**Create and configure a widget:**

Devextreme UI component must be placed in a div container. Add the widget by chaining them on the selector with prefix as ‘dx’.

**Get and Set properties:**

We can get single property, all properties, set single property or get all properties. On the instance of the widget, one can retrieve or set the property or multiple properties by passing them in an object.

**Call methods:**

One can call various methods from the widget instance such as reset, focus, etc.

**Handle Events:**

There are various events that can be attached to a widget by subscribing them using ‘on’ method. To unsubscribe an event, pass the event or function name as 2nd parameter in the ‘off’ method.

There are various events such as :onCellClick, onKeyDown, onFocus, etc.

**Destroy a widget:**

A widget can be destroyed by invoking the ‘dispose()’ method on the widget instance and further use .remove() method for removing the corresponding ‘div’ tag of that widget from the DOM.

**option in DevExtreme?**

In DevExtreme, option is a method used to **get or set** configuration options dynamically for UI components **after initialization**. This allows you to update the component's properties without needing to recreate it.

Options: can be modified. Option method for initialization.

Properties: read-only. Use widget instance methods to get the property details.

var dataGridInstance = $("#dataGridContainer").dxDataGrid("instance");

        var dataSource = dataGridInstance.option("dataSource");

        var editMode = dataGridInstance.option("editing.mode");

        // ---------- or ----------

        var dataSource = $("#dataGridContainer").dxDataGrid("option", "dataSource");

        var editMode = $("#dataGridContainer").dxDataGrid("option", "editing.mode");

A screenshot of a black screen

Description automatically generated

**Which One Should You Use?**

1. **Use Version 1 (instance) when:**
   * You are **accessing multiple options** from the DataGrid.
   * You might need to **call other methods** on the DataGrid instance later.
   * You want better **performance** for repeated interactions.
2. **Use Version 2 (Direct .option) when:**
   * You only need to retrieve **one or two options**.
   * You prefer a **shorter** and simpler syntax.
   * Performance is not a major concern.

Widgets:

1. **Checkbox:**

**Options**

* text
* value
* enableThreeStateBehavior
* disabled
* hint
* iconSize
* activeStateEnabled
* focusStateEnabled
* hoverStateEnabled
* isValid
* name
* readOnly
* rtlEnabled
* tabIndex
* validationStatus
* validationErrors
* visible
* width
* accessKey
* elementAttr

**Events**

* onValueChanged
* onOptionChanged
* onInitialized
* onContentReady
* onDisposing

**Methods**

* option()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* reset()
* resetOption()
* element()
* focus()
* registerKeyHandler()
* off()
* on()

1. **Datebox:**

**Options**

* value
* min
* max
* displayFormat
* type
* opened
* readOnly
* disabled
* showClearButton
* placeholder
* useMaskBehavior
* showAnalogClock
* show24HourFormat
* focusStateEnabled
* activeStateEnabled
* tabIndex
* validationStatus
* validationErrors
* hint
* elementAttr
* rtlEnabled
* width
* height
* name
* displayExpr
* valueExpr

**Events**

* onValueChanged
* onOpened
* onClosed
* onFocusIn
* onFocusOut
* onInitialized
* onOptionChanged
* onContentReady
* onDisposing

**Methods**

* option()
* reset()
* resetOption()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* element()
* focus()
* selectText()
* clear()
* show()
* hide()

1. **Dropdownbox:**

**Options**

* value
* valueExpr
* displayExpr
* dataSource
* placeholder
* searchEnabled
* showClearButton
* inputAttr
* opened
* readOnly
* disabled
* clearButtonText
* useSearch
* grouped
* groupTemplate
* contentTemplate
* height
* width
* rtlEnabled
* tabIndex
* validationStatus
* validationErrors
* showPopupButton
* popupHeight
* popupWidth
* searchMode
* minSearchLength
* showTitle

**Events**

* onValueChanged
* onOpened
* onClosed
* onContentReady
* onDisposing
* onOptionChanged
* onInitialized
* onSelectionChanged
* onSearch

**Methods**

* option()
* reset()
* resetOption()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* element()
* focus()
* selectText()
* clear()
* show()
* hide()
* open()
* close()

