EXT JS

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

ExtJS stands for Extended JavaScript. It is a JavaScript framework and a product of Sencha, based on YUI (Yahoo User Interface). It is basically a desktop application development platform with modern UI. This tutorial gives a complete understanding of Ext JS. This reference will take you through simple and practical approaches while learning Ext JS.

Audience

This tutorial has been prepared for beginners to help them understand the concepts of ExtJS to build dynamic web UI.

Prerequisites

For this tutorial, the reader should have prior knowledge of HTML, CSS, and JavaScript coding. It would be helpful if the reader knows the concepts of object-oriented programming and has a general idea on creating web applications.

Execute ExtJS Online

For most of the examples given in this tutorial you will find a **Try it** option. Make use of this option to execute your ExtJS programs on the spot and enjoy your learning.

Try the following example using the **Try it** option available at the top right corner of the following sample code box –

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function() {
         Ext.create('Ext.Panel', {
            renderTo: 'helloWorldPanel',
            height: 100,
            width: 200,
            title: 'Hello world',
            html: 'First Ext JS Hello World Program'
            });
         });
      </script>
   </head>
   <body>
      <div id="helloWorldPanel"></div>
   </body>
</html>
```

Copyright & Disclaimer

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

1. Ext.js – Overview

Ext JS is a popular JavaScript framework which provides rich UI for building web applications with cross-browser functionality. Ext JS is basically used for creating desktop applications. It supports all the modern browsers such as IE6+, FF, Chrome, Safari 6+, Opera 12+, etc. Whereas another product of Sencha, Sencha Touch is used for mobile applications.

Ext JS is based on MVC/MVVM architecture. The latest version of Ext JS 6 is a single platform, which can be used for both desktop and mobile application without having different code for different platform.

History

Ext JS 1.1

The first version of Ext JS was developed by Jack Slocum in 2006. It was a set of utility classes, which is an extension of YUI. He named the library as YUI-ext.

Ext JS 2.0

Ext JS version 2.0 was released in 2007. This version had a new API documentation for desktop application with limited features. This version doesn't have backward compatibility with previous version of Ext JS.

Ext JS 3.0

Ext JS version 3.0 was released in 2009. This version added new features as chart and list view but at the cost of speed. It had backward compatibility with version 2.0.

Ext JS 4.0

After the release of Ext JS 3, the developers of Ext JS had the major challenge of ramping up the speed. Ext JS version 4.0 was released in 2011. It had the complete revised structure, which was followed by MVC architecture and a speedy application.

Ext JS 5.0

Ext JS version 5.0 was released in 2014. The major change in this release was to change the MVC architecture to MVVM architecture. It includes the ability to build desktop apps on touch-enabled devices, two-way data binding, responsive layouts, and many more features.

Ext JS 6.0

Ext JS 6 merges the Ext JS (for desktop application) and Sencha Touch (for mobile application) framework.

Features

Following are the highlighted features of Ext JS.

- Customizable UI widgets with collection of rich UI such as grids, pivot grids, forms, charts, trees.
- Code compatibility of new versions with the older one.
- A flexible layout manager helps to organize the display of data and content across multiple browsers, devices, and screen sizes.
- Advance data package decouples the UI widgets from the data layer. The data package allows client-side collection of data using highly functional models that enable features such as sorting and filtering.
- It is protocol agnostic, and can access data from any back-end source.
- Customizable Themes Ext JS widgets are available in multiple out-of-the-box themes that are consistent across platforms.

Benefits

Sencha Ext JS is the leading standard for business-grade web application development. Ext JS provides the tools necessary to build robust applications for desktop and tablets.

- Streamlines cross-platform development across desktops, tablets, and smartphones - for both modern and legacy browsers.
- Increases the productivity of development teams by integrating into enterprise development environments via IDE plugins.
- Reduces the cost of web application development.
- Empowers the teams to create apps with a compelling user experience.
- Offers a set of widgets to easily make a powerful UI.
- Follows MVC architecture, hence the code is highly readable.

Limitations

- The size of the library is large, around 500 KB, which makes initial loading time more and makes application slow.
- HTML is full of tags that makes it complex and difficult to debug.
- According to general public license policy, it is free for open source applications but paid for commercial applications.
- Sometimes for loading even simple things require few lines of coding, which is simpler in plain html or JQuery.

- Need quite experienced developer for developing Ext JS applications.

Tools

Following are the tools provided by Sencha used for Ext JS application development mainly at the production level.

Sencha CMD

Sencha CMD is a tool which provides the features of Ext JS code minification, scaffolding, and production build generation.

Sencha IDE Plugins

Sencha IDE plugins integrates Sencha frameworks into IntelliJ, WebStorm IDEs, which helps in improving the developer’s productivity by providing features such as code completion, code inspection, code navigation, code generation, code refactoring, template creation, spell-checking, etc.

Sencha Inspector

Sencha Inspector is a debugging tool which helps the debugger to debug any issue while development.

2. Ext.js – Environment Setup

Try it Option Online

We have set up ExtJS Programming environment online, so that you can compile and execute all the available examples online. It gives you confidence in what you are reading and enables you to verify the programs with different options. Feel free to modify any example and execute it online.

Try the following example using **Try it** option available at the top right corner of the following sample code box.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function() {
         Ext.create('Ext.Panel', {
            renderTo: 'helloWorldPanel',
            height: 100,
            width: 200,
            title: 'Hello world',
            html: 'First Ext JS Hello World Program'
            });
         });
      </script>
   </head>
   <body>
      <div id="helloWorldPanel"></div>
   </body>
</html>
```

For most of the examples given in this tutorial, you will find a Try It option in our website code sections at the top right corner that will take you to the online compiler. Make use of it and enjoy your learning.

Local Environment Setup

This section guides you on how to download and set up Ext JS on your machine. Please follow the steps to set up the environment.

Downloading Library Files

Download the trial version of Ext JS library files from Sencha https://www.sencha.com. You will get the trial version from the site on your registered mail id, which will be a zipped folder named ext-6.0.1-trial.

Unzip the folder and you will find various JavaScript and CSS files, which you will include in our application. We will mostly include the following files:

JavaScript Files: JS file which you can find under the folder \ext-6.0.1-trial\ext-6.0.1\build are:

File	Description
ext.js	This is the core file which contains all the functionalities to run the application.
ext-all.js	This file contains all the code minified with no comments in the file.
ext-all-debug.js	This is the unminified version of ext-all.js for debugging purpose.
ext-all-dev.js	This file is also unminified and is used for development purpose as it contains all the comments and console logs to check any errors/issue.
ext-all.js	This file is used for production purpose mostly as it is much smaller than any other.

You can add these files to your projects JS folder or you can provide a direct path where the file resides in your system.

CSS Files: There are number of theme-based files, which you can find under folder \ext-6.0.1-trial\ext-6.0.1\build\classic\theme-classic\resources\theme-classic-all.css

- If you are going to use desktop application, then you can use classic themes under folder \ext-6.0.1-trial\ext-6.0.1\build\classic
- If we are going to use mobile application, then you can use modern themes which can be found under folder \ext-6.0.1-trial\ext-6.0.1\build\modern

The following library files will be added in an Ext JS application.

```
<html>
   <head>
      <link rel = "stylesheet" type ="text/css" href= "..\ext-6.0.1-trial\ext-
6.0.1\build\classic\theme-classic\resources\theme-classic-all.css" />
      <script type ="text/javascript" src = "..\ext-6.0.1-trial\ext-
6.0.1\build\ext-all.js" > </script>
      <script type ="text/javascript" src = "app.js" > </script>
   </head>
</html>
```

You will keep ExtJS application code in app.js file.

CDN Setup

CDN is content delivery network with which you do not need to download the Ext JS library files, instead you can directly add CDN link for ExtJS to your program as follows:

```
<html>
   <head>
      <link rel = "stylesheet" type ="text/css" href=
"https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
crisp/resources/theme-crisp-all.css" / >
      <script type ="text/javascript" src =
"https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js" > </script>
      <script type ="text/javascript" src = "app.js" > </script>
   </head>
</html>
```

Popular Editors

As it is a JavaScript framework, which is used for developing web applications, in our project we will have HTML, JS files. To write our Ext JS programs, we will need a text editor. There are even multiple IDEs available in the market. But for now, we can consider one of the following:

- **Notepad:** On Windows machine, you can use any simple text editor such as Notepad (Recommended for this tutorial), Notepad++, sublime.

- **Eclipse:** It is an IDE developed by the eclipse open-source community and can be downloaded from http://www.eclipse.org/

Browser

Ext JS supports cross-browser compatibility, it supports all major browsers such as:

- IE 6 and above

- Firefox 3.6 and above
- Chrome10 and above
- Safari 4 and above
- Opera 11 and above

You can use any browser for running Ext JS application.

3. Ext.js – Naming Convention

Naming convention is a set of rule to be followed for identifiers. It makes the code more readable and understandable to other programmers as well.

Naming convention in Ext JS follows the standard JavaScript convention, which is not mandatory but a good practice to follow. It should follow the camel case syntax for naming the class, method, variable and properties.

If the name is combined with two words, the second word will start with an uppercase letter always. For example, doLayout(), StudentForm, firstName, etc.

Name	Convention
Class Name	It should start with an uppercase letter, followed by camel case. For example, StudentClass
Method Name	It should start with a lowercase letter, followed by camel case. For example, doLayout()
Variable Name	It should start with a lowercase letter, followed by camel case. For example, firstName
Constant Name	It should be in uppercase only. For example, COUNT, MAX_VALUE
Property Name	It should start with a lowercase letter, followed by camel case. For example, enableColumnResize = true

4. Ext.js – Architecture

Ext JS follows MVC/ MVVM architecture.

MVC – Model View Controller architecture (version 4)

MVVM – Model View Viewmodel (version 5)

This architecture is not mandatory for the program, however, it is a best practice to follow this structure to make your code highly maintainable and organized.

Project Structure with Ext JS App

```
----------src
----------resources
-------------------CSS files
-------------------Images
----------JavaScript
--------------------App Folder
-------------------------------Controller
------------------------------------Contoller.js
-------------------------------Model
------------------------------------Model.js
-------------------------------Store
------------------------------------Store.js
-------------------------------View
------------------------------------View.js
-------------------------------Utils
------------------------------------Utils.js
--------------------------------app.js
-----------HTML files
```

Ext JS app folder will reside in JavaScript folder of your project.

The App will contain controller, view, model, store, and utility files with app.js.

app.js: The main file from where the flow of program will start, which should be included in the main HTML file using <script> tag. App calls the controller of application for the rest of the functionality.

Controller.js: It is the controller file of Ext JS MVC architecture. This contains all the control of the application, the events listeners, and most of the functionality of the code. It has the path defined for all the other files used in that application such as store, view, model, require, mixins.

View.js: It contains the interface part of the application, which shows up to the user. Ext JS uses various UI rich views, which can be extended and customized here according to the requirement.

Store.js: It contains the data locally cached which is to be rendered on the view with the help of model objects. Store fetches the data using proxies which has the path defined for services to fetch the backend data.

Model.js: It contains the objects which binds the store data to view. It has the mapping of backend data objects to the view dataIndex. The data is fetched with the help of store.

Utils.js: It is not included in MVC architecture but a best practice to use to make the code clean, less complex, and more readable. We can write methods in this file and call them in the controller or the view renderer wherever required. It is helpful for code reusability purpose as well.

In MVVM architecture, the controller is replaced by ViewModel.

ViewModel: It basically mediates the changes between view and model. It binds the data from the model to the view. At the same time, it does not have any direct interaction with the view. It has only knowledge of the model.

