, Height: 24 },});
}
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
documentLoad={documentLoaded}
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormDesigner, FormFields ] />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}

```

Note: To set up the **server-backed PDF Viewer**, add the following **serviceUrl** within the **<div>** element in either the **index.tsx** or **index.jsx** file:

serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer".

Edit/Update form field programmatically

Using **updateFormField** method, Form Field can be updated programmatically. We should get the Form Field object/Id from **FormFieldCollections** property that you would like to edit and pass it as a parameter to **updateFormField** method. The second parameter should be the properties that you would like to update for Form Field programmatically. We have updated the value and background Color properties of Textbox Form Field.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormDesigner, FormFields,
Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
function documentLoaded() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }}});
viewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
bounds: { X: 338, Y: 229, Width: 150, Height: 24 }}});
viewer.formDesignerModule.addFormField('Textbox', {name: 'Last Name',bounds:
{ X: 530, Y: 229, Width: 150, Height: 24 },});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Month',bounds:
{ X: 146, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Date',bounds: {
X: 193, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Year',bounds: {
X: 242, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Agree',bounds:
{ X: 148, Y: 408, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds: {
X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Email',bounds: {
X: 146, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Phone',bounds: {
X: 482, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Month',bounds:
{ X: 386, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Date',bounds: {
X: 434, Y: 923, Width: 35, Height: 24 },});

```

```
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Year', bounds: {
X: 482, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.updateFormField(viewer.formFieldCollections[0], {
backgroundColor: 'red' });
}
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
documentLoad={documentLoaded}
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormDesigner, FormFields] />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormDesigner, FormFields,
Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
function documentLoaded() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
bounds: { X: 338, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField('Textbox', {name: 'Last Name', bounds:
{ X: 530, Y: 229, Width: 150, Height: 24 },});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Month', bounds:
{ X: 146, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Date', bounds: {
X: 193, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Year', bounds: {
X: 242, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Agree', bounds:
{ X: 148, Y: 408, Width: 200, Height: 43 },});
```

```

viewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds: {
X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Email',bounds: {
X: 146, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Phone',bounds: {
X: 482, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Month',bounds:
{ X: 386, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Date',bounds: {
X: 434, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Year',bounds: {
X: 482, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.updateFormField(viewer.formFieldCollections[0], {
backgroundColor: 'red' });
}
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
documentLoad={documentLoaded}
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormDesigner, FormFields] />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Note: To set up the **server-backed PDF Viewer**, add the following **serviceUrl** within the **<div>** element in either the **index.tsx** or **index.jsx** file:

serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer".

Delete form field programmatically

Using deleteFormField method, the form field can be deleted programmatically. We should retrieve the Form Field object/Id from FormFieldCollections property that you would like to delete and pass it as a parameter to deleteFormField method. To delete a Form Field programmatically, use the following code.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormDesigner, FormFields,
Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
function documentLoaded(){
var viewer = document.getElementById('container').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }}});
viewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
bounds: { X: 338, Y: 229, Width: 150, Height: 24 }}});
viewer.formDesignerModule.addFormField('Textbox', {name: 'Last Name',bounds:
{ X: 530, Y: 229, Width: 150, Height: 24 },});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Month',bounds:
{ X: 146, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Date',bounds: {
X: 193, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Year',bounds: {
X: 242, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Agree',bounds:
{ X: 148, Y: 408, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds: {
X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Email',bounds: {
X: 146, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Phone',bounds: {
X: 482, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});

```



```
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Month', bounds:
{ X: 386, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Date', bounds: {
X: 434, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Year', bounds: {
X: 482, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.deleteFormField(viewer.formFieldCollections[0]);
}
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
documentLoad={documentLoaded}
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields]
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormDesigner, FormFields,
Inject } from '@syncfusion/ej2-react-pdfviewer';
export function App() {
function documentLoaded() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
bounds: { X: 338, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField('Textbox', {name: 'Last Name', bounds:
{ X: 530, Y: 229, Width: 150, Height: 24 },});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Month', bounds:
{ X: 146, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Date', bounds: {
X: 193, Y: 320, Width: 35, Height: 24 },});
```

```

viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Year',bounds: {
X: 242, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Agree',bounds:
{ X: 148, Y: 408, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds: {
X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Email',bounds: {
X: 146, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Phone',bounds: {
X: 482, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Month',bounds:
{ X: 386, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Date',bounds: {
X: 434, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Year',bounds: {
X: 482, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.deleteFormField(viewer.formFieldCollections[0]);
}
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
documentLoad={documentLoaded}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}

```

Note: To set up the **server-backed PDF Viewer**, add the following **serviceUrl** within the <div> element in either the **index.tsx** or **index.jsx** file:

serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer".

setFormFieldMode programmatically

The **setFormFieldMode** method is a function in the Syncfusion React PDF Viewer library that allows you to add a form field dynamically by passing the type of the form field. You can pass the form fields as a parameter like below.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function addPasswordField() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.formDesignerModule.setFormFieldMode("Password");
}
return (<div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<button onClick={addPasswordField}>Add Password Field</button>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Magnification, Navigation, LinkAnnotation,
Annotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields]
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function addPasswordField() {
```

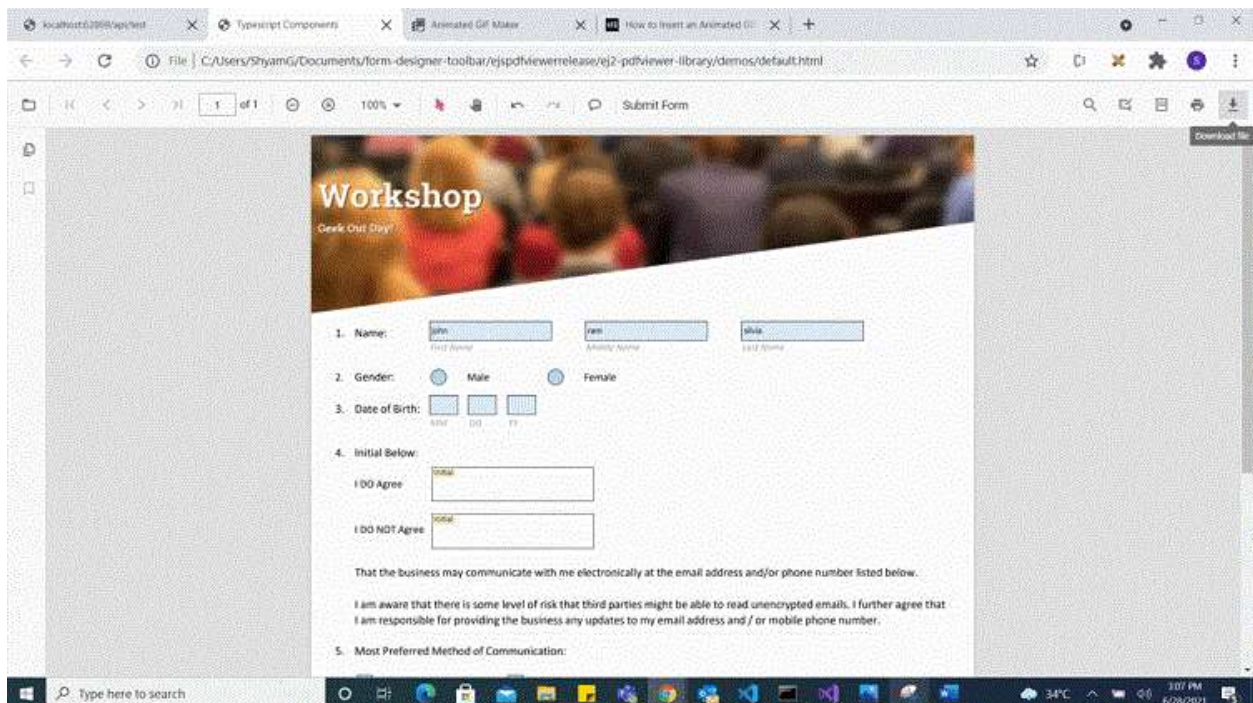
```

var viewer = document.getElementById('container').ej2_instances[0];
viewer.formDesignerModule.setFormFieldMode("Password");
}
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<button onClick={addPasswordField}>Add Password Field</button>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
Annotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Saving the form fields

When the download icon is selected on the toolbar, the Form Fields will be saved in the PDF document and this action will not affect the original document. Refer the below GIF for further reference.



You can invoke download action using following code snippet.,

STANDALONE

```
{% raw %}
```

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
  function downloadClicked() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.download();
  }
  return (<div>
    <div className='control-section'>
      { /* Render the PDF Viewer */ }
      <button onClick={downloadClicked}>Download</button>
      <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; } }
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
      style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
      Annotation, BookmarkView,
      ThumbnailView, Print, TextSelection, TextSearch]} />
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
  function downloadClicked() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.download();
  }
  return (<div>
    <div className='control-section'>
      { /* Render the PDF Viewer */ }
      <button onClick={downloadClicked}>Download</button>
      <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; } }
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
      style={{ 'height': '640px' }}>

```



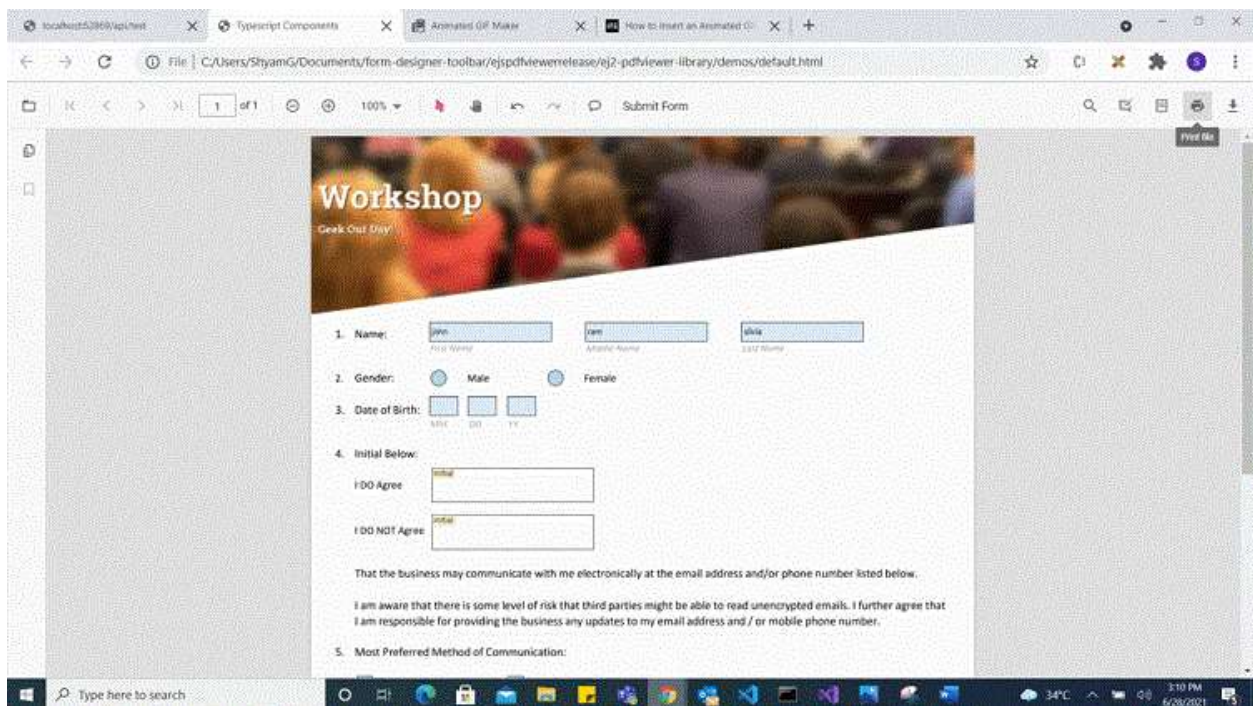
```

<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
Annotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Printing the form fields

When the print icon is selected on the toolbar, the PDF document will be printed along with the Form Fields added to the pages and this action will not affect the original document. Refer the below GIF for further reference.



STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
return <div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"

```

```

documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
enablePrint={true}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

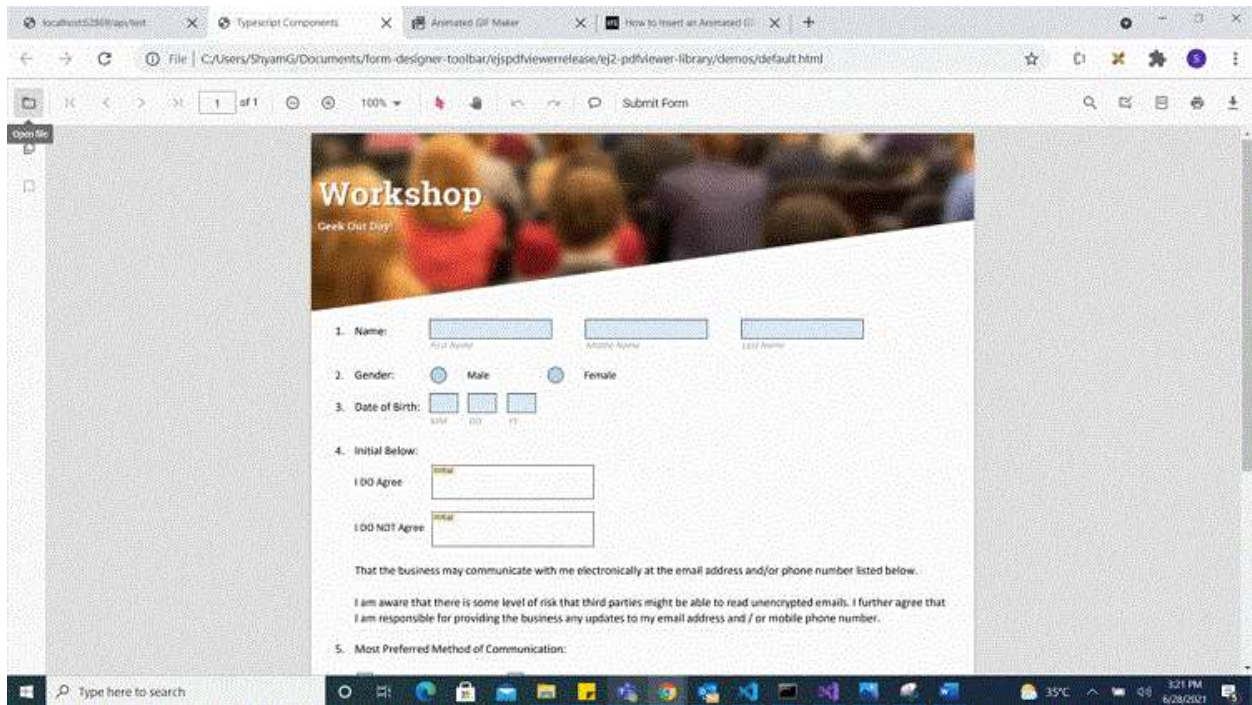
```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
return (<div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
enablePrint={true}
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Open the existing PDF document

We can open the already saved PDF document contains Form Fields in it by clicking the open icon in the toolbar. Refer the below GIF for further reference.



Validate form fields

The form fields in the PDF Document will be validated when the `enableFormFieldsValidation` is set to true and hook the `validateFormFields`. The `validateFormFields` will be triggered when the PDF document is downloaded or printed with the non-filled form fields. The non-filled fields will be obtained in the `nonFillableFields` property of the event arguments of `validateFormFields`.

Add the following code snippet to validate the form fields,

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner,
FormFields } from '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function documentLoaded () {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
    { X: 146, Y: 229, Width: 150, Height: 24 }});
  }
  function validateFormFields(args) {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.nonfilledFormFields = args.nonFillableFields
  }
  return (<div>
    <div className='control-section'>
      { /* Render the PDF Viewer */ }
      <PdfViewerComponent ref={ (scope) => { pdfviewer = scope; } }
      id="container"
```



```

documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
documentLoad={documentLoaded}
enableFormFieldsValidation={true}
ValidateFormFields= {validateFormFields}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields ]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner,
FormFields } from '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function documentLoaded () {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
{ X: 146, Y: 229, Width: 150, Height: 24 }});
}
function validateFormFields(args) {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.nonfilledFormFields = args.nonFillableFields
}
return (<div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
documentLoad={documentLoaded}
enableFormFieldsValidation={true}
ValidateFormFields= {validateFormFields}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields ]}
/>
</PdfViewerComponent>
</div>
</div>);

```

```

}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Export and import form fields

The PDF Viewer control provides the support to export and import the form field data in the following formats using the methods `importFormFields`, `exportFormFields`, `exportFormFieldsAsObject`.

- FDF
- XFDF
- JSON

Export and import as FDF

Using the `exportFormFields` method, the form field data can be exported in the specified data format. This method accepts two parameters:

- The first one must be the destination path for the exported data. If path is not specified, it will ask for the location while exporting.
- The second parameter should be the format type of the form data.

The following code explains how to export and import the form field data as FDF.

```

<button onclick="OnExportFdf()">Export FDF</button>
<button onclick="OnImportFdf()">Import FDF</button>
`ts
// Event triggers on the Export FDF button click.
function OnExportFdf() {
var viewer = document.getElementById('container').ej2_instances[0];
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Fdf);
}
// Event triggers on the Import FDF button click.
function OnImportFdf() {
var viewer = document.getElementById('container').ej2_instances[0];
// The file for importing the form fields should be placed in the desired location and the path should be
provided correctly
viewer.importFormFields('File', FormFieldDataFormat.Fdf);
}

```

`

Export and import as XFDF

The following code explains how to export and import the form field data as XFDF.

`

```
<button onclick="OnExportXfdf()">Export XFDF</button>
```

```
<button onclick="OnImportXfdf()">Import XFDF</button>
```

`

```
`ts
```

```
// Event triggers on the Export XFDF button click.
```

```
function OnExportXfdf(){
```

```
var viewer = document.getElementById('container').ej2_instances[0];
```

```
// Data must be the desired path for the exported document.
```

```
viewer.exportFormFields('Data', FormFieldDataFormat.Xfdf);
```

```
}
```

```
// Event triggers on the Import XFDF button click.
```

```
function OnImportXfdf(){
```

```
var viewer = document.getElementById('container').ej2_instances[0];
```

```
// The file for importing the form fields should be placed in the desired location and the path should be provided correctly
```

```
viewer.importFormFields('File', FormFieldDataFormat.Xfdf);
```

```
}
```

`

Export and import as JSON

The following code explains how to export and import the form field data as JSON.

`

```
<button onclick="OnExportJson()">Export JSON</button>
```

```
<button onclick="OnImportJson()">Import JSON</button>
```

`

```
`ts
```

```
// Event triggers on the Export JSON button click.
```

```
function OnExportJson(){
```

```
var viewer = document.getElementById('container').ej2_instances[0];
```

```
// Data must be the desired path for the exported document.
```

```
viewer.exportFormFields('Data', FormFieldDataFormat.Json);
```

```

}
// Event triggers on the Import JSON button click.
function OnImportJson(){
var viewer = document.getElementById('container').ej2_instances[0];
// The file for importing the form fields should be placed in the desired location and the path should be
provided correctly
viewer.importFormFields('File', FormFieldDataFormat.Json);
}
`

```

Export and import as Object

The PDF Viewer control supports exporting the form field data as an object, and the exported data will be imported into the current PDF document from the object.

The following code shows how to export the form field data as an object and import the form field data from that object into the current PDF document via a button click.

```

`
<button onclick="exportDataAsObject()">Export Object</button>
<button onclick="importData()">Import Data</button>
`

`ts
var exportedData;
// Event triggers on the Export Object button click.
function exportDataAsObject(){
var viewer = document.getElementById('container').ej2_instances[0];
// Export the form fields data to an FDF object.
exportedData = viewer.exportFormFieldsAsObject(FormFieldDataFormat.Fdf);
//// Export the form fields data to an XFDF object.
//exportedData = viewer.exportFormFieldsAsObject(FormFieldDataFormat.Xfdf);
//// Export the form fields data to an JSON object.
//exportedData = viewer.exportFormFieldsAsObject(FormFieldDataFormat.Json);
}
// Event triggers on Import Data button click.
function importData(){
var viewer = document.getElementById('container').ej2_instances[0];
// Import the form fields data from the FDF object into the current PDF document.

```

```
viewer.importFormFields(exportedData, FormFieldDataFormat.Fdf);
//// Import the form fields data from the XFDF object into the current PDF document.
//viewer.importFormFields(exportedData, FormFieldDataFormat.Xfdf);
//// Import the form fields data from the JSON object into the current PDF document.
//viewer.importFormFields(exportedData, FormFieldDataFormat.Json);
}
`
```

Form field properties

Form field properties in Syncfusion PDF Viewer allow you to customize and interact with form fields embedded within PDF documents. This documentation provides an overview of the form field properties supported by the Syncfusion PDF Viewer and explains how to use them effectively.

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Signature and initial fields settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the signature field properties on a button click.

