Basics of Devextreme

1.1 Introduction to DevExtreme

- DevExtreme is a comprehensive suite of high-performance HTML5 and JavaScript-based UI components and tools for building responsive web applications.
- It provides a wide range of widgets and frameworks designed to meet the needs of various types of applications, from simple websites to complex enterprise solutions.
- The DevExtreme jQuery Component Suite is a feature-complete set of 65+ responsive and touch-enabled UI components implemented as jQuery plugins.
- The components are a **data grid**, **interactive charts**, **data editors**, **navigation** and **multi-purpose UI components**.

- **Wide Range of Widgets:**
  - **Data Visualization**: Charts, gauges, sparklines, and maps.
  - **Data Management:** Data grids, pivot grids, tree lists, and data editors.
  - **Navigation & Layout:** Tabs, menus, toolbars, navigation panels, and layouts.
  - **Form Elements:** Text boxes, buttons, checkboxes, radio buttons, sliders, date pickers, and more.
- **Responsive and Adaptive:**

  DevExtreme components are designed to be responsive, automatically adjusting to different screen sizes and orientations. This ensures a consistent and optimized user experience across various devices, including desktops, tablets, and smartphones.

- **Cross-Platform Support:**

  DevExtreme is compatible with major front-end frameworks such as Angular, React, Vue, and ASP.NET Core. It also supports integration with jQuery.

- **Theming and Customization:**

  DevExtreme provides extensive theming capabilities, allowing developers to customize the appearance of components to match their application's design. Themes can be easily switched or customized using the DevExtreme ThemeBuilder.

1.2 Installation – NuGet Package

- DevExtreme sources are scripts and stylesheets. You can get them from a Content Delivery Network (CDN) or download and use them locally.

1.2.1 CDN

- Link DevExtreme scripts and stylesheets within the <head> tag on your index page. The order of the scripts and stylesheets is important.

```
<head>
        <script type="text/javascript"
        src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

          <!-- DevExtreme theme →
        <link rel="stylesheet"
        href="https://cdn3.devexpress.com/jslib/21.1.11/css/dx.light.css">

           <!-- DevExtreme library -->
           <script type="text/javascript"
        src="https://cdn3.devexpress.com/jslib/21.1.11/js/dx.all.js"></script>
           <!-- <script type="text/javascript"
        src="https://cdn3.devexpress.com/jslib/21.1.11/js/dx.web.js"></script> -->
           <!-- <script type="text/javascript"
        src="https://cdn3.devexpress.com/jslib/21.1.11/js/dx.viz.js"></script> →
</head>
<body class="dx-viewport">
   <!-- ... -->
</body>
```

1.2.2 Local Files

- You can find all required files in the DevExtreme ZIP archive or in the DevExtreme folder (%ProgramFiles(x86)%\DevExpress 21.1\DevExtreme\Sources) if you used the Windows installer.
- Copy the Lib folder into the folder with your application.
- Then, link jQuery and DevExtreme stylesheets and scripts in the index page's <head> tag in the following order:

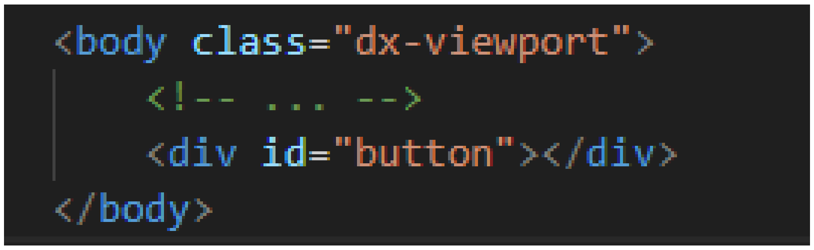
```html
<head>
    <!-- ... -->
    <script type="text/javascript" src="js/jquery-3.5.1.min.js"></script>

    <!-- DevExtreme theme -->
    <link rel="stylesheet" href="css/dx.light.css">

    <!-- DevExtreme library -->
    <script type="text/javascript" src="js/dx.all.js"></script>
    <!-- <script type="text/javascript" src="js/dx.web.js"></script> -->
    <!-- <script type="text/javascript" src="js/dx.viz.js"></script> -->
</head>
<body class="dx-viewport">
    <!-- ... -->
</body>
```

1.3 Widget Basics - jQuery

- To initialize a DevExtreme widget, you need to create an HTML element and use jQuery to apply the widget to that element.

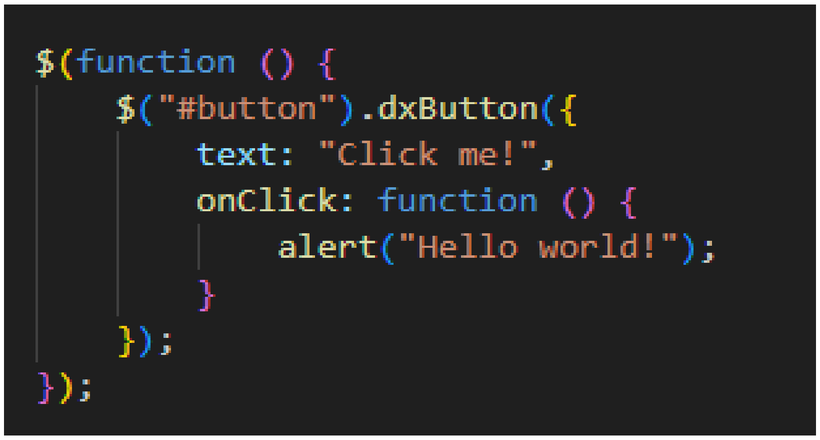
1.3.1 Create and Configure a Widget

- Any DevExtreme UI component must be placed in a container. This role is played by a <div> HTML element.
- Add a <div> to the <body> tag of your page. Make sure that this <div> has the id attribute specified.



```html
<body class="dx-viewport">
    <!-- ... -->
    <div id="button"></div>
</body>
```