How It Works

For example, if we are using one model object at two-three places in UI. If we change the value at one place of UI, we can see without even saving that change. The value of model changes and so gets reflected in all the places in the UI, wherever the model is used.

It makes the developers’ effort much lesser and easier as no extra coding is required for binding data.

5. Ext.js – First Program

This chapter lists down the steps to write the first Hello World program in Ext JS.

Step 1

Create an index.htm page in the editor of our choice. Include the required library files in the head section of html page as follows.

index.htm

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function() {
         Ext.create('Ext.Panel', {
            renderTo: 'helloWorldPanel',
            height: 200,
            width: 600,
            title: 'Hello world',
            html: 'First Ext JS Hello World Program'
            });
         });
      </script>
   </head>
   <body>
      <div id="helloWorldPanel" />
   </body>
</html>
```

Explanation

- Ext.onReady() method will be called once the Ext JS is ready to render the Ext JS elements.

- Ext.create() method is used to create an object in Ext JS. Here we are creating an object of simple panel class Ext.Panel.

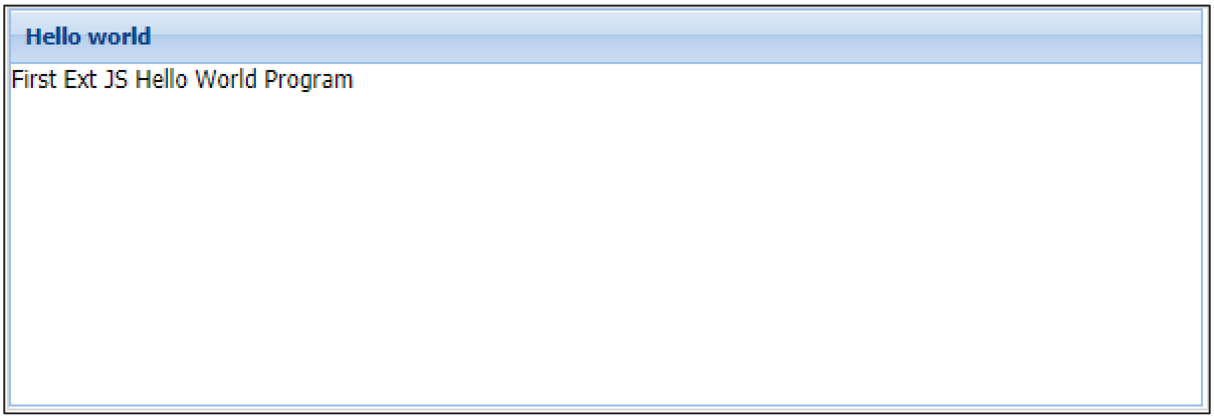
- Ext.Panel is the predefined class in Ext JS for creating a panel.
- Every Ext JS class has different properties to perform some basic functionalities.

Ext.Panel class has various properties such as:

- **renderTo** is the element where this panel has to render. 'helloWorldPanel' is the div id in Index.html file.
- **Height** and **width** properties are for customizing the size of the panel.
- **Title** property is to provide the title to the panel.
- **Html** property is the html content to be shown in the panel.

Step 2

Open the index.htm file in a standard browser and you will get the following output on the browser.



Hello world

First Ext JS Hello World Program

6. Ext.js – Class System

Ext JS is a JavaScript framework having functionalities of object oriented programming. Ext is the namespace, which encapsulates all the classes in Ext JS.

Defining a Class in Ext JS

Ext provides more than 300 classes, which we can use for various functionalities.

Ext.define() is used for defining the classes in Ext JS.

Syntax

```
Ext.define(class name, class members/properties, callback function);
```

Class name is the name of the class according to app structure. For example, appName.folderName.ClassName studentApp.view.StudentView.

Class properties/members defines the behavior of class.

Callback function is optional. It is called when the class has loaded properly.

Example of Ext JS Class Definition

```
Ext.define(studentApp.view.StudentDeatilsGrid, {
   extend : 'Ext.grid.GridPanel',
   id : 'studentsDetailsGrid',
   store : 'StudentsDetailsGridStore',
   renderTo : 'studentsDetailsRenderDiv',
   layout : 'fit',
   columns : [{
      text : 'Student Name',
      dataIndex : 'studentName'
   },{
      text : 'ID',
      dataIndex : 'studentId'
   },{

 text : 'Department',
      dataIndex : 'department'
   }]
});
```

Creating Objects

As like other OOPS based languages, we can create objects in Ext JS as well.

Following are the different ways of creating objects in Ext JS.

Using new keyword

```
var studentObject = new student();
studentObject.getStudentName();
```

Using Ext.create()

```
Ext.create('Ext.Panel', {
   renderTo : 'helloWorldPanel',
   height : 100,
   width : 100,
   title : 'Hello world',
   html :   'First Ext JS Hello World Program'
});
```

Inheritance in Ext JS

Inheritance is the principle of using functionality defined in class A into class B.

In Ext JS, inheritance can be done using two methods -

Ext.extend

```
Ext.define(studentApp.view.StudentDetailsGrid, {
   extend : 'Ext.grid.GridPanel',
   ...
});
```

Here, our custom class StudentDetailsGrid is using the basic features of Ext JS class GridPanel.

Using Mixins

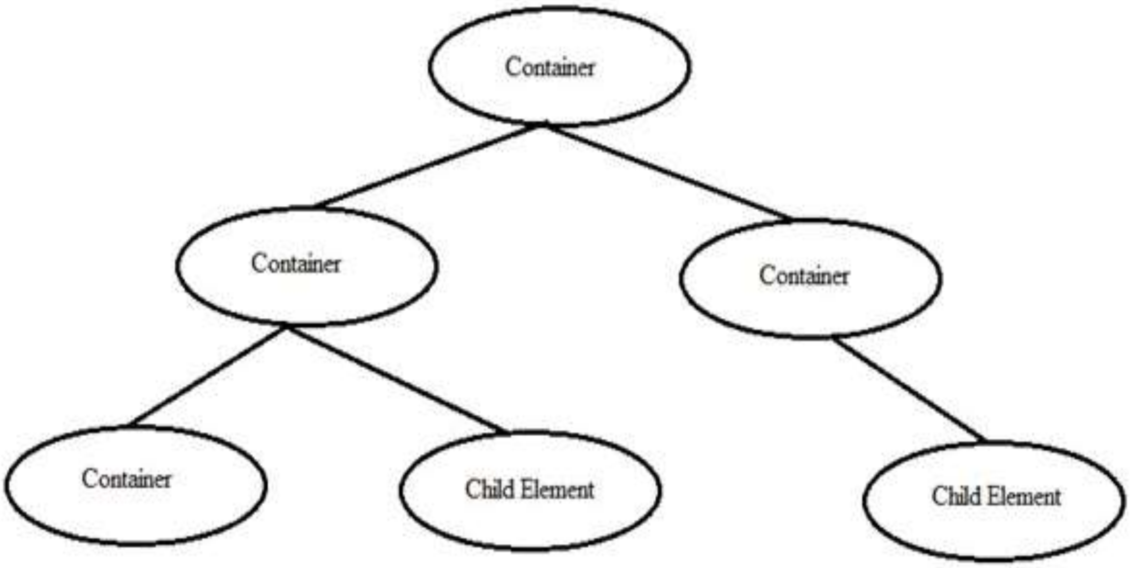
Mixins is a different way of using class A in class B without extend.

```
mixins : {
   commons : 'DepartmentApp.utils.DepartmentUtils'
},
```

Mixins are added in the controller where we declare all the other classes such as store, view, etc. In this way, we can call DepartmentUtils class and use its functions in the controller or in this application.

7. Ext.js - Containers

Container in Ext JS is the component where we can add other container or child components. These containers can have multiple layout to arrange the components in the containers. We can add or remove the components from the container and from its child elements. Ext.container.Container is the base class for all the containers in Ext JS.



Sr. No.	Description
1	**Components inside Container** This example shows how to define the components inside a container
2	**Container inside container** This example shows how to define a container inside a container with other components

Components Inside Container

Components inside container: We can have multiple components inside the container.

Syntax

Following is the simple syntax to use Components inside the container.

```
   var component1 = Ext.create('Ext.Component', {
      html:'First Component'
   });
   Ext.create('Ext.container.Container', {
      renderTo: Ext.getBody(),
      items: [component1]
   });
```

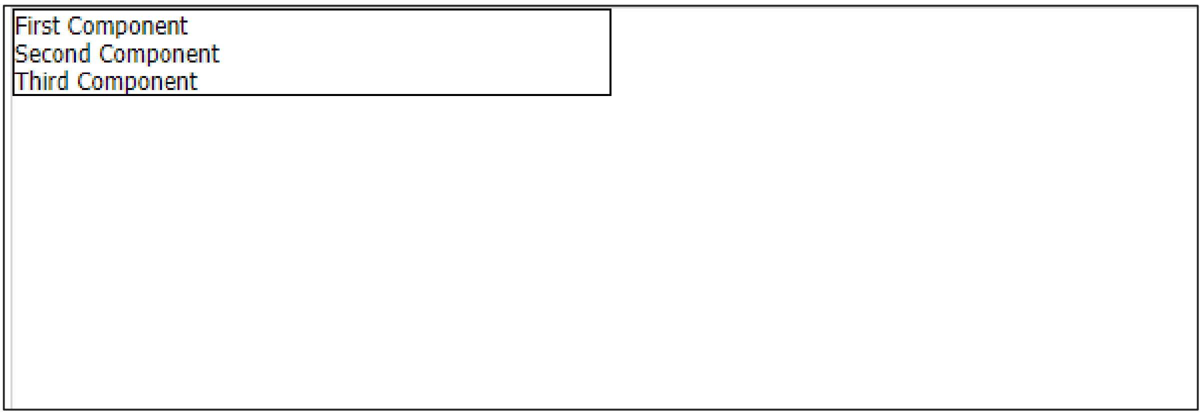
You can have components as an item inside a container.

Example

Following is a simple example showing the components inside the container.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function () {
         var component1 = Ext.create('Ext.Component', {
            html:'First Component'
         });

         var component2 = Ext.create('Ext.Component', {
            html: 'Second Component'
         });

         var component3 = Ext.create('Ext.Component', {
            html: 'Third Component'
         });

         Ext.create('Ext.container.Container', {
            renderTo: Ext.getBody(),
            title: 'Container',
            border: 1,
            width: '50%',
            style: {borderStyle: 'solid', borderWidth: '2px' },
            items: [component1, component2,  component3]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



First Component
Second Component
Third Component

Container Inside Container

Container inside container: We can have containers inside other containers as a component for the parent container along with other components.

Syntax

Following is the simple syntax to use Container inside the container.