```
`html
<button onClick={updateProperties}>Update Properties</button>
`

`ts
// Event triggers on the Update Properties button click.
function updateProperties(){
var viewer = document.getElementById('container').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Initial',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
```

tooltip: 'Initial',

thickness: 4

});

}

,

The following code example explains how to update the properties of the signature field added to the document from the form designer toolbar.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, DisplayMode, Inject} from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad() {
var viewer = document.getElementById('container').ej2_instances[0];
/* Defines the signature field settings */
viewer.signatureFieldSettings =
{
// Set the name of the form field element.
name: 'Signature',
// Specify whether the signature field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the signature field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Signature',
// Set the thickness of the signature field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specify the properties of the signature Dialog Settings in the signature
field.
signatureDialogSettings: {
displayMode: DisplayMode.Draw | DisplayMode.Upload | DisplayMode.Text,
hideSaveSignature: false,
},
// Specify the properties of the signature indicator in the signature field.
signatureIndicatorSettings: {
opacity: 1,
backgroundColor: '#daeaf7ff',
height: 50,
fontSize: 15,
text: 'Signature Field',
color: 'white'

```

```

}
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}
documentLoad={documentLoad}>
{/* Inject the required services */}
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, DisplayMode, Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad() {
var viewer = document.getElementById('container').ej2_instances[0];
{/* Defines the signature field settings */}
viewer.signatureFieldSettings =
{
// Set the name of the form field element.
name: 'Signature',
// Specify whether the signature field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the signature field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Signature',
// Set the thickness of the signature field. To hide the borders, set the
value to 0 (zero).
thickness: 4,

```

```
// Specify the properties of the signature Dialog Settings in the signature
field.
signatureDialogSettings: {
  displayMode: DisplayMode.Draw | DisplayMode.Upload | DisplayMode.Text,
  hideSaveSignature: false,
},
// Specify the properties of the signature indicator in the signature field.
signatureIndicatorSettings: {
  opacity: 1,
  backgroundColor: '#daeaf7ff',
  height: 50,
  fontSize: 15,
  text: 'Signature Field',
  color: 'white'
}
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/pdfviewer"
style={{ 'height': '640px' }}
documentLoad={documentLoad}>
{/* Inject the required services */}
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```



The following code example explains how to update the properties of the initial field added to the document from the form designer toolbar.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
```



```

Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, DisplayMode, Inject} from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad(){
var viewer = document.getElementById('container').ej2_instances[0];
/* Defines the initial field settings */
viewer.initialFieldSettings = {
name: 'Initial',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Initial',
thickness: 4,
// Specify the properties of the initial indicator in the initial field.
initialIndicatorSettings: {
opacity: 1,
backgroundColor: '#daeaf7ff',
height: 50,
fontSize: 15,
text: 'Initial Field',
color: 'white'
},
// Specify the properties of the initial Dialog Settings in the initial field.
initialDialogSettings: {
displayMode: DisplayMode.Draw | DisplayMode.Upload | DisplayMode.Text,
hideSaveSignature: false
}
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}
documentLoad={documentLoad}>
/* Inject the required services */
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]]
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

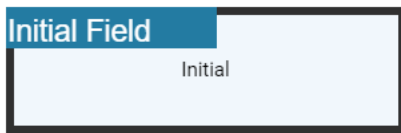
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';

```

```

import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, DisplayMode, Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad() {
var viewer = document.getElementById('container').ej2_instances[0];
/* Defines the initial field settings */
viewer.initialFieldSettings = {
name: 'Initial',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Initial',
thickness: 4,
// Specify the properties of the initial indicator in the initial field.
initialIndicatorSettings: {
opacity: 1,
backgroundColor: '#daeaf7ff',
height: 50,
fontSize: 15,
text: 'Initial Field',
color: 'white'
},
// Specify the properties of the initial Dialog Settings in the initial field.
initialDialogSettings: {
displayMode: DisplayMode.Draw | DisplayMode.Upload | DisplayMode.Text,
hideSaveSignature: false
}
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/pdfviewer"
style={{ 'height': '640px' }}
documentLoad={documentLoad}>
/* Inject the required services */
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



Textbox field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the Textbox field properties on a button click.

```
`html
```

```
<button onClick={updateProperties}>Update Properties</button>
```

```
,
```

```
`ts
```

```
// Event triggers on the Update Properties button click.
```

```
function updateProperties() {
```

```
var viewer = document.getElementById('container').ej2_instances[0];
```

```
var formField = viewer.retrieveFormFields();
```

```
viewer.formDesignerModule.updateFormField(formField[0], {
```

```
name: 'Textbox',
```

```
isReadOnly: true,
```

```
visibility: 'visible',
```

```
isRequired: false,
```

```
isPrint: true,
```

```
tooltip: 'Textbox',
```

```
thickness: 4,
```

```
value: 'Textbox',
```

```
fontFamily: 'Courier',
```

```
fontSize: 10,
```

```
fontStyle: 'None',
```

```
color: 'black',
```

```
borderColor: 'black',
```

```
backgroundColor: '#daeaf7f',
```

```
alignment: 'Left',
```

```
maxLength: 0,
```

```
isMultiline: false,
```

```
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }
```

```
});
```

```
}
```

```
,
```

The following code example explains how to update the properties of the Textbox field added to the document from the form designer toolbar.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad() {
var viewer = document.getElementById('container').ej2_instances[0];
/* Defines the Textbox field settings */
viewer.textFieldSettings = {
// Set the name of the form field element.
name: 'Textbox',
// Specify whether the Textbox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Textbox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Textbox',
// Set the thickness of the Textbox field. To hide the borders, set the value
to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'Textbox',
// Set the font family of the textbox field.
fontFamily: 'Courier',
// Set the font size of the textbox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the textbox field.
color: 'black',
// Set the border color of the textbox field.
borderColor: 'black',
// Set the background color of the textbox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
```

```

maxLength: 0,
// Allows multiline input in the text field. FALSE, by default.
isMultiline: false
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}
documentLoad={documentLoad}>
{/* Inject the required services */}
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

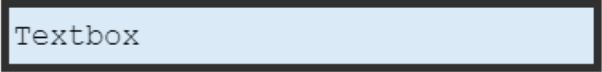
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad() {
var viewer = document.getElementById('container').ej2_instances[0];
{/* Defines the Textbox field settings */}
viewer.textFieldSettings = {
// Set the name of the form field element.
name: 'Textbox',
// Specify whether the Textbox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Textbox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Textbox',
// Set the thickness of the Textbox field. To hide the borders, set the value
to 0 (zero).
thickness: 4,

```

```

// Set the value of the form field element.
value: 'Textbox',
// Set the font family of the textbox field.
fontFamily: 'Courier',
// Set the font size of the textbox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the textbox field.
color: 'black',
// Set the border color of the textbox field.
borderColor: 'black',
// Set the background color of the textbox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
// Allows multiline input in the text field. FALSE, by default.
isMultiline: false
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/pdfviewer"
style={{ 'height': '640px' }}
documentLoad={documentLoad}>
{/* Inject the required services */}
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



Textbox

Password field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the Password field properties on a button click.

`html

```
<button onClick={updateProperties}>Update Properties</button>
```

```
,
```

```
`typescript
```

```
// Event triggers on the Update Properties button click.
```

```
function updateProperties() {
```

```
var viewer = document.getElementById('container').ej2_instances[0];
```

```
var formField = viewer.retrieveFormFields();
```

```
viewer.formDesignerModule.updateFormField(formField[0], {
```

```
name: 'Password',
```

```
isReadOnly: true,
```

```
visibility: 'visible',
```

```
isRequired: false,
```

```
isPrint: true,
```

```
tooltip: 'Password',
```

```
thickness: 4,
```

```
value:'Password',
```

```
fontFamily: 'Courier',
```

```
fontSize: 10,
```

```
fontStyle: 'None',
```

```
color: 'black',
```

```
borderColor: 'black',
```

```
backgroundColor: '#daeaf7f',
```

```
alignment: 'Left',
```

```
maxLength: 0,
```

```
bounds: { X: 148, Y: 229, Width: 150, Height: 24 }
```

```
});
```

```
}
```

```
,
```

The following code example explains how to update the properties of the Password field added to the document from the form designer toolbar.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
```

```

import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject} from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad(){
var viewer = document.getElementById('container').ej2_instances[0];
/* Defines the password field settings */
viewer.passwordFieldSettings = {
// Set the name of the form field element.
name: 'Password',
// Specify whether the Password field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Password field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Password',
// Set the thickness of the Password field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'Password',
// Set the font family of the Password field.
fontFamily: 'Courier',
// Set the font size of the Password field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the Password field.
color: 'black',
// Set the border color of the Password field.
borderColor: 'black',
// Set the background color of the Password field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
}
};
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}
documentLoad={documentLoad}>
/* Inject the required services */
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]]
/>

```



```

</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

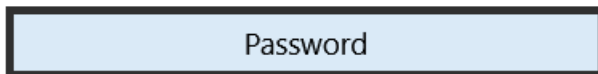
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad() {
var viewer = document.getElementById('container').ej2_instances[0];
/* Defines the password field settings */
viewer.passwordFieldSettings = {
// Set the name of the form field element.
name: 'Password',
// Specify whether the Password field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Password field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Password',
// Set the thickness of the Password field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'Password',
// Set the font family of the Password field.
fontFamily: 'Courier',
// Set the font size of the Password field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the Password field.
color: 'black',
// Set the border color of the Password field.
borderColor: 'black',
// Set the background color of the Password field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,

```

```

}
};
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/pdfviewer"
style={{ 'height': '640px' }}
documentLoad={documentLoad}>
{ /* Inject the required services */}
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}

```



CheckBox field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the CheckBox field properties on a button click.

`html

```
<button onClick={updateProperties}>Update Properties</button>
```

,

`typescript

```
// Event triggers on the Update Properties button click.
```

```
function updateProperties() {
var viewer = document.getElementById('container').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'CheckBox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
```

```

isPrint: true,
tooltip: 'CheckBox',
thickness: 4,
isChecked: true,
backgroundColor: '#daeaf7ff',
borderColor: 'black',
value:'CheckBox'
});
}
`

```

The following code example explains how to update the properties of the CheckBox field added to the document from the form designer toolbar.

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject} from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad() {
var viewer = document.getElementById('container').ej2_instances[0];
// Properties to customize the RadioButton field settings
viewer.checkBoxFieldSettings = {
// Set the name of the form field element.
name: 'CheckBox',
// Specify whether the CheckBox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the CheckBox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'CheckBox',
// Set the thickness of the CheckBox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies whether the check box is in checked state or not.
isChecked: true,
// Set the background color of the check box in hexadecimal string format.
backgroundColor: '#daeaf7ff',
// Set the border color of the check box field.
borderColor: 'black',
// Set the value of the form field element.

```

```

value: 'CheckBox'
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}
documentLoad={documentLoad}
>
  {/* Inject the required services */}
  <Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]
  />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
  function documentLoad() {
    var viewer = document.getElementById('container').ej2_instances[0];
    // Properties to customize the RadioButton field settings
    viewer.checkBoxFieldSettings = {
      // Set the name of the form field element.
      name: 'CheckBox',
      // Specify whether the CheckBox field is in read-only or read-write mode.
      isReadOnly: false,
      // Set the visibility of the form field.
      visibility: 'visible',
      // Specify whether the field is mandatory or not.
      isRequired: false,
      // Specify whether to print the CheckBox field.
      isPrint: true,
      // Set the text to be displayed as a tooltip.
      tooltip: 'CheckBox',
      // Set the thickness of the CheckBox field. To hide the borders, set the
value to 0 (zero).
      thickness: 4,
      // Specifies whether the check box is in checked state or not.

```

```

isChecked: true,
// Set the background color of the check box in hexadecimal string format.
backgroundColor: '#daeaf7ff',
// Set the border color of the check box field.
borderColor: 'black',
// Set the value of the form field element.
value: 'CheckBox'
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/pdfviewer"
style={{ 'height': '640px' }}
documentLoad={documentLoad}
>
{/* Inject the required services */}
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



RadioButton field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the RadioButton field properties on a button click.

`html

```
<button onClick={updateProperties}>Update Properties</button>
```

,

`typescript

```
// Event triggers on the Update Properties button click.
```

```
function updateProperties() {
```

```

var viewer = document.getElementById('container').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
  name: 'RadioButton',
  isReadOnly: true,
  visibility: 'visible',
  isRequired: false,
  isPrint: true,
  tooltip: 'RadioButton',
  thickness: 4,
  isSelected: true,
  backgroundColor: '#daeaf7ff',
  borderColor: 'black',
  value: 'RadioButton'
});
}
`

```

The following code example explains how to update the properties of the RadioButton field added to the document from the form designer toolbar.

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
  function documentLoad() {
    var viewer = document.getElementById('container').ej2_instances[0];
    // Properties to customize the RadioButton field settings
    viewer.radioButtonFieldSettings = {
      // Set the name of the form field element.
      name: 'RadioButton',
      // Specify whether the RadioButton field is in read-only or read-write mode.
      isReadOnly: false,
      // Set the visibility of the form field.
      visibility: 'visible',
      // Specify whether the field is mandatory or not.
      isRequired: false,
      // Specify whether to print the RadioButton field.
      isPrint: true,

```

```

// Set the text to be displayed as a tooltip.
tooltip: 'RadioButton',
// Set the thickness of the RadioButton field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies whether the radio button is in checked state or not.
isSelected: true,
// Set the background color of the radio button in hexadecimal string format.
backgroundColor: '#daeaf7ff',
// Set the border color of the radio button field.
borderColor: 'black',
// Set the value of the form field element.
value: 'RadioButton'
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}
documentLoad={documentLoad}
>
/* Inject the required services */
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad() {
var viewer = document.getElementById('container').ej2_instances[0];
// Properties to customize the RadioButton field settings
viewer.radioButtonFieldSettings = {
// Set the name of the form field element.
name: 'RadioButton',
// Specify whether the RadioButton field is in read-only or read-write mode.
isReadOnly: false,

```

```

// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the RadioButton field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'RadioButton',
// Set the thickness of the RadioButton field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies whether the radio button is in checked state or not.
isSelected: true,
// Set the background color of the radio button in hexadecimal string format.
backgroundColor: '#daeaf7ff',
// Set the border color of the radio button field.
borderColor: 'black',
// Set the value of the form field element.
value: 'RadioButton'
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/pdfviewer"
style={{ 'height': '640px' }}
documentLoad={documentLoad}
>
{/* Inject the required services */}
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



ListBox field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the ListBox field properties on a button click.

`html


```

<button onClick={updateProperties}>Update Properties</button>
,
`ts
// Event triggers on the Update Properties button click.
var customOptions = [{itemName:'item1',itemValue:'item1'}, {itemName:'item2',itemValue:'item2'},
{itemName:'item3',itemValue:'item3'}]
function updateProperties() {
var viewer = document.getElementById('container').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'ListBox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'ListBox',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
options: customOptions,
});
}
,

```

The following code example explains how to update the properties of the ListBox field added to the document from the form designer toolbar.

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';

```

```

import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject} from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad(){
var viewer = document.getElementById('container').ej2_instances[0];
var customOptions = [{itemName:'item1',itemValue:'item1'},
{itemName:'item2',itemValue:'item2'}, {itemName:'item3',itemValue:'item3'}]
// Properties to customize the RadioButton field settings
viewer.listBoxFieldSettings = {
// Set the name of the form field element.
name: 'ListBox',
// Specify whether the ListBox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the ListBox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'ListBox',
// Set the thickness of the ListBox field. To hide the borders, set the value
to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'ListBox',
// Set the font family of the ListBox field.
fontFamily: 'Courier',
// Set the font size of the ListBox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the ListBox field.
color: 'black',
// Set the border color of the ListBox field.
borderColor: 'black',
// Set the background color of the ListBox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the listbox items.
options: customOptions
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}
documentLoad={documentLoad}
>
</div>
</div>
/* Inject the required services */

```

```

<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject} from '@syncfusion/ej2-react-pdfviewer';
function App() {
function documentLoad(){
var viewer = document.getElementById('container').ej2_instances[0];
var customOptions = [{itemName:'item1',itemValue:'item1'},
{itemName:'item2',itemValue:'item2'}, {itemName:'item3',itemValue:'item3'}]
// Properties to customize the RadioButton field settings
viewer.listBoxFieldSettings = {
// Set the name of the form field element.
name: 'ListBox',
// Specify whether the ListBox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the ListBox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'ListBox',
// Set the thickness of the ListBox field. To hide the borders, set the value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'ListBox',
// Set the font family of the ListBox field.
fontFamily: 'Courier',
// Set the font size of the ListBox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the ListBox field.
color: 'black',
// Set the border color of the ListBox field.
borderColor: 'black',

```

```
// Set the background color of the ListBox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the listbox items.
options: customOptions
};
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/pdfviewer"
style={{ 'height': '640px' }}
documentLoad={documentLoad}
>
{/* Inject the required services */}
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}
```



DropDown field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the DropDown field properties on a button click.

`typescript

// Event triggers on the Update Properties button click.

```
var customOptions = [{itemName:'item1',itemValue:'item1'}, {itemName:'item2',itemValue:'item2'},
{itemName:'item3',itemValue:'item3'}]
```

```
function updateProperties() {
```

```
var viewer = document.getElementById('container').ej2_instances[0];
```

```
var formField = viewer.retrieveFormFields();
```

```
viewer.formDesignerModule.updateFormField(formField[0], {
  name: 'DropDown',
  isReadOnly: true,
  visibility: 'visible',
  isRequired: false,
  isPrint: true,
  tooltip: 'DropDown',
  thickness: 4,
  fontFamily: 'Courier',
  fontSize: 10,
  fontStyle: 'None',
  color: 'black',
  borderColor: 'black',
  backgroundColor: '#daeaf7ff',
  alignment: 'Left',
  options: customOptions,
});
}
,
```

The following code example explains how to update the properties of the DropDown field added to the document from the form designer toolbar.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
  var customOptions = [{itemName:'item1',itemValue:'item1'},
  {itemName:'item2',itemValue:'item2'}, {itemName:'item3',itemValue:'item3'}]
  function documentLoad() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.DropDownFieldSettings = {
      // Set the name of the form field element.
      name: 'DropDown',
      // Specify whether the DropDown field is in read-only or read-write mode.
      isReadOnly: false,
      // Set the visibility of the form field.
```

```

visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the DropDown field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'DropDown',
// Set the thickness of the DropDown field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'DropDown',
// Set the font family of the DropDown field.
fontFamily: 'Courier',
// Set the font size of the DropDown field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the DropDown field.
color: 'black',
// Set the border color of the DropDown field.
borderColor: 'black',
// Set the background color of the DropDown field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the DropDown items.
options: customOptions
}
}
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}
documentLoad={documentLoad}
>
{/* Inject the required services */}
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';

```

```

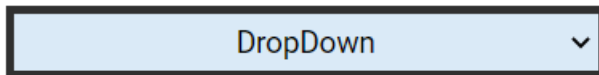
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, FormFields, FormDesigner,
Inject } from '@syncfusion/ej2-react-pdfviewer';
function App() {
  var customOptions = [{itemName:'item1',itemValue:'item1'},
  {itemName:'item2',itemValue:'item2'}, {itemName:'item3',itemValue:'item3'}]
  function documentLoad() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.DropDownFieldSettings = {
      // Set the name of the form field element.
      name: 'DropDown',
      // Specify whether the DropDown field is in read-only or read-write mode.
      isReadOnly: false,
      // Set the visibility of the form field.
      visibility: 'visible',
      // Specify whether the field is mandatory or not.
      isRequired: false,
      // Specify whether to print the DropDown field.
      isPrint: true,
      // Set the text to be displayed as a tooltip.
      tooltip: 'DropDown',
      // Set the thickness of the DropDown field. To hide the borders, set the
      value to 0 (zero).
      thickness: 4,
      // Set the value of the form field element.
      value: 'DropDown',
      // Set the font family of the DropDown field.
      fontFamily: 'Courier',
      // Set the font size of the DropDown field.
      fontSize: 10,
      // Specify the font style
      fontStyle: 'None',
      // Set the font color of the DropDown field.
      color: 'black',
      // Set the border color of the DropDown field.
      borderColor: 'black',
      // Set the background color of the DropDown field.
      backgroundColor: '#daeaf7ff',
      // Set the alignment of the text.
      alignment: 'Left',
      // Set the DropDown items.
      options: customOptions
    }
  }
  return (<div>
    <div className='control-section'>
    <PdfViewerComponent
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      serviceUrl="https://ej2services.syncfusion.com/production/web-
      services/api/pdfviewer"
      style={{ 'height': '640px' }}
      documentLoad={documentLoad}
    >

```

```

{ /* Inject the required services */}
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



Create with user interface interaction in React Pdfviewer component

The PDF viewer control provides the option for interaction with Form Fields such as Drag and resize. you can draw a Form Field dynamically by clicking the Form Field icon on the toolbar and draw it in the PDF document. The Form Fields type supported by the PDF Viewer Control are:

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Enable or Disable form designer toolbar

We should inject FormDesigner module and set enableFormDesignerToolbar as true to enable the Form designer icon on the toolbar. By default, enableFormDesignerToolbar is set as true. Use the following code to inject FormDesigner module and to enable the enableFormDesignerToolbar property.

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner,
FormFields } from '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
return <div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<PdfViewerComponent

```



```

id="container"
ref={ (scope) => { pdfviewer = scope; }}
documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
enableFormDesignerToolbar={true}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormDesigner, FormFields]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

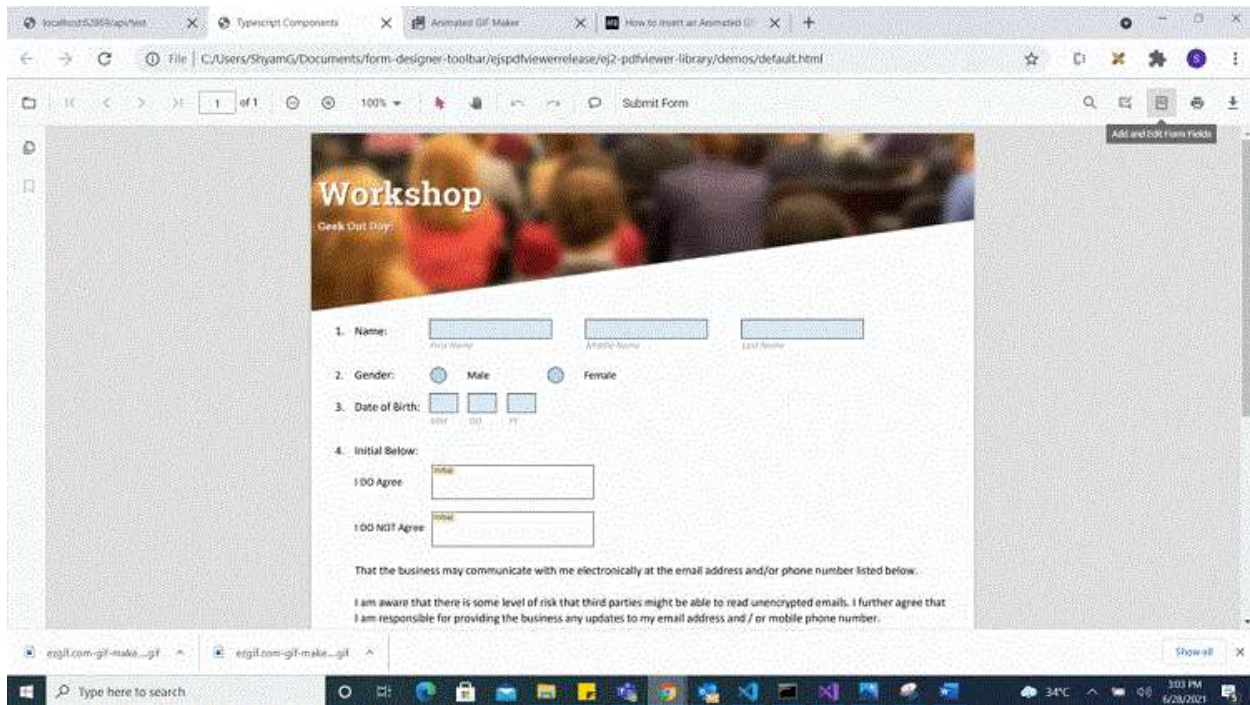
```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner,
FormFields } from '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
return (<div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
ref={ (scope) => { pdfviewer = scope; }}
documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
enableFormDesignerToolbar={true}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormDesigner, FormFields]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

