**Ext JS Event Handling Documentation**

**Introduction to Ext JS and Event Handling**

**Overview of Ext JS**

Ext JS is a powerful JavaScript framework for building rich, interactive web applications.

**Features:**

* Comprehensive UI components (grids, forms, trees, charts, etc.).
* Supports MVC/MVVM architecture for structured code.
* Advanced data handling and robust cross-browser compatibility.

**What is Event Handling?**

* Event handling is the process of responding to user interactions such as clicks, key presses, and form submissions.
* Ext JS provides a robust event-handling mechanism to manage user interactions efficiently.
* Helps enhance user experience by making applications dynamic and interactive..
* Supports event delegation to optimize performance by reducing multiple event listeners.
* Provides built-in event listeners that simplify handling common UI interaction.

**Types of Events in Ext JS**

**Mouse Events**

* **click**: Triggered when an element is clicked.

**Ext.get('myButton').on('click', function() {**

**alert('Button clicked!');**

**});**

* **dblclick: Fires when an element is double-clicked.**
  + Useful for triggering actions that require confirmation, such as renaming a file or opening a detailed view.
  + Can be used to toggle UI components between expanded and collapsed states.
* **mouseover / mouseout: Detects mouse hovering.**
  + Helps in displaying tooltips or highlighting elements when a user hovers over them.
  + Can be used to dynamically change element styles for better user engagement.
  + Useful for creating interactive menus where items expand on hover.

**Keyboard Events**

Keyboard events are triggered when a user interacts with a keyboard. The primary events are:

1. **keydown**: Fires when a key is initially pressed down.
2. **keypress (Deprecated)**: Fires when a key that produces a character value is pressed.
3. **keyup**: Fires when a key is released after being pressed.

**Example of Keydown Event:**

**Ext.getBody().on('keydown', function(event) {**

**console.log('Key pressed: ' + event.getKey());**

**});**

**Form Events**

* change: Detects input changes.
* submit: Fires when a form is submitted.

**Window Events**

* resize: Triggered when the window is resized.
* beforeclose: Fires before a window is closed.

**Handling Events in Ext JS**

**Button Click Event Handling**

**Ext.create('Ext.Button', {**

**text: 'Click Me',**

**renderTo: Ext.getBody(),**

**handler: function() {**

**alert('Button was clicked!');**

**}**

**});**

**Event Delegation**

* Instead of attaching event listeners to multiple elements, delegate them to a common ancestor.

Example:

**Ext.get('container').on('click', function(event, target) {**

**if (target.tagName === 'BUTTON') {**

**alert('Button inside container clicked!');**

**}**

**});**

**Custom Events in Ext JS**

**Creating and Firing Custom Events**

**Ext.define('CustomComponent', {**

**extend: 'Ext.Component',**

**initComponent: function() {**

**this.addEvents('customEvent');**

**this.callParent(arguments);**

**}**

**});**

**var myComponent = Ext.create('CustomComponent');**

**myComponent.on('customEvent', function() {**

**console.log('Custom event triggered!');**

**});**

**myComponent.fireEvent('customEvent');**

* **addEvents()**: Declares a custom event.
* **fireEvent()**: Triggers the event.

**Demo Project and Conclusion**

**Demo Project Overview**

**Features:**

* Button click events.
* Form validation.
* Dynamic grid updates.