```
    var container = Ext.create('Ext.container.Container', {
       items: [component3, component4]
    });
    Ext.create('Ext.container.Container', {
       renderTo: Ext.getBody(),
       items: [container]
    });
```

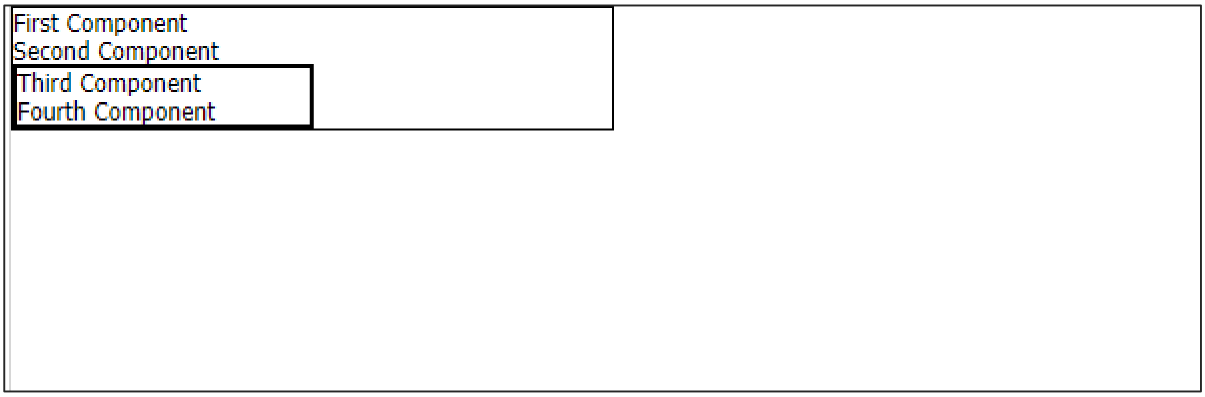
You can have a container as an item inside the other container.

Example

Following is a simple example showing a container inside a container.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
       Ext.onReady(function () {
         var component1 = Ext.create('Ext.Component', {
            html:'First Component'
         });

         var component2 = Ext.create('Ext.Component', {
            html: 'Second Component'
         });

         var component3 = Ext.create('Ext.Component', {
            html: 'Third Component'
         });

         var component4 = Ext.create('Ext.Component', {
            html: 'Fourth Component'
         });

         var container = Ext.create('Ext.container.Container', {
            style: {borderStyle: 'solid', borderWidth: '2px' },
            width: '50%',
            items: [component3, component4]
         });

         Ext.create('Ext.container.Container', {
            renderTo: Ext.getBody(),
            title: 'Container',
            border: 1,
            width: '50%',
            style: {borderStyle: 'solid', borderWidth: '2px' },
            items: [component1, component2,  container]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



```
First Component
Second Component
Third Component
Fourth Component
```

There are various type of containers Ext.panel.Panel, Ext.form.Panel, Ext.tab.Panel and Ext.container.Viewport are frequently used containers in Ext JS. Below are the example which shows how to use these containers.

Sr. No.	Type of Containers & Description
1	**Ext.panel.Panel** This example shows a Ext.panel.Panel container
2	**Ext.form.Panel** This example shows a Ext.form.Panel container
3	**Ext.tab.Panel** This example shows a Ext.tab.Panel container
4	**Ext.container.Viewport** This example shows a Ext.container.Viewport container

Ext.panel.Panel Container

Ext.panel.Panel: It is the basic container which allows to add items in a normal panel.

Syntax

Following is the simple syntax to create Ext.panel.Panel container.

```
Ext.create('Ext.panel.Panel', {
   items: [child1, child2]
   // this way we can add different child elements to the container as container items.
});
```

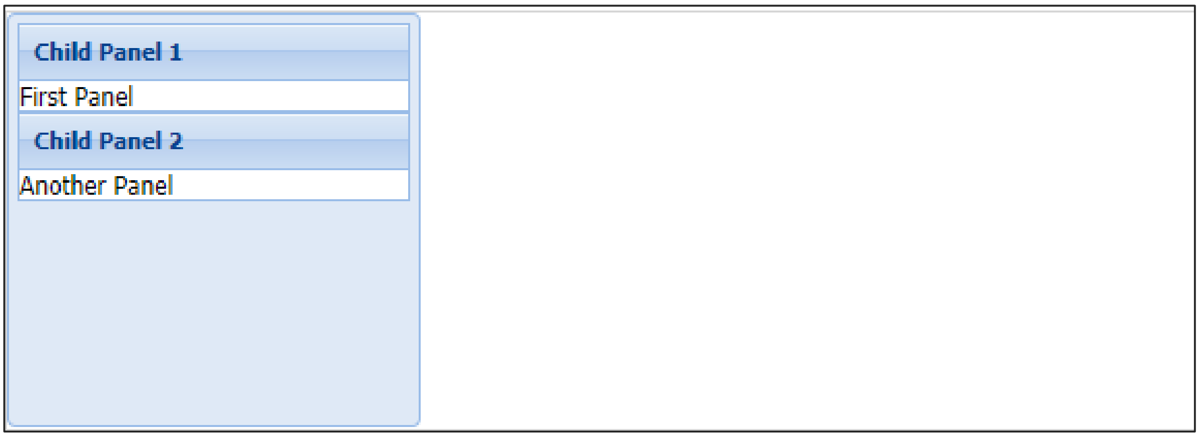
Example

Following is a simple example showing Ext.panel.Panel Container

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function () {
         var childPanel1 = Ext.create('Ext.Panel', {
             html: 'First Panel'
         });

         var childPanel2 = Ext.create('Ext.Panel', {
             html: 'Another Panel'
         });

         Ext.create('Ext.panel.Panel', {
         renderTo: Ext.getBody(),
            width: 100,
            height : 100,
            border : true,
            frame : true,
            items: [ childPanel1, childPanel2 ]
```

```
          });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Child Panel 1

First Panel

Child Panel 2

Another Panel

Ext.form.Panel Container

Ext.form.Panel: Form panel provides a standard container for forms. It is essentially a standard Ext.panel.Panel, which automatically creates a BasicForm for to manage any Ext.form.field.Field objects.

Syntax

Following is a simple syntax to create Ext.form.Panel container.

```
Ext.create('Ext.form.Panel', {
   items: [child1, child2]
   // this way we can add different child elements to the container as container items.
});
```

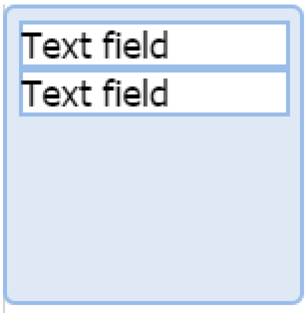
Example

Following is a simple example showing Ext.form.Panel container.

```
<!DOCTYPE html>
<html>
<head>
```

```
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />

   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>

   <script type="text/javascript">

      Ext.onReady(function () {

         var child1 = Ext.create('Ext.Panel',{

            html: 'Text field'

         });


         var child2 = Ext.create('Ext.Panel',{

            html: 'Text field'

         });

         Ext.create('Ext.form.Panel', {

            renderTo: Ext.getBody(),

            width: 100,

            height : 100,

            border : true,

            frame : true,

            layout: 'auto',// auto is one of the layout type.

            items: [child1, child2]

         });

      });

   </script>

</head>

<body>

</body>

</html>
```

The above program will produce the following result –



Ext.tab.Panel Container

Ext.tab.Panel: Tab panel is like a normal panel but has support for card tab panel layout.

Syntax

Following is a simple syntax to create Ext.tab.Panel container.

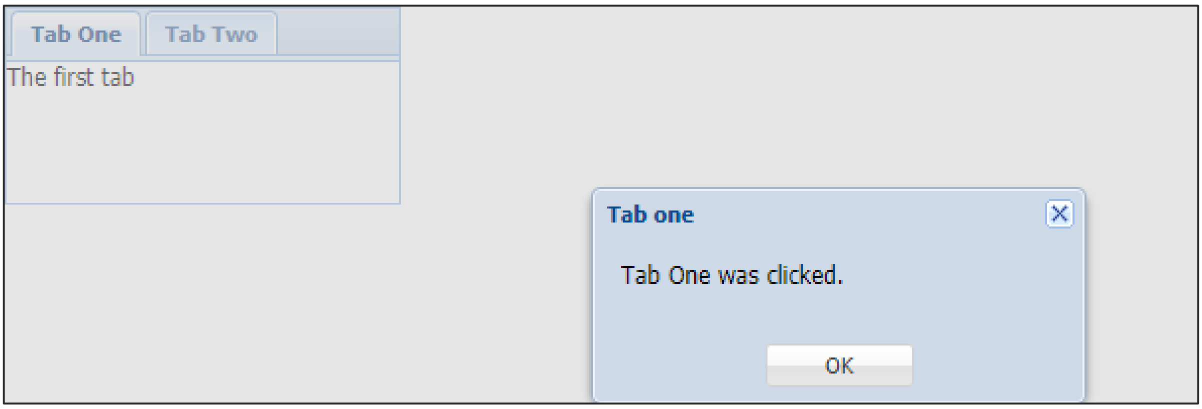
```
Ext.create('Ext.tab.Panel', {
   items: [child1, child2]
   // this way we can add different child elements to the container as container items.
});
```

Example

Following is a simple example showing Ext.tab.Panel container.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function () {
         Ext.create('Ext.tab.Panel', {
            renderTo: Ext.getBody(),
            height: 100,
            width: 200,
            items: [{
               xtype: 'panel',
               title: 'Tab One',
               html: 'The first tab',
               listeners: {
                  render: function() {
                     Ext.MessageBox.alert('Tab one', 'Tab One was clicked.');
                  }
               }
               },{
               // xtype for all Component configurations in a Container
               title: 'Tab Two',
               html: 'The second tab',
               listeners: {
                  render: function() {
                     Ext.MessageBox.alert('Tab two', 'Tab Two was clicked.');
                  }
               }
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Ext.container.Viewport Container

Ext.container.Viewport: Viewport is a container that automatically resizes itself to the size of the whole browser window. You can then add other ExtJS UI components and containers in it.

Syntax

Following is the simple syntax to create Ext.container.Viewport container.