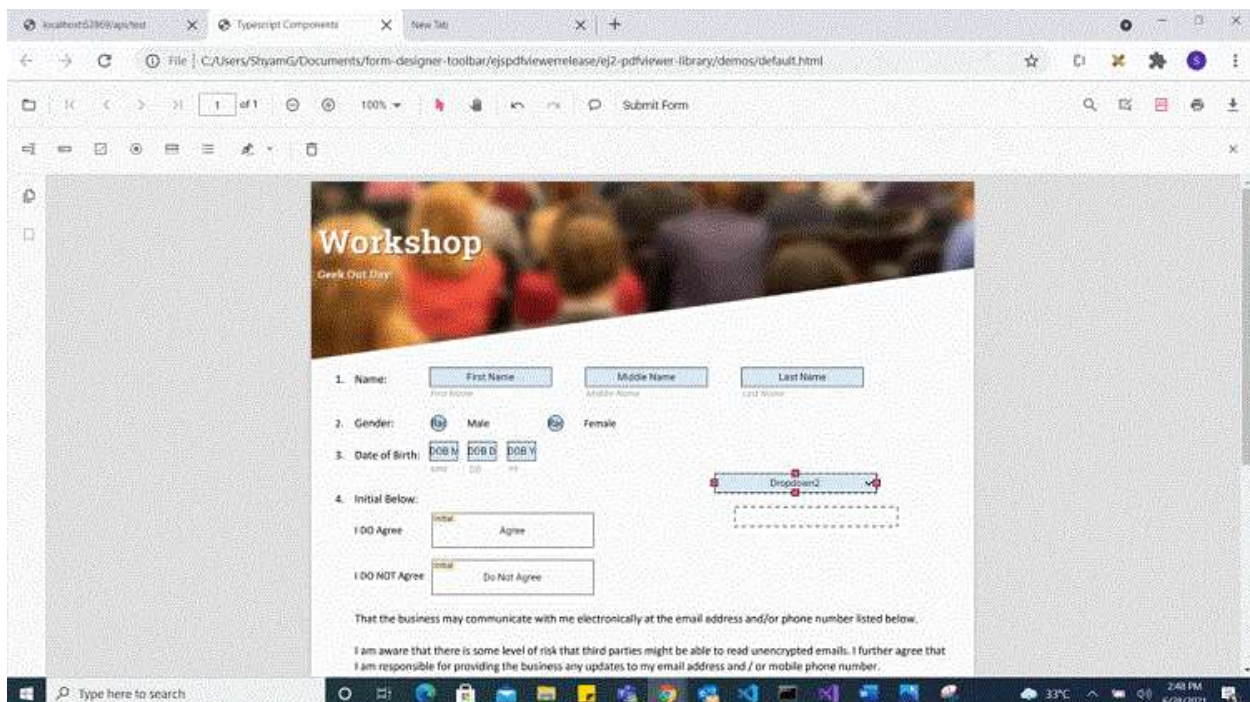
Add the form field dynamically

Click the Form Field icon on the toolbar and then click on to the PDF document to draw a Form Field. Refer the below GIF for further reference.



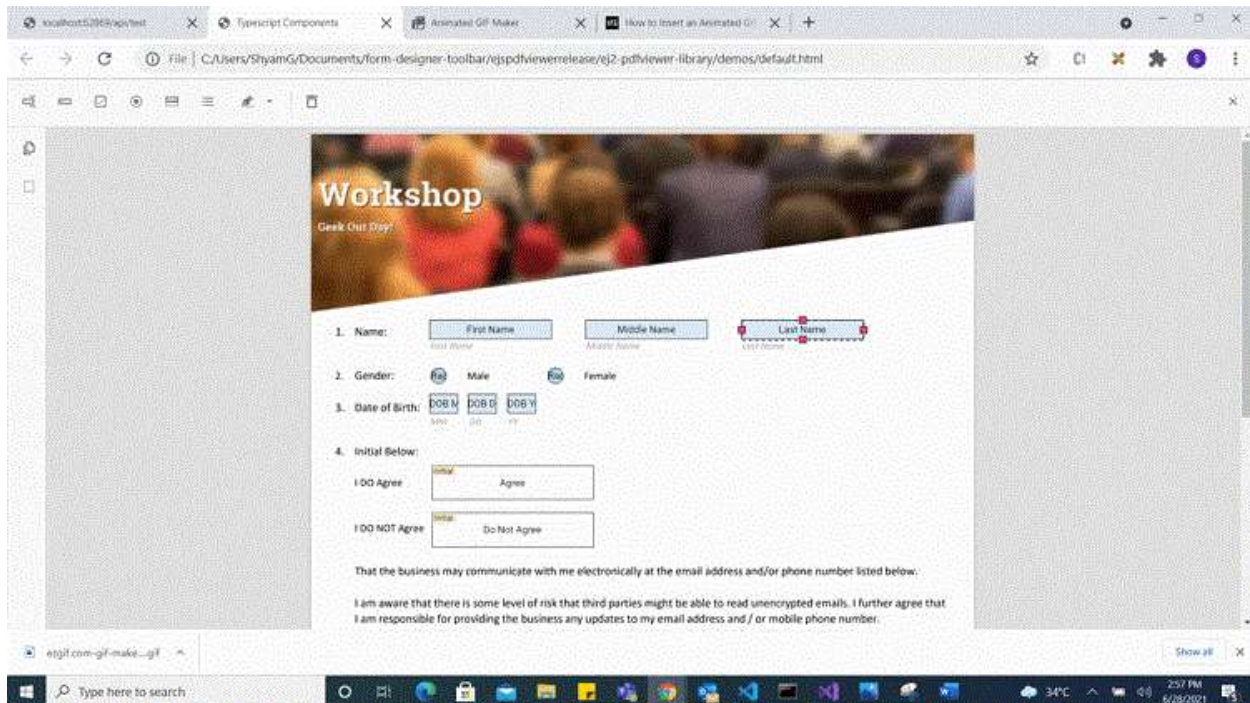
Drag the form field

We provide options to drag the Form Field which is currently selected in the PDF document. Refer the below GIF for further reference.



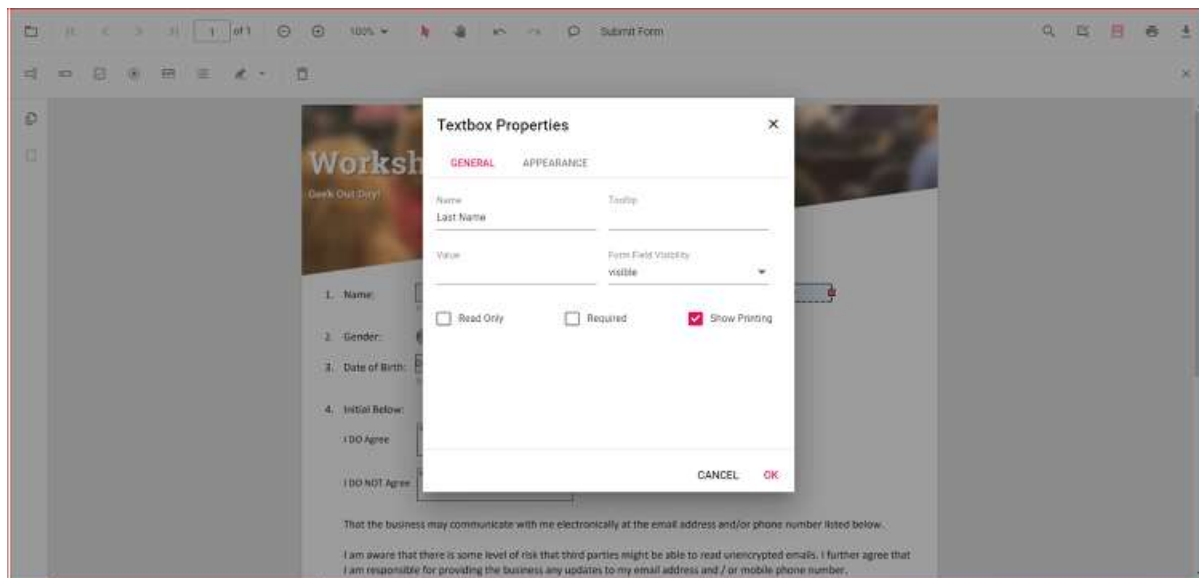
Resize the form field

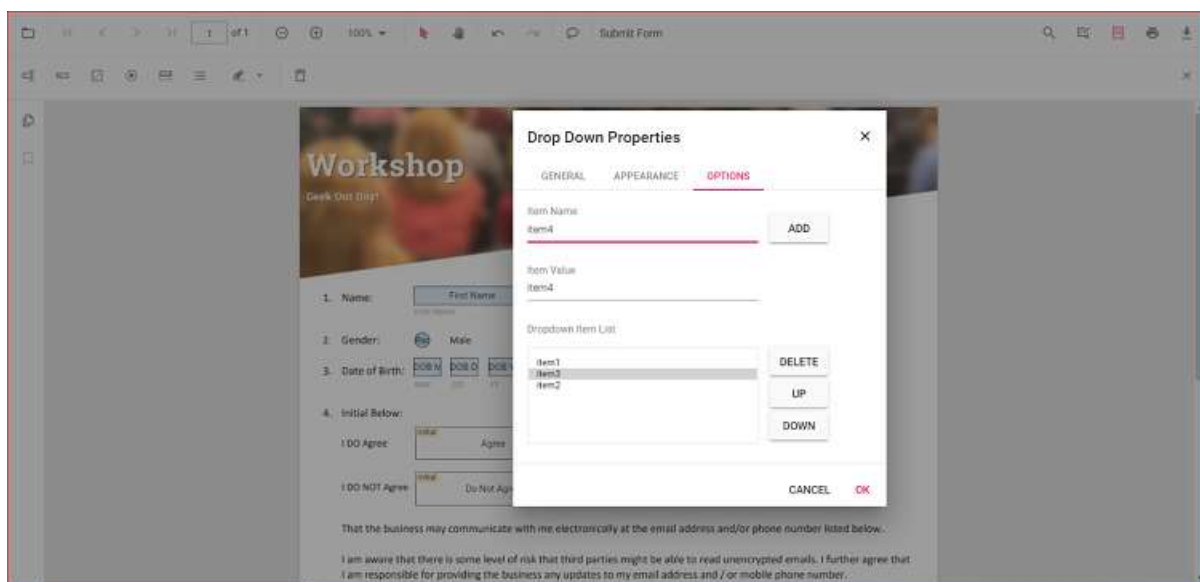
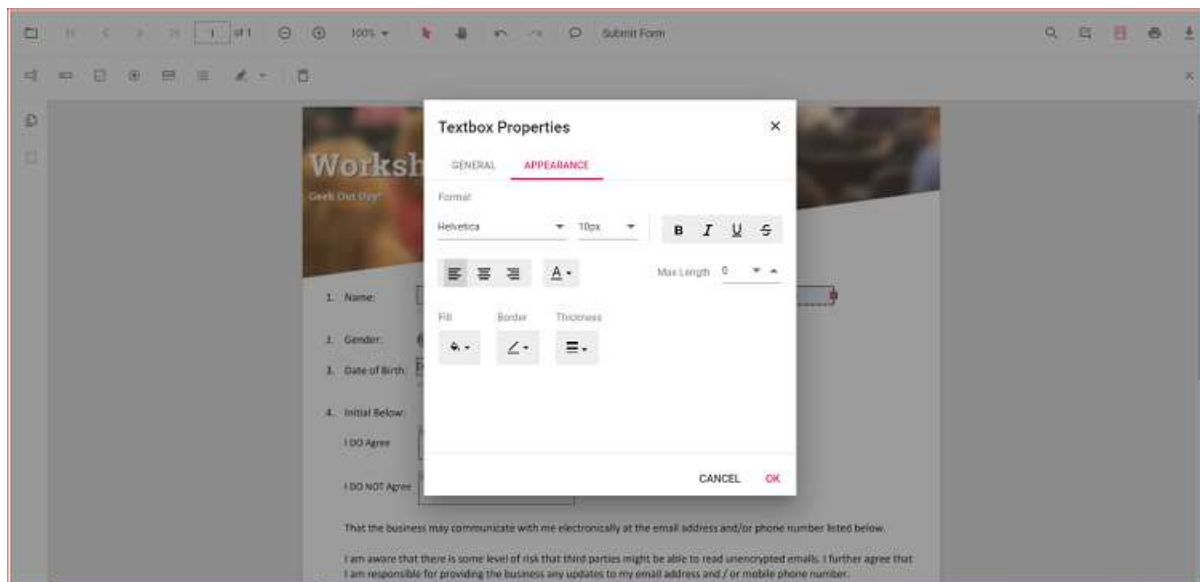
We provide options to resize the Form Field which is currently selected in the PDF document. Refer the below GIF for further reference.



Edit or Update the form field dynamically

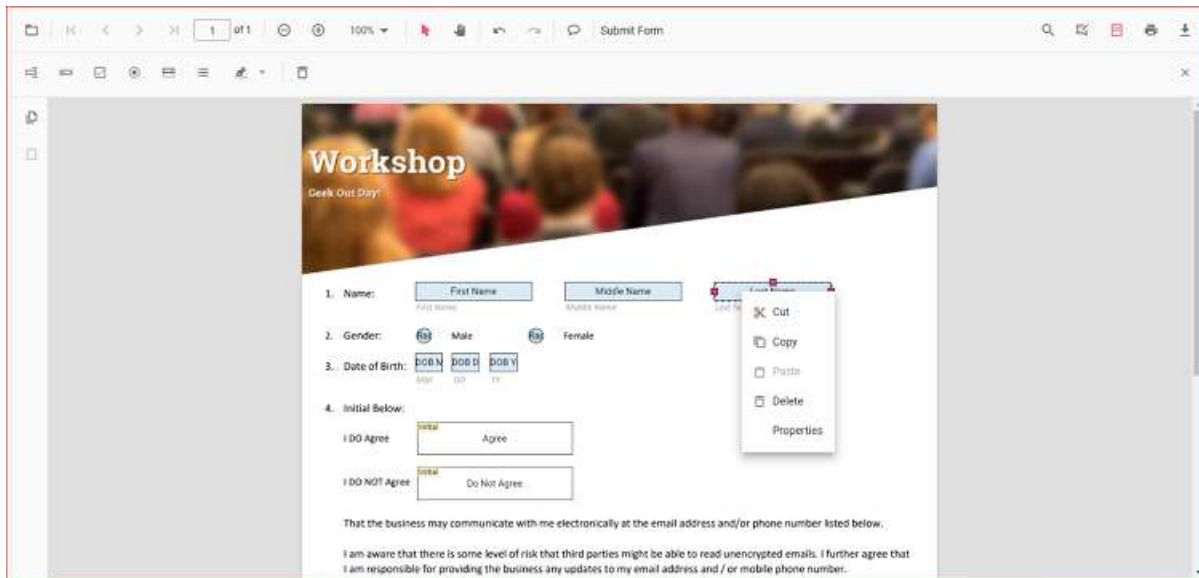
The properties of the Form Fields can be edited using the Form Field Properties window. It can be opened by selecting the Properties option in the context menu that appears on the right by clicking the Form Field object. Refer the below image for the properties available to customize the appearance of the Form Field.





Clipboard operation with form field

The PDF Viewer control supports the clipboard operations such as cut, copy and paste for Form Fields. You can right click on the Form Field object to view the context menu and select to the clipboard options that you would like to perform. Refer the below image for the options in the context menu.



Undo and Redo

We provided support to undo/redo the Form Field actions that are performed at runtime. Use the following code example to perform undo/redo actions.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner,
FormFields } from '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function undoClicked() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.undo();
  }
  function redoClicked() {
    var viewer = document.getElementById('container').ej2_instances[0];
    viewer.redo();
  }
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <button onClick={undoClicked}>Undo</button>
      <button onClick={redoClicked}>Redo</button>
      <PdfViewerComponent
        id="container"
        ref={ (scope) => { pdfviewer = scope; } }
        documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Magnification, Navigation, Annotation,
        LinkAnnotation, BookmarkView, ThumbnailView,
```

```
Print, TextSelection, TextSearch, FormDesigner, FormFields]] />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, Annotation, TextSearch, Inject, FormDesigner,
FormFields } from '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function undoClicked() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.undo();
}
function redoClicked() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.redo();
}
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */}
<button onClick={undoClicked}>Undo</button>
<button onClick={redoClicked}>Redo</button>
<PdfViewerComponent
id="container"
ref={ (scope) => { pdfviewer = scope; }}
documentPath="https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormDesigner, FormFields]] />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Print in React Pdfviewer component

The PDF Viewer supports printing the loaded PDF file. You can enable/disable the print using the following code snippet.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
        enablePrint={true}
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

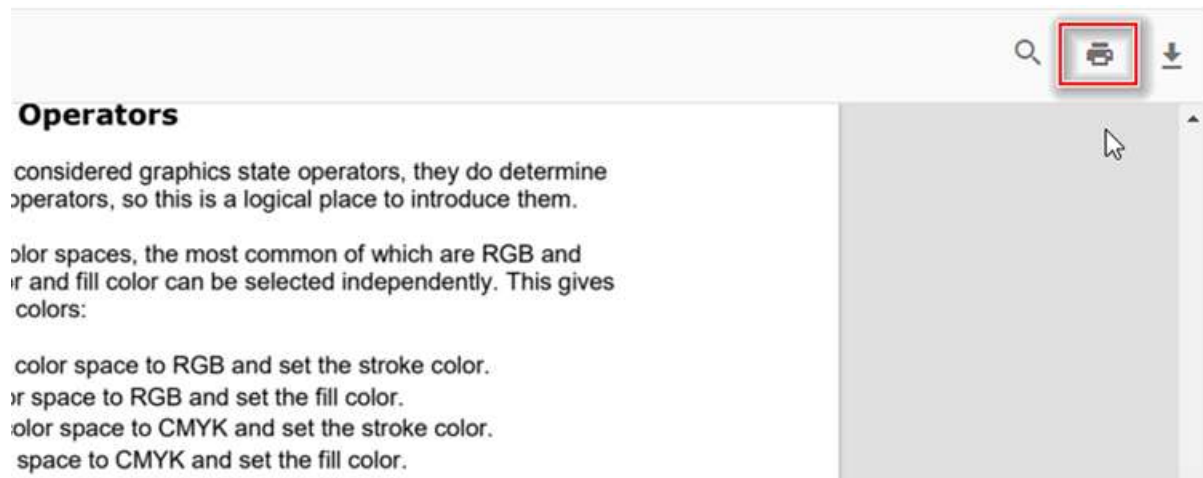
SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        id="container"
        documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
        enablePrint={true}
        serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
        style={{ 'height': '640px' }}>
        <Inject services={[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch]}/>
      </PdfViewerComponent>
    </div>
  </div>);
}
```

```

</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```



You can invoke print action using the following code snippet.,

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
function printClicked() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.print.print()
}
return (<div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<button onClick={printClicked}>Print</button>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services=[ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch ] />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));

```



```
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

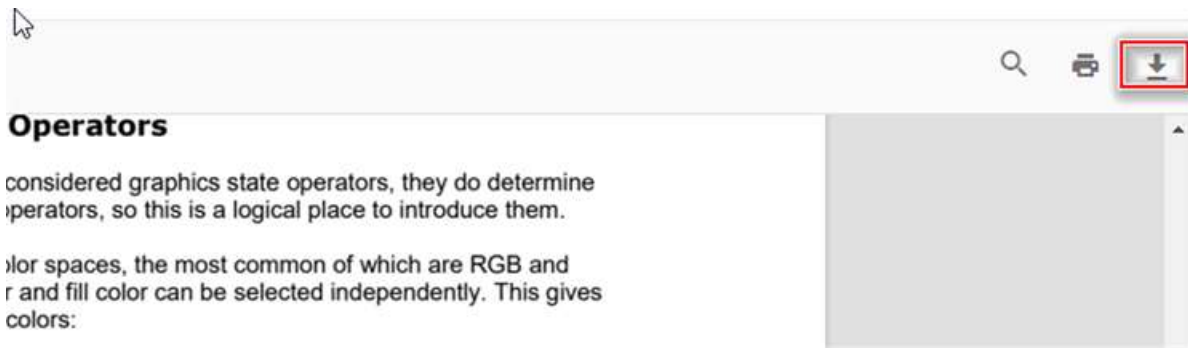
```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
function printClicked() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.print.print()
}
return (<div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<button onClick={printClicked}>Print</button>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch ]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

See also

- [Toolbar items](#)
- [Feature Modules](#)

Download in React Pdfviewer component

The PDF Viewer supports downloading the loaded PDF file. You can enable/disable the download using the following code snippet.



You can invoke download action using following code snippet.,

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function downloadClicked() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.download();
}
return (<div>
<div className='control-section'>
{ /* Render the PDF Viewer */ }
<button onClick={downloadClicked}>Download</button>
<PdfViewerComponent
ref={ (scope) => { pdfviewer = scope; } }
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
Annotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
```

```

import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function downloadClicked() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.download();
}
return (<div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<button onClick={downloadClicked}>Download</button>
<PdfViewerComponent
ref={ (scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
Annotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}

```

See also

- [Toolbar items](#)
- [Feature Modules](#)

Globalization in React Pdfviewer component

The text contents provided in the PDF Viewer can be localized using the collection of localized strings for different cultures. By default, the PDF Viewer is localized in “**en-US**”.