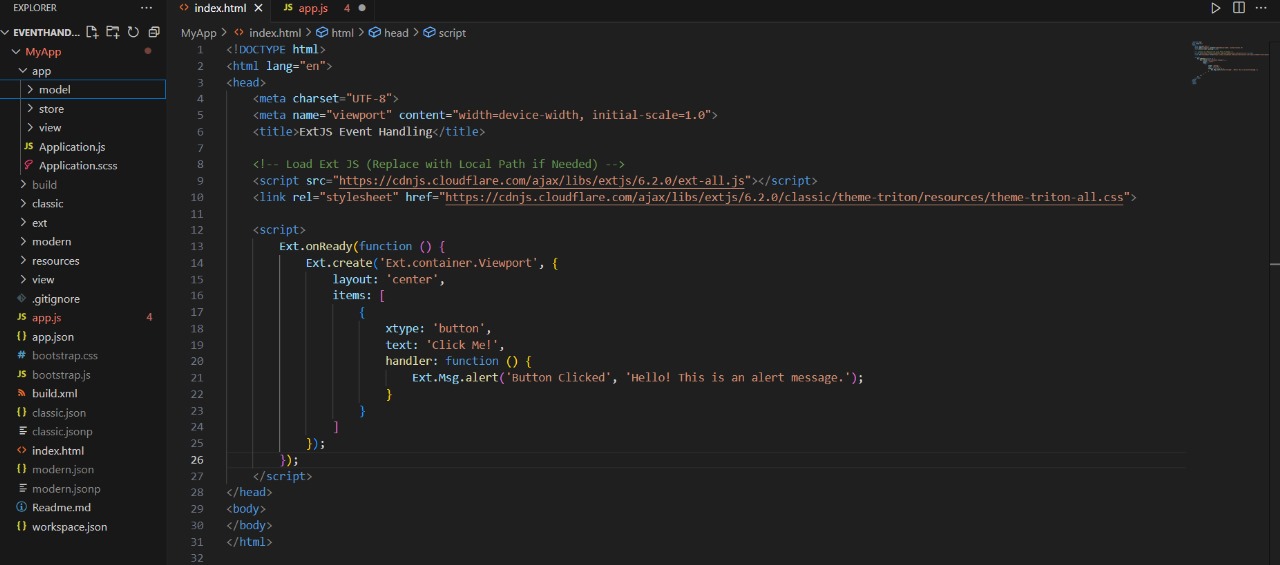
**Key Takeaways**

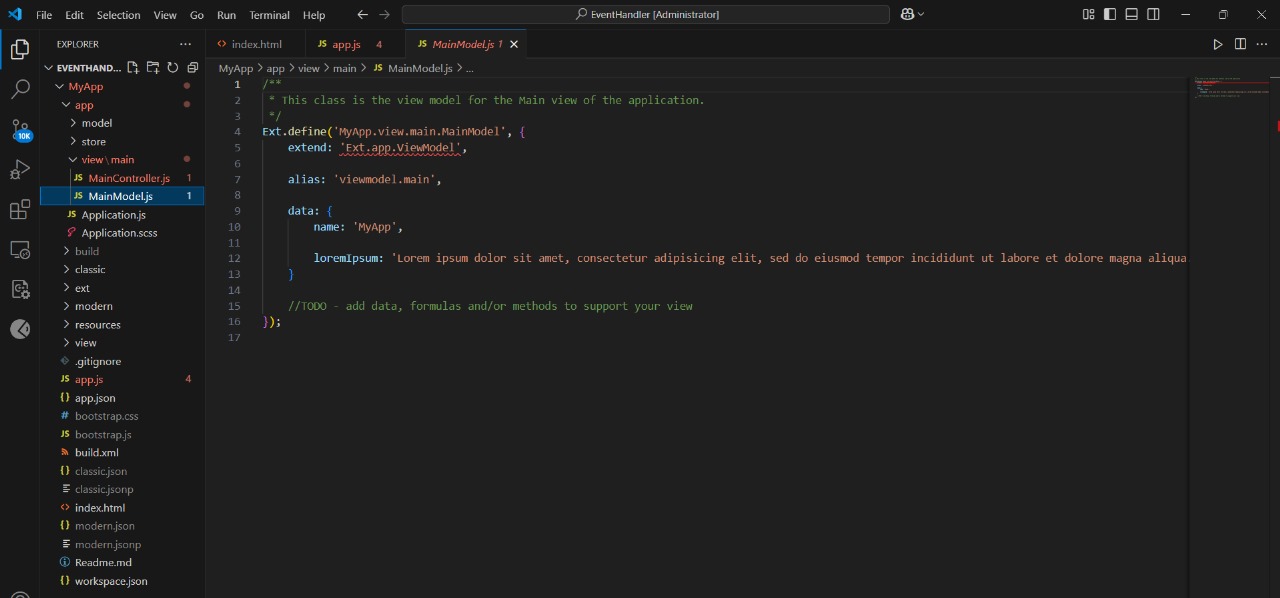
* Ext JS provides a structured approach to event handling.
* Supports built-in, delegated, and custom events.
* Helps build interactive and responsive web applications efficiently.