```
Ext.create('Ext.container.Viewport', {
   items: [child1, child2]
   // this way we can add different child elements to the container as container items.
});
```

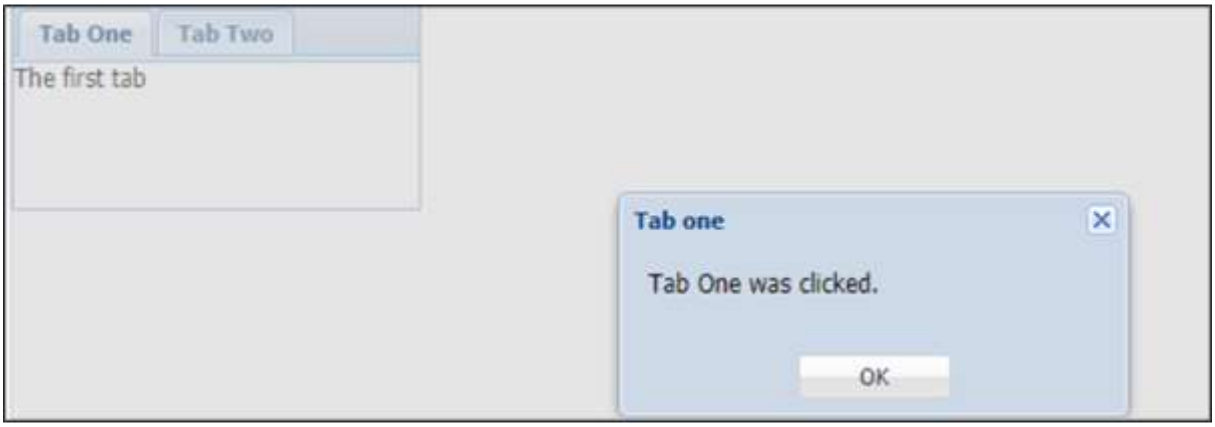
Example

Following is a simple example showing Ext.container.Viewport container.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function () {
         var childPanel1 = Ext.create('Ext.panel.Panel', {
             title: 'Child Panel 1',
             html: 'A Panel'
         });

         var childPanel2 = Ext.create('Ext.panel.Panel', {
             title: 'Child Panel 2',
             html: 'Another Panel'
         });

         Ext.create('Ext.container.Viewport', {
         renderTo: Ext.getBody(),
             items: [ childPanel1, childPanel2 ]
```

```
          });
       });
    </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



The first tab

Tab one

Tab One was clicked.

8. Ext.js – Layouts

Layout is the way the elements are arranged in a container. It can be horizontal, vertical, or any other way. Ext JS has a different layout defined in its library but we can always write custom layouts as well.

Sr. No.	Layout & Description
1	**Absolute** This layout allows to position the items using XY coordinates in the container.
2	**Accordion** This layout allows to place all the items in stack fashion (one on top of the other) inside the container.
3	**Anchor** This layout gives the privilege to the user to specify the size of each element with respect to the container size.
4	**Border** In this layout various panels are nested and separated by borders.
5	**Auto** This is the default layout that decides the layout of the elements based on the number of elements.
6	**Card(TabPanel)** This layout arranges different components in tab fashion. Tabs will be displayed on top of the container. Every time only one tab is visible and each tab is considered as a different component.
7	**Card(Wizard)** In this layout, every time the elements come for full container space. There is a bottom tool bar in the wizard for navigation.
8	**Column** This layout is to show multiple columns in the container. We can define a fixed or percentage width to the columns. The percentage width will be calculated based on the full size of the container.
9	**Fit** In this layout, the container is filled with a single panel. When there is no specific requirement related to the layout, then this layout is used.
10	**Table** As the name implies, this layout arranges the components in a container in the HTML table format.

11	**vBox** This layout allows the element to be distributed in a vertical manner. This is one of the most used layout.
12	**hBox** This layout allows the element to be distributed in a horizontal manner.

Absolute Layout

This layout allows to position the items using XY coordinates in the container.

Syntax

Following is the simple syntax to use for Absolute layout.

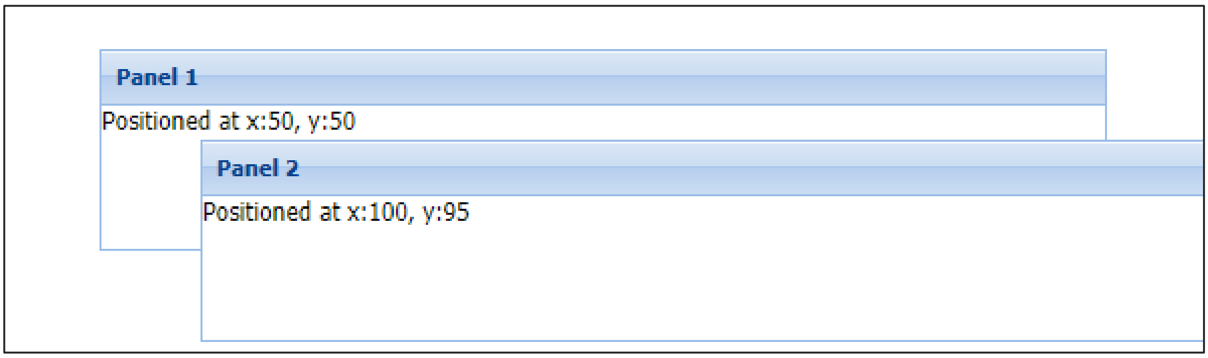
```
layout: 'absolute'
```

Example

Following is a simple example showing the usage of Absolute layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.container.Container', {
            renderTo: Ext.getBody(),
            layout: 'absolute' ,
            items: [{
               title: 'Panel 1',
               x: 50,
               y: 50,
               html: 'Positioned at x:50, y:50',
               width: 500,
               height: 100
            },{
               title: 'Panel 2',
               x: 100,
               y: 95,
               html: 'Positioned at x:100, y:95',
               width: 500,
               height: 100
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Accordion Layout

This layout allows to place all the items in stack fashion (one on top of the other) inside a container.

Syntax

Following is a simple syntax to use Accordion layout.

```
  layout: 'accordion'
```

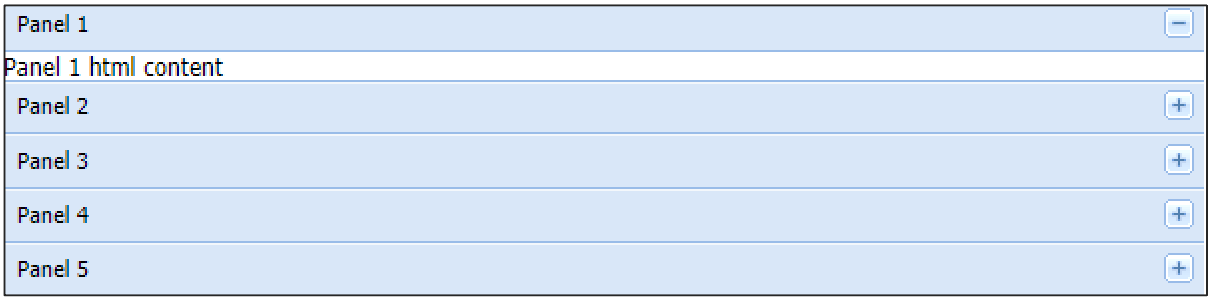
Example

Following is a simple example showing the usage of Accordion layout.

```
<!DOCTYPE html>
<html>
```

```
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.container.Container', {
            renderTo : Ext.getBody(),
            layout : 'accordion' ,
            width : 600,
            items : [{
               title : 'Panel 1',
               html : 'Panel 1 html content'
            },{
               title : 'Panel 2',
               html : 'Panel 2 html content'
            },{
               title : 'Panel 3',
               html : 'Panel 3 html content'
            },{
               title : 'Panel 4',
               html : 'Panel 4 html content'
            },{
               title : 'Panel 5',
               html : 'Panel 5 html content'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –


Panel 1 −
Panel 1 html content
Panel 2 +
Panel 3 +
Panel 4 +
Panel 5 +

Anchor Layout

This layout gives the privilege to the user to specify the size of each element with respect to the container size.

Syntax

DDFollowing is a simple syntax to use Anchor layout.

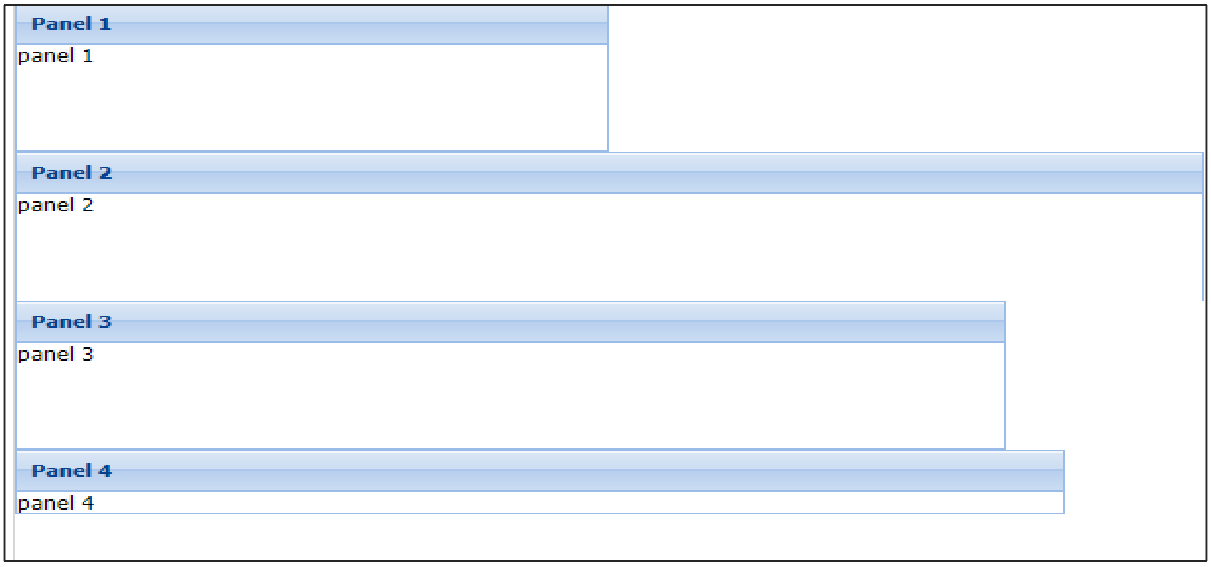
```
layout: 'anchor'
```

Example

Following is a simple example showing the usage of Anchor layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.container.Container', {
            renderTo : Ext.getBody(),
            layout : 'anchor' ,
            width : 600,
            items : [{
               title : 'Panel 1',
               html : 'panel 1',
               height : 100,
               anchor : '50%'
            },{
               title : 'Panel 2',
               html : 'panel 2',
               height : 100,
               anchor : '100%'
            },{
               title : 'Panel 3',
               html : 'panel 3',
               height : 100,
               anchor : '-100'
            },{
               title : 'Panel 4',
               html : 'panel 4',
               anchor : '-70, 500'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Border Layout

In this layout, various panels are nested and separated by borders.

Syntax

Following is a simple syntax to use Border layout.

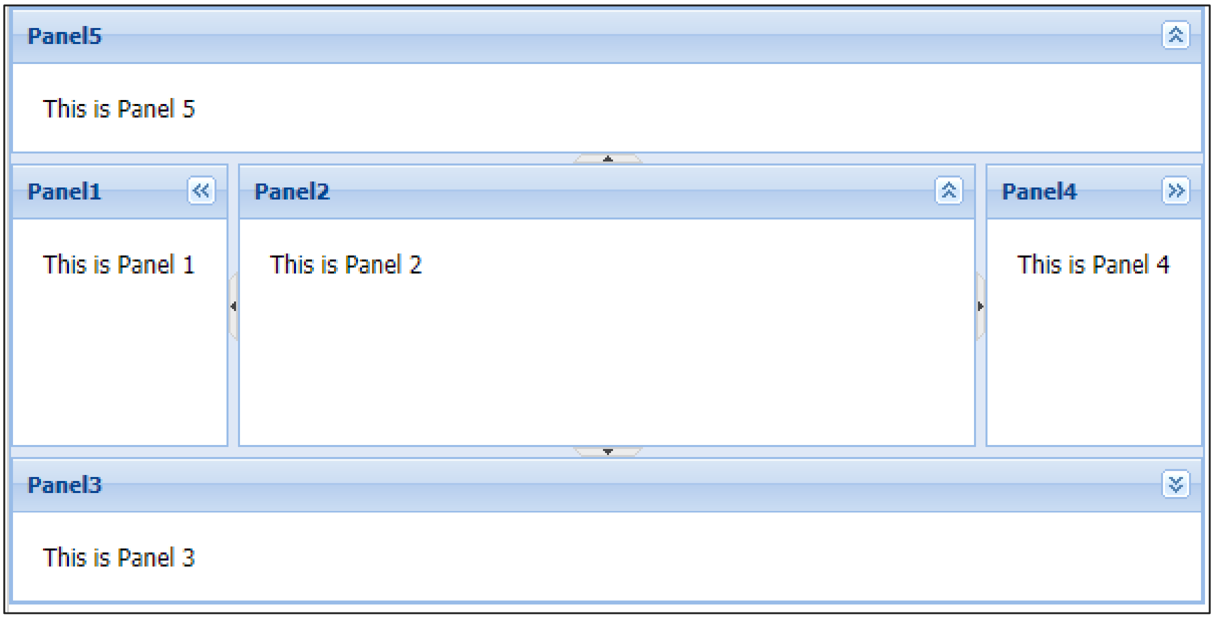
```
layout: 'border'
```

Example

Following is a simple example showing the usage of Border layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.panel.Panel', {
            renderTo: Ext.getBody(),
            height: 300,
            width: 600,
            layout:'border',
            defaults: {
               collapsible: true,
               split: true,
               bodyStyle: 'padding:15px'
            },
            items: [{
               title: 'Panel1',
               region:'west',
               html: 'This is Panel 1'
            },{
               title: 'Panel2',
               region:'center',
               html: 'This is Panel 2'
            },{
               title: 'Panel3',
               region: 'south',
               html: 'This is Panel 3'
            },{
               title: 'Panel4',
               region: 'east',
               html: 'This is Panel 4'
            },{
               title: 'Panel5',
               region: 'north',
               html: 'This is Panel 5'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Panel5

This is Panel 5

Panel1

This is Panel 1

Panel2

This is Panel 2

Panel4

This is Panel 4

Panel3

This is Panel 3

Auto Layout

This is the default layout that decides the layout of the elements based on the number of elements.