1. **Numberbox:**

**Options**

* value
* valueExpr
* displayExpr
* min
* max
* step
* placeholder
* showSpinButtons
* format
* mode
* useLargeSpinButtons
* showClearButton
* inputAttr
* disabled
* readOnly
* rtlEnabled
* tabIndex
* width
* height
* validationStatus
* validationErrors

**Events**

* onValueChanged
* onInput
* onFocusIn
* onFocusOut
* onKeyDown
* onKeyUp
* onContentReady
* onDisposing
* onInitialized
* onOptionChanged

**Methods**

* option()
* reset()
* resetOption()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* element()
* focus()
* clear()
* show()
* hide()

1. **Selectbox:**

**Options**

* value
* dataSource
* displayExpr
* valueExpr
* placeholder
* searchEnabled
* searchMode
* searchTimeout
* minSearchLength
* itemTemplate
* grouped
* groupTemplate
* showClearButton
* readOnly
* disabled
* showDataBeforeSearch
* inputAttr
* clearButtonText
* showSelectionControls
* dropDownButtonTemplate
* width
* height
* validationStatus
* validationErrors
* hint
* tabIndex
* itemHoldTimeout
* deferRendering
* openOnFieldClick
* opened
* dropDownOptions
* popupHeight
* popupWidth
* useItemTextAsTitle

**Events**

* onValueChanged
* onOpened
* onClosed
* onItemClick
* onItemContextMenu
* onItemHold
* onContentReady
* onDisposing
* onInitialized
* onOptionChanged

**Methods**

* option()
* reset()
* resetOption()
* beginUpdate()
* endUpdate()
* dispose()
* repaint()
* element()
* focus()
* clear()
* show()
* hide()
* open()
* close()

**6) Textarea:**

**Options**

* accessKey
* activeStateEnabled
* autoResizeEnabled
* disabled
* elementAttr
* focusStateEnabled
* height
* hint
* hoverStateEnabled
* inputAttr
* isValid
* maxHeight
* maxLength
* minHeight
* name
* placeholder
* readOnly
* rtlEnabled
* spellCheck
* stylingMode
* tabIndex
* text
* validationError
* validationErrors
* validationMessageMode
* validationStatus
* value
* valueChangeEvent
* visible
* width

**Methods**

* blur()
* focus()
* reset()
* select()
* updateDimensions()

**Events**

* onChange
* onContentReady
* onCopy
* onCut
* onDisposing
* onEnterKey
* onFocusIn
* onFocusOut
* onInitialized
* onInput
* onKeyDown
* onKeyPress
* onKeyUp
* onOptionChanged
* onPaste
* onValueChanged

**7) Textbox:**

**Options**

* accessKey
* activeStateEnabled
* buttons
* disabled
* elementAttr
* focusStateEnabled
* height
* hint
* hoverStateEnabled
* inputAttr
* isValid
* label
* labelMode
* mask
* maskChar
* maskInvalidMessage
* maskRules
* maxLength
* mode
* name
* placeholder
* readOnly
* rtlEnabled
* showClearButton
* showMaskMode
* spellCheck
* stylingMode
* tabIndex
* text
* useMaskedValue
* validationError
* validationErrors
* validationMessageMode
* validationStatus
* value
* valueChangeEvent
* visible
* width

**Methods**

* blur()
* focus()
* getButton(name)
* reset()
* select()
* updateDimensions()

**Events**

* onChange
* onContentReady
* onCopy
* onCut
* onDisposing
* onEnterKey
* onFocusIn
* onFocusOut
* onInitialized
* onInput
* onKeyDown
* onKeyPress
* onKeyUp
* onOptionChanged
* onPaste
* onValueChanged

--

Validation Rules:

1. Async rules:

A custom validation rule that is checked asynchronously. Use async rules for server-side validation. (Eg: This username is already taken.)

1. Compare rules:

A validation rule that demands that a validated editor has a value that is equal to a specified expression.

