6. DataGrid

- The DataGrid is a UI component that represents data from a local or remote source in the form of a grid. This UI component offers such basic features as sorting, grouping, filtering, as well as more advanced capabilities, like state storing, client-side exporting, master-detail interface, and many others.

6.1. Data Binding

The DataGrid component displays data from a local or remote store.
For binding data into the datagrid we have to specify *dataSource* option which can have store's or simple array as value.
Simple Array : In this kind of binding, we assigns *array or ArrayStore* to the *dataSource.*
Ajax Request : We can create *CustomStore or DataStore* and calling *APIs or fetching remote data using Ajax request* and then assigns store to the *dataSource*.

6.2. Paging and Scrolling

Paging allows the UI component to load data in portions instead of loading it simultaneously. To enable paging, set the *paging.enabled* property to *true*.
Users can switch between pages and change paging settings using the *pager* or they can scroll the pages. Paging settings apply with any *scrolling* mode.
The pager is an element that allows users to navigate through pages and change their size at runtime.
Paging Options
enabled (Boolean-true) : Enables paging
pageIndex (Number-0) : Specifies page to be displayed using a zero based index
pageSize (Number-20) : Specifies page size
Pager Options
allowedPageSizes(Array<N|S>|String-'auto') : Specifies available page sizes in page size selector
displayMode (String-'adaptive') : Specifies pager's display mode ('adaptive' | 'compact' | 'full')
infoText (String) : Specifies page information text ({0} current page,{1} total page,{2} total row)
showInfo (Boolean-false) : Specifies whether to show the page information
showNavigationButtons (Boolean-false) : Specifies whether to show navigation buttons
showPageSizeSelector (Boolean-false) : Specifies whether to show the page size selector
visible (Boolean|Sting-'auto') : Specifies whether the pager is visible
Scrolling allows a user to browse data left outside the current viewport.

Scrolling Options

columnRenderingMode (String-‘standard’) : Specifies the rendering mode for columns
('standard' | 'virtual')

mode (String-‘standard’) : Specifies the scrolling mode ('infinite' | 'standard' | 'virtual')

preloadEnabled (Boolean-false) : Specifies whether UI component should load adjacent pages
Applies only if scrolling.mode is *"virtual"* or *"infinite"*

rowRenderingMode (String-‘standard’) : Specifies the rendering mode for loaded rows
('standard' | 'virtual')

scrollByContent(Boolean),scrollByThumb(Boolean),showScrollBar(String),useNative(Boolean)

6.3. Editing

The UI component can allow a user to add, update and delete data.

Editing Options

allowAdding (Boolean-false) : Specifies whether a user can add new rows

allowDeleting (Boolean|Func) : Specifies whether a user can delete rows. It is called for each data
row when defined as a function

allowUpdating (Boolean|Func) : Specifies whether a user can update rows. It is called for each data row when defined as a function

mode (String-‘row’) : Specifies how a user edits data ('batch' | 'cell' | 'row' | 'form' | 'popup)

refreshMode (String-‘full’) : Specifies operations that are performed after saving changes
('full' | 'reshape' | 'repaint')

Changes[], confirmDelete, editColumnName, editRowKey, form, popup, selectTextOnEditStart, startEditAction, texts, useIcons

6.4. Grouping

Data in DataGrid can be grouped by one column or by several. Once a column is used for grouping, it is added to the group panel.

We can create group by specifying groupIndex, from contextMenu or by dragging columns into groupPanel.