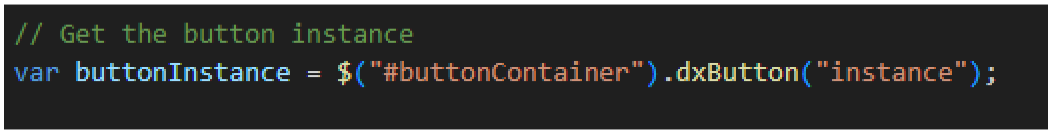
- DevExtreme supplies a jQuery plugin for each UI component. To create, for example, the Button UI component within the buttonContainer element, use the dxButton() plugin as the following code shows.



```javascript
$(function () {
    $("#button").dxButton({
        text: "Click me!",
        onClick: function () {
            alert("Hello world!");
        }
    });
});
```

1.3.2 Get a Widget Instance

- Use the following code to get a UI component instance:



```javascript
// Get the button instance
var buttonInstance = $("#buttonContainer").dxButton("instance");
```

- If the UI component is not yet instantiated, this code throws an E0009 exception that you can handle with a try...catch block:

```
try {
    var chartInstance = $("#chartContainer").dxChart("instance");
}
catch (err) {
    alert("Exception handled: " + err.message);
}
```

1.3.3 Get and Set Options

- All operations with UI component properties are carried out using the **option()** method. You can use it to do the following:

- **Get Single Property**

```
var buttonText = buttonInstance.option("text");
alert("Button text: " + buttonText);
```

- **Get All Properties**

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

- **Set a Single Property**

```
buttonInstance.option("text", "New Text");
alert("Button text has been changed");
```

- **Set All Properties**

```
var dataGridInstance = $("#dataGridContainer").dxDataGrid("instance");
dataGridInstance.option({
    dataSource: [],
    editing: {
        mode: "cell"
    }
});
```

1.3.4 Call Methods

- To call a UI component method, pass its name to the jQuery plugin.

```
var allSeries = $("#chartContainer").dxChart("getAllSeries");
```

- If a method accepts arguments, pass them right after the method's name.

```
var fruitsSeries = $("#chartContainer").dxChart("getSeriesByName", "fruits");
```

- As an alternative, you can obtain the UI component instance first, and then call any method of this instance.

```
var chartInstance = $("#chartContainer").dxChart("instance");
var allSeries = chartInstance.getAllSeries();
var fruitsSeries = chartInstance.getSeriesByName("fruits");
```

1.3.5 Handle Events

- **Subscribe to an Event:**
  You can subscribe to an event using a configuration property. All event handling properties are given names that begin with on.

```
$("#dataGridContainer").dxDataGrid({
    onCellClick: function (e) {
        // Handles the "cellClick" event
    },
    onSelectionChanged: function (e) {
        // Handles the "selectionChanged" event
    }
});
```

- As a more flexible solution, you can use the **on()** method. It allows you to subscribe to events at runtime and attach several handlers to a single event.

```
var dataGridInstance = $("#dataGridContainer").dxDataGrid("instance");
// Subscribes to the "cellClick" and "selectionChanged" events
dataGridInstance
    .on({
        "cellClick": cellClickHandler,
        "selectionChanged": selectionChangedHandler
    });
```

- **Unsubscribe from an Event**
  To detach a specific handler from an event, call the **off(eventName, handler)** method.

```
var dataGridInstance = $("#dataGridContainer").dxDataGrid("instance");
// Detaches the "cellClickHandler1" from the "cellClick" event leaving other handle
dataGridInstance.off("cellClick", cellClickHandler1)
```

- You can also use this method to detach all handlers from a particular event.

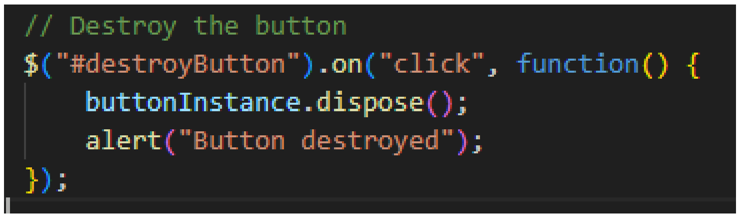
```
var dataGridInstance = $("#dataGridContainer").dxDataGrid("instance");
// Detaches all handlers from the "cellClick" event
dataGridInstance.off("cellClick")
```

- If you subscribed to an event using an **onEventName** property, you can unsubscribe from it by setting this property to undefined.

```
var dataGridInstance = $("#dataGridContainer").dxDataGrid("instance");
dataGridInstance.option("onCellClick", undefined);
```

1.3.6 Destroy a Widget

- To dispose of a DevExtreme UI component, free up the allocated resources by calling the **dispose()** method. Then, remove the UI component's associated DOM node:



```
// Destroy the button
$("#destroyButton").on("click", function() {
    buttonInstance.dispose();
    alert("Button destroyed");
});
```

```
$("#dataGridContainer").dxDataGrid("dispose");
$("#dataGridContainer").remove();
```