The following table shows the default text values used in PDF Viewer in 'en-US' culture:

Keywords	Values
---	---
PdfViewer	PDF Viewer
Cancel	Cancel
Download file	Download file
Download	Download

Enter Password	This document is password protected. Please enter a password.
File Corrupted	File corrupted
File Corrupted Content	The file is corrupted and cannot be opened.
Fit Page	Fit page
Fit Width	Fit width
Automatic	Automatic
Go To First Page	Show first page
Invalid Password	Incorrect password. Please try again.
Next Page	Show next page
OK	OK
Open	Open file
Page Number	Current page number
Previous Page	Show previous page
Go To Last Page	Show last page
Zoom	Zoom
Zoom In	Zoom in
Zoom Out	Zoom out
Page Thumbnails	Page thumbnails
Bookmarks	Bookmarks
Print	Print file
Password Protected	Password required
Copy	Copy
Text Selection	Text selection tool
Panning	Pan mode
Text Search	Find text
Find in document	Find in document
Match case	Match case
Apply	Apply
GoToPage	Go to page
No matches	Viewer has finished searching the document. No more matches were found
No Text Found	No Text Found
Undo	Undo
Redo	Redo

Annotation	Add or Edit annotations
FormDesigner	Add and Edit Form Fields
Highlight	Highlight Text
Underline	Underline Text
Strikethrough	Strikethrough Text
Delete	Delete annotation
Opacity	Opacity
Color edit	Change Color
Opacity edit	Change Opacity
Highlight context	Highlight
Underline context	Underline
Strikethrough context	Strike through
Server error	Web-service is not listening. PDF Viewer depends on web-service for all it's features.
Please start the web service to continue.	
Client error	Client-side error is found. Please check the custom headers provided in the
AjaxRequestSettings property and web action methods in the ServerActionSettings property.	
Open text	Open
First text	First Page
Previous text	Previous Page
Next text	Next Page
Last text	Last Page
Zoom in text	Zoom In
Zoom out text	Zoom Out
Selection text	Selection
Pan text	Pan
Print text	Print
Search text	Search
Annotation Edit text	Edit Annotation
FormDesigner Edit text	Add and Edit Form Fields
Line Thickness	Line Thickness
Line Properties	Line Properties
Start Arrow	Start Arrow
End Arrow	End Arrow
Line Style	Line Style

Fill Color	Fill Color
Line Color	Line Color
None	None
Open Arrow	Open
Closed Arrow	Closed
Round Arrow	Round
Square Arrow	Square
Diamond Arrow	Diamond
Butt	Butt
Cut	Cut
Paste	Paste
Delete Context	Delete
Properties	Properties
Add Stamp	Add Stamp
Add Shapes	Add Shapes
Stroke edit	Change Stroke Color
Change thickness	Change Border Thickness
Add line	Add Line
Add arrow	Add Arrow
Add rectangle	Add Rectangle
Add circle	Add Circle
Add polygon	Add Polygon
Add Comments	Add Comments
Comments	Comments
SubmitForm	Submit Form
No Comments Yet	No Comments Yet
Accepted	Accepted
Completed	Completed
Cancelled	Cancelled
Rejected	Rejected
Leader Length	Leader Length
Scale Ratio	Scale Ratio
Calibrate	Calibrate

Calibrate Distance	Calibrate Distance
Calibrate Perimeter	Calibrate Perimeter
Calibrate Area	Calibrate Area
Calibrate Radius	Calibrate Radius
Calibrate Volume	Calibrate Volume
Depth	Depth
Closed	Closed
Round	Round
Square	Square
Diamond	Diamond
Edit	Edit
Comment	Comment
Comment Panel	Comment Panel
Set Status	Set Status
Post	Post
Page	Page
Add a comment	Add a comment
Add a reply	Add a reply
Import Annotations	Import annotations from JSON file
Export Annotations	Export annotation to JSON file
Export XFDF	Export annotation to XFDF file
Import XFDF	Import annotations from XFDF file
Add	Add
Clear	Clear
Bold	Bold
Italic	Italic
Strikethroughs	Strikethrough
Underlines	Underline
Superscript	Superscript
Subscript	Subscript
Align left	Align Left
Align right	Align Right
Center	Center

Justify	Justify
Font color	Font Color
Text Align	Text Align
Text Properties	Font Style
SignatureFieldDialogHeaderText	Add Signature
HandwrittenSignatureDialogHeaderText	Add Signature
InitialFieldDialogHeaderText	Add Initial
HandwrittenInitialDialogHeaderText	Add Initial
Draw Ink	Draw Ink
Create	Create
Font family	Font Family
Font size	Font Size
Free Text	Free Text
Import Failed	Invalid JSON file type or file name; please select a valid JSON file
File not found	Imported JSON file is not found in the desired location
Export Failed	Export annotations action has failed; please ensure annotations are added properly
of	of
Dynamic	Dynamic
Standard Business	Standard Business
Sign Here	Sign Here
Custom Stamp	Custom Stamp
Enter Signature as Name	Enter your name
Draw-hand Signature	DRAW
Type Signature	TYPE
Upload Signature	UPLOAD
Browse Signature Image	BROWSE
Save Signature	Save Signature
Save Initial	Save Initial
Textbox	Textbox
Password	Password
Check Box	Checkbox
Radio Button	Radio Button
Dropdown	Drop Down

List Box	List Box
Signature	Signature
Delete FormField	Delete Form Field
Textbox Properties	Textbox Properties
Name	Name
Tooltip	Tooltip
Value	Value
Form Field Visibility	Form Field Visibility
Read Only	Read Only
Required	Required
Checked	Checked
Show Printing	Show Printing
Formatting	Format
Fill	Fill
Border	Border
Border Color	Border Color
Thickness	Thickness
Max Length	Max Length
List Item	Item Name
Export Value	Item Value
Dropdown Item List	Dropdown Item List
List Box Item List	List Box Item List
General	GENERAL
Appearance	APPEARANCE
Options	OPTIONS
Delete Item	Delete
Up	Up
Down	Down
Multiline	Multiline
Revised	Revised
Reviewed	Reviewed
Received	Received
Confidential	Confidential

Approved	Approved
Not Approved	Not Approved
Witness	Witness
Initial Here	Initial Here
Draft	Draft
Final	Final
For Public Release	For Public Release
Not For Public Release	Not For Public Release
For Comment	For Comment
Void	Void
Preliminary Results	Preliminary Results
Information Only	Information Only
in	in
m	m
ftin	ftin
ft	ft
p	p
cm	cm
mm	mm
pt	pt
cu	cu
sq	sq
Initial	Initial

The different locale value for the PDF Viewer can be specified using the locale property.

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormDesigner,
FormFields, Inject } from '@syncfusion/ej2-react-pdfviewer';
import { L10n } from '@syncfusion/ej2-base';
//PDF Viewer Arabic Sample Locale
L10n.load({
  'ar-AE': {
    'PdfViewer' : {
      'PdfViewer': 'قوات الدفاع الشعبي المشاهد',
    }
  }
});
  
```

```

'Cancel': 'إلغاء',
'Download file': 'تحميل الملف',
'Download': 'تحميل',
'Enter Password': 'هذا المستند محمي بكلمة مرور. يرجى إدخال كلمة مرور.',
'File Corrupted': 'ملف تالف',
'File Corrupted Content': 'الملف تالف ولا يمكن فتحه',
'Fit Page': 'لائق بدنيا الصفحة',
'Fit Width': 'لائق بدنيا عرض',
'Automatic': 'تلقائي',
'Go To First Page': 'عرض الصفحة الأولى',
'Invalid Password': 'كلمة سر خاطئة. حاول مرة اخرى',
'Next Page': 'عرض الصفحة التالية',
'OK': 'حسنًا',
'Open': 'فتح الملف',
'Page Number': 'رقم الصفحة الحالية',
'Previous Page': 'عرض الصفحة السابقة',
'Go To Last Page': 'عرض الصفحة الأخيرة',
'Zoom': 'تكبير',
'Zoom In': 'تكبير في',
'Zoom Out': 'تكبير خارج',
'Page Thumbnails': 'مصغرات الصفحة',
'Bookmarks': 'المرجعية',
'Print': 'اطبع الملف',
'Password Protected': 'كلمة المرور مطلوبة',
'Copy': 'نسخ',
'Text Selection': 'أداة اختيار النص',
'Panning': 'وضع عموم',
'Text Search': 'بحث عن نص',
'Find in document': 'ابحث في المستند',
'Match case': 'حالة مبالاة',
'Apply': 'تطبيق',
'GoToPage': 'انتقل إلى صفحة',
'No matches': 'انتهى العارض من البحث في المستند. لم يتم العثور على مزيد من',
'التطابقات',
'No Text Found': 'لم يتم العثور على نص',
'Undo': 'فك',
'Redo': 'فعل ثانية',
'Annotation': 'إضافة أو تعديل التعليقات التوضيحية',
'Highlight': 'تسليط الضوء على النص',
'Underline': 'تسطير النص',
'Strikethrough': 'نص يتوسطه خط',
>Delete': 'حذف التعليق التوضيحي',
'Opacity': 'غموض',
'Color edit': 'غير اللون',
'Opacity edit': 'تغيير التعتيم',
'highlight': 'تسليط الضوء',
'underline': 'أكد',
'strikethrough': 'يتوسطه',
// tslint:disable-next-line:max-line-length
'Server error': 'خدمة الانترنت لا يستمع. يعتمد قوات الدفاع الشعبي المشاهد على',
'خدمة الويب لجميع ميزاته. يرجى بدء خدمة الويب للمتابعة',
'Open text': 'افتح',
'First text': 'الصفحة الأولى',
'Previous text': 'الصفحة السابقة',
'Next text': 'الصفحة التالية',
'Last text': 'آخر صفحة',
'Zoom in text': 'تكبير',

```

```

'Zoom out text': 'تصغير',
'Selection text': 'اختيار',
'Pan text': 'مقللة',
'Print text': 'طباعة',
'Search text': 'بحث',
'Annotation Edit text': 'تحرير التعليق التوضيحي',
'Line Thickness': 'سمك الخط',
'Line Properties': 'خط الخصائص',
'Start Arrow': 'ابدأ السهم',
'End Arrow': 'نهاية السهم',
'Line Style': 'أسلوب الخط',
'Fill Color': 'ملء اللون',
'Line Color': 'الخط اللون',
'None': 'لا شيء',
'Open Arrow': 'افتح',
'Closed Arrow': 'مغلق',
'Round Arrow': 'مستدير',
'Square Arrow': 'مربع',
'Diamond Arrow': 'الماس',
'Cut': 'يقطع',
'Paste': 'معجون',
>Delete Context': 'حذف',
'Properties': 'الخصائص',
'Add Stamp': 'إضافة الطوابع',
'Add Shapes': 'أضف الأشكال',
'Stroke edit': 'تغيير لون السكتة الدماغية',
'Change thickness': 'تغيير سمك الحدود',
'Add line': 'إضافة خط',
'Add arrow': 'سهم إضافة',
'Add rectangle': 'أضف مستطيل',
'Add circle': 'إضافة دائرة',
'Add polygon': 'أضف مضلع',
'Add Comments': 'أضف تعليقات',
'Comments': 'تعليقات',
'No Comments Yet': 'لا توجد تعليقات حتى الآن',
'Accepted': 'وافقت',
'Completed': 'منجز',
'Cancelled': 'ألغيت',
'Rejected': 'مرفوض',
'Leader Length': 'زعيم الطول',
'Scale Ratio': 'نسبة مقياس',
'Calibrate': 'عاير',
'Calibrate Distance': 'معايرة المسافة',
'Calibrate Perimeter': 'معايرة محيط',
'Calibrate Area': 'عاير منطقة',
'Calibrate Radius': 'معايرة نصف القطر',
'Calibrate Volume': 'معايرة الحجم',
'Depth': 'عمق',
'Closed': 'مغلق',
'Round': 'مستدير',
'Square': 'ميدان',
'Diamond': 'الماس',
'Edit': 'تصحیح',
'Comment': 'تعليقات',
'Comment Panel': 'لوحة التعليقات',
'Set Status': 'تعيين الحالة',
'Post': 'بريد',

```

```

'Page': 'صفحة',
'Add a comment': 'أضف تعليق',
'Add a reply': 'أضف رد',
'Import Annotations': 'استيراد التعليقات التوضيحية',
'Export Annotations': 'شروح التصدير',
'Add': 'أضف',
'Clear': 'واضح',
'Bold': 'بالخط العريض',
'Italic': 'مائل',
'Strikethroughs': 'يتوسطه خط',
'Underlines': 'تحت الخط',
'Superscript': 'حرف فوقي',
'Subscript': 'الفرعية النصي',
'Align left': 'محاذاة اليسار',
'Align right': 'محاذاة اليمين',
'Center': 'مركز',
'Justify': 'برر',
'Font color': 'لون الخط',
'Text Align': 'محاذاة النص',
'Text Properties': 'نوع الخط',
'Draw Signature': 'ارسم التوقيع',
'Create': 'خلق',
'Font family': 'خط العائلة',
'Font size': 'حجم الخط',
'Free Text': 'نص حر',
'Import Failed': 'نوع ملف سلمان أو اسم الملف غير صالح ؛ يرجى تحديد ملف',
'سلمان صالح',
'File not found': 'لم يتم العثور على ملف سلمان المستورد في الموقع المطلوب',
'Export Failed': 'شل إجراء تصدير التعليقات التوضيحية ؛ يرجى التأكد من إضافة',
'التعليقات التوضيحية بشكل صحيح',
'Dynamic': 'متحرك',
'Standard Business': 'الأعمال القياسية',
'Sign Here': 'وقع هنا',
'Custom Stamp': 'ختم مخصص',
'InitialFieldDialogHeaderText': 'إضافة الأولية',
'HandwrittenInitialDialogHeaderText': 'إضافة الأولية',
'SignatureFieldDialogHeaderText': 'أضف التوقيع',
'HandwrittenSignatureDialogHeaderText': 'أضف التوقيع',
'Draw-hand Signature': 'يرسم',
'Type Signature': 'نوع',
'Upload Signature': 'تحميل',
'Browse Signature Image': 'تصفح',
'Save Signature': 'احفظ التوقيع',
'Save Initial': 'حفظ الأولي',
'Highlight context': 'تسليط الضوء',
'Underline context': 'تسطير',
'Strikethrough context': 'يتوسطه خط',
'FormDesigner': 'إضافة وتحرير حقل النموذج',
'SubmitForm': 'تقديم النموذج',
'Search text': 'بحث',
'Draw Ink': 'ارسم الحبر',
'Revised': 'مراجعة',
'Reviewed': 'تمت المراجعة',
'Received': 'تم الاستلام',
'Confidential': 'مؤتمن',
'Approved': 'وافق',
'Not Approved': 'غير مقبول',

```

```

'Witness': 'الشاهد',
'Initial Here': 'المبدئي هنا',
'Draft': 'مشروع',
'Final': 'أخير',
'For Public Release': 'للنشر العام',
'Not For Public Release': 'ليس للنشر العام',
'For Comment': 'للتعليق',
'Void': 'فارغ',
'Preliminary Results': 'نتائج أولية',
'Information Only': 'المعلومات فقط',
'Enter Signature as Name': 'أدخل أسمك',
'Textbox': 'مربع الكتابة',
'Password': 'كلمه السر',
'Check Box': 'خانة اختيار',
'Radio Button': 'زر الراديو',
'Dropdown': 'اسقاط',
'List Box': 'مربع القائمة',
'Signature': 'إمضاء',
>Delete FormField': 'حذف حقل النموذج',
'FormDesigner Edit text': 'إضافة وتحرير حقل النموذج',
'in': 'في',
'm': 'م',
'ft_in': 'قدم',
'ft': 'قدم',
'p': 'ص',
'cm': 'سم',
'mm': 'مم',
'pt': 'نقطة',
'cu': 'مكعب',
'sq': 'قدم مربع',
'General': 'جنرال لواء',
'Appearance': 'مظهر خارجي',
'Options': 'والخيارات',
'Textbox Properties': 'خصائص مربع النص',
'Name': 'اسم',
'Tooltip': 'تلميح',
'Value': 'القيمة',
'Form Field Visibility': 'رؤية حقل النموذج',
'Read Only': 'يقرأ فقط',
'Required': 'مطلوب',
'Checked': 'التحقق',
>Show Printing': 'عرض الطباعة',
'Formatting': 'صيغة',
'Fill': 'يملأ',
'Border': 'الحدود',
'Border Color': 'لون الحدود',
'Thickness': 'السماكة',
'Max Length': 'الحد الاقصى للطول',
'List Item': 'اسم العنصر',
'Export Value': 'قيمة البند',
'Dropdown Item List': 'قائمة العناصر المنسدلة',
'List Box Item List': 'قائمة عناصر مربع القائمة',
>Delete Item': 'حذف',
'Up': 'فوق',
'Down': 'تحت',
'Multiline': 'متعدد الأسطر',
'Initial': 'أولي',

```

```
'Export XFDF': 'تصدير التعليق التوضيحي إلى ملف',
'Import XFDF': 'استيراد التعليقات التوضيحية من ملف'
}
}
});
function App() {
  return ( <div>
    <div className='control-section'>
      { /* Render the PDF Viewer */ }
    <PdfViewerComponent
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
      locale="ar-AE"
      style={{ 'height': '640px' }}>
      <Inject services=[ Toolbar, Annotation, Magnification, Navigation,
        LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields ]>
      />
    </PdfViewerComponent>
  </div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView,
  ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormDesigner,
  FormFields, Inject } from '@syncfusion/ej2-react-pdfviewer';
import { L10n } from '@syncfusion/ej2-base';
//PDF Viewer Arabic Sample Locale
L10n.load({
  'ar-AE': {
    'PdfViewer' : {
      'PdfViewer': 'قوات الدفاع الشعبي المشاهد',
      'Cancel': 'إلغاء',
      'Download file': 'تحميل الملف',
      'Download': 'تحميل',
      'Enter Password': 'هذا المستند محمي بكلمة مرور. يرجى إدخال كلمة مرور',
      'File Corrupted': 'ملف تالف',
      'File Corrupted Content': 'الملف تالف ولا يمكن فتحه',
      'Fit Page': 'لائق بدنيا الصفحة',
      'Fit Width': 'لائق بدنيا عرض',
      'Automatic': 'تلقائي',
      'Go To First Page': 'عرض الصفحة الأولى',
      'Invalid Password': 'كلمة سر خاطئة. حاول مرة أخرى',
      'Next Page': 'عرض الصفحة التالية',
      'OK': 'حسنًا',

```

```

'Open': 'فتح الملف',
'Page Number': 'رقم الصفحة الحالية',
'Previous Page': 'عرض الصفحة السابقة',
'Go To Last Page': 'عرض الصفحة الأخيرة',
'Zoom': 'تكبير',
'Zoom In': 'تكبير في',
'Zoom Out': 'تكبير خارج',
'Page Thumbnails': 'مصفغات الصفحة',
'Bookmarks': 'المرجعية',
'Print': 'اطبع الملف',
'Password Protected': 'كلمة المرور مطلوبة',
'Copy': 'نسخ',
'Text Selection': 'أداة اختيار النص',
'Panning': 'وضع عموم',
'Text Search': 'بحث عن نص',
'Find in document': 'ابحث في المستند',
'Match case': 'حالة مباراة',
'Apply': 'تطبيق',
'GoToPage': 'انتقل إلى صفحة',
'No matches': 'انتهى العارض من البحث في المستند. لم يتم العثور على مزيد من',
'التطابقات',
'No Text Found': 'لم يتم العثور على نص',
'Undo': 'فك',
'Redo': 'فعل ثانية',
'Annotation': 'إضافة أو تعديل التعليقات التوضيحية',
'Highlight': 'تسليط الضوء على النص',
'Underline': 'تسطير النص',
'Strikethrough': 'نص يتوسطه خط',
>Delete': 'حذف التعليق التوضيحي',
'Opacity': 'غموض',
'Color edit': 'غير اللون',
'Opacity edit': 'تغيير التعقيم',
'highlight': 'تسليط الضوء',
'underline': 'أكد',
'strikethrough': 'يتوسطه',
// tslint:disable-next-line:max-line-length
'Server error': 'خدمة الانترنت لا يستمع. يعتمد قوات الدفاع الشعبي المشاهد على',
'خدمة الويب لجميع ميزاته. يرجى بدء خدمة الويب للمتابعة',
'Open text': 'افتح',
'First text': 'الصفحة الأولى',
'Previous text': 'الصفحة السابقة',
'Next text': 'الصفحة التالية',
'Last text': 'آخر صفحة',
'Zoom in text': 'تكبير',
'Zoom out text': 'تصغير',
'Selection text': 'اختيار',
'Pan text': 'مقلدة',
'Print text': 'طباعة',
'Seatch text': 'بحث',
'Annotation Edit text': 'تحرير التعليق التوضيحي',
'Line Thickness': 'سمك الخط',
'Line Properties': 'خط الخصائص',
'Start Arrow': 'ابدأ السهم',
'End Arrow': 'نهاية السهم',
'Line Style': 'أسلوب الخط',
'Fill Color': 'ملء اللون',
'Line Color': 'الخط اللون',

```



```

'None': 'لا شيء',
'Open Arrow': 'افتح',
'Closed Arrow': 'مغلق',
'Round Arrow': 'مستدير',
'Square Arrow': 'مربع',
'Diamond Arrow': 'الماس',
'Cut': 'يقطع',
'Paste': 'معجون',
>Delete Context': 'حذف',
'Properties': 'الخصائص',
>Add Stamp': 'إضافة الطوابع',
>Add Shapes': 'أضف الأشكال',
'Stroke edit': 'تغيير لون السكتة الدماغية',
'Change thickness': 'تغيير سمك الحدود',
'Add line': 'إضافة خط',
'Add arrow': 'سهم إضافة',
'Add rectangle': 'أضف مستطيل',
'Add circle': 'إضافة دائرة',
'Add polygon': 'أضف مضلع',
'Add Comments': 'أضف تعليقات',
'Comments': 'تعليقات',
'No Comments Yet': 'لا توجد تعليقات حتى الآن',
'Accepted': 'وافقت',
'Completed': 'منجز',
'Cancelled': 'ألغيت',
'Rejected': 'مرفوض',
'Leader Length': 'زعيم الطول',
'Scale Ratio': 'نسبة مقياس',
'Calibrate': 'عاير',
'Calibrate Distance': 'معايرة المسافة',
'Calibrate Perimeter': 'معايرة محيط',
'Calibrate Area': 'عاير منطقة',
'Calibrate Radius': 'معايرة نصف القطر',
'Calibrate Volume': 'معايرة الحجم',
'Depth': 'عمق',
'Closed': 'مغلق',
'Round': 'مستدير',
'Square': 'ميدان',
'Diamond': 'الماس',
>Edit': 'تصحیح',
'Comment': 'تعليقات',
'Comment Panel': 'لوحة التعليقات',
'Set Status': 'تعيين الحالة',
'Post': 'بريد',
'Page': 'صفحة',
'Add a comment': 'أضف تعليق',
'Add a reply': 'أضف رد',
'Import Annotations': 'استيراد التعليقات التوضيحية',
'Export Annotations': 'شروح التصدير',
'Add': 'أضف',
'Clear': 'واضح',
'Bold': 'بالخط العريض',
'Italic': 'مائل',
'Strikethroughs': 'يتوسطه',
'Underlines': 'تحت الخط',
'Superscript': 'حرف فوقي',
'Subscript': 'الفرعية النصي',

```