Grouping Options

allowCollapsing(Boolean-true), autoExpandAll(Boolean-true), contextMenuEnabled(Boolean-false), expandMode(String), texts(Object)

Group Panel Options

allowColumnDragging(Boolean-true), emptyPanelText(String), visible(Boolean|String-false)

6.5. Filtering

Data in DataGrid can be filtered using filterValue,headerFilters,filterRow,filterPannel etc.

filterValue (filter configuration-null) : Specifies a filter expression

Filter Panel Options

customizeText(function), filterEnabled(Boolean-true), texts(Object), visible(Boolean-false)

Filter Row Options

applyFilter(String-'auto'), applyFilterText(String), betweenEndText(String), betweenStartText(String), operationDescriptions(Object), resetOperationsText(String), showAllText(String), showOperationChooser(Boolean-true), visible(Boolean-false)

Header Filter Options

allowSearch(Boolean-false), height(Number-325/315), searchTimeout(Number-500), texts(Object), visible(Boolean-false), width(Number-252)

filterBuilder (filterBuilder configuration) : Configures the integrated filter builder

filterBuilderPopup (popup configuration) : Configures the popup in which the integrated filter builder is shown

6.6. Sorting

A user can sort rows by values of a single or multiple columns depending on the value of the **sorting**.mode property.

Sorting Options

mode(String-'single') : Specifies the sorting mode (Accepts : 'multiple' | 'none' | 'single')
ascendingText(String), cleartext(String), descendingText(String), showSortIndexes(Boolean-true)
Also we can specifying sortOrder, sortIndex(sort.mode=multiple) and sortingMethod into columns for default sorting.

6.7. Selection

A user can select rows in a single or multiple mode. In multiple mode, a user can select all rows at once.
By default, once a user selects a row, the data source is instantly notified about it. This may lower the UI component performance if the data source is remote and the user is allowed to select all rows at once. In this case, we can make the selection deferred.

Selection Options

allowSelectAll(Boolean-true), deferred(Boolean-false), selectAllMode(String-'allPages')('allPages'|'page'), showCheckBoxesMode(String-'onClick')('always'|'none'|"onClick'|' onLongTap') mode(String-'none') : Specifies selection mode (Accepts : 'multiple'|'single'|'none')

Selection Filter (filter expression)

Specifies filters for the rows that must be selected initially. Applies only if **selection**.deferred is **true**.

6.8. Columns

By default, a column is created for each field of a data source object, but in most cases, it is redundant. To specify a set of columns to be created in a grid, assign an array specifying these columns to the **columns** property. Each grid column is represented in this array by an object containing column settings or by a data source field that this column is bound to.

Column properties define the behavior and appearance of a grid column. One of the other capabilities allows you to control the sorting of column values using the allowSorting and sortOrder properties, apply a filter to grid records using the allowFiltering and filterOperations properties, and group grid records using the allowGrouping and groupIndex properties. In addition, you can change the visibility and width of a column using corresponding properties.

To get or set a property or several properties for a column at runtime, use the columnOption method with the required arguments.

Column Customization

customizeColumns (function(columns)) : Customizes columns after they are created

Columns based on a Data Source

We can specify columns using columns[] option

Multi-level Headers (Band)

We can specify multiple columns inside one column or band, by enabling isBand to true and assign columns of band

Column Resizing

User can resize columns if allowResizing for column/columns is true. Also we can set columnAutoWidth, columnMinWidth etc.

Command Column Customization

We can customize command columns by specifying column's configuration inside columns[] options. Columns like buttons,adaptive,selection etc known as command columns

ColumnChooser and ColumnFixing

The column chooser allows a user to hide columns at runtime. To enable it, assign **true** to the **columnChooser**.enabled property.

When the width of all columns exceeds the UI component width, horizontal scrolling appears. If specific columns should be on screen constantly regardless of how far the UI component is scrolled, allow a user to fix them at runtime using the context menu. For this, set the **columnFixing.**enabled property to **true**.

6.9. State Persistence

State storing enables the UI component to save applied settings and restore them the next time the UI component is loaded. Assign true to the stateStoring.enabled property to enable this functionality.

