1. Basics of DevExtreme

1.1 Introduction to DevExtreme

DevExtreme is a front-end ui framework for creating high impact, responsive desktop and mobile web apps with ease. DevExtreme provides components for Angular, React, Vue, and Jquery. It provides cdn and local files for creating apps etc.

Template for creating the app using jQuery.

```
<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 Installation - Nuget Package

Installation of DevExtreme can be done in two ways.

1. **Use Nuget package** :- Go to project's nuget package manager and then search DevExtreme.Web then choose which version you want to install and download it.

2. **Use Package Manager Console:-**
   In that type **Install-Package DevExteme.Web -Version 21.1.3** for the specific version

1.3 Widget Basics

1.3.1 Create and Configure 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.

<div id="buttonContainer"></div>

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.

```
$(function () {
   $("#buttonContainer").dxButton();
});
```

To configure a UI component, pass an object to the plugin as shown in the following code. Properties of this object mirror the properties of the UI component.

```
$("#buttonContainer").dxButton({
   text: "Click Me!", // Show the text to user
   onClick: function () {
     console.log("Click me button clicked."); // log the text when button clicked.
   },
   onInitialized: function () {
     console.log("Button initiated.");
   },
   onContentReady: function () {
     console.log("Content ready.");
   },
 });
```

1.3.2 Get a Widget Instance

Instance can be get by using two ways

```
 let buttonInst = $("#buttonContainer").dxButton("instance");
 let buttonInstUsingDx = $('#buttonContainer').dxButton().instance();
```

1.3.3 Get and Set Options

```
// Using instance
let btnVal1 = buttonInst.option("text");
console.log("With Instance ", btnVal1);

// Without instance
let btnVal2 = $("#buttonContainer").dxButton("option", "text");
console.log("Without instance", btnVal2);

// Get all properties
let buttonOptions = buttonInst.option();
const _ = $("#buttonContainer").dxButton("option");
console.log("All properties", buttonOptions);

// Set the property
buttonInst.option("text", "value set by instance.");

// Set the property without instance
$("#buttonContainer").dxButton("option", "text", "Without instance");

// Set all properties
buttonInst.option({
  text: "Setting all properties using instance.",
  onClick: function () {
    console.log("New updated event is clicked.");
  },
});

// Without instance
$("#buttonContainer").dxButton({
  text: "Setting all properties without instance.",
  onClick: function () {
    console.log("New updated event is clicked.");
  },
});
```

1.3.4 Call Methods

For calling the method of the button or any other component using

```
// Call Methods
$("#buttonContainer").dxButton().click();
```

```
  $("#buttonContainer").dxButton().click(function(){console.log("from call
methods")});
```

1.3.5 Handle Events

Events of the components are handled using on and off methods of any instance.

If you use one then you can bind the event with the specified callback function. Off is used to detach the callback which is associated with that event. If you specify which callback function to detach then it detaches it and if zero arguments then it off all the callback functions which are attached by one method of instance. For handling the onClick event you need to pass undefined set the value of that method.

```
// binding event using on
  buttonInst.on({
    click: function () {
      console.log("button clicked using multiple event setter.");
    },
  });

  buttonInst.on("click", clickHandler1).on("click", clickHandler2);

  function clickHandler1() {
    console.log("Click handler 1");
  }

  function clickHandler2() {
    console.log("Click handler 2");
  }

  // unsubscribe the function which is given as the second argument.
  buttonInst.off("click", clickHandler1);

  // unsubscribe all click event of button
  // buttonInst.off("click");

// unsubscribe using undefined
  buttonInst.option("onClick", undefined);
```

1.3.6 Destroy a Widget

Destroying a widget can be done in two ways.

1. It removes all the classes which are added during initialization and simply give a div at the user side.

```
// dispose the button using instance
  buttonInst.dispose();
  $("#buttonContainer").dxButton("dispose");
```

2. In other way it removes that div from the html page permanently.

   ```
   $("#buttonContainer").remove();
   ```