```

'Align left': 'محاذاة اليسار',
'Align right': 'محاذاة اليمين',
'Center': 'مركز',
'Justify': 'برر',
'Font color': 'لون الخط',
'Text Align': 'محاذاة النص',
'Text Properties': 'نوع الخط',
'Draw Signature': 'ارسم التوقيع',
'Create': 'خلق',
'Font family': 'خط العائلة',
'Font size': 'حجم الخط',
'Free Text': 'نص حر',
'Import Failed': 'نوع ملف سلمان أو اسم الملف غير صالح ؛ يرجى تحديد ملف',
'Sلمانصالح',
'File not found': 'لم يتم العثور على ملف سلمان المستورد في الموقع المطلوب',
'Export Failed': 'شل إجراء تصدير التعليقات التوضيحية ؛ يرجى التأكد من إضافة',
'tعليقات التوضيحية بشكل صحيح',
'Dynamic': 'متحرك',
'Standard Business': 'الأعمال القياسية',
'Sign Here': 'وقع هنا',
'Custom Stamp': 'ختم مخصص',
'InitialFieldDialogHeaderText': 'إضافة الأولية',
'HandwrittenInitialDialogHeaderText': 'إضافة الأولية',
'SignatureFieldDialogHeaderText': 'أضف التوقيع',
'HandwrittenSignatureDialogHeaderText': 'أضف التوقيع',
'Draw-hand Signature': 'يرسم',
'Type Signature': 'نوع',
'Upload Signature': 'تحميل',
'Browse Signature Image': 'تصفح',
'Save Signature': 'احفظ التوقيع',
'Save Initial': 'حفظ الأولي',
'Highlight context': 'تسليط الضوء',
'Underline context': 'تسطير',
'Strikethrough context': 'يتوسطه خط',
'FormDesigner': 'إضافة وتحرير حقل النموذج',
'SubmitForm': 'تقديم النموذج',
'Search text': 'بحث',
'Draw Ink': 'ارسم الحبر',
'Revised': 'مراجعة',
'Reviewed': 'تمت المراجعة',
'Received': 'تم الاستلام',
'Confidential': 'مؤتمن',
'Approved': 'وافق',
'Not Approved': 'غير مقبول',
'Witness': 'الشاهد',
'Initial Here': 'المبدئي هنا',
'Draft': 'مشروع',
'Final': 'أخير',
'For Public Release': 'للنشر العام',
'Not For Public Release': 'ليس للنشر العام',
'For Comment': 'للتعليق',
'Void': 'فارغ',
'Preliminary Results': 'نتائج اولية',
'Information Only': 'المعلومات فقط',
'Enter Signature as Name': 'أدخل أسمك',
'Textbox': 'مربع الكتابة',
'Password': 'كلمه السر',

```

```

'Check Box': 'خانة اختيار',
'Radio Button': 'زر الراديو',
'Dropdown': 'اسقاط',
'List Box': 'مربع القائمة',
'Signature': 'إمضاء',
>Delete FormField': 'حذف حقل النموذج',
'FormDesigner Edit text': 'إضافة وتحرير حقل النموذج',
'in': 'في',
'm': 'م',
'ft_in': 'قدم',
'ft': 'قدم',
'p': 'ص',
'cm': 'سم',
'mm': 'مم',
'pt': 'نقطة',
'cu': 'مكعب',
'sq': 'قدم مربع',
'General': 'جنرال لواء',
'Appearance': 'مظهر خارجي',
'Options': 'والخيارات',
'Textbox Properties': 'خصائص مربع النص',
'Name': 'اسم',
'Tooltip': 'تلميح',
'Value': 'القيمة',
'Form Field Visibility': 'رؤية حقل النموذج',
'Read Only': 'يقرأ فقط',
'Required': 'مطلوب',
'Checked': 'التحقق',
>Show Printing': 'عرض الطباعة',
'Formatting': 'صيغة',
'Fill': 'يملأ',
'Border': 'الحدود',
'Border Color': 'لون الحدود',
'Thickness': 'السماكة',
'Max Length': 'الحد الأقصى للطول',
'List Item': 'اسم العنصر',
'Export Value': 'قيمة البند',
'Dropdown Item List': 'قائمة العناصر المنسدلة',
'List Box Item List': 'قائمة عناصر مربع القائمة',
>Delete Item': 'حذف',
'Up': 'فوق',
'Down': 'تحت',
'Multiline': 'متعدد الأسطر',
'Initial': 'أولي',
'Export XFDF': 'تصدير التعليق التوضيحي إلى ملف',
'Import XFDF': 'استيراد التعليقات التوضيحية من ملف'
}
}
});
function App() {
return ( <div>
<div className='control-section'>
{/* Render the PDF Viewer */}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
locale="ar-AE"

```

```

serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields ]}
/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}

```

Accessibility in Syncfusion React PDF Viewer components

The PDF Viewer 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 PDF Viewer 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

[WAI-ARIA](#) (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components. The following ARIA attributes are used in the PDF Viewer component

| Attributes | Purpose |
|-------------------------------|--|
| --- | --- |
| <code>aria-disabled</code> | Indicates whether the PDF Viewer component is in a disabled state or not. |
| <code>aria-expanded</code> | Indicates whether the suggestion list has expanded or not. |
| <code>aria-readonly</code> | Indicates the readonly state of the PDF Viewer element. |
| <code>aria-haspopup</code> | Indicates whether the PDF Viewer input element has a suggestion list or not. |
| <code>aria-label</code> | Indicates the breadcrumb item text. |
| <code>aria-labelledby</code> | Provides a label for the PDF Viewer. Typically, the "aria-labelledby" attribute will contain the id of the element used as the PDF Viewer's title. |
| <code>aria-describedby</code> | This attribute points to the PDF Viewer element describing the one it's set on. |
| <code>aria-orientation</code> | Indicates whether the PDF Viewer element is oriented horizontally or vertically. |
| <code>aria-valuetext</code> | Returns the current text of the PDF Viewer. |
| <code>aria-valuemax</code> | Indicates the Maximum value of the PDF Viewer. |
| <code>aria-valuemin</code> | Indicates the Minimum value of the PDF Viewer. |
| <code>aria-valuenow</code> | Indicates the current value of the PDF Viewer. |
| <code>aria-controls</code> | Attribute is set to the button and it points to the corresponding content. |

Keyboard interaction

The PDF Viewer 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 Message component.

| Press (Windows) | Press (Macintosh) | To do this |
|-------------------------------|-------------------|------------|
| --- | --- | --- |
| Shortcuts for page navigation | | |

| CONTROL + Left Arrow (or) CONTROL + Up Arrow | COMMAND + Left Arrow (or) COMMAND + Up Arrow | Navigate to the first page |

| CONTROL + Right Arrow (or) CONTROL + Down Arrow | COMMAND + Right Arrow (or) COMMAND + Down Arrow | Navigate to the last page |

| Left Arrow | Left Arrow (or) Shift + Space | Navigate to the previous page |

| Right Arrow | Right Arrow (or) Space | Navigate to the next page |

| CONTROL + G | COMMAND + G | Go To The Page |

| Up Arrow | Up Arrow | Scroll up |

| Down Arrow | Down Arrow | Scroll down |

|| Shortcuts for Zooming |

| CONTROL + = | COMMAND + = | Perform zoom-in operation |

| CONTROL + - | COMMAND + - | Perform zoom-out operation |

| CONTROL + 0 | COMMAND + 0 | Retain the zoom level to 1 |

|| Shortcut for Text Search |

| CONTROL + F | COMMAND + F | Open the search toolbar |

|| Shortcut for Text Selection |

| CONTROL + C | CONTROL + C | Copy the selected text or annotation or form field |

| CONTROL + X | CONTROL + X | Cut the selected text or annotation of the form field |

| CONTROL + Y | CONTROL + Y | Paste the selected text or annotation or form field |

|| Shortcuts for the general operation |

| CONTROL + Z | CONTROL + Z | Undo the action |

| CONTROL + Y | CONTROL + Y | Redo the action |

| CONTROL + P | COMMAND + P | Print the document |

| Delete | Delete | Delete the annotations and form fields |

| CONTROL + Shift + A | COMMAND + Shift + A | Toggle Annotation Toolbar |

| CONTROL + Alt + 0 | COMMAND + Option + 0 | Show Command panel |

| CONTROL + Alt + 2 | COMMAND + Option + 2 | Show Bookmarks |

| CONTROL + Alt + 1 | COMMAND + Option + 1 | Show Thumbnails |

| CONTROL + S | COMMAND + S | Download |

| Shift + H | Shift + H | Enable pan mode |

| Shift + V | Shift + V | Enable text selection mode |

The current implementation of our PDF Viewer includes keyboard shortcuts for various functions like scrolling, zooming, text search, printing, and annotation deletion.

To enhance user experience, we're adding additional keyboard shortcuts for actions such as navigating between pages, accessing specific pages, toggling annotation tools, and displaying PDF elements like outlines, annotations, bookmarks, and thumbnails.

To support this, we're introducing a new class called **commandManager**, which handles custom commands triggered by specific key gestures. These custom commands will be defined by users and executed accordingly.

The **commandManager** will have a parameter called **Commands**, which will hold the collection of custom keyboard commands specified by users. Each custom command will be represented by a **KeyboardCommand** class, containing the **command name** and associated **keyboard combination**.

Additionally, we're introducing the **keyboardCustomCommands** parameter for the **CommandManager**, which will utilize the **EventCallback** to handle keyboard events and trigger appropriate methods when specific key combinations are pressed.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import {PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function commandManager() {
keyboardCommand: [{
name: 'customCopy',
gesture: {
pdfKeys: PdfKeys.G,
modifierKeys: ModifierKeys.Shift | ModifierKeys.Alt
}
},
{
name: 'customPaste',
gesture: {
pdfKeys: PdfKeys.H,
modifierKeys: ModifierKeys.Shift | ModifierKeys.Alt
}
},
{
name: 'customCut',
gesture: {
pdfKeys: PdfKeys.Z,
modifierKeys: ModifierKeys.Control
}
},
{
name: 'customSelectAll',
gesture: {
pdfKeys: PdfKeys.E,
```

```

modifierKeys: ModifierKeys.Control
}
},
]
}
return (<div>
<div className='control-section'>
  { /* Render the PDF Viewer */ }
  <PdfViewerComponent
    ref={ (scope) => { pdfviewer = scope; }}
    id="container"
    documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
    resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
    commandManager = {commandManager}
    style={{ 'height': '640px' }}>
    <Inject services=[ Toolbar, Magnification, Navigation, LinkAnnotation,
    Annotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch] />
  </PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function commandManager() {
keyboardCommand: [{
name: 'customCopy',
gesture: {
pdfKeys: PdfKeys.G,
modifierKeys: ModifierKeys.Shift | ModifierKeys.Alt
}
}],
{
name: 'customPaste',
gesture: {
pdfKeys: PdfKeys.H,
modifierKeys: ModifierKeys.Shift | ModifierKeys.Alt
}
}],
{
name: 'customCut',
gesture: {

```



```

pdfKeys: PdfKeys.Z,
modifierKeys: ModifierKeys.Control
},
{
name: 'customSelectAll',
gesture: {
pdfKeys: PdfKeys.E,
modifierKeys: ModifierKeys.Control
}
},
]
}

return (<div>
<div className='control-section'>
  { /* Render the PDF Viewer */ }
  <PdfViewerComponent
  ref={(scope) => { pdfviewer = scope; }}
  id="container"
  documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
  serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
  commandManager = {commandManager}
  style={{ 'height': '640px' }}>
    <Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
    Annotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch]} />
  </PdfViewerComponent>
</div>
</div>);
}

const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Each **keyboardCommand** object consists of a **name** property, specifying the **name** of the **custom command**, and a **gesture** property, defining the key gesture associated with the command.

For example, the first command named **customCopy** is associated with the **G** key and requires both the **Shift** and **Alt** modifier keys to be pressed simultaneously.

Additionally, there's an explanation of the key modifiers used in the gestures:

- Ctrl corresponds to the Control key, represented by the value **1**.
- Alt corresponds to the Alt key, represented by the value **2**.
- Shift corresponds to the Shift key, represented by the value **4**.
- Meta corresponds to the Command key on macOS or the Windows key on Windows, represented by the value **8**.

This setup allows users to perform custom actions within the PDF viewer by pressing specific key combinations, enhancing the user experience and providing more efficient navigation and interaction options.

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

How To

Toolbar customization in React Pdfviewer component

The PDF Viewer provides API for user interactions options provided in it's built-in toolbar. Using this we can create our own User Interface for toolbar actions in application level by hiding the default toolbar. The following steps are used to create the custom toolbar for PDF Viewer,

Step 1: Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

Step 2: Now, add an HTML div element in template to act as the custom toolbar PDF Viewer using the following code.

STANDALONE

```
{% raw %}
render() {
function template() {
return (
<div><span className='e-pv-total-page-number' id='totalPage'>of
0</span></div>
);
}
function inputTemplate() {
return (
<div><input type='text' className='e-input-group e-pv-current-page-number'
id='currentPage' /></div>
);
}
return (<div>
<div className='control-section'>
<div>
<div className='e-pdf-toolbar'>
<ToolbarComponent ref={(scope) => { this.toolbar = scope; }}
clicked={ this.clickHandler.bind(this) }>
<ItemsDirective>
<ItemDirective prefixIcon='e-pv-open-document-icon' id='file_Open'
tooltipText='Open'></ItemDirective>
<ItemDirective prefixIcon="e-pv-previous-page-navigation-icon"
id='previous_page' tooltipText="Previous Page"
align="Center"></ItemDirective>
<ItemDirective prefixIcon="e-pv-next-page-navigation-icon" id='next_page'
tooltipText="Next Page" align="Center"></ItemDirective>
<ItemDirective template={inputTemplate} tooltipText="Page Number"
type="Input" align="Center"></ItemDirective>
```

```

<ItemDirective template={template} align="Center" tooltipText="Page
Number"></ItemDirective>
<ItemDirective prefixIcon="e-pv-print-document-icon" tooltipText="Print"
id='print' align="Right"></ItemDirective>
<ItemDirective prefixIcon="e-pv-download-document-icon"
tooltipText="Download" id='download' align="Right"></ItemDirective>
</ItemsDirective>
</ToolbarComponent>
</div>
{ /* Render the PDF Viewer */ }
<PdfViewerComponent
id="container"
ref={ (scope) => { this.viewer = scope; } }
enableToolbar={false}
documentLoad={this.documentLoaded}
pageChange={this.onPageChange}
documentPath="https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'display': 'block', 'height': '640px' }}>
<Inject services={[ Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
<input type="file"
id="fileUpload"
accept=".pdf"
onChange={this.readFile.bind(this)}
style={{ 'display': 'block', 'visibility': 'hidden', 'width': '0', 'height':
'0' }} />
<div className='e-pdf-toolbar' id="magnificationToolbarItems">
<ToolbarComponent id="magnificationToolbar"
clicked={this.clickHandler.bind(this)}>
<ItemsDirective >
<ItemDirective prefixIcon="e-pv-fit-page" id='fit_to_page' tooltipText="Fit
to page" ></ItemDirective>
<ItemDirective prefixIcon="e-pv-zoom-in-icon" id='zoom_in' tooltipText="Zoom
in"></ItemDirective>
<ItemDirective prefixIcon="e-pv-zoom-out-sample" id='zoom_out'
tooltipText="Zoom out" ></ItemDirective>
</ItemsDirective>
</ToolbarComponent>
</div>
</div>
</div>
</div>
);
}
{% enddraw %}

```

SERVER-BACKED

```

{% raw %}
render() {
function template() {
return (
<div ><span className='e-pv-total-page-number' id='totalPage'>of
0</span></div>

```

```

);
}
function inputTemplate() {
  return (
    <div><input type='text' className='e-input-group e-pv-current-page-number'
    id='currentPage' /></div>
  );
}
return (
  <div className='control-section'>
    <div>
      <div className='e-pdf-toolbar'>
        <ToolbarComponent ref={(scope) => { this.toolbar = scope; }}
        clicked={this.clickHandler.bind(this)}>
          <ItemsDirective>
            <ItemDirective prefixIcon='e-pv-open-document-icon' id='file_Open'
            tooltipText='Open'></ItemDirective>
            <ItemDirective prefixIcon="e-pv-previous-page-navigation-icon"
            id='previous_page' tooltipText="Previous Page"
            align="Center"></ItemDirective>
            <ItemDirective prefixIcon="e-pv-next-page-navigation-icon" id='next_page'
            tooltipText="Next Page" align="Center"></ItemDirective>
            <ItemDirective template={inputTemplate} tooltipText="Page Number"
            type="Input" align="Center"></ItemDirective>
            <ItemDirective template={template} align="Center" tooltipText="Page
            Number"></ItemDirective>
            <ItemDirective prefixIcon="e-pv-print-document-icon" tooltipText="Print"
            id='print' align="Right"></ItemDirective>
            <ItemDirective prefixIcon="e-pv-download-document-icon"
            tooltipText="Download" id='download' align="Right"></ItemDirective>
          </ItemsDirective>
        </ToolbarComponent>
      </div>
      { /* Render the PDF Viewer */ }
      <PdfViewerComponent
      id="container"
      ref={(scope) => { this.viewer = scope; }}
      enableToolbar={false}
      documentLoad={this.documentLoaded}
      pageChange={this.onPageChange}
      documentPath="https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf"
      serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
      style={{ 'display': 'block', 'height': '640px' }}>
        <Inject services={[ Magnification, Navigation, LinkAnnotation, BookmarkView,
        ThumbnailView, Print, TextSelection, TextSearch]} />
      </PdfViewerComponent>
      <input type="file"
      id="fileUpload"
      accept=".pdf"
      onChange={this.readFile.bind(this)}
      style={{ 'display': 'block', 'visibility': 'hidden', 'width': '0', 'height':
      '0' }} />
      <div className='e-pdf-toolbar' id="magnificationToolbarItems">
        <ToolbarComponent id="magnificationToolbar"
        clicked={this.clickHandler.bind(this)}>
          <ItemsDirective >

```

```

<ItemDirective prefixIcon="e-pv-fit-page" id='fit_to_page' tooltipText="Fit
to page" ></ItemDirective>
<ItemDirective prefixIcon="e-pv-zoom-in-icon" id='zoom_in' tooltipText="Zoom
in"></ItemDirective>
<ItemDirective prefixIcon="e-pv-zoom-out-sample" id='zoom_out'
tooltipText="Zoom out" ></ItemDirective>
</ItemsDirective>
</ToolbarComponent>
</div>
</div>
</div>
</div>
);
}
{% endraw %}

```

Step 3: Import and inject the modules used for the custom toolbar,

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import {
```

```
PdfViewerComponent, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
```

```
ThumbnailView, Print, TextSelection, TextSearch, Inject
```

```
} from '@syncfusion/ej2-react-pdfviewer';
```

```
import { ToolbarComponent, ItemsDirective, ItemDirective, ClickEventArgs } from '@syncfusion/ej2-
react-navigations';
```

```
import { RouteComponentProps } from 'react-router';
```

```
,
```

Step 4: Hide the default toolbar of PDF Viewer using below code snippet,

STANDALONE

```

{% raw %}
<PdfViewerComponent
id="container"
ref={(scope) => { this.viewer = scope; }}
enableToolbar={false}
documentLoad={this.documentLoaded}
pageChange={this.onPageChange}
documentPath="https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'display': 'block', 'height': '640px' }}>
<Inject services={[ Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
{% endraw %}

```

SERVER-BACKED

```
{% raw %}
<PdfViewerComponent
id="container"
ref={ (scope) => { this.viewer = scope; }}
enableToolbar={false}
documentLoad={this.documentLoaded}
pageChange={this.onPageChange}
documentPath="https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'display': 'block', 'height': '640px' }}>
<Inject services={[ Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
{% endraw %}
```

Step 5: Add the following style to achieve the custom toolbar styling,

```
magnificationToolbarItems {
position: absolute;
bottom: 66px;
display: block;
width: auto;
transform: rotate(90deg);
right: 7.5px;
z-index: 1001;
}

magnificationToolbar {
background: transparent;
}

.e-pv-zoom-out-sample {
transform: rotate(-90deg);
}

div#magnificationToolbar.e-toolbar .e-toolbar-items {
background: transparent;
padding: 2px 3px 2px 2px;
}

magnificationToolbar.e-toolbar .e-tbar-btn {
border-radius: 50%;
min-height: 30px;
```

```
min-width: 30px;
border: 0.5px solid #c8c8c8;
}

topToolbar {
top: 0px;
z-index: 1001;
}

.e-pv-current-page-number {
width: 46px;
height: 28px;
text-align: center;
}

.material .e-pv-current-page-number {
border-width: 1px;
}

.e-pv-icons {
font-family: "e-pv-icons";
font-style: normal;
font-variant: normal;
font-weight: normal;
line-height: 1;
text-transform: none;
}

.e-pdf-toolbar .e-icons::before {
font-family: 'e-pv-icons';
}

.e-pv-icon-search::before {
font-family: 'e-pv-icons';
font-size: 12px;
}

topToolbar .e-pv-download-document-icon::before {
padding-left: 4px;
content: '\ed05';
}
```

```

topToolbar .e-pv-print-document-icon::before {
padding-left: 1px;
content: '\ed08';
}

.e-pv-previous-page-navigation-icon::before {
content: '\ed01';
}

.e-pv-next-page-navigation-icon::before {
content: '\ed02';
}

.e-pv-zoom-out-sample::before {
content: '\ed03';
}

.e-pv-zoom-in-icon::before {
content: '\ed04';
}

.e-pv-fit-page::before {
content: '\ed12';
}

.e-pv-open-document-icon::before {
content: '\ed13';
}

@font-face {
font-family: "e-pv-icons";
font-style: normal;
font-weight: normal;
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgT1MvMkMhTzoAAAEoAAAAVmNtYXDae9qvAAABuAAAAFZnbHlmok0Nt
wAAAjAAAAPkaGVhZBN5FAcAAADQAAAAANmhoZWEHrwNhAAAArAAAACRobXR4NsgAAAAAYAAAAA4b
G9jYQdkBmQAAAIQAAAAHm1heHABHAAwAAABCAAAACBuYW1lsXg1swAABhQAAAJ5cG9zdFG4mE4AA
AiQAAAAyAABAAADUv9qAFoEAAAA/+gEAAABAAAAAAAAAAAAAAAAAADgABAAAAAQAAaoJDif8PPP
UACwPoAAAAANGtZ5EAAAAA2C1nkQAAAAAEAAQAAAAACAACAAAAAAAAAAAAEAAAAOACQABAAAAAAA
gAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA7QHtEwNS/2oAWgQAAJYAAAAABAAAAAAAAABAAAAAPoAAAD
6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAAAACAAAAAwA
AABQAAwABAAAFAAEAEIAAAAGAAQAAQAC7QntE///AADtAe0Q//8AAAAAAAAEABgAWAAAAAQACAAM