State storing saves the following properties :

filterValue, focusedRowKey, selectedRowKeys, selectionFilter, filterPanel.filterEnabled, paging.pageSize, paging.pageIndex, searchPanel.text, columns.dataField, columns.dataType, columns.filterType, columns.filterValue, columns.filterValues, columns.fixed, columns.fixedPosition, columns.groupIndex, columns.name, columns.selectedFilterOperation, columns.sortIndex, columns.sortOrder, columns.visible (only if the column chooser is enabled), columns.visibleIndex, columns.width

StateStoring Options

customLoad(function-R.promise) : pecifies a function that is executed on state loading. Applies only if the type is *'custom'*.

customSave(function) : Specifies a function that is executed on state change. Applies only if the type is *"custom"*.

type(String-'localStorage') : Specifies the type of storage where the state is saved.

enabled(Boolean-false), savingTimeout(Number-2000), storageKey(String-null)

6.10. Appearance

We can customize appearance of DataGrid by setting height, width, showBorders, showColumnHeaders, showColumnLine, showRowLines, rowAlternationEnabled, etc. properties.

6.11. Template

We can customize DataGrid's different portions templates like column template, cell template, row template, toolbar template, etc.

Column Template

We can customize particular column's template by setting cellTemplate on that column

Row Template

We can customize how rows appears into a DataGrid by setting rowTemplate as we want

Cell Customization

Cells can be customized by setting onCellPrepared property in association with cellTemplate

Toolbar Customization

We can also customize toolbar by setting onToolbarPreparing, we can add custom toolbar items into the toolbar.

6.12. Data Summaries

A summary is a grid feature that provides a synopsis of data contained in the grid. A summary consists of several items. A summary item displays a value that is a product of applying an aggregate function to the data of a specific column.
There are two types of summary in DataGrid: group and total. The group summary is calculated on a group of data, which is segregated during grouping. To specify the items of the group summary, declare an array of objects and assign it to the summary.groupItems field.
The total summary is calculated on all data contained in the grid. To specify the items of the total summary, declare an array of objects and assign it to the summary.totalItems field.

Summary Options

groupItems(Array<Object>) : Specifies items of the group summary
totalItems(Array<Object>) : Specifies items of the total summary
calculateCustomSummary(function(Object)), recalculateWhileEditing(Boolean-false), skipEmptyValues(Boolean-true), texts(Object)

6.13. Master-Detail View

In DataGrid, a master-detail interface supplies a usual data row with an expandable section that contains the details on this data row. In that case, the data row is called "master row", while the section is called "detail section".
To enable the master-detail interface, assign **true** to the **masterDetail**.enabled property. After that, specify the template for detail sections using the **masterDetail.template** property. Templates allow you to place virtually anything into the detail sections. For example, you can display another DataGrid or any other UI component there.

Master-Detail Options

autoExpandAll(Boolean-false), enabled(Boolean-false), template(Template(container,data))

6.14. Export

We can export DataGrid into PDF format or Excel sheet. Also user can export all data of DataGrid or selected rows of DataGrid as needed.

Export Options

allowExportSelectedData(Boolean-false), customizeExcelCell(Function(Object)), enabled(Boolean-false), excelFilterEnabled(Boolean-false), excelWrapTextEnabled(Boolean), filename(String-‘DataGrid’), ignoreExcelErrors(Boolean-true), proxyUrl(String), texts(Object)

Export PDF

We can export DataGrid to PDF using jsPdf library, we can include it using cdn or download it, after refrencing jsPdf, we can make object of it and write our logic in onExporting function for exporting DataGrid.

Export Excel

We can export DataGrid to Excel using ExcelJs and FileSaver libraries, we can include it using cdn or download it, after refrencing them, we can make object of it and write our logic in onExporting function for exporting DataGrid.

onExporting (Function(Object)) : A function that is executed before data is exported

6.15. Adaptability

Adaptability refers to the concept of adaptiveness of the DataGrid into various devices, We can make our DataGrid adaptive by setting hidingPriority of columns in combination with column of type ‘adaptive’. If we set columnAuto to ture then adaptive column will not work.

hidingPriority (Number) : Specifies the order in which columns are hidden when the UI component adapts to the screen or container size.

Ignored if allowColumnResizing is true and columnResizingMode is "widget".

The hidingPriority is a unique positive integer that ascends from right to left beginning with 0 by default. Columns with low hidingPriority are hidden first.