1. Custom rules: Pass custom logic in validationCallback.
2. Email rule: Built-in for email verification.
3. Numeric rule: Built-in numeric check verification.
4. Pattern rule: Pass custom reg-ex for verification.
5. Range rule: Specifies range for integral input.
6. Required rule: Specifies that the following widget value is required.
7. Stringlength rule: Same as range but for string length

Data Layer:

**1. ArrayStore**

**Options:**

* data – The in-memory array of objects managed by the store.
* key – The unique key field in objects for CRUD operations.
* onModified – A callback function that executes when data is modified.
* errorHandler – Handles errors that occur during data operations.

**Methods:**

* load(options) – Fetches data with filtering, sorting, and paging applied.
* byKey(key) – Retrieves a specific record by key.
* insert(values) – Adds a new record to the store.
* update(key, values) – Modifies an existing record by key.
* remove(key) – Deletes a record by key.
* clear() – Removes all data from the store.

**2. DataSource**

**Options:**

* store – The underlying store (e.g., ArrayStore, CustomStore).
* filter – Defines filtering conditions.
* sort – Specifies sorting criteria.
* group – Groups data based on a specified field.
* pageSize – Defines the number of items per page.
* paginate – Enables or disables pagination.
* searchExpr – Specifies fields for searching.
* searchOperation – Defines the search operation (e.g., contains, startswith).
* searchValue – The value used for searching.

**Methods:**

* load() – Fetches data from the store.
* reload() – Refreshes the data source.
* filter(criteria) – Applies filtering conditions dynamically.
* sort(ordering) – Sorts the data dynamically.
* group(field) – Groups data by a specific field.
* totalCount() – Returns the total number of records.
* isLastPage() – Checks if the last page has been reached.

**3. CustomStore**

**Options:**

* load(options) – Defines how data is fetched from the server.
* byKey(key) – Retrieves a specific record by key.
* insert(values) – Sends an insert request to the server.
* update(key, values) – Sends an update request.
* remove(key) – Sends a delete request.
* cacheRawData – Enables caching of raw data.
* loadMode – Determines whether to load data in raw or processed form.
* onLoading – Callback triggered before loading data.
* onLoaded – Callback triggered after data is loaded.
* onInserting – Callback before inserting a new record.
* onInserted – Callback after inserting a new record.
* onUpdating – Callback before updating a record.
* onUpdated – Callback after updating a record.
* onRemoving – Callback before removing a record.
* onRemoved – Callback after removing a record.

**Methods:**

* load(options) – Fetches data from the server with filtering, sorting, and paging.
* byKey(key) – Retrieves a record based on its key.
* insert(values) – Calls the server to add a new record.
* update(key, values) – Updates a record on the server.
* remove(key) – Deletes a record from the server.

**4. LocalStore**

**Options:**

* name – The key under which data is stored in localStorage.
* data – Initial data stored in localStorage.
* immediate – Determines whether changes are saved immediately.

**Methods:**

* load() – Fetches data from localStorage.
* insert(values) – Adds a new record to localStorage.
* update(key, values) – Updates an existing record.
* remove(key) – Deletes a record from localStorage.
* clear() – Removes all data from localStorage.

**5. Query**

**Methods:**

* filter(criteria) – Filters data based on specified conditions.
* sort(field, order) – Sorts data in ascending or descending order.
* groupBy(field) – Groups data by a specific field.
* select(fields) – Selects specific fields from the dataset.
* aggregate(method, field) – Performs aggregation (sum, avg, count, etc.).
* toArray() – Returns the final processed data as an array.
* count() – Returns the number of records in the dataset.
* slice(skip, take) – Paginates data by skipping and taking a specified number of records.

DataGrid:

**Data Binding**

* **dataSource** – Defines the data source of the grid.
* **dateSerializationFormat** – Specifies the format for serializing date values.
* **getDataSource()** – Retrieves the grid's data source instance.

**Paging and Scrolling**

* **pager** – Configures the pagination controls.
* **paging** – Enables or disables paging.
* **scrolling** – Configures scrolling behavior (standard, virtual, infinite).
* **pageCount()** – Returns the total number of pages.
* **pageIndex()** – Gets or sets the current page index.
* **pageSize()** – Gets or sets the number of rows per page.