```



```

ABAAFAAYABwAIAAkACgALAAwADQAAAAAAAAAAUACoAZACkAL4A7gEuAVwBcAGEAZ4ByAHyAAAAAQA
AAAAD6gMuAAUAAkBBwkBJwIAAet0/on+iXQDL/4VcwF3/olzAAEAAAAAAAA+oDLgAFAAATCQEXCQGJAXc
Bd3T+FF4VAY/+iQF3c/4VAesAAAAAAwAAAAAEAAQAAAMADwAbAAAABITUhBQ4BBy4BJz4BNx4BBRYAFzY
AnYyAJwYAAQACAP4AAoAE2aOj2QQE2aOj2fyEBgEh2dkBIQYG/t/Z2f7fAcCAQKPZBATZo6PZBATZo9n+3w
YGASHZ2QEhBgb+3wAAAAADAAAAAAQABAAACwAXACMAAAEjFTMVMzUzNSM1lwEOAQcuASc+ATceA
QUWABc2ADcmACcGAAHAWMCAwMCAAcAE2aOj2QQE2aOj2fyEBgEh2dkBIQYG/t/Z2f7fAkCAwMCAwP8
Ao9kEBNmjo9kEBNmj2f7fBgYBlDnZASEGBv7fAAIAAAAAAAwAEAAADAAoAADEhNSEBIQkBIREhAwD9AAEA
/wABgAGA/wD/AIACAP6AAYABgAACAAAAANABAAADgAaAAABMh4CFREIBREONz4BMycGFREIBRE0JiM
hlgKdCwwHBf7g/uAJBAwKdC8BoAGgX0T+BkQDgAYGCwr9YH2ZAqAOCQQGUS9D/KGrqwNfRIsAAAACAA
AAAAP/BAAACwAjAAABDgEHLgEnPgE3HgEFHgEXMjY/ARcVATcBlYc3PgE1LgEnDgECgAOQbW2QAwoQb
W2Q/YME2aNGfDIDJAEYf78MyMCKi4E2aOj2QKAbZADA5BtbZADA5Bto9kELioDJDp+/GEBBCQDMnxGo9
kEBNkAAAQAAAAABAAEAAADAACAFQAZAAABFSE1JRujNSERMxUhNTMRLgEnIQ4BNyE1IQLA/oACQID9A
MACgMABSDf9ADdlvwKA/YABwMDAwICA/sDAwAFAN0gBAUmKwAAAAQAAAAACQAQAAAUABEBNwk
BJwHsU/6HAXpSAmD+YGIBPgE+YgAAAAEAAAAAAkAEAAFAAAARCQEXCQEBEv6HUwHs/hMDnv7C/sJiAa
ABoAABAAAAAAKABAAACwAAERcHFzcXNyc3Jwcn9fVM9PVL9PRL9fQDtfX0TPX1TPT0TPT0AAAAABAAAA
AD8APwAAUACwARABcAAcEzNTM1IQUzFTMRISUhNSM1lwUjFSElWj2fvz+hv2K/H7+hgJ2AXr8fv6G/AF6
fvx+fvwBevx+/Px+AXoAAAAAAgAAAAEAAQAAAMAFgAAAREhEScGFREUFhchPgE1ETQmlyEnIQYDgP0AY
h48LQMULtW8Lf5pa/7ULQMA/gACAN8eLf1YLTwDAzwtAigvPYACAAAAAASAN4AAQAAAAAABAAAA
AAQAAAAAAQAQAAEAAQAAAAAAAgAHAA8AAQAAAAAAAwAOABYAAQAAAAAABAAOACQAQAAAAAA
ABQALADIAAQAAAAAABgAOAD0AAQAAAAAACgAsAEsAAQAAAAAACwASAHcAAwABBakAAAAACAlkAAw
ABBakAAQAcAlSAAwABBakAAgAOAKcAAwABBakAAwAcALUAAwABBakABAACANEAAwABBakABQAWA
O0AAwABBakABgAcAQMAAwABBakACgBYAR8AAwABBakACwAkAXcgY3VzdG9tLXRvb2xiYXJSZWd1bGfy
Y3VzdG9tLXRvb2xiYXJjdXN0b20tdG9vbGJhclZlcnNpb24gMS4wY3VzdG9tLXRvb2xiYXJGb250IGdlbmVyYXR
lZCB1c2luZyBTeW5jZnVzaW9uE1ldHjVlFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGMAdQBzAHQA
bwBtAC0AdABvAG8AbABiAGEAcgBSAGUAZwB1AGwAYQByAGMAdQBzAHQAbwBtAC0AdABvAG8AbABiA
GEAcgBjAHUAcwB0AG8AbQAtAHQAbwBvAGwAYgBhAHIAVgBIAHIAcwBpAG8AbgAgADEALgAwAGMAdQ
BzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBGAG8AbgB0ACAAZwBIAg4AZQByAGEAdABIAGQAIAB1AHMA
aQBuaGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3A
HcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAAaKAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAOAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDwAIVG9wLWljY24LZG93bi1hcnJ
vdzIKUFZfWm9vbW91dAlQVI9ab29taW4LUFZfRG93bmxxvYwQLUFZfQm9va21hcmsJUFZfU2VhcmNoCFB
WX1ByaW50C1BWx1ByZXZpb3VzB1BWx05leHQlUFZfQ2xvc2UMUFZfRml0VG9QYWdlB1BWx09wZW4A
AA==) format('trueype');
}

```

The icons are embedded in the font file used in above code snippet.

Step 6: Add the following scripts for performing user interaction in PDF Viewer in code behind

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
PdfViewerComponent, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Inject

```

```
} from '@syncfusion/ej2-react-pdfviewer';

import { ToolbarComponent, ItemsDirective, ItemDirective, ClickEventArgs } from '@syncfusion/ej2-react-navigations';

import { RouteComponentProps } from 'react-router';

export class CustomToolbar extends SampleBase<{}, {}> {

  public viewer: PdfViewerComponent;

  public toolbar: ToolbarComponent;

  public currentPageNumber: string = '1';

  public fileName: string = '';

  rendereComplete() {
    this.wireEvent();
  }

  render() {
    // template code from step 2
  }

  wireEvent() {

    let inputElement: HTMLInputElement = document.getElementById('currentPage') as HTMLInputElement;

    inputElement.addEventListener('click', this.currentPageClicked.bind(this));
    inputElement.addEventListener('keypress', this.onCurrentPageBoxKeyPress.bind(this));
    inputElement.value = this.currentPageNumber;
  }

  onPageChange = () => {
    this.currentPageNumber = this.viewer.currentPageNumber.toString();

    let inputElement: HTMLInputElement = document.getElementById('currentPage') as HTMLInputElement;
    inputElement.value = this.currentPageNumber;
    this.updatePageNavigation();
  }

  clickHandler(args: ClickEventArgs) {
    switch (args.item.id) {
      case 'file_Open':
        document.getElementById('fileUpload').click();
        break;
    }
  }
}
```

```
case 'previous_page':
this.viewer.navigation.goToPreviousPage();
break;
case 'next_page':
this.viewer.navigation.goToNextPage();
break;
case 'print':
this.viewer.print.print();
break;
case 'download':
this.viewer.download();
break;
case 'fittopage':
this.viewer.magnification.fitToPage();
break;
case 'zoom_in':
this.viewer.magnification.zoomIn();
break;
case 'zoom_out':
this.viewer.magnification.zoomOut();
break;
}
}
documentLoaded = () => {
var pageCount = document.getElementById('totalPage');
pageCount.textContent = 'of ' + this.viewer.pageCount;
this.updatePageNavigation();
}
updatePageNavigation() {
if (this.viewer.currentPageNumber === 1) {
this.toolbar.enableItems(document.getElementById('previous_page').parentElement, false);
this.toolbar.enableItems(document.getElementById('next_page').parentElement, true);
} else if (this.viewer.currentPageNumber === this.viewer.pageCount) {
```

```
this.toolbar.enableItems(document.getElementById('previous_page').parentElement, true);
this.toolbar.enableItems(document.getElementById('next_page').parentElement, false);
} else {
this.toolbar.enableItems(document.getElementById('previous_page').parentElement, true);
this.toolbar.enableItems(document.getElementById('next_page').parentElement, true);
}
}

onCurrentPageBoxKeypress(event) {
let currentPageBox: HTMLInputElement = document.getElementById('currentPage') as HTMLInputElement;
if ((event.which < 48 || event.which > 57) && event.which !== 8 && event.which !== 13) {
event.preventDefault();
return false;
}
else {
var currentPageNumber = parseInt(currentPageBox.value);
if (event.which === 13) {
if (currentPageNumber > 0 && currentPageNumber <= this.viewer.pageCount) {
this.viewer.navigation.goToPage(currentPageNumber);
}
else {
currentPageBox.value = this.viewer.currentPageNumber.toString();
}
}
return true;
}
}

currentPageClicked() {
let currentPage: HTMLInputElement = document.getElementById('currentPage') as HTMLInputElement;
currentPage.select();
}

readFile(evt) {
let uploadedFiles = evt.target.files;
```

```

let uploadedFile = uploadedFiles[0];
this.fileName = uploadedFile.name;
let reader = new FileReader();
reader.readAsDataURL(uploadedFile);
let viewer: PdfViewerComponent = this.viewer;
let uploadedFileName: string = this.fileName;
reader.onload = function () {
let uploadedFileUrl: string = this.result as string;
viewer.load(uploadedFileUrl, null);
viewer.fileName = uploadedFileName;
var pageCount = document.getElementById('totalPage');
pageCount.textContent = 'of ' + viewer.pageCount;
}
}
}
,

```

Sample :

<https://ej2.syncfusion.com/react/demos/#/material/pdfviewer/custom-toolbar>

Load document in React Pdfviewer component

The PDF Viewer library allows to switch or load PDF documents dynamically after the initial load operation. To achieve this, load the PDF document as a base64 string or the file name into the PDF Viewer control using the [Load\(\)](#) method dynamically.

The following steps are used to load the PDF document dynamically.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to load the PDF document using a base64 string.

```

<button id='load1'>LoadDocumentFromBase64</button>
<script>
// Load PDF document from Base64 string
function load_1(){
var viewer = document.getElementById('container').ej2_instances[0];
viewer.load('data:application/pdf;base64,'+ AddBase64String, null);
}
</script>

```

Step 3: Use the following code snippet to load the PDF document using the document name.

```
<button id='load2'>LoadDocument</button>

<script>
// Load PDF document using file name
function load_2(){
var viewer = document.getElementById('container').ej2_instances[0];
viewer.load('https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf', null);
}
</script>
```

Find the sample [how to load PDF documents dynamically](#)

Step 4: Use the following code snippet to load the PDF document using the documentPath.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
return (<div>
<div className='control-section'>
<PdfViewerComponent ref={ (scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch ]}/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, Inject } from '@syncfusion/ej2-
react-pdfviewer';
let pdfviewer;
function App() {
return (<div>
<div className='control-section'>
<PdfViewerComponent ref={(scope) => { pdfviewer = scope; }}
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Annotation, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch ]}/>
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

Find the sample [how to load PDF documents using documentPath](#)

Unload document in React Pdfviewer component

The PDF Viewer library allows you to unload the PDF document being displayed in the PDF Viewer control programmatically using the [unload\(\)](#) method.

The following steps are used to unload the PDF document programmatically.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Add the following code snippet to perform the unload operation.

```
<button onclick="unload()">Unload Document</button>
<script>
function unload(){
var viewer = document.getElementById('container').ej2_instances[0];
// Unload the PDF document.
viewer.unload();
}
</script>
```

Find the sample [how to unload the PDF document programmatically](#)

Lock annotation in React Pdfviewer component

The PDF Viewer library allows you to lock the rectangle or square annotations using the **isLock** property in the **rectangleSettings**.

Step 1: Follow the steps provided in the [link](#) to create simple PDF Viewer sample in React.

Step 2: Add the following code snippet to lock the rectangle or square annotations.

STANDALONE

```
{% raw %}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
// lock annotation
rectangleSettings={{ isLock: true }}
style={{ height: '640px' }}>
</PdfViewerComponent>
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
// lock annotation
rectangleSettings={{ isLock: true }}
style={{ height: '640px' }}>
</PdfViewerComponent>
{% endraw %}
```

Find the Sample [how to lock square or rectangle annotations](#)

Add signature in React Pdfviewer component

The PDF Viewer library allows you to add signature in the signature field of the loaded PDF document programmatically using the **formFieldClick** event.

Step 1: Follow the steps provided in the [link](#) to create simple PDF Viewer sample in React.

Step 2: Add the following code snippet to add signature in signature field.

STANDALONE

```
{% raw %}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/form-filling-
document.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ height: '640px' }}
formFieldClick={fieldClick}>
{% endraw %}
```



```
<Inject services=[ [ Toolbar, Magnification, Navigation, Annotation,  
LinkAnnotation, BookmarkView,  
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields ] ]  
>  
</PdfViewerComponent>  
function fieldClick(args) {  
var viewer = document.getElementById('container').ej2_instances[0];  
if (viewer) {  
args.cancel = true;  
if (args.field.type === 'SignatureField') {  
var forms = viewer.formFieldCollections;  
forms.map((r) => {  
if (r.id === args.field.id) {  
console.log(args.field.value);  
var el = document.getElementById(r.id);  
if (el) {  
if (el.style.textAlign !== 'center') {  
el.style.textAlign = 'center';  
}  
if (el.style.fontStyle !== 'italic') {  
el.style.fontStyle = 'italic';  
}  
if (el.style.fontWeight !== 'normal') {  
el.style.fontWeight = 'normal';  
}  
if (args.field.value !== '' && args.field.value) {  
args.field.value = '';  
viewer.updateFormFieldsValue(args.field);  
} else {  
args.field.signatureType = ['Type'];  
args.field.value = 'DA FIRMARE';  
args.cancel = true;  
viewer.updateFormFieldsValue(args.field);  
}  
}  
});  
}  
}  
}  
}{% enddraw %}
```

SERVER-BACKED

```
{% raw %}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/form-filling-
document.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ height: '640px' }}
formFieldClick={fieldClick}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields ]}
/>
```

```
</PdfViewerComponent>
function fieldClick(args) {
var viewer = document.getElementById('container').ej2_instances[0];
if (viewer) {
args.cancel = true;
if (args.field.type === 'SignatureField') {
var forms = viewer.formFieldCollections;
forms.map((r) => {
if (r.id === args.field.id) {
console.log(args.field.value);
var el = document.getElementById(r.id);
if (el) {
if (el.style.textAlign !== 'center') {
el.style.textAlign = 'center';
}
if (el.style.fontStyle !== 'italic') {
el.style.fontStyle = 'italic';
}
if (el.style.fontWeight !== 'italic') {
el.style.fontWeight = 'italic';
}
if (args.field.value !== '' && args.field.value) {
args.field.value = '';
viewer.updateFormFieldsValue(args.field);
} else {
args.field.signatureType = ['Type'];
args.field.value = 'DA FIRMARE';
args.cancel = true;
viewer.updateFormFieldsValue(args.field);
}
}
});
}
}
}
}{% enddraw %}
```

Find the Sample [how to add signature in signature field](#)

Extract text in React Pdfviewer component

The PDF Viewer library allows you to extract the text from a page along with the bounds. Text extraction can be done using the `isExtractText` property and `extractTextCompleted` event.

Here is an example of how you can use the `isExtractText` property and `extractTextCompleted` event:

```
{% raw %}
```

```
`javascript
```

<PdfViewerComponent

```
id="container"
```

```
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
```

```
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
```

```

isExtractText={true}
extractTextCompleted={extractTextCompleted}
style={{ height: '640px' }}>
</PdfViewerComponent>

function extractTextCompleted(args){
// Extract the Complete text of load document
console.log(args);
console.log(args.documentTextCollection[1]);
// Extract the Text data.
console.log(args.documentTextCollection[1][1].TextData);
// Extract Text in the Page.
console.log(args.documentTextCollection[1][1].PageText);
// Extract Text along with Bounds
console.log(args.documentTextCollection[1][1].TextData[0].Bounds);
};
`

```

```
{% endraw %}
```

Find the sample [how to Extract Text](#)

Import export annotation object in React Pdfviewer component

The PDF Viewer library allows you to import annotations from objects or streams instead of loading it as a file. To import such annotation objects, the PDF Viewer control must export the PDF annotations as objects using the [ExportAnnotationsAsObject\(\)](#) method. Only the annotations objects that are exported from the PDF Viewer can be imported.

The following steps are used to import and export annotation as object.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to perform import and export annotation.

```

<button onclick="exportAnnotation()">Export Annotation</button>
<button onclick="importAnnotation()">Import Annotation</button>
<script>
var exportObject;
// Export annotation as object.
function exportAnnotation(){
var viewer = document.getElementById('container').ej2_instances[0];

```

```
viewer.exportAnnotationsAsObject().then(function(value) {
exportObject = value;
});
}

// Import annotation that are exported as object.
function importAnnotation() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.importAnnotation(JSON.parse(exportObject));
}
</script>
,
```

Find the sample [how to import and export annotation as object](#)

Import and export annotations programmatically

The PDF Viewer library allows you to import annotations via code behind by using the [importAnnotation\(\)](#) method.

The following steps are used to import and export annotation programmatically.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom/client';
import * as React from 'react';
import './index.css'
import {
PdfViewerComponent,
Toolbar,
Magnification,
Navigation,
LinkAnnotation,
BookmarkView,
ThumbnailView,
Print,
TextSelection,
TextSearch,
Annotation,
FormFields,
Inject,
} from '@syncfusion/ej2-react-pdfviewer';
function App() {
return (
<div>
<button onclick="exportAnnotation()">Export Annotation</button>
<button onclick="importAnnotation()">Import Annotation</button>
<div className="control-section">
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
```

```

documentLoad={documentLoad}
style={{ height: '640px' }}>
<Inject
services={ [
  Toolbar,
  Magnification,
  Navigation,
  LinkAnnotation,
  BookmarkView,
  ThumbnailView,
  Print,
  TextSelection,
  TextSearch,
  Annotation,
  FormFields,
] }
/>
</PdfViewerComponent>
</div>
</div>
);
}
function documentLoad(event) {
var viewer = document.getElementById('container').ej2_instances[0];
//API to add annotation programmatically for initial loading.
viewer.importAnnotation({
pdfAnnotation: {
0: {
shapeAnnotation: [
{
ShapeAnnotationType: 'Square',
Author: 'Guest',
AnnotationSelectorSettings: {
selectionBorderColor: '',
resizerBorderColor: 'black',
resizerFillColor: '#FF4081',
resizerSize: 8,
selectionBorderThickness: 1,
resizerShape: 'Square',
selectorLineDashArray: [],
resizerLocation: 3,
resizerCursorType: null,
},
ModifiedDate: '4/22/2021, 10:33:04 AM',
Subject: 'Rectangle',
Note: '',
IsCommentLock: false,
StrokeColor: 'rgba(255,0,0,1)',
FillColor: 'rgba(255,255,255,0)',
Opacity: 1,
Bounds: {
X: 124,
Y: 76,
Width: 202,
Height: 154,
Location: { X: 124, Y: 76 },
Size: { IsEmpty: false, Width: 202, Height: 154 },

```

```

Left: 124,
Top: 76,
Right: 326,
Bottom: 230,
},
Thickness: 2,
BorderStyle: 'Solid',
BorderDashArray: 0,
RotateAngle: 'RotateAngle0',
IsCloudShape: false,
CloudIntensity: 0,
RectangleDifference: null,
VertexPoints: null,
LineHeadStart: null,
LineHeadEnd: null,
IsLocked: false,
AnnotName: 'e9a14dbe-5d09-4226-329e-c6edab201284',
Comments: null,
State: '',
StateModel: '',
AnnotType: 'shape',
EnableShapeLabel: false,
LabelContent: null,
LabelFillColor: null,
LabelBorderColor: null,
FontColor: null,
FontSize: 0,
CustomData: null,
LabelBounds: {
X: 0,
Y: 0,
Width: 0,
Height: 0,
Location: { X: 0, Y: 0 },
Size: { IsEmpty: true, Width: 0, Height: 0 },
Left: 0,
Top: 0,
Right: 0,
Bottom: 0,
},
LabelSettings: null,
AnnotationSettings: {
minWidth: 0,
maxWidth: 0,
minHeight: 0,
maxHeight: 0,
isLock: false,
isPrint: true,
},
AllowedInteractions: ['None'],
IsPrint: true,
ExistingCustomData: null,
},
],
},
},
});

```

```

}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom/client';
import * as React from 'react';
import './index.css'
import {
PdfViewerComponent,
Toolbar,
Magnification,
Navigation,
LinkAnnotation,
BookmarkView,
ThumbnailView,
Print,
TextSelection,
TextSearch,
Annotation,
FormFields,
Inject,
} from '@syncfusion/ej2-react-pdfviewer';
function App() {
return (
<div>
<button onclick="exportAnnotation()">Export Annotation</button>
<button onclick="importAnnotation()">Import Annotation</button>
<div className="control-section">
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
documentLoad={documentLoad}
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ height: '640px' }}>
<Inject
services={[
Toolbar,
Magnification,
Navigation,
LinkAnnotation,
BookmarkView,
ThumbnailView,
Print,
TextSelection,
TextSearch,
Annotation,
FormFields,
]}
/>
</PdfViewerComponent>
</div>
</div>

```

```

);
}
function documentLoad(event) {
var viewer = document.getElementById('container').ej2_instances[0];
//API to add annotation programmatically for initial loading.
viewer.importAnnotation({
pdfAnnotation: {
0: {
shapeAnnotation: [
{
ShapeAnnotationType: 'Square',
Author: 'Guest',
AnnotationSelectorSettings: {
selectionBorderColor: '',
resizerBorderColor: 'black',
resizerFillColor: '#FF4081',
resizerSize: 8,
selectionBorderThickness: 1,
resizerShape: 'Square',
selectorLineDashArray: [],
resizerLocation: 3,
resizerCursorType: null,
},
ModifiedDate: '4/22/2021, 10:33:04 AM',
Subject: 'Rectangle',
Note: '',
IsCommentLock: false,
StrokeColor: 'rgba(255,0,0,1)',
FillColor: 'rgba(255,255,255,0)',
Opacity: 1,
Bounds: {
X: 124,
Y: 76,
Width: 202,
Height: 154,
Location: { X: 124, Y: 76 },
Size: { IsEmpty: false, Width: 202, Height: 154 },
Left: 124,
Top: 76,
Right: 326,
Bottom: 230,
},
Thickness: 2,
BorderStyle: 'Solid',
BorderDashArray: 0,
RotateAngle: 'RotateAngle0',
IsCloudShape: false,
CloudIntensity: 0,
RectangleDifference: null,
VertexPoints: null,
LineHeadStart: null,
LineHeadEnd: null,
IsLocked: false,
AnnotName: 'e9a14dbe-5d09-4226-329e-c6edab201284',
Comments: null,
State: '',
StateModel: '',

```



```

AnnotType: 'shape',
EnableShapeLabel: false,
LabelContent: null,
LabelFillColor: null,
LabelBorderColor: null,
FontColor: null,
FontSize: 0,
CustomData: null,
LabelBounds: {
  X: 0,
  Y: 0,
  Width: 0,
  Height: 0,
  Location: { X: 0, Y: 0 },
  Size: { IsEmpty: true, Width: 0, Height: 0 },
  Left: 0,
  Top: 0,
  Right: 0,
  Bottom: 0,
},
LabelSettings: null,
AnnotationSettings: {
  minWidth: 0,
  maxWidth: 0,
  minHeight: 0,
  maxHeight: 0,
  isLock: false,
  isPrint: true,
},
AllowedInteractions: ['None'],
IsPrint: true,
ExistingCustomData: null,
},
],
},
},
});
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% enddraw %}

```

The following code example represents how to export annotation in a button click.

