Ext – JS – Components :

Made By :

Prajwal Gajanan Ponarkar
Nikhil Aryan Kohli
Diya Jain
Elvina Francy H
Pooja D

Ext JS Overview and Key Features Explained :

What is ext js?

Ext JS is a comprehensive JavaScript framework for building interactive web applications. Developed by Sencha, it is particularly well-suited for creating data-intensive, cross-platform web applications. Ext JS provides a rich set of UI components, such as grids, trees, forms, and charts, which can be used to build complex user interfaces with minimal effort.

Key Features of Ext JS:

- **Rich UI Components:** Ext JS offers a wide range of pre-built, customizable UI components like grids, forms, trees, and charts, which are highly optimized for performance.

- **Cross-Browser Compatibility:** It ensures consistent behavior across different browsers, including older versions.

- **MVC/MVVM Architecture:** Ext JS supports the Model-View-Controller (MVC) and Model-View-ViewModel (MVVM) architectural patterns, making it easier to organize and maintain large-scale applications.

- **Data Package:** It includes a robust data package for managing data models, stores, and proxies, which simplifies data binding and synchronization with backend services.

- **Theming and Styling:** Ext JS provides a theming system that allows developers to customize the look and feel of their applications using Sass.

- **Responsive Design:** The framework supports responsive design, enabling applications to adapt to different screen sizes and devices.

- **Integration with Other Libraries:** Ext JS can be integrated with other JavaScript libraries and frameworks, such as React and Angular, for enhanced functionality.

- **Comprehensive Documentation:** Ext JS is known for its extensive documentation, which includes guides, examples, and API references.

Use Cases:

- **Enterprise Applications:** Ext JS is often used for building enterprise-level applications that require complex data handling and rich user interfaces.

- **Single-Page Applications (SPAs):** It is well-suited for SPAs where a seamless user experience is critical.

- **Data Visualization:** With its powerful charting and graphing capabilities, Ext JS is ideal for applications that require extensive data visualization.

- **Example:**

Here's a simple example of an Ext JS application that creates a window with a button:

Javascript :

```
Ext.application({
    name: 'MyApp',
    launch: function() {
        Ext.create('Ext.window.Window', {
```

```
        title: 'Hello Ext JS',
        height: 200,
        width: 400,
        layout: 'fit',
        items: {
            xtype: 'button',
            text: 'Click Me',
            handler: function() {
                Ext.Msg.alert('Message', 'Button clicked!');
            }
        }
    }).show();
  }
});
```

Licensing:

Ext JS is a commercial product, and you need to purchase a license to use it in your projects. Sencha offers different licensing options, including developer licenses and enterprise licenses.

Alternatives:

- **React:** A popular open-source library for building user interfaces.
- **Angular:** A full-fledged framework for building web applications.
- **Vue.js:** A progressive JavaScript framework for building UIs and single-page applications.

Ext JS is a powerful tool for developers who need to build complex, data-driven web applications with rich user interfaces. However, its commercial nature and steep learning curve may be considerations when choosing a framework for your project.

Ext JS is a powerful JavaScript framework that provides a rich set of built-in components for building modern, data-intensive web applications. Below is a detailed explanation of its components, an overview of built-in components, and a hands-on guide to building and implementing simple components in a project.

Overview of Ext JS Components :

Ext JS components are the building blocks of the framework. They are reusable UI elements that can be customized and combined to create complex user interfaces. Components are divided into several categories:

Categories of Built-in Components :

Containers:

Used to hold and organize other components.

Examples: Panel, Window, TabPanel, Viewport, FormPanel.

Forms:

Used for data input and validation.

Examples: TextField, ComboBox, DateField, Checkbox, RadioButton.

Grids:

Used to display and manipulate tabular data.

Examples: Grid, PagingToolbar, RowExpander.

Trees:

Used to display hierarchical data.

Examples: TreePanel, TreeGrid.

Charts:

Used for data visualization.

Examples: LineChart, BarChart, PieChart.

Toolbars and Menus:

Used for navigation and actions.

Examples: Toolbar, Menu, Button.

Layouts:

Used to define the arrangement of components within containers.

Examples: BorderLayout, HBox, VBox, CardLayout.

Utilities:

Helper components for common tasks.

Examples: DataView, MessageBox, ProgressBar.

2. Hands-On: Building Simple Components

Let’s build a simple Ext JS application step by step.

Step 1: Set Up the Environment

Download Ext JS from the Sencha website.

Include the Ext JS library in your project:

html

Copy

```
<link rel="stylesheet" type="text/css" href="path/to/extjs/resources/css/ext-all.css">
<script type="text/javascript" src="path/to/extjs/ext-all.js"></script>
```

Run HTML

Step 2: Create a Simple Component

Let’s create a Panel with a Button inside it.

javascript

Copy

```
Ext.onReady(function() {
    Ext.create('Ext.Panel', {
        title: 'My First Panel',
        width: 300,
        height: 200,
        renderTo: Ext.getBody(), // Render to the body of the page
        items: [
            {
                xtype: 'button', // Use the button component
                text: 'Click Me',
                handler: function() {
```

```
                Ext.Msg.alert('Message', 'Button clicked!');
            }
        }
    ]
  });
});
```

Step 3: Add a Grid Component

Let’s add a Grid to display tabular data.

```
javascript
Copy
Ext.onReady(function() {
  // Define a model
  Ext.define('User', {
    extend: 'Ext.data.Model',
    fields: ['name', 'email', 'phone']
  });

  // Create a store (data source)
  var store = Ext.create('Ext.data.Store', {
    model: 'User',
    data: [
      { name: 'John Doe', email: 'john@example.com', phone: '123-456-7890'
},
      { name: 'Jane Smith', email: 'jane@example.com', phone: '987-654-
3210' }
    ]
```

```
    });

    // Create a grid
    Ext.create('Ext.grid.Panel', {
        title: 'User List',
        store: store,
        columns: [
            { text: 'Name', dataIndex: 'name', flex: 1 },
            { text: 'Email', dataIndex: 'email', flex: 1 },
            { text: 'Phone', dataIndex: 'phone', flex: 1 }
        ],
        width: 600,
        height: 300,
        renderTo: Ext.getBody()
    });
});
```

3. Implementing Components in a Project

To implement Ext JS components in a real-world project, follow these steps:

Step 1: Define the Application Structure

Use the MVC/MVVM architecture to organize your code:

Model: Define data models.