Syntax

Following is a simple syntax to use Auto layout.

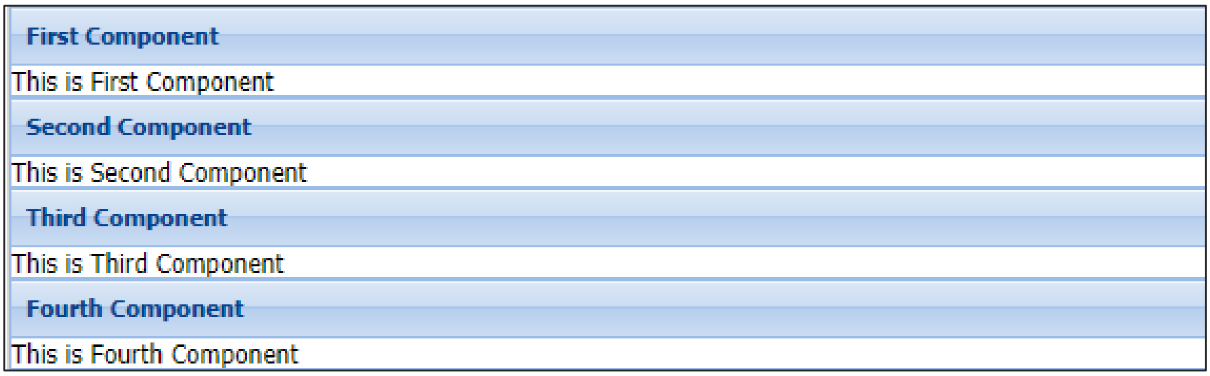
```
layout: 'auto'
```

Example

Following is a simple example showing the usage of Auto layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.container.Container', {
            renderTo : Ext.getBody(),
            layout : 'auto',
            width : 600,
            items : [{
               title: 'First Component',
               html : 'This is First Component'
            },{
               title: 'Second Component',
               html : 'This is Second Component'
            },{
               title: 'Third Component',
               html : 'This is Third Component'
            },{
               title: 'Fourth Component',
               html : 'This is Fourth Component'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



First Component

This is First Component

Second Component

This is Second Component

Third Component

This is Third Component

Fourth Component

This is Fourth Component

card_panel Layout

This layout allows to position the items using XY coordinates in the container.

Syntax

Following is a simple syntax to use card tab panel layout.

```
  layout: 'layout-cardtabs'
```

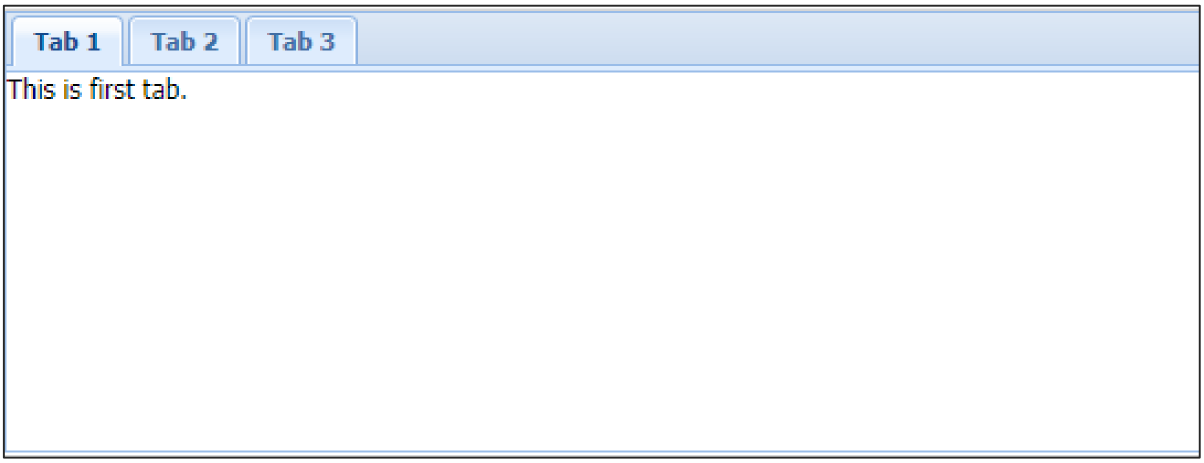
Example

Following is a simple example showing the usage of card tab panel layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
```

```
      Ext.onReady(function() {
         Ext.create('Ext.tab.Panel', {
            renderTo: Ext.getBody(),
            requires: ['Ext.layout.container.Card'],
            xtype: 'layout-cardtabs',
            width: 600,
            height: 200,
            items:[{
               title: 'Tab 1',
               html:   'This is first tab.'
            },{
               title: 'Tab 2',
               html: 'This is second tab.'
            },{
               title: 'Tab 3',
               html: 'This is third tab.'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



This is first tab.

card_wizard Layout

This layout allows to position the items using XY coordinates in the container.

Syntax

Following is a simple syntax to use card wizard layout.

```
layout: 'card'
```

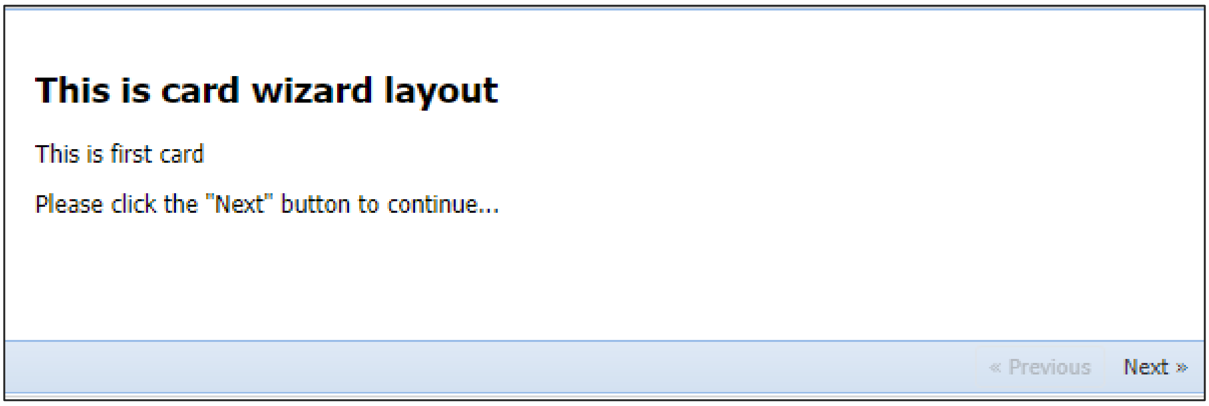
Example

Following is a simple example showing the usage of card wizard layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.panel.Panel', {
            renderTo: Ext.getBody(),
            requires: [
            'Ext.layout.container.Card'],
            layout: 'card',
            width: 600,
            height: 200,
            bodyPadding: 15,
            defaults: {
               border:false
            },
            defaultListenerScope: true,
            bbar: ['->',
            {
               itemId: 'card-prev',
               text: '« Previous',
               handler: 'showPrevious',
               disabled: true
            },{
               itemId: 'card-next',
               text: 'Next »',
               handler: 'showNext'
            }],

            items: [{
               id: 'card0',
               html: '<h2> This is card wizard layout </h2> <p> This is first 
card </p> <p> Please click the "Next" button to continue... </p>'
            },{
               id: 'card1',
               html: '<p> This is second card </p> <p> Please click the "Next" 
button for next card and "Previous" button for previous card... </p>'
            },{
               id: 'card2',
               html: '<p> This is final card </p> <p> Please click the 
"Previous" button for previous card... </p>'
            }],

            showNext: function () {
               this.cardPanelNavigation(1);
            },

            showPrevious: function (btn) {
               this.cardPanelNavigation(-1);
            },

            cardPanelNavigation: function (incr) {
               var me = this;
               var l = me.getLayout();
               var i = l.activeItem.id.split('card')[1];
               var next = parseInt(i, 10) + incr;
               l.setActiveItem(next);
               me.down('#card-prev').setDisabled(next===0);
               me.down('#card-next').setDisabled(next===2);
```

```
             }
          });
       });
    </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



This is card wizard layout

This is first card

Please click the "Next" button to continue...

Column Layout

This layout is to show multiple columns in the container. We can define a fixed or percentage width to the columns. The percentage width will be calculated based on the full size of the container.

Syntax

Following is a simple syntax to use the Column layout.

```
 layout: 'column'
```

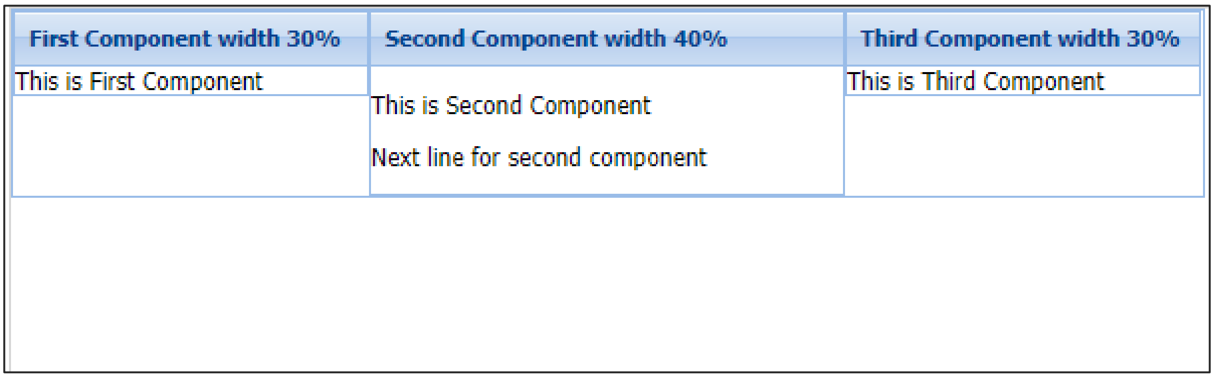
Example

Following is a simple example showing the usage of Column layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
```

```
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.panel.Panel', {
            renderTo : Ext.getBody(),
            layout : 'column' ,
              xtype: 'layout-column',
            requires: ['Ext.layout.container.Column'],
            width : 600,
            items: [{
               title : 'First Component width 30%',
               html : 'This is First Component',
               columnWidth : 0.30
            },{
               title : 'Second Component width 40%',
               html : '<p> This is Second Component </p> <p> Next line for
second component </p>',
               columnWidth : 0.40
            },{
               title : 'Third Component width 30%',
               html : 'This is Third Component' ,
               columnWidth : 0.30
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



First Component width 30%
This is First Component

Second Component width 40%
This is Second Component

Next line for second component

Third Component width 30%
This is Third Component

Fit Layout

In this layout, the container is filled with a single panel and when there is no specific requirement related to the layout then this layout is used.

Syntax

Following is a simple syntax to use the Fit layout.