,

```
<button onclick="exportAnnotation()">ExportAnnotation</button>
```

```
<script>
```

```
//Event triggers when you click the ExportAnnotation button.
```

```
function exportAnnotation() {
```

```
var viewer = document.getElementById('container').ej2_instances[0];
```

```
//API to export annotation.
```

```
viewer.exportAnnotation();
}
</script>
`
```

[View sample in GitHub.](#)

Authorization token in React Pdfviewer component

The PDF Viewer library allows you to include the authorization token in the PDF viewer AJAX request using the properties of the ajaxRequest header available in `AjaxRequestSettings`, and it will be included in every AJAX request send from PDF Viewer.

Here is an example of how you can use the **AjaxRequestSettings** property to include the authorization token to the PDF viewer control:

```
{% raw %}
`javascript
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ height: '640px' }}
ajaxRequestSettings={{
  ajaxHeaders: [
    {
      headerName: 'Authorization',
      headerValue: 'Bearerabcdefghijklmnopqrstuvwxyz',
    },
  ],
  withCredentials: false,
}}>
<Inject services={[ Toolbar, Magnification, Navigation, Annotation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormDesigner, FormFields ]} />
</PdfViewerComponent>
`

{% endraw %}
```

Find the sample [how to include authorization token](#)

Delete annotation in React Pdfviewer component

The PDF Viewer library allows you to delete a specific annotation from a PDF document. Deleting a specific annotation can be done using the **deleteAnnotationById()** method. This method is used to delete a specific annotation using its id.

The following steps are used to delete a specific annotation from PDF Document.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to delete a specific annotation using `deleteAnnotationById()` method.

```
,
<button onclick="deleteAnnotationById()">Delete Annotation by ID</button>
<script>
// Delete Annotation by id.
function deleteAnnotationById() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.annotationModule.deleteAnnotationById(viewer.annotationCollection[0].annotationId);
}
</script>
,
```

Find the sample [how to delete a specific annotation using deleteAnnotationById](#)

Open thumbnail in React Pdfviewer component

The PDF Viewer library allows you to open the thumbnail pane programmatically using the [openThumbnailPane\(\)](#) method.

The following steps are used to open the thumbnail.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to open thumbnail.

```
,
<button onclick="openThumbnail()">Open Thumbnail Pane</button>
<script>
function openThumbnail() {
var viewer = document.getElementById('container').ej2_instances[0];
// Open Thumbnail pane
viewer.thumbnailViewModule.openThumbnailPane();
}
</script>
,
```

Find the sample [how to open the thumbnail pane programmatically](#)

Convert pixel to point in server side in React Pdfviewer component

Syncfusion PDF viewer will get the bounds of the annotations as the pixel in the Client-side. But while using it in the back end, we need to convert the pixel into point by using the below calculation. And the 0.75 is constant for all the calculations in the back end.

`ts

$X = x * 72 / 96$

$Y = y * 72 / 96$

$Width = width * 72 / 96$

$Height = height * 72 / 96$

,

Retrieve id of the document in React Pdfviewer component

View the PDF document's id by passing the `hashId` value to the `getItem()` method of session storage API.

Refer to the following code to get the id of a PDF document in a button click.

,

```
<button onclick="uniqueId()">Uniqueid</button>
```

```
<script>
```

```
//Event triggers when you click the Uniqueid button.
```

```
function uniqueId() {
```

```
//Prints the PDF document id in the console window.
```

```
console.log(window.sessionStorage.getItem("hashId"));
```

```
}
```

```
</script>
```

,

[View sample in GitHub.](#)

Perform form field double click event in React Pdfviewer component

The PDF Viewer library allows you to trigger an event when you double click on the form field using the `formFieldDoubleClick` event.

Step 1: Follow the steps provided in the [link](#) to create simple PDF Viewer sample in React.

Step 2: Add the following code snippet in the `index.js` file to add the form field double click event in the PDF Viewer.

STANDALONE

```
{% raw %}
<PdfViewerComponent
id="container"
```

```
documentPath="https://cdn.syncfusion.com/content/pdf/form-filling-
document.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
formFieldDoubleClick={formFieldDoubleClick}
style={{ height: '640px' }}>
</PdfViewerComponent>
function formFieldDoubleClick(args) {
  //Prints the argument value in the console window
  console.log(args);
}
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/form-filling-
document.pdf"
formFieldDoubleClick={formFieldDoubleClick}
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ height: '640px' }}>
</PdfViewerComponent>
function formFieldDoubleClick(args) {
  //Prints the argument value in the console window
  console.log(args);
}
{% endraw %}
```

Clear annotation in React Pdfviewer component

To clear all the annotations in a PDF document using the Syncfusion PDF Viewer, you can use the **deleteAnnotations()** method. This method is used to clear all the annotations present in the currently loaded document.

Here is an example of how you can clear all the annotations present in the currently loaded document:

```
`html
<button onclick="deleteAnnotations()">Delete Annotations</button>

<script>
// clear Annotations.
function deleteAnnotations() {
  var viewer = document.getElementById("container").ej2_instances[0];
  viewer.deleteAnnotations();
}
</script>
`
```

We can also delete specific annotation with the **deleteAnnotationById()** method. This method is used to delete a specific annotation using its id.

Here is an example of how you can delete specific annotation with the **deleteAnnotationById()** method:

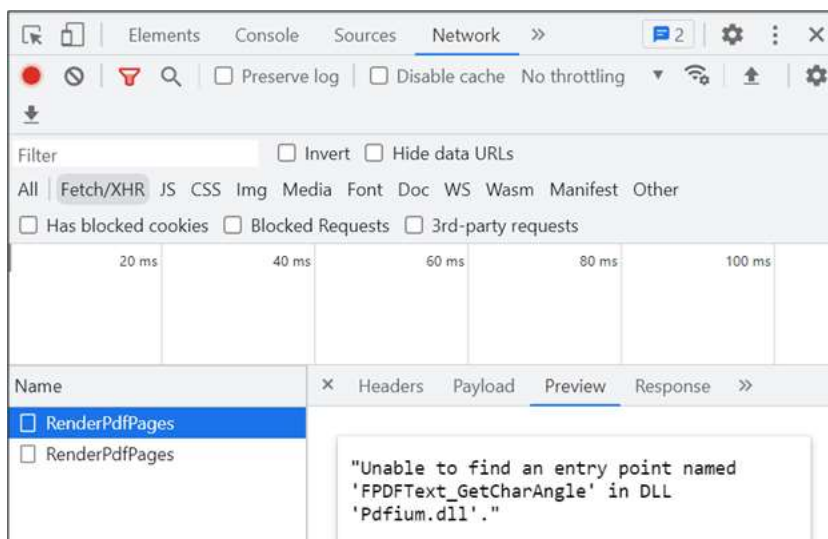
```
`html
<button onclick="deleteAnnotationById()">Delete Annotation by ID</button>
<script>
// Delete Annotation by ID.
function deleteAnnotationById() {
var viewer = document.getElementById("container").ej2_instances[0];
viewer.annotationModule.deleteAnnotationById(viewer.annotationCollection[0].annotationId);
}
</script>
`
```

Find the sample [how to clear annotations using deleteAnnotations](#)

Resolve "Unable to find an entry point named FPDFText_GetCharAngle" error

From the release of version **21.1.0.35 (2023 Volume 1)** of Essential Studio, the Pdfium package has been upgraded to improve various functionalities like text search, text selection, rendering, and even performance. If you are updating your project to this version of the Syncfusion PDF Viewer, you may encounter the **"Web-Service is not listening"** error. The Network tab can help you identify the root cause of the issue, which is typically caused by an old version of pdfium assembly being referenced in the local web service project. Below are the assemblies to be referred to in the respective operating systems.

- Windows – pdfium.dll
- Linux – libpdfium.so
- OSX – libpdfium.dylib



To solve this issue, you should follow the below steps:

1. Clear the bin and object files of the web-service project.
2. Re-publish the web-service project.

Note: Note: If you are hosting your application in Azure, AWS, or in Linux environments, delete the older published files and republish the application.

How to clear the "Web-service is not listening" to error

PDF Viewer

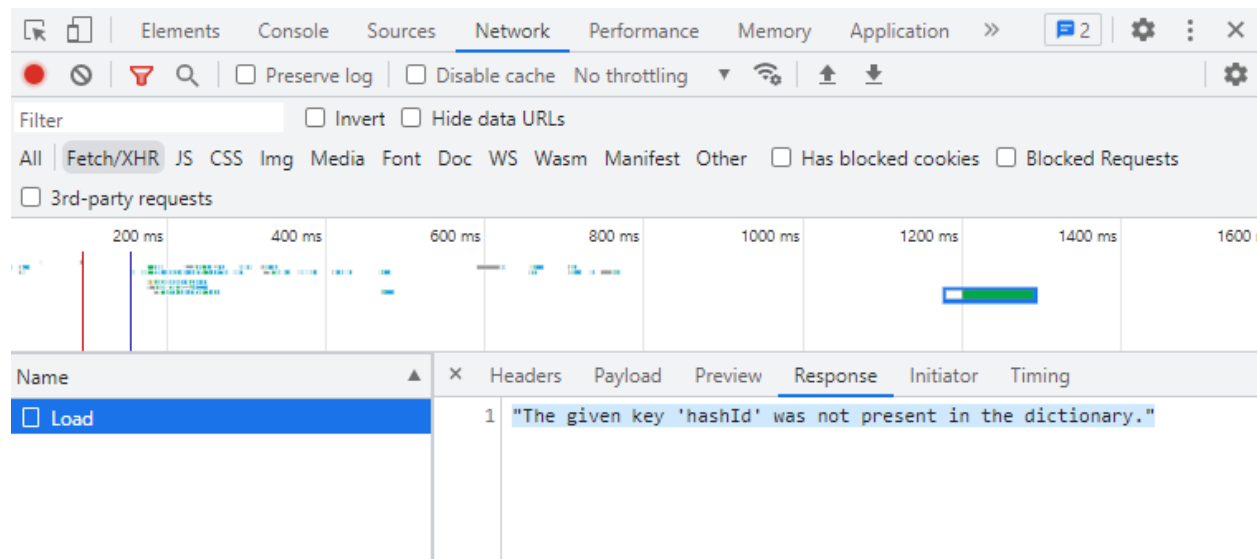


Web-service is not listening. PDF Viewer depends on web-service for all it's features. Please start the web service to continue.

OK

If you are facing a **Web-service is not listening** to error in the Syncfusion PDF Viewer, there could be several reasons for this. To troubleshoot the issue, you can use the Network tab in your browser's developer tools to gather more information. Here are the steps you can follow:

Step 1: Open the browser's developer tools by right-clicking on the page and selecting **Inspect** from the dropdown menu. Then Navigate to the **Network** tab. This will show you all of the requests that are being made by the page.



Step 2: Try to request the web service. If the service is not listening, the request will fail, and you should see an error message in the Network tab. Click on the failing request to see the details of the error, such as the error message or stack trace. This can help you identify the root cause of the issue. Check the server logs for any errors or warnings that may indicate the cause of the issue and help you to troubleshoot the problem.

Step 3: Check the request URL and parameters to see if they are correct. If there is a type or an incorrect parameter, the web service may be unable to process the request.

By following these steps and using the Network tab in your browser's developer tools, you can gather more information about the issue and troubleshoot the problem more effectively.

Note: Make sure you are connected to the internet and that your connection is stable. You can try accessing other websites or services to see if they are working, and make sure the URL you are using to access the web service is correct and properly formatted.

Here are some common exceptions

- File not found.
- Document cache not found.
- Document pointer does not exist in the cache.

File not found

If you are encountering an error message stating that the web service is not listening due to a file not being found in the Syncfusion PDF viewer, you can try the following steps to resolve the issue:

Check the file path

Ensure that the file path you use to access the PDF file is correct and that the file exists in that location. You will need to update the file path if the file does not exist.

Document cache not found

The **Document cache not found** exception in Syncfusion PDF Viewer typically occurs when the cache used to store the rendered pages of a PDF document is not found or has been deleted. This can happen if the cache directory is changed or deleted or if the application is running in a different environment than it was previously.

Check for multiple instances

It's possible that you have multiple instances of the Syncfusion PDF Viewer running simultaneously, which can cause issues with the document cache. To check for this, open the Task Manager on your computer and look for any instances of the Syncfusion PDF Viewer running. If you find multiple instances, try closing them all and reopening the viewer.

We can use Redis cache and distributive cache for this issue.

Check your network connection

Ensure that your network connection is stable and strong enough to support the web service you are trying to use. Sometimes, simply restarting the web service can resolve the issue. Try stopping and starting the service again to see if it resolves the problem.

The document pointer does not exist in the cache.

The **Document pointer does not exist in the cache** exception in the Syncfusion PDF Viewer usually occurs when there is an issue with loading or caching the PDF document. This error can be caused by a variety of reasons, including:

To clear this error in the Syncfusion PDF Viewer, you can try the following steps:

Step 1: Clearing the cache may help resolve the issue. To clear the cache, navigate to the cache location, which can be found in the Syncfusion PDF Viewer's settings or configuration files. Once you locate the cache folder, delete its contents.

Step 2: Try reloading the document to ensure it is loaded correctly. You can do this by calling the controller's Load() method. Ensure the document is not already loaded before attempting to load it again.

Step 3: Restart the application. If clearing the cache does not work, you can try restarting the PDF Viewer application. This will reload all the necessary components and may resolve the error.

Internal server error

Server-side exceptions happen for various use cases. We can't just define them if they are document-specific, provide the document, or you may need to contact Syncfusion support for further assistance.

Retry Timeout

The **Retry Timeout** feature allows you to specify a duration for the PDF Viewer to retry failed AJAX requests before considering them permanent failures. It helps in handling temporary failures due to network issues or server unavailability.

The retryTimeout allows developers to specify a duration after which the AJAX request should retry failed requests automatically. Developers can ensure a more resilient and fault-tolerant PDF viewing experience by configuring an appropriate retry timeout value.

By default, when an AJAX request fails, the Retry Timeout property is set to **0**, indicating that no timeout is set. In this case, the PDF Viewer will wait indefinitely for a response, potentially leading to a hanging request. However, you can set the Retry Timeout property to a positive number, specifying the maximum number of seconds the PDF Viewer should wait for a response. If the response is not received within the specified time, the request will be aborted, triggering an appropriate error or timeout property.

To set the retry timeout, use the **retryTimeout** property in the PDF Viewer configuration. This property takes a value in seconds.

```
{% raw %}
`javascript
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
retryTimeout={10}
retryCount={5}
style={{ height: '640px' }}>
</PdfViewerComponent>
`

{% endraw %}
```

In the given example, the `retryTimeout` is set to 10 seconds, and the `retryCount` is set to 5. This means that if a request made by the PDF Viewer takes longer than 10 seconds to receive a response, it will be considered a timeout. In such cases, The PDF Viewer will resend the same request based on the `retryCount`. Here, this process will repeat up to maximum of 5 retries.

When an exception occurs during the AJAX request in the context of the PDF Viewer, the request will wait for the specified `retryTimeout` duration. If the timeout duration is exceeded, the PDF Viewer will decrement the `retryCount` and attempt to load the document again. This retry process continues until the document is successfully loaded or the `retryCount` limit is reached.

The `retryCount` property of the PDF Viewer allows you to set the number of retries for a specific request. This feature is particularly useful for handling temporary errors such as network timeouts or server issues. By initiating new requests according to the retry count, ensure a smoother user experience and efficiently handle network or server problems.

If the timeout duration specified by `retryTimeout` is exceeded during the AJAX request, the PDF Viewer will decrement the `retryCount` and initiate a new request. This process will continue until the document is successfully loaded or the retry count limit is reached. This functionality helps improve the handling of temporary errors and ensures a more efficient and user-friendly experience.

Find the sample [Retry Timeout](#)

Load N number of pages on initial loading

The initial rendering in a PDF viewer allows users to control the number of pages displayed when opening a PDF document. This improves the user experience by providing flexibility in loading a specific number of pages initially, while additional pages are dynamically rendered as the user scrolls through the document. This approach enhances the responsiveness of the PDF viewer and minimizes delays as users can access specific pages without waiting for the entire document to load.

To utilize this capability in Syncfusion PDF Viewer, use the `initialRenderPages` property. You can achieve the desired outcome by setting this property to the desired number of pages you want to load initially. However, it's important to exercise caution when setting a high value for the `initialRenderPages` in large documents with numerous pages. Rendering a large number of pages simultaneously can increase memory usage and slow down loading times, impacting the performance of the PDF viewer.

Using the `initialRenderPages` property judiciously is advisable, especially when dealing with larger documents. It is more suitable for scenarios where a smaller range of pages, such as 10-20, can be loaded to provide a quick initial view of the document.

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
  return (
    <div>
      <div className='control-section'>
        <PdfViewerComponent
          id='container'
```

```

documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
style={{ 'height': '640px' }}
initialRenderPages = {10}>
<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
Annotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
function App() {
return <div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
style={{ 'height': '640px' }}
initialRenderPages = {10}>
<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
Annotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

Find the sample [how to load n number of pages on initial loading](#)

Customize toolbar in PDF Viewer component

[How to customize existing toolbar in PDF Viewer component](#)

PDF Viewer allows you to customize(add, show, hide, enable, and disable) existing items in a toolbar.

- Add - New items can be defined by [CustomToolbarItemModel](#) and with existing items in [ToolbarSettings](#) property. Newly added item click action can be defined in [toolbarclick](#).
- Show, Hide - Existing items can be shown or hidden using the [ToolbarSettings](#) property. Pre-defined toolbar items are available with [ToolbarItem](#).
- Enable, Disable - Toolbar items can be enabled or disabled using [enabletoolbaritem](#)

STANDALONE

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
import { ComboBox } from '@syncfusion/ej2-dropdowns';
import { TextBox } from '@syncfusion/ej2-inputs';
export function App() {
  // Add OnCreateSearch outside the App function
  function OnCreateSearch() {
    this.addIcon('prepend', 'e-icons e-search');
  }
  var toolItem1 = {
    prefixIcon: 'e-icons e-paste',
    id: 'print',
    tooltipText: 'Custom toolbar item',
  };
  var toolItem2 = {
    id: 'download',
    text: 'Save',
    tooltipText: 'Custom toolbar item',
    align: 'right'
  };
  var LanguageList = ['Typescript', 'Javascript', 'Angular', 'C#', 'C',
    'Python'];
  var toolItem3 = {
    type: 'Input',
    tooltipText: 'Language List',
    cssClass: 'percentage',
    align: 'Left',
    id: 'dropdown',
    template: new ComboBox({ width: 100, value: 'TypeScript', dataSource:
    LanguageList, popupWidth: 85, showClearButton: false, readonly: false })
  };
  var toolItem4 = {
    type: 'Input',
    tooltipText: 'Text',
    align: 'Right',
    cssClass: 'find',
    id: 'textbox',
    template: new TextBox({ width: 125, placeholder: 'Type Here', created:
    OnCreateSearch })
  }
  function toolbarClick(args) {
    var viewer = document.getElementById('container').ej2_instances[0];
```

```

if (args.item && args.item.id === 'print') {
  viewer.printModule.print();
}
else if (args.item && args.item.id === 'download') {
  viewer.download();
}
};
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
resourcePath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
toolbarSettings={{ showTooltip : true, toolbarItems: [toolItem1, toolItem2,
'OpenOption', 'PageNavigationTool', 'MagnificationTool', toolItem3,
'PanTool', 'SelectionTool', 'SearchOption', 'PrintOption', 'DownloadOption',
'UndoRedoTool', 'AnnotationEditTool', 'FormDesignerEditTool', toolItem4,
'CommentTool', 'SubmitForm']}}
toolbarClick={toolbarClick}
style={{ 'height': '640px' }}>
/* Inject the required services */
<Inject services={[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

SERVER-BACKED

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, FormFields,
FormDesigner, Inject } from '@syncfusion/ej2-react-pdfviewer';
import { ComboBox } from "@syncfusion/ej2-dropdowns";
import { TextBox } from "@syncfusion/ej2-inputs";
export function App() {
  // Add OnCreateSearch outside the App function
  function OnCreateSearch() {
    this.addIcon('prepend', 'e-icons e-search');
  }
  var toolItem1 = {
    prefixIcon: 'e-icons e-paste',
    id: 'print',
    tooltipText: 'Custom toolbar item',
  };
  var toolItem2 = {
    id: 'download',

```

```

text: 'Save',
tooltipText: 'Custom toolbar item',
align: 'right'
});
var LanguageList = ['Typescript', 'Javascript', 'Angular', 'C#', 'C',
'Python'];
var toolItem3 = {
type: 'Input',
tooltipText: 'Language List',
cssClass: 'percentage',
align: 'Left',
id: 'dropdown',
template: new ComboBox({ width: 100, value: 'TypeScript', dataSource:
LanguageList, popupWidth: 85, showClearButton: false, readonly: false })
};
var toolItem4 = {
type: 'Input',
tooltipText: 'Text',
align: 'Right',
cssClass: 'find',
id: 'textbox',
template: new TextBox({ width: 125, placeholder: 'Type Here', created:
OnCreateSearch})
}
function toolbarClick(args){
var viewer = document.getElementById('container').ej2_instances[0];
if (args.item && args.item.id === 'print') {
viewer.printModule.print();
}
else if (args.item && args.item.id === 'download') {
viewer.download();
}
};
return (<div>
<div className='control-section'>
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/pdfviewer"
toolbarSettings={{ showTooltip : true, toolbarItems: [toolItem1, toolItem2,
'OpenOption', 'PageNavigationTool', 'MagnificationTool', toolItem3,
'PanTool', 'SelectionTool', 'SearchOption', 'PrintOption', 'DownloadOption',
'UndoRedoTool', 'AnnotationEditTool', 'FormDesignerEditTool', toolItem4,
'CommentTool', 'SubmitForm']}}
toolbarClick={toolbarClick}
style={{ 'height': '640px' }}>
{/* Inject the required services */}
<Inject services=[ Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner] />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

```
{% endraw %}
```

Note : Default value of toolbar items is ['OpenOption', 'PageNavigationTool', 'MagnificationTool', 'PanTool', 'SelectionTool', 'SearchOption', 'PrintOption', 'DownloadOption', 'UndoRedoTool', 'AnnotationEditTool', 'FormDesignerEditTool', 'CommentTool', 'SubmitForm']

Align Property

The align property is used to specify the alignment of a toolbar item within the toolbar.