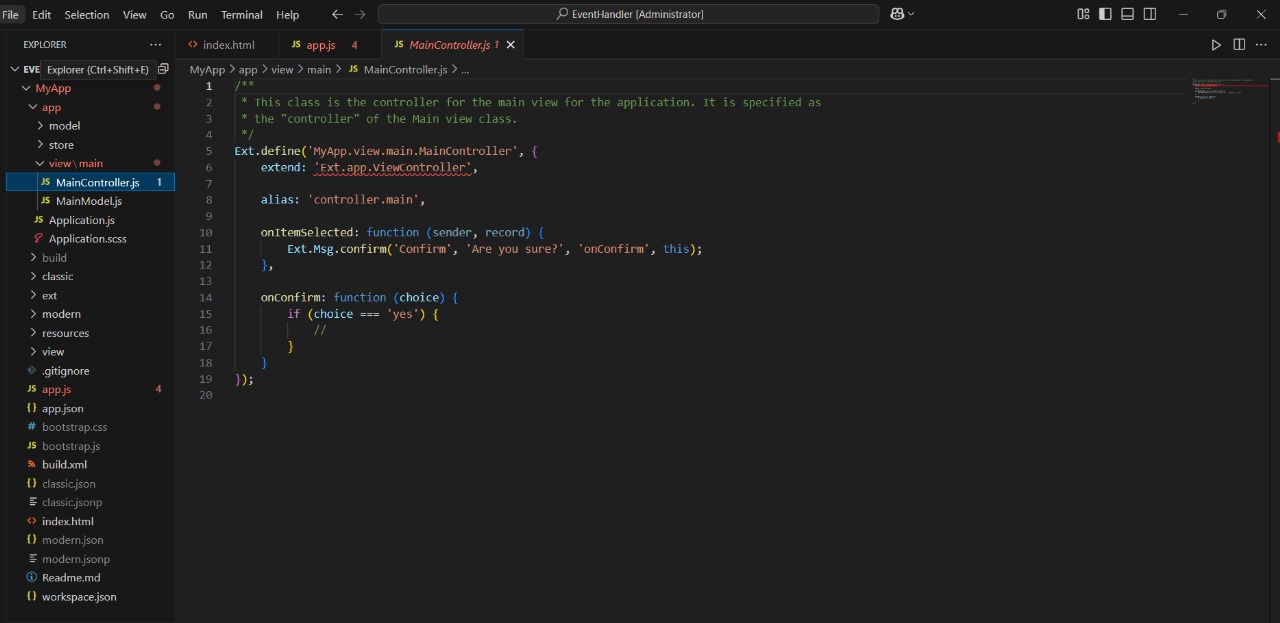
**Code Availability**

* All sample codes and demo project details are provided in the presentation.
* Follow the structured examples to implement event handling in Ext JS projects effectively.

**CODE:**

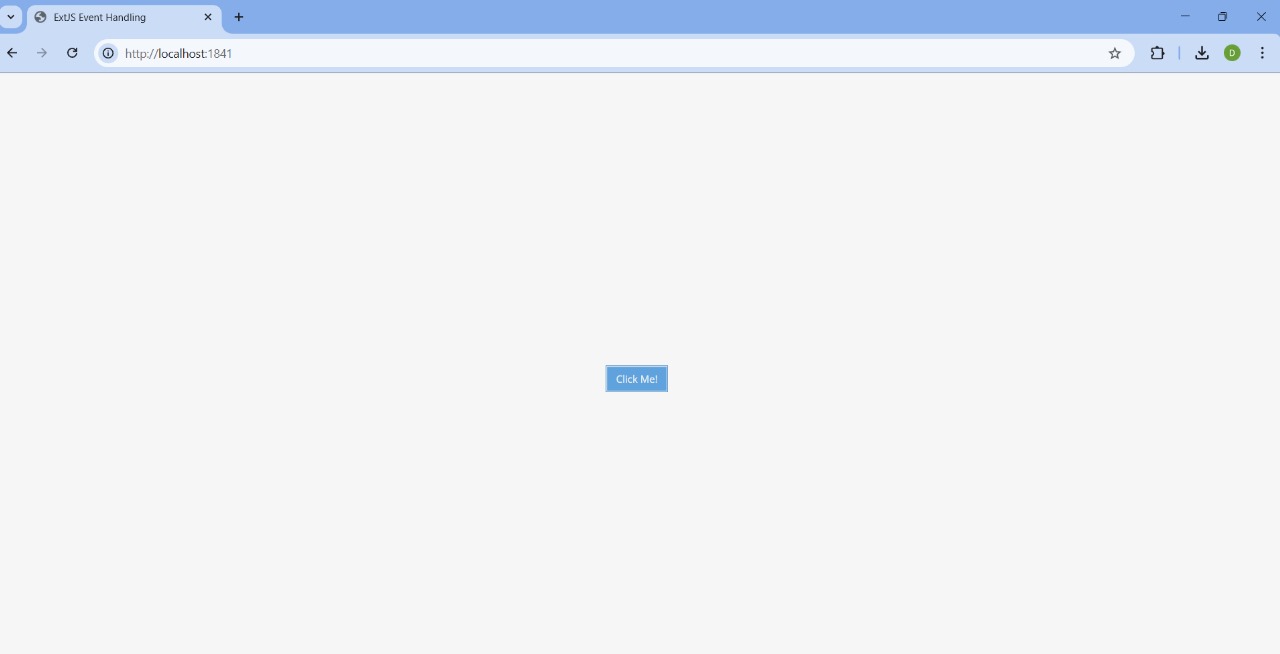






**OUTPUT:**

Before clicking button



After clicking button