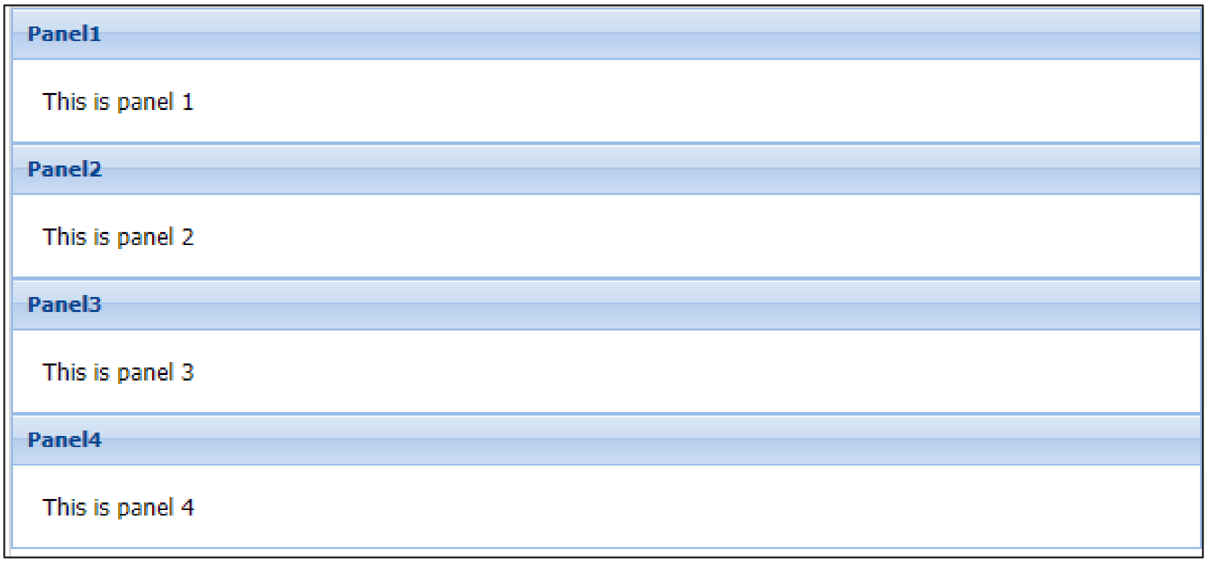
```
 layout: 'fit'
```

Example

Following is a simple example showing the usage of Fit layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.container.Container', {
            renderTo : Ext.getBody(),
            layout : {
               type :'fit'
            },
            width : 600,
            defaults: {
               bodyPadding: 15
            },
            items : [{
               title: 'Panel1',
               html : 'This is panel 1'
            },{
               title: 'Panel2',
               html : 'This is panel 2'
            },{
               title: 'Panel3',
               html : 'This is panel 3'
            },{
               title: 'Panel4',
               html : 'This is panel 4'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Panel1

This is panel 1

Panel2

This is panel 2

Panel3

This is panel 3

Panel4

This is panel 4

Table Layout

As the name implies, this layout arranges the components in a container in the HTML table format.

Syntax

Following is a simple syntax to use the Table layout.

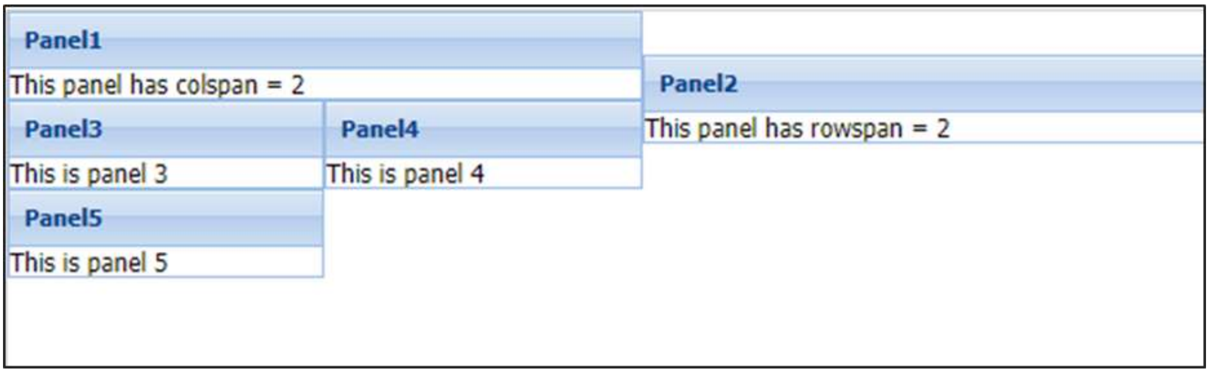
```
layout: 'table'
```

Example

Following is a simple example showing the usage of Table layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.container.Container', {
            renderTo : Ext.getBody(),
            layout : {
               type :'table',
               columns : 3,
               tableAttrs: {
                  style: {
                     width: '100%'
                  }
               }
            },
            width:600,
            height:200,
            items : [{
               title : 'Panel1',
               html : 'This panel has colspan = 2',
               colspan :2
            },{
               title : 'Panel2',
               html : 'This panel has rowspan = 2',
               rowspan: 2
            },{
               title : 'Panel3',
               html : 'This  s panel 3'
            },{
               title : 'Panel4',
               html : 'This is panel 4'
            },{
               title : 'Panel5',
               html : 'This is panel 5'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



vbox Layout

This layout allows the element to be distributed in a vertical manner. This is one of the most used layout.

Syntax

Following is a simple syntax to use the vbox layout.

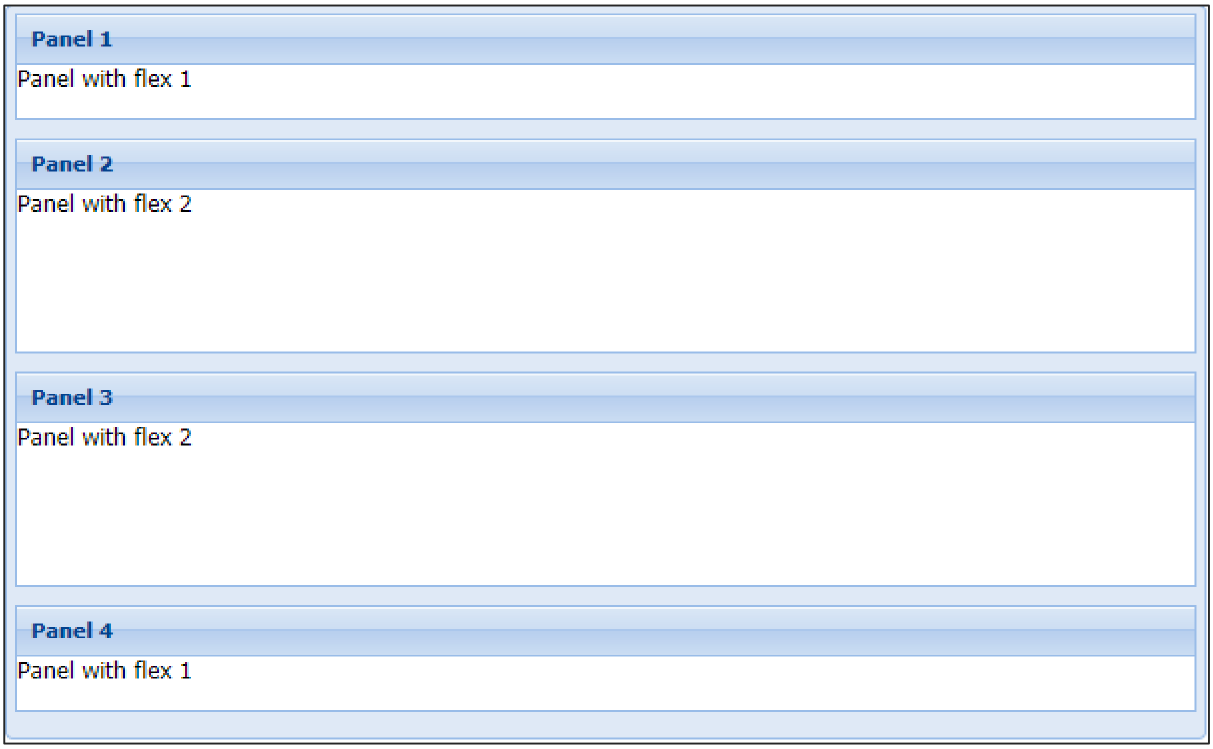
```
layout: 'vbox'
```

Example

Following is a simple example showing the usage of vbox layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.panel.Panel', {
            renderTo : Ext.getBody(),
            layout : {
               type :'vbox',
               align: 'stretch'
            },
            requires: ['Ext.layout.container.VBox'],
            xtype: 'layout-vertical-box',
            width : 600,
            height :400,
            frame :true,
            items : [{
               title: 'Panel 1',
               html : 'Panel with flex 1',
               margin: '0 0 10 0',
               flex : 1
            },{
               title: 'Panel 2',
               html : 'Panel with flex 2',
               margin: '0 0 10 0',
               flex : 2
            },{
               title: 'Panel 3',
               flex : 2,
               margin: '0 0 10 0',
               html : 'Panel with flex 2'
            },{
               title: 'Panel 4',
               html : 'Panel with flex 1',
               margin: '0 0 10 0',
               flex : 1
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Panel 1

Panel with flex 1

Panel 2

Panel with flex 2

Panel 3

Panel with flex 2

Panel 4

Panel with flex 1

hbox Layout

This layout allows the element to be distributed in a horizontal manner.

Syntax

Following is a simple syntax to use the hbox layout.

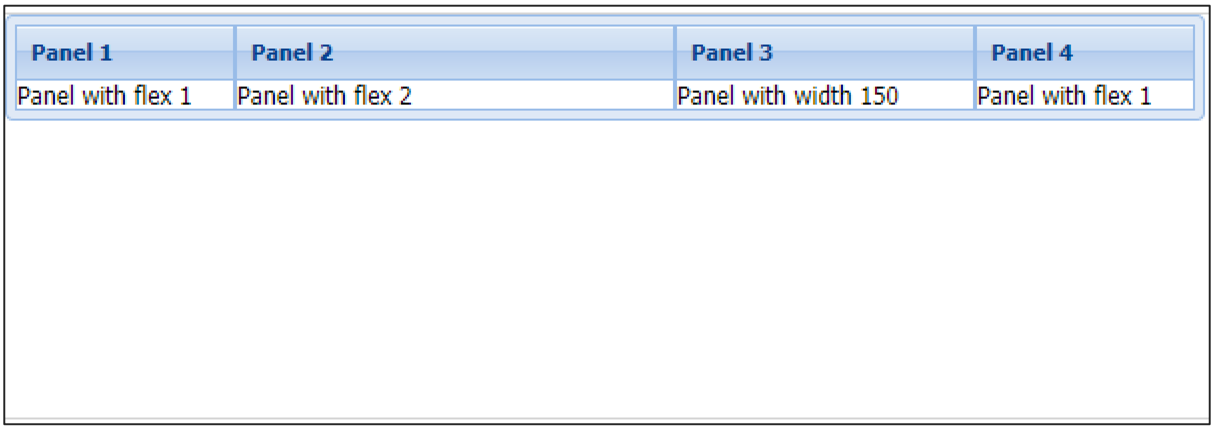
```
layout: 'hbox'
```

Example

Following is a simple example showing the usage of hbox layout.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.panel.Panel', {
            renderTo : Ext.getBody(),
            layout : {
               type :'hbox'
            },
            requires: ['Ext.layout.container.HBox'],
            xtype: 'layout-horizontal-box',
            width : 600,
            frame :true,
            items : [{
               title: 'Panel 1',
               html : 'Panel with flex 1',
               flex : 1
            },{
               title: 'Panel 2',
               html : 'Panel with flex 2',
               flex : 2
            },{
               title: 'Panel 3',
               width: 150,
               html : 'Panel with width 150'
            },{
               title: 'Panel 4',
               html : 'Panel with flex 1',
               flex : 1
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



9. Ext.js – Components

ExtJS UI is made up of one or many widgets called Components. Ext Js has various UI components defined that can be customised as per your requirements.