Left: Aligns the item to the left side of the toolbar.

Right: Aligns the item to the right side of the toolbar.

Tooltip Property

The tooltip property is used to set the tooltip text for a toolbar item. Tooltip provides additional information when a user hovers over the item.

CssClass Property

The cssClass property is used to apply custom CSS classes to a toolbar item. It allows custom styling of the toolbar item.

Prefix Property

The prefix property is used to set the CSS class or icon that should be added as a prefix to the existing content of the toolbar item.

ID Property

The id property within a CustomToolbarItemModel is a compulsory attribute that plays a vital role in toolbar customization. It serves as a unique identifier for each toolbar item, facilitating distinct references and interactions.

When defining or customizing toolbar items, it is mandatory to assign a specific and descriptive id to each item.

These properties are commonly used when defining custom toolbar items with the CustomToolbarItemModel in the context of Syncfusion PDF Viewer. When configuring the toolbar using the ToolbarSettings property, you can include these properties to customize the appearance and behavior of each toolbar item.

Note: When customizing toolbar items, you have the flexibility to include either icons or text based on your design preference.

[View sample in GitHub](#)

Know the supported conformance PDF documents in React PDF Viewer component

The PDF Viewer supports the below conformance documents:

- **PDF/A-1a conformance**
- **PDF/A-1b conformance**
- **PDF/X-1a conformance**
- **PDF/A-2a conformance**
- **PDF/A-2b conformance**
- **PDF/A-2u conformance**
- **PDF/A-3a conformance**

- **PDF/A-3b conformance**
- **PDF/A-3u conformance**
- **PDF/A-4 conformance**
- **PDF/A-4e conformance**
- **PDF/A-4f conformance**

Organize Pages Feature in React Pdfviewer component

Introduction

Welcome to the User Guide for the Organize Pages feature in JS2 PDF Viewer. This powerful feature allows you to manage your PDF documents efficiently by organizing pages seamlessly. Whether you need to add new pages, remove unnecessary ones, or adjust page orientation, this feature has got you covered.

Getting Started

To access the Organize Pages feature, simply open the PDF document in the JS2 PDF Viewer and navigate to the toolbar. Look for the **Organize Pages** option to begin utilizing these capabilities.

Key Functionalities

- **Add New Pages:** Easily integrate additional content by adding new pages to your document. Simply select the option to insert new pages and customize them as needed.
- **Remove Pages:** Streamline your document management process by removing unnecessary pages with ease. Select the pages you wish to delete and confirm to remove them from the document.
- **Rotate Pages:** Resolve orientation issues by rotating pages clockwise or counterclockwise as required. This feature ensures that your document displays correctly, maintaining clarity and readability.
- **Select All Pages:** Make uniform adjustments and modifications by conveniently selecting all pages at once. This allows for efficient editing and formatting across the entire document.
- **Real-Time Updates:** Experience instant updates as any changes made to the page organization are instantly reflected within the PDF Viewer. Simply click the **Save** button to ensure that your modifications are preserved.
- **Save As Feature:** Preserve your edits by utilizing the **Save As** feature. This allows you to download the modified version of the PDF document for future reference, ensuring that your changes are stored securely.

API's supported

enableOrganizePdf: This API enables or disables the page organizer feature in the PDF Viewer. By default, it is set to **true**, indicating that the page organizer is enabled.

isPageOrganizerOpen: This API determines whether the page organizer dialog will be displayed automatically when a document is loaded into the PDF Viewer. By default, it is set to **false**, meaning the dialog is not displayed initially.

pageOrganizerSettings: This API allows control over various page management functionalities within the PDF Viewer. It includes options to enable or disable actions such as deleting, inserting, rotating, and moving pages. By default, all these actions are enabled.

showPageOrganizerTool: This API controls the visibility of the page organizer tool within the PDF Viewer. If set to **false**, the tool is hidden by default.

openPageOrganizer: This API opens the page organizer dialog within the PDF Viewer, providing access to manage PDF pages.

closePageOrganizer: This API closes the currently open page organizer dialog within the PDF Viewer, if it is present. It allows users to dismiss the dialog when done with page organization tasks.

Conclusion

With the Organize Pages feature in JS2 PDF Viewer, managing your PDF documents has never been easier. Whether you're adding new content, adjusting page orientation, or removing unnecessary pages, this feature provides the tools you need to streamline your document management workflow. Explore these capabilities today and take control of your PDF documents with ease!

Customize context menu in React Pdfviewer component

PDF Viewer allows you to add custom option in context menu. It can be achieved by using the `addCustomMenu()` method and custom action is defined using the `customContextMenuSelect()` method.

Add Custom Option

The following code shows how to add custom option in context menu.

```
`js
export function App() {
  let menuItems = [
    {
      text: 'Search In Google',
      id: 'searchingoogle',
      iconCss: 'e-icons e-de-ctnr-find'
    },
    {
      text: 'Lock Annotation',
      iconCss: 'e-icons e-lock',
      id: 'lock_annotation'
    },
    {
      text: 'Unlock Annotation',
      iconCss: 'e-icons e-unlock',
      id: 'unlock_annotation'
    },
    {
      text: 'Lock Form Field',
      iconCss: 'e-icons e-lock',
```

```

id: 'readonlytrue'
},
{
text: 'Unlock Form Field',
iconCss: 'e-icons e-unlock',
id: 'readonlyfalse'
},
];
function documentLoad(args) {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.addCustomMenu(menuitems, false);
}
return (
<div>
<div className='control-section'>
{/ Render the PDF Viewer /}
<PdfViewerComponent
id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
documentLoad={documentLoad}
customContextMenuSelect={customContextMenuSelect}
customContextMenuBeforeOpen={customContextMenuBeforeOpen}
height='640px'>
<Inject services={[Toolbar, Magnification, Navigation, Annotation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]} />
</PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
`

```

[Customize custom option in context menu](#)

The PDF Viewer feature enables customization of custom options and the ability to toggle the display of the default context menu. When the `addCustomMenu` parameter is set to `true`, the default menu is hidden; conversely, when it is set to `false`, the default menu items are displayed.

```
`js
export function App() {
  let menuItems = [
    {
      text: 'Search In Google',
      id: 'searchingoogle',
      iconCss: 'e-icons e-de-ctnr-find'
    },
    {
      text: 'Lock Annotation',
      iconCss: 'e-icons e-lock',
      id: 'lock_annotation'
    },
    {
      text: 'Unlock Annotation',
      iconCss: 'e-icons e-unlock',
      id: 'unlock_annotation'
    },
    {
      text: 'Lock Form Field',
      iconCss: 'e-icons e-lock',
      id: 'readonlytrue'
    },
    {
      text: 'Unlock Form Field',
      iconCss: 'e-icons e-unlock',
      id: 'readonlyfalse'
    },
  ];
  function documentLoad(args) {
```

```

var viewer = document.getElementById('container').ej2_instances[0];
viewer.addCustomMenu(menuitems, true);
}
return (
<div>
<div className='control-section'>
  {/ Render the PDF Viewer /}
  <PdfViewerComponent
    id="container"
    documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
    resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
    documentLoad={documentLoad}
    customContextMenuSelect={customContextMenuSelect}
    customContextMenuBeforeOpen={customContextMenuBeforeOpen}
    height='640px'>
    <Inject services={[Toolbar, Magnification, Navigation, Annotation, LinkAnnotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]} />
  </PdfViewerComponent>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
`

```

[Customize added context menu items](#)

The following code shows how to hide/show added custom option in context menu using the `customContextMenuBeforeOpen()` method.

```

`js
export function App() {
let menuitems = [
{
text: 'Search In Google',
id: 'searchingoogle',
iconCss: 'e-icons e-de-ctnr-find'

```

```
},
{
  text: 'Lock Annotation',
  iconCss: 'e-icons e-lock',
  id: 'lock_annotation'
},
{
  text: 'Unlock Annotation',
  iconCss: 'e-icons e-unlock',
  id: 'unlock_annotation'
},
{
  text: 'Lock Form Field',
  iconCss: 'e-icons e-lock',
  id: 'readonlytrue'
},
{
  text: 'Unlock Form Field',
  iconCss: 'e-icons e-unlock',
  id: 'readonlyfalse'
},
];

function documentLoad(args) {
  var viewer = document.getElementById('container').ej2_instances[0];
  viewer.addCustomMenu(menuItems, false, false);
}

function customContextMenuSelect(args) {
  var viewer = document.getElementById('container').ej2_instances[0];
  switch (args.id) {
    case 'searchingoogle':
      for (var i = 0; i < viewer.textSelectionModule.selectionRangeArray.length; i++) {
        var content = viewer.textSelectionModule.selectionRangeArray[i].textContent;
        if ((viewer.textSelectionModule.isTextSelection) && (/S/.test(content))) {
```

```
window.open('http://google.com/search?q=' + content);
}
}
break;
case 'lock_annotation':
lockAnnotations(args);
break;
case 'unlock_annotation':
unlockAnnotations(args);
break;
case 'readonlytrue':
setReadOnlyTrue(args);
break;
case 'readonlyfalse':
setReadOnlyFalse(args);
break;
default:
break;
}
}

function customContextMenuBeforeOpen(args) {
for (var i = 0; i < args.ids.length; i++) {
var search = document.getElementById(args.ids[i]);
var viewer = document.getElementById('container').ej2_instances[0];
if (search) {
search.style.display = 'none';
if (args.ids[i] === 'searchingoogle' && (viewer.textSelectionModule) &&
viewer.textSelectionModule.isTextSelection) {
search.style.display = 'block';
} else if (args.ids[i] === "lockannotation" || args.ids[i] === "unlockannotation") {
var isLockOption = args.ids[i] === "lock_annotation";
for (var j = 0; j < viewer.selectedItems.annotations.length; j++) {
var selectedAnnotation = viewer.selectedItems.annotations[j];
```

```
if (selectedAnnotation && selectedAnnotation.annotationSettings) {
var shouldDisplay = (isLockOption && !selectedAnnotation.annotationSettings.isLock) ||
(!isLockOption && selectedAnnotation.annotationSettings.isLock);
search.style.display = shouldDisplay ? 'block' : 'none';
}
}
} else if (args.ids[i] === "readonlytrue" && viewer.selectedItems.formFields.length !== 0) {
var selectedFormField = viewer.selectedItems.formFields[0].isReadOnly;
search.style.display = selectedFormField ? 'none' : 'block';
} else if (args.ids[i] === "readonlyfalse" && viewer.selectedItems.formFields.length !== 0) {
var selectedFormField = viewer.selectedItems.formFields[0].isReadOnly;
search.style.display = selectedFormField ? 'block' : 'none';
} else if (args.ids[i] === 'formfield properties' && viewer.selectedItems.formFields.length !== 0) {
search.style.display = 'block';
}
}
}
}
}

function lockAnnotations(args) {
var viewer = document.getElementById('container').ej2_instances[0];
var selectedAnnotations = viewer.selectedItems.annotations;
for (var i = 0; i < selectedAnnotations.length; i++) {
var annotation = selectedAnnotations[i];
if (annotation && annotation.annotationSettings) {
annotation.annotationSettings.isLock = true;
viewer.annotationModule.editAnnotation(annotation);
args.cancel = false;
}
}
}

function unlockAnnotations(args) {
var viewer = document.getElementById('container').ej2_instances[0];
var selectedAnnotations = viewer.selectedItems.annotations;
```

```
for (var i = 0; i < selectedAnnotations.length; i++) {
    var annotation = selectedAnnotations[i];
    if (annotation && annotation.annotationSettings) {
        annotation.annotationSettings.isLock = false;
        viewer.annotationModule.editAnnotation(annotation);
        args.cancel = false;
    }
}

function setReadOnlyTrue(args) {
    var viewer = document.getElementById('container').ej2_instances[0];
    var selectedFormFields = viewer.selectedItems.formFields;
    for (var i = 0; i < selectedFormFields.length; i++) {
        var selectedFormField = selectedFormFields[i];
        if (selectedFormField) {
            viewer.formDesignerModule.updateFormField(selectedFormField, {
                isReadOnly: true,
            });
        }
        args.cancel = false;
    }
}

function setReadOnlyFalse(args) {
    var viewer = document.getElementById('container').ej2_instances[0];
    var selectedFormFields = viewer.selectedItems.formFields;
    for (var i = 0; i < selectedFormFields.length; i++) {
        var selectedFormField = selectedFormFields[i];
        if (selectedFormField) {
            viewer.formDesignerModule.updateFormField(selectedFormField, {
                isReadOnly: false,
            });
        }
        args.cancel = false;
    }
}
```



```

}
}
return (
<div>
<div className='control-section'>
  {/ Render the PDF Viewer /}
  <PdfViewerComponent
    id="container"
    documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
    resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
    documentLoad={documentLoad}
    customContextMenuSelect={customContextMenuSelect}
    customContextMenuBeforeOpen={customContextMenuBeforeOpen}
    height='640px'>
    <Inject services={[Toolbar, Magnification, Navigation, Annotation, LinkAnnotation, BookmarkView,
    ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner]} />
  </PdfViewerComponent>
</div>
</div>);
}

const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);

```

The following is the output of custom context menu with customization.

INDEX.JSX

STANDALONE

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
'@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {

```

```
function pageRenderInitiate(args) {
  // This method is called when the page rendering starts
  console.log('Rendering of pages started' + args);
};
function pageRenderComplete(args) {
  // This method is called when the page rendering completes
  console.log('Rendering of pages completed' + args);
};
return (<div>
  <div className='control-section'>
    {/ * Render the PDF Viewer */}
    <PdfViewerComponent
      ref={(scope) => { pdfviewer = scope; }}
      id="container"
      documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
      resourceUrl="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
      pageRenderStart={pageRenderStart}
      pageRenderComplete={pageRenderComplete}
      style={{ 'height': '640px' }}>
      <Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
        Annotation,
        BookmarkView, ThumbnailView, Print, TextSelection, TextSearch]} />
    </PdfViewerComponent>
  </div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}
```

SERVER-BACKED

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
  LinkAnnotation, BookmarkView,
  ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from
  '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
  function pageRenderStart(args) {
    // This method is called when the page rendering starts
    console.log('Rendering of pages started' + args);
  };
  function pageRenderComplete(args) {
    // This method is called when the page rendering completes
    console.log('Rendering of pages completed' + args);
  };
  return (<div>
    <div className='control-section'>
      {/ * Render the PDF Viewer */}
      <PdfViewerComponent
        ref={(scope) => { pdfviewer = scope; }}

```

```

id="container"
documentPath="https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
serviceUrl="https://services.syncfusion.com/react/production/api/pdfviewer"
pageRenderStart={pageRenderStart}
pageRenderComplete={pageRenderComplete}
style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation,
Annotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
{% endraw %}

```

The provided code demonstrates how to subscribe to the `pageRenderStart` and `pageRenderComplete` events in the Syncfusion PDF Viewer component.

[View sample in GitHub]()

Open and Close Bookmark pane programmatically

The PDF Viewer library allows you to open the Bookmark pane programmatically using the `openBookmarkPane()` method.

The following steps are used to open the Bookmark.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Insert the following code snippet to implement the functionality for opening the Bookmark pane:

```

<button onClick={openBookmark}>Open Bookmark Pane</button>

`ts
<script>
function openBookmark() {
var viewer = document.getElementById('container').ej2_instances[0];
// Open Bookmark pane
viewer.bookmarkViewModule.openBookmarkPane();
}
</script>

```

Similarly, to close the Bookmark pane programmatically, employ the following code snippet:

```

`
<button onClick={closeBookmark}>Close Bookmark Pane</button>
`
`ts
<script>
function closeBookmark() {
var viewer = document.getElementById('container').ej2_instances[0];
// close Bookmark pane
viewer.bookmarkViewModule.closeBookmarkPane();
}
</script>
`

```

[View sample in GitHub]()

Troubleshooting

Why Do I Have to Manually Copy Files from node_modules into My App?

PDF Viewer offers flexibility across different build systems, remaining both framework-agnostic and independent of bundlers. Even without a bundler, you can seamlessly integrate the PDF Viewer by directly linking its assets through standard HTML tags.

Moreover, our codebase is meticulously divided into distinct files, enabling selective loading of components when required. This strategic approach to lazy loading prevents unwieldy file sizes that a single bundle might impose, which is often impractical.

While 'pdfium.js,' the primary entry point, is commonly bundled automatically, the supplementary assets from 'ej2-pdfviewer-lib' need to be manually incorporated due to their on-demand loading. This necessity arises because the host application lacks inherent awareness of these assets' lazy loading behavior.

Troubleshoot error 'cp' is not recognized as a command

The error message you're seeing, "'cp' is not recognized as an internal or external command," is because the `cp` command you're trying to use is not recognized on Windows command prompt.

On Windows, you should use the `copy` command to copy files and directories instead of `cp`. The equivalent command in Windows to copy a directory and its contents recursively is:

```

`batch
xcopy /s /e /i .\node_modules\@syncfusion\ej2-pdfviewer\dist\ej2-pdfviewer-lib public\ej2-pdfviewer-
lib
`

```

Here, `/s` indicates that you want to copy directories and subdirectories recursively. Also, note that Windows uses backslashes `\` as path separators, not forward slashes `/`.

Make sure to run this command in the appropriate directory where you want to perform the copy operation.

Note: If you encounter other issues or error messages while working with the Windows Command Prompt, make sure to double-check your command syntax and file paths for accuracy. Additionally, ensure that you have the necessary permissions to perform the copy operation in the specified directories.

Document Loading Issues in Version 23.1 or Newer

If you're experiencing problems with your document not rendering in the viewer, especially when using version 23.1 or a newer version, follow these troubleshooting steps to resolve the issue:

1. **Check for viewer.dataBind() Requirement:** Ensure that you have called `viewer.dataBind()` as required in version 23.1 or newer. This explicit call is essential for initializing data binding and document rendering correctly. It is must to call the `dataBind()` method before load.

```
{% raw %}
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import './index.css';
import { PdfViewerComponent, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, Annotation, TextSearch, Inject } from '@syncfusion/ej2-react-pdfviewer';
let pdfviewer;
function App() {
function documentLoad() {
var viewer = document.getElementById('container').ej2_instances[0];
viewer.serviceUrl = "https://services.syncfusion.com/react/production/api/pdfviewer";
viewer.dataBind();
viewer.load("https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",null);
}
return (<div>
<div className='control-section'>
{ / Render the PDF Viewer /}
<button onClick={documentLoad}>Load</button>
<PdfViewerComponent
ref={({scope}) => { pdfviewer = scope; }}
id="container"
```

```

style={{ 'height': '640px' }}>
<Inject services={[ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch]} />
</PdfViewerComponent>
</div>
</div>
);
}

const root = ReactDOM.createRoot(document.getElementById('sample'));
root.render(<App />);
,

{% endraw %}

```

2. **Verify Document Source:** Confirm that the document source or URL you're trying to display is valid and accessible. Incorrect URLs or document paths can lead to loading issues.
3. **Network Connectivity:** Ensure that your application has a stable network connection. Document rendering may fail if the viewer can't fetch the document due to network issues.
4. **Console Errors:** Use your browser's developer tools to check for any error messages or warnings in the console. These messages can provide insights into what's causing the document not to load.
5. **Loading Sequence:** Make sure that you're calling `viewer.dataBind()` and initiating document loading in the correct sequence. The viewer should be properly initialized before attempting to load a document.
6. **Update Viewer:** Ensure that you're using the latest version of the viewer library or framework. Sometimes, issues related to document loading are resolved in newer releases.
7. **Cross-Origin Resource Sharing (CORS):** If you're loading documents from a different domain, ensure that CORS headers are correctly configured to allow cross-origin requests.
8. **Content Security Policies (CSP):** Check if your application's Content Security Policy allows the loading of external resources, as this can affect document loading. Refer [here](#) to troubleshoot.

By following these troubleshooting steps, you should be able to address issues related to document loading in version 23.1 or newer, ensuring that your documents render correctly in the viewer.