**Editing**

* **editing** – Enables editing mode (batch, row, cell, popup, etc.).
* **addRow()** – Adds a new row to the grid.
* **deleteRow()** – Deletes a row by key or index.
* **editRow()** – Puts a row into edit mode.
* **editCell()** – Puts a specific cell into edit mode.
* **saveEditData()** – Saves all pending changes.
* **cancelEditData()** – Cancels all pending changes.
* **cellValue()** – Gets or sets a cell’s value.
* **closeEditCell()** – Closes the currently edited cell.
* **hasEditData()** – Checks if there are unsaved changes.

**Data Validation**

* **errorRowEnabled** – Displays an error row if validation fails.

**Cascading Lookups**

* **customizeColumns** – Allows modifying column settings dynamically.

**Grouping**

* **groupPanel** – Enables the group panel for drag-and-drop grouping.
* **grouping** – Configures group-related settings.
* **expandAll()** – Expands all grouped rows.
* **collapseAll()** – Collapses all grouped rows.
* **expandRow()** – Expands a specific grouped row.
* **collapseRow()** – Collapses a specific grouped row.

**Filtering**

* **filterBuilder** – Enables advanced filtering with a UI-based filter builder.
* **filterBuilderPopup** – Configures the filter builder popup.
* **filterRow** – Enables filtering within column headers.
* **filterPanel** – Displays an interactive filter panel.
* **filterSyncEnabled** – Synchronizes filter states across UI components.
* **clearFilter()** – Clears all applied filters.
* **searchPanel** – Enables a search bar for quick filtering.
* **searchByText()** – Filters grid data based on a text query.
* **getCombinedFilter()** – Retrieves the current filter expression.

**Sorting**

* **sorting** – Enables or disables sorting.
* **clearSorting()** – Resets sorting on all columns.

**Selection**

* **selection** – Configures row selection (single, multiple, checkbox).
* **selectedRowKeys** – Holds the selected row keys.
* **selectAll()** – Selects all rows.
* **deselectAll()** – Deselects all rows.
* **selectRows()** – Selects specific rows by key.
* **selectRowsByIndexes()** – Selects rows by index.
* **deselectRows()** – Deselects specific rows by key.
* **getSelectedRowKeys()** – Retrieves the keys of selected rows.
* **getSelectedRowsData()** – Retrieves the data of selected rows.
* **isRowSelected()** – Checks if a row is selected.

**Columns**

* **columns** – Defines column configurations.
* **columnAutoWidth** – Automatically adjusts column width.
* **columnChooser** – Enables a UI for selecting visible columns.
* **columnFixing** – Allows fixing columns to the left or right.
* **columnHidingEnabled** – Enables automatic hiding of columns on small screens.
* **columnMinWidth** – Sets the minimum width of columns.
* **columnResizingMode** – Defines how column resizing works.
* **columnResizing** – Enables or disables column resizing.
* **columnCustomization** – Allows customizing column settings dynamically.
* **addColumn()** – Adds a new column to the grid.
* **deleteColumn()** – Removes a column from the grid.
* **getCellElement()** – Retrieves the HTML element of a specific cell.

**Appearance**

* **hoverStateEnabled** – Enables hover effects on rows.
* **rowAlternationEnabled** – Enables alternate row coloring.
* **showBorders** – Displays borders around grid elements.
* **showColumnHeaders** – Shows or hides column headers.
* **showColumnLines** – Displays lines between columns.
* **showRowLines** – Displays lines between rows.
* **wordWrapEnabled** – Enables or disables text wrapping.

**Templates**

* **rowTemplate** – Defines a custom row layout.
* **cellCustomization** – Enables custom cell rendering.
* **columnTemplate** – Allows customizing column content.

**Toolbar Customization**

* **toolbarCustomization** – Configures toolbar buttons and elements.

**Data Summaries**

* **summary** – Enables summary rows with aggregate functions.
* **gridSummaries** – Displays summaries at the bottom of the grid.
* **groupSummaries** – Displays summaries within groups.
* **customSummaries** – Allows defining custom summary functions.

**Master-Detail**

* **masterDetail** – Enables nested grids or custom detail views.
* **masterDetailView** – Configures the master-detail relationship.

**Export**

* **export** – Enables exporting grid data.
* **exportToExcel()** – Exports data to an Excel file.

**Adaptability**

* **adaptiveDetailRow** – Enables adaptive row details for small screens.
* **isAdaptiveDetailRowExpanded()** – Checks if an adaptive detail row is expanded.
* **collapseAdaptiveDetailRow()** – Collapses an adaptive detail row.
* **expandAdaptiveDetailRow()** – Expands an adaptive detail row.
* **gridAdaptabilityOverview** – Provides an overview of adaptability settings.
* **gridColumnsHidingPriority** – Sets priority for hiding columns on small screens.

**Demo with Custom Store and Concept of DataGrid**

* **stateStoring** – Saves and restores grid state.
* **state()** – Retrieves or sets the current grid state.

**Menu**

**Options**

* **dataSource** – Defines the menu items.
* **displayExpr** – Specifies which field in the data source should be displayed.
* **items** – Defines the menu structure manually.
* **itemTemplate** – Customizes the menu item rendering.
* **orientation** – Sets the menu orientation (horizontal/vertical).
* **rtlEnabled** – Enables right-to-left support.
* **showFirstSubmenuMode** – Configures how the first submenu appears.
* **submenuDirection** – Defines the direction of submenus.
* **adaptivityEnabled** – Enables menu responsiveness.
* **animation** – Configures submenu animations.
* **cssClass** – Applies a custom CSS class.
* **hideSubmenuOnMouseLeave** – Hides submenus when the mouse leaves.
* **selectByClick** – Enables menu selection by clicking.
* **selectedItem** – Stores the currently selected menu item.

**Methods**

* **open()** – Opens a specific submenu.
* **close()** – Closes an open submenu.
* **selectItem()** – Selects a menu item.
* **unselectItem()** – Unselects a menu item.
* **option()** – Gets or sets an option dynamically.
* **getSelectedItem()** – Returns the currently selected item.
* **updateDimensions()** – Updates menu layout when the container size changes.

**Events**

* **onItemClick** – Fires when a menu item is clicked.
* **onItemRendered** – Fires after an item is rendered.
* **onSelectionChanged** – Fires when the selection changes.
* **onSubmenuShowing** – Fires before a submenu is shown.
* **onSubmenuShown** – Fires after a submenu is shown.
* **onSubmenuHiding** – Fires before a submenu is hidden.
* **onSubmenuHidden** – Fires after a submenu is hidden.

**TreeView**

**Options**

* **dataSource** – Binds the tree to a data source.
* **displayExpr** – Specifies which field in the data source should be displayed.
* **items** – Defines the tree structure manually.
* **searchEnabled** – Enables searching within the tree.
* **searchExpr** – Specifies the fields used for searching.
* **searchMode** – Sets the search mode (contains, startsWith, etc.).
* **selectionMode** – Defines how selection works (single, multiple).
* **showCheckBoxesMode** – Enables checkboxes for selection.
* **expandNodesRecursive** – Expands all child nodes when a parent is expanded.
* **expandAllEnabled** – Enables a button to expand/collapse all nodes.
* **parentIdExpr** – Specifies the field for parent-child relationships.
* **expandedExpr** – Determines whether a node should be expanded.
* **hasItemsExpr** – Indicates whether a node has children.
* **hoverStateEnabled** – Enables hover effects.
* **focusStateEnabled** – Enables keyboard navigation focus.
* **animationEnabled** – Enables or disables node expansion animation.

**Methods**

* **selectItem()** – Selects a tree node.
* **unselectItem()** – Unselects a tree node.
* **collapseItem()** – Collapses a tree node.
* **expandItem()** – Expands a tree node.
* **getSelectedNodes()** – Retrieves selected nodes.
* **getSelectedNodeKeys()** – Gets selected node keys.
* **getNodes()** – Retrieves all tree nodes.
* **getNodeByKey()** – Finds a node by its key.
* **refresh()** – Reloads the tree data.
* **scrollToItem()** – Scrolls to a specific item.
* **updateDimensions()** – Updates tree layout when the container size changes.

**Events**

* **onItemClick** – Fires when a tree item is clicked.
* **onItemExpanded** – Fires when an item is expanded.
* **onItemCollapsed** – Fires when an item is collapsed.
* **onItemRendered** – Fires after a tree item is rendered.
* **onSelectionChanged** – Fires when selection changes.
* **onContentReady** – Fires when the tree is fully loaded.

**Load Indicator**

A visual indicator that represents a loading state.

**Options**

* **indicatorSrc** – Defines a custom loading indicator image.
* **rtlEnabled** – Enables right-to-left layout support.
* **visible** – Controls visibility.

**Methods**

* **option()** – Gets or sets an option dynamically.
* **dispose()** – Destroys the instance.

**Events**

* **onContentReady** – Fires when the widget is fully loaded.

**Load Panel**

A loading screen overlay that prevents user interaction during processing.

**Options**

* **animation** – Configures show/hide animations.
* **container** – Specifies the container for the load panel.
* **delay** – Sets a delay before showing the panel.
* **hideOnOutsideClick** – Determines if clicking outside closes the panel.
* **message** – Displays a loading message.
* **position** – Sets the panel’s position.
* **shading** – Enables a background overlay.
* **shadingColor** – Specifies the shading color.
* **visible** – Controls visibility.

**Methods**

* **show()** – Displays the load panel.
* **hide()** – Hides the load panel.
* **toggle()** – Toggles visibility.
* **option()** – Gets or sets an option dynamically.

**Events**

* **onShown** – Fires when the panel is shown.
* **onHidden** – Fires when the panel is hidden.
* **onContentReady** – Fires when the panel is fully rendered.

**Popup**

A floating container for displaying dialogs, forms, or messages.

**Options**

* **animation** – Configures show/hide animations.
* **closeOnOutsideClick** – Determines if clicking outside closes the popup.
* **contentTemplate** – Defines custom content rendering.
* **dragEnabled** – Allows dragging.
* **fullScreen** – Displays the popup in full-screen mode.
* **height** – Sets height.
* **position** – Defines placement.
* **resizeEnabled** – Allows resizing.
* **shading** – Enables a background overlay.
* **shadingColor** – Sets the shading color.
* **showCloseButton** – Displays a close button.
* **visible** – Controls visibility.
* **width** – Sets width.

**Methods**

* **show()** – Opens the popup.
* **hide()** – Closes the popup.
* **toggle()** – Toggles visibility.
* **option()** – Gets or sets an option dynamically.

**Events**

* **onShown** – Fires when the popup is shown.
* **onHidden** – Fires when the popup is hidden.
* **onContentReady** – Fires when the popup is fully rendered.
* **onResize** – Fires when the popup is resized.
* **onResizeEnd** – Fires when resizing is finished.
* **onResizeStart** – Fires when resizing begins.

**Popover**

A small floating popup used for contextual messages or additional information.

**Options**

* **animation** – Configures show/hide animations.
* **closeOnOutsideClick** – Determines if clicking outside closes the popover.
* **contentTemplate** – Defines custom content rendering.
* **position** – Specifies placement relative to the target element.
* **target** – Defines the element that triggers the popover.
* **visible** – Controls visibility.
* **width** – Sets width.

**Methods**

* **show()** – Opens the popover.
* **hide()** – Closes the popover.
* **toggle()** – Toggles visibility.
* **option()** – Gets or sets an option dynamically.

**Events**

* **onShown** – Fires when the popover is shown.
* **onHidden** – Fires when the popover is hidden.
* **onContentReady** – Fires when the popover is fully rendered.

**Toast**

A lightweight notification message that disappears after a few seconds.

**Options**

* **animation** – Configures show/hide animations.
* **closeOnClick** – Closes the toast when clicked.
* **displayTime** – Sets how long the toast is visible.
* **message** – The text displayed in the toast.
* **position** – Defines where the toast appears.
* **type** – Sets the toast type (success, error, warning, etc.).
* **visible** – Controls visibility.

**Methods**

* **show()** – Displays the toast.
* **hide()** – Hides the toast.
* **toggle()** – Toggles visibility.
* **option()** – Gets or sets an option dynamically.

**Events**

* **onShown** – Fires when the toast appears.
* **onHidden** – Fires when the toast disappears.
* **onContentReady** – Fires when the toast is fully loaded.