View: Define UI components.

Controller: Handle business logic and interactions.

Step 2: Create a View

Define a view (e.g., a Panel with a Grid).

javascript

Copy

```
Ext.define('MyApp.view.UserGrid', {
    extend: 'Ext.grid.Panel',
    xtype: 'usergrid',
    title: 'User List',
    store: 'Users',
    columns: [
        { text: 'Name', dataIndex: 'name', flex: 1 },
        { text: 'Email', dataIndex: 'email', flex: 1 },
        { text: 'Phone', dataIndex: 'phone', flex: 1 }
    ]
});
```

Step 3: Create a Store

Define a store to manage data.

javascript

Copy

```
Ext.define('MyApp.store.Users', {
    extend: 'Ext.data.Store',
    model: 'MyApp.model.User',
    data: [
        { name: 'John Doe', email: 'john@example.com', phone: '123-456-7890' },
```

```
        { name: 'Jane Smith', email: 'jane@example.com', phone: '987-654-3210' }
    ]
});
```

Step 4: Create a Controller

Define a controller to handle interactions.

javascript

Copy

```
Ext.define('MyApp.controller.UserController', {
    extend: 'Ext.app.Controller',
    views: ['UserGrid'],
    init: function() {
        this.control({
            'usergrid button': {
                click: this.onButtonClick
            }
        });
    },
    onButtonClick: function(button) {
        Ext.Msg.alert('Info', 'Button clicked in ' + button.getText());
    }
});
```

Step 5: Launch the Application

Define the application and launch it.

javascript

Copy

```
Ext.application({
    name: 'MyApp',
    controllers: ['UserController'],
    stores: ['Users'],
    launch: function() {
        Ext.create('Ext.container.Viewport', {
            layout: 'fit',
            items: [
                { xtype: 'usergrid' }
            ]
        });
    }
});
```

4. Best Practices

Use MVC/MVVM: Organize your code into models, views, and controllers for better maintainability.

Lazy Loading: Load components and data only when needed to improve performance.

Theming: Use Sass to create custom themes for your application.

Testing: Write unit tests for your components and controllers.

By following this guide, you can start building Ext JS applications with its rich set of components and implement them in real-world projects.

Include the Ext JS library in your project:

how to do these step ?

Including the Ext JS library in your project is a crucial step to start building applications with the framework. Below is a step-by-step guide to help you include Ext JS in your project:

Step 1: Download Ext JS

Visit the Sencha website.

Sign up or log in to your Sencha account.

Download the Ext JS framework. You can choose either the Trial Version (for evaluation) or the Commercial Version (if you have a license).

Step 2: Extract the Ext JS Package

After downloading, extract the ZIP file to a folder in your project directory.

Example: Extract to C:\Projects\MyApp\extjs.

The extracted folder will contain the following:

build/: Minified and debug versions of the Ext JS library.

examples/: Sample applications and code snippets.

packages/: Additional packages and themes.

resources/: CSS files, images, and other assets.

src/: Source code (for debugging purposes).

welcome/: Welcome page and documentation.

Step 3: Include Ext JS in Your Project

To include Ext JS in your project, you need to link the Ext JS JavaScript file and CSS file in your HTML file.

Example: Basic HTML File

html

Copy

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Ext JS App</title>
    <!-- Include Ext JS CSS -->
    <link rel="stylesheet" type="text/css" href="path/to/extjs/resources/css/ext-all.css">
</head>
<body>
    <!-- Placeholder for rendering Ext JS components -->
    <div id="app"></div>
```

```
    <!-- Include Ext JS Library -->
    <script type="text/javascript" src="path/to/extjs/ext-all.js"></script>

    <!-- Your Application Script -->
    <script type="text/javascript">
        Ext.onReady(function() {
            Ext.create('Ext.Panel', {
                title: 'Hello Ext JS',
                width: 300,
                height: 200,
                renderTo: 'app',
                html: 'Welcome to Ext JS!'
            });
        });
    </script>
</body>
</html>
```

Run HTML

Explanation:

CSS File:

Include the Ext JS CSS file to apply the default styling to components.

Example: <link rel="stylesheet" type="text/css" href="path/to/extjs/resources/css/ext-all.css">

JavaScript File:

Include the Ext JS library file to enable the framework's functionality.

Example: <script type="text/javascript" src="path/to/extjs/ext-all.js"></script>

Application Script:

Write your Ext JS code inside a <script> tag or in a separate JavaScript file.

Step 4: Verify the Setup

Open your HTML file in a web browser.

If everything is set up correctly, you should see a panel with the title "Hello Ext JS" and the text "Welcome to Ext JS!".

Step 5: Organize Your Project

For larger projects, organize your files into folders:

extjs/: Ext JS framework files.

app/: Your application code (models, views, controllers, etc.).

index.html: Main HTML file.

Example Project Structure:

Copy

```
MyApp/
├── extjs/              # Ext JS framework
├── app/                # Your application code
│   ├── model/
│   ├── view/
│   ├── controller/
│   └── store/
├── resources/          # Custom CSS, images, etc.
└── index.html          # Main HTML file
```