Sr. No.	Methods & Description
1	**Grid** Grid component can be used to show the data in a tabular format.
2	**Form** Form widget is to get the data from the user.
3	**Message Box** Message box is basically used to show data in the form of alert box.
4	**Chart** Charts are used to represent data in pictorial format.
5	**Tool tip** Tool tip is used to show some basic information when any event occurs.
6	**Window** This UI widget is to create a window, which should pop up when any event occurs.
7	**HTML editor** HTML Editor is one of the very useful UI component, which is used for styling the data that the user enters in terms of fonts, color, size, etc.
8	**Progress bar** Shows the progress of the backend work.

Grid

This is a simple component to display data, which is a collection of record stored in Ext.data.Store in a tabular format.

Syntax

Following is a simple syntax to create grid.

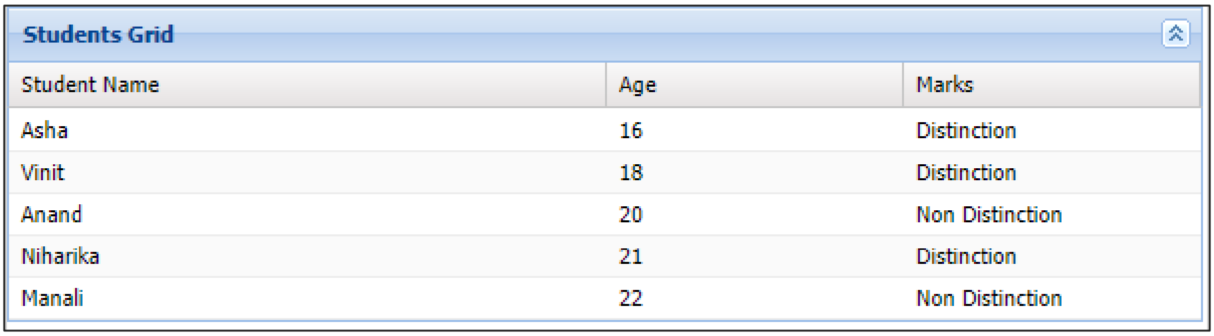
```
Ext.create('Ext.grid.Panel',{
grid properties..
columns : [Columns]
});
```

Example

Following is a simple example showing grid.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      // Creation of data model
      Ext.define('StudentDataModel', {
         extend: 'Ext.data.Model',
         fields: [
         {name: 'name', mapping : 'name'},
         {name: 'age', mapping : 'age'},
         {name: 'marks', mapping : 'marks'}
         ]
      });

      Ext.onReady(function(){
         // Store data
         var myData = [
            { name : "Asha", age : "16", marks : "90" },
            { name : "Vinit", age : "18", marks : "95" },
            { name : "Anand", age : "20", marks : "68" },
            { name : "Niharika", age : "21", marks : "86" },
            { name : "Manali", age : "22", marks : "57" }
         ];
         // Creation of first grid store
         var gridStore = Ext.create('Ext.data.Store', {
         model: 'StudentDataModel',
         data: myData
         });
         // Creation of first grid
         Ext.create('Ext.grid.Panel', {
            id                : 'gridId',
            store             : gridStore,
            stripeRows        : true,
            title             : 'Students Grid', // Title for the grid
            renderTo          :'gridDiv', // Div id where the grid has to be rendered
            width             : 600,
            collapsible       : true, // property to collapse grid
            enableColumnMove  :true,  // property which alllows column to move
to different position by dragging that column.
            enableColumnResize:true,  // property which allows to resize column run time.
            columns           :
               [{
                  header: "Student Name",
                  dataIndex: 'name',
                  id : 'name',
                  flex:  1,       // property defines the amount of space this
column is going to take in the grid container with respect to all.
                  sortable: true, // property to sort grid column data.
                  hideable: true  // property which allows column to be hidden
run time on user request.
               },{
                  header: "Age",
                  dataIndex: 'age',
                  flex: .5,
                  sortable: true,
                   hideable: false  // this column will not be available to be hidden.
               },{
                  header: "Marks",
                  dataIndex: 'marks',
                  flex: .5,
                  sortable: true,
                  // renderer is used to manipulate data based on some
conditions here who ever has marks greater than 75 will be displayed as
'Distinction' else 'Non Distinction'.
                  renderer :  function (value, metadata, record, rowIndex,
colIndex, store) {
                     if (value > 75) {
                        return "Distinction";
                     } else {
                        return "Non Distinction";
                     }
                  }
            }]
         });
      });
   </script>
</head>
<body>
   <div id = "gridDiv"></div>
</body>
</html>
```

The above program will produce the following result –



Students Grid

Student Name	Age	Marks
Asha	16	Distinction
Vinit	18	Distinction
Anand	20	Non Distinction
Niharika	21	Distinction
Manali	22	Non Distinction

Grid Poperties

Collapsible: This property is to add a collapse feature to the grid. Add **"Collapsible : true"** feature in grid properties to add this feature.

Sorting: This property is to add a sorting feature to the grid. Add Column property**"sortable : true"** in the grid to apply sorting ASC/DESC. By default, it is true. It can be made false, if you don’t want this feature to appear.

```
columns : [{
   header: "Student Name",
   dataIndex: 'name',
   id : 'name',
   flex:  1,
   sortable: true // property to sort grid column data
}]
```

By default, sorting can be applied with property **sorters : {property: 'id', direction : 'ASC'}** in store. It will sort grid data based on the property provided in the sorters and the direction given, before rendering data into the grid.

Enable Column resize: Column can be resized (its width can be increased or decreased) using grid properties **"enableColumnResize: true"**.

Column hideable: Add Column property **"hideable : true"** in a grid to make the column appear or hide. By default, it is true. It can be made false, if you don’t want this feature to appear.

```
columns : [{
   header: "Student Name",
   dataIndex: 'name',
   id : 'name',
   flex:  1,
   sortable: true,
   hideable: true // property which allows column to be hidden run time on user request
}]
```

Draggable column: Add Column property **"enableColumnMove: true"** is grid property with which we can move columns in a grid.

Renderer: This is the property to customize the view of grid data based on the data we get from the store.

```
columns : [{
   header: "Marks",
   dataIndex: 'marks',
   flex: .5,
   sortable: true,
   // renderer is used to manipulate data based on some conditions here who
ever has marks greater than 75 will be displayed as 'Distinction' else 'Non
Distinction'.
   renderer :  function (value, metadata, record, rowIndex, colIndex, store) {
      if (value > 75) {
         return "Distinction";
      } else {
         return "Non Distinction";
      }
   }
}]
```

Note: All the properties are added in the above grid example. Try them in try it editor.

Form

In most of the web applications, forms are the most important widget to get the information from the user such as login form/feedback form so that the value can be saved in the database for future reference. Form widget is used for this purpose.

Before creating a form, we should know about xTypes.

xType defines the type of Ext JS UI component, which is determined during rendering of the component. For example, the element can be a textbox for which we have xType as textField or the element can have a numeric value only for which we have Numeric xType.

Different Types of xType

TEXTFIELD:

This xType is to represent a text field, where the user can enter a value. Ext JS class for textfield is 'Ext.Form.Field.Text'.

```
{
   xtype: 'textfield',
   fieldLabel: 'Text field'
}
```

TEXT AREA:

This is to represent a text area. Ext JS class for this is 'Ext.form.field.TextArea'.

```
{
   xtype: 'textarea',
   fieldLabel: 'Text Area'
}
```

DISPLAY FIELD:

This is to represent a display text field. Ext JS class for this is 'Ext.form.Label'.

```
{
   xtype: 'displayfield',
   fieldLabel: 'Display Field''
}
```

DATE FIELD:

This is to represent a date field which has the date picker to select the date. Ext JS class for this is 'Ext.form.field.Date'.

```
{
   xtype: 'datefield',
   fieldLabel: 'Date picker'
}
```

BUTTON:

This is to represent a button. Ext JS class for this is 'Ext.button.Button'.

```
{
   xtype: 'button',
   text : 'Button'
}
```

RADIO FIELD:

This is to represent a radio field, where we can select any one among the all radio buttons or it can be customized to select multiple at a time. Ext JS class for this is 'Ext.form.field.Radio'.

```
{
   xtype      : 'fieldcontainer',
   fieldLabel : 'Radio field',
   defaultType: 'radiofield',
   defaults: {
      flex: 1
   },
   layout: 'hbox',
   items: [{
      boxLabel  : 'A',
      inputValue: 'a',
      id        : 'radio1'
   },{
      boxLabel  : 'B',
      inputValue: 'b',
      id        : 'radio2'
   },{
      boxLabel  : 'C',
      inputValue: 'c',
      id          : 'radio3'
   }]
}
```

CHECKBOX FIELD:

This is to represent a checkbox field, where we can select multiple values at a time. Ext JS class for this is 'Ext.form.field.Checkbox'.

```
{
   xtype: 'fieldcontainer',
   fieldLabel: 'Check Box Field',
   defaultType: 'checkboxfield',
   items: [{
      boxLabel  : 'HTML',
      inputValue: 'html',
      id        : 'checkbox1'
   },{
      boxLabel  : 'CSS',
      inputValue: 'css',
      checked   : true,
      id        : 'checkbox2'
   },{
      boxLabel  : 'JavaScript',
      inputValue: 'js',
      id        : 'checkbox3'
   }]
}
```

COMBO BOX:

This is to represent a combo box. Combo box contains a list of items. Ext JS class for this is 'Ext.form.field.ComboBox'.

```
{
   xtype : 'combobox',
   fieldLabel: 'Combo box',
   store: store,
   valueField: 'name'
}
// store for drop down values
var store = Ext.create('Ext.data.Store', {
   id : 'statesid',
   fields: ['abbr', 'name'],
   data : [
      {"abbr":"HTML", "name":"HTML"},
      {"abbr":"CSS", "name":"CSS"},
      {"abbr":"JS", "name":"JavaScript"}
   ]
});
```

TIME FIELD:
This is to represent a time field, where max and min time value can be predefined. Ext JS class for this is 'Ext.form.field.Time'.

```
{
   xtype: 'timefield',
   fieldLabel: 'Time field',
   minValue: '6:00 AM',
   maxValue: '8:00 PM',
   increment: 30,
   anchor: '100%'
}
```

FILE FIELD:
This is to represent a file upload field where we can browse and upload files. Ext JS class for this is 'Ext.form.field.File'.

```
{
    xtype: 'filefield',
    fieldLabel: 'File field',
    labelWidth: 50,
    msgTarget: 'side',
    allowBlank: false,
    anchor: '100%',
    buttonText: 'Select File...'
}
```

SPINNER:
This is to represent a spinner field, where the list can be spin and added. Ext JS class for this is 'Ext.form.field.Spinner'.

```
{
    xtype: 'spinnerfield',
    fieldLabel: 'Spinner field'
}
```

NUMERIC FIELD:
This is to represent a Numeric field, where we can have max and min value predefined. Ext JS class for this is 'Ext.form.field.Number'.

```
{
    xtype: 'numberfield',
    anchor: '100%',
    fieldLabel: 'Numeric field',
    maxValue: 99,
    minValue: 0
}
```

HIDDEN FIELD:
As the name explains, this xtype is to keep values hidden.

```
{
    xtype: 'hiddenfield',
   value: 'value from hidden field'
}
```

Syntax for Form Panel

Following is a simple syntax to create a form.

```
Ext.create('Ext.form.Panel');
```

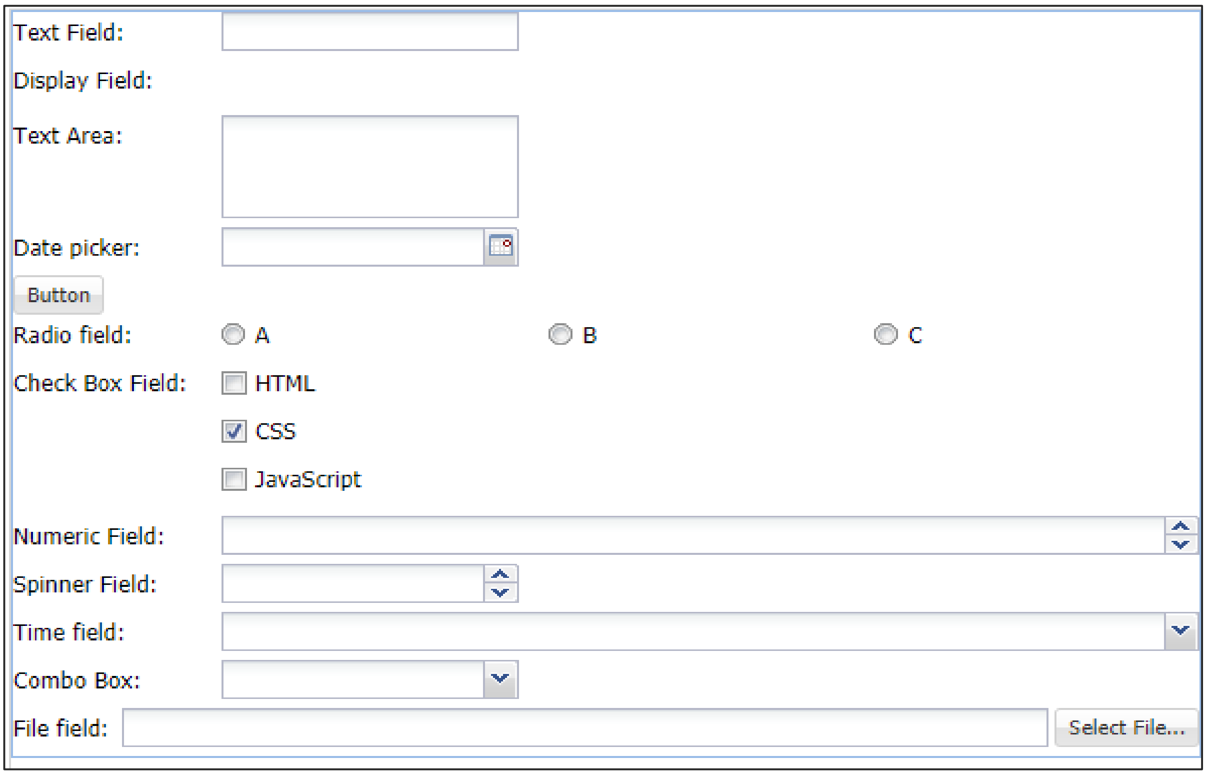
Example

Following is a simple example showing form with all the xTypes.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.form.Panel', {
            id: 'newForm',
            renderTo: 'formId',
            border: true,
            width: 600,
            items: [{
               xtype: 'textfield',
               fieldLabel: 'Text Field'
            },{
               xtype: 'displayfield',
               fieldLabel: 'Display Field'
            },{
               xtype: 'textarea',
               fieldLabel: 'Text Area'
            },{
               xtype: 'datefield',
               fieldLabel: 'Date picker'
            },{
               xtype: 'button',
               text: 'Button'
            },{
               xtype: 'fieldcontainer',
               fieldLabel: 'Radio field',
               defaultType: 'radiofield',
               defaults: {
                  flex: 1
               },
               layout: 'hbox',
               items: [{
                  boxLabel: 'A',
                  inputValue: 'a',
                  id: 'radio1'
               },{
                  boxLabel: 'B',
                  inputValue: 'b',
                  id: 'radio2'
               },{
                  boxLabel: 'C',
                  inputValue: 'c',
                  id: 'radio3'
               }]
            },{
               xtype: 'fieldcontainer',
               fieldLabel: 'Check Box Field',
               defaultType: 'checkboxfield',
               items: [{
                  boxLabel: 'HTML',
                  inputValue: 'html',
                  id: 'checkbox1'
               },{
                  boxLabel: 'CSS',
                  inputValue: 'css',
                  checked: true,
                  id: 'checkbox2'
               },{
                  boxLabel: 'JavaScript',
                  inputValue: 'js',
                  id: 'checkbox3'
               }]
            },{
               xtype: 'hiddenfield',
               name: 'hidden field ',
               value: 'value from hidden field'
            },{
               xtype: 'numberfield',
               anchor: '100%',
               fieldLabel: 'Numeric Field',
               maxValue: 99,
               minValue: 0
            },{
               xtype: 'spinnerfield',
               fieldLabel: 'Spinner Field',
               step: 6,
               // override onSpinUp (using step isn't necessary)
               onSpinUp: function() {
                 var me = this;
                 if (!me.readOnly) {
                     var val = me.step; // set the default value to the step value
                     if(me.getValue() !== '') {
                         val = parseInt(me.getValue().slice(0, -5));
                         // gets rid of " Pack"
                     }
                     me.setValue((val + me.step) + ' Pack');
                 }
               },

               // override onSpinDown
               onSpinDown: function() {
                 var me = this;
```

```
     if (!me.readOnly) {
         if(me.getValue() !== '') {
             val = parseInt(me.getValue().slice(0, -5));
             // gets rid of " Pack"
         }
         me.setValue((val - me.step) + ' Pack');
     }
   }
},{
   xtype: 'timefield',
   fieldLabel: 'Time field',
   minValue: '6:00 AM',
   maxValue: '8:00 PM',
   increment: 30,
   anchor: '100%'
},{
   xtype: 'combobox',
   fieldLabel: 'Combo Box',
   store: Ext.create('Ext.data.Store', {
            fields: ['abbr', 'name'],
            data: [{
               'abbr': 'HTML',
               'name': 'HTML'
            },{
               'abbr': 'CSS',
               'name': 'CSS'
            },{
               'abbr': 'JS',
               'name': 'JavaScript'
            }]
         }),
   valueField: 'abbr',
   displayField: 'name'
},{
   xtype: 'filefield',
   fieldLabel: 'File field',
               labelWidth: 50,
               msgTarget: 'side',
               allowBlank: false,
               anchor: '100%',
               buttonText: 'Select File...'
            }]
         });
      });
   </script>
</head>
<body>
   <div id = "formId"></div>
</body>
</html>
```

The above program will produce the following result –



Text Field:

Display Field:

Text Area:

Date picker:

Radio field: ○ A ○ B ○ C

Check Box Field: □ HTML

☑ CSS

□ JavaScript

Numeric Field:

Spinner Field:

Time field:

Combo Box:

File field:

Message box

Message box is to show some alert information on the occurrence of some event. Ext JS has different message boxes as listed in the following table.

Sr. No.	Message Box
1	**Basic alert box** This is the simplest alert box just to show some information on some event.
2	**Confirm box** This message box asks for user confirmation and different methods called on different options. The user selects as yes or no.
3	**Prompt box** This message box prompts for user input.
4	**multiline User input box** This is also like the prompt box but it allows the user to enter multiline information instead of just one line.
5	**Yes No Cancle alert box** This is like a confirm box which asks the user for some confirmation, such as whether the user wants to carry out the task or decline or cancel the task. Based on user selection, different methods get called.

Basic Alert Box

This is the simplest alert box just to show some information on some event.

Syntax

Following is a simple syntax for basic alert box.

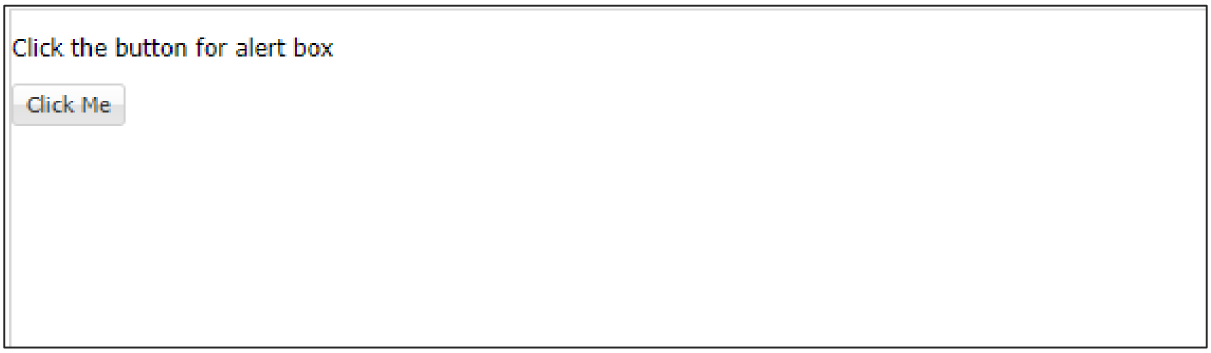
```
Ext.Msg.alert('Title', 'Basic message box in ExtJS');
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.Button', {
            renderTo: Ext.getElementById('msgBox'),
            text: 'Click Me',
            listeners: {
               click: function() {
                  Ext.Msg.alert('Title', 'Basic message box in ExtJS');
               }
            }
         });
      });
   </script>
</head>
<body>
   <p> Click the button for alert box </p>
   <div id = "msgBox" ></div>
</body>
</html>
```

The above program will produce the following result –



Click the button for alert box

Confirm Box

This message box asks for user confirmation and different methods called on different options the user selects as yes or no.

Syntax

Following is a simple syntax for confirm box.

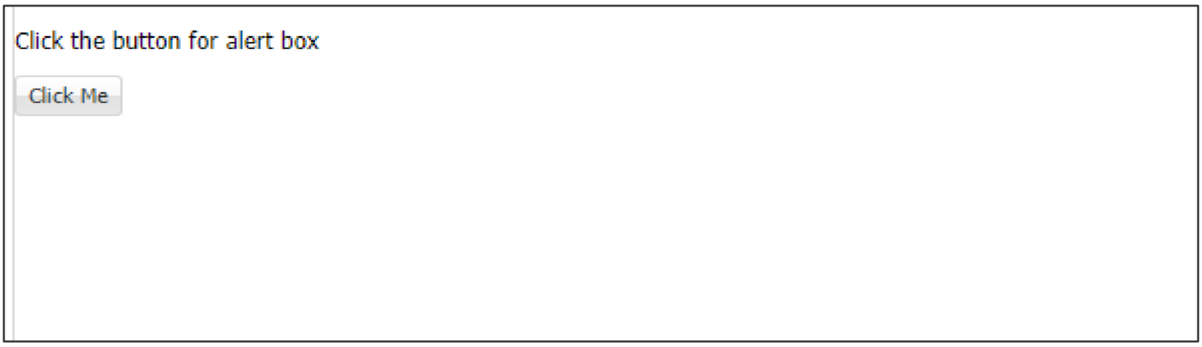
```
Ext.MessageBox.confirm('Confirm', Msg, optional callbackFunction());
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.Button', {
            renderTo: Ext.getElementById('msgBox'),
            text: 'Click Me',
            listeners: {
               click: function() {
                  Ext.MessageBox.confirm('Confirm', 'Are you sure you want to
do this ?', callbackFunction);
                  function callbackFunction(btn){
                  if(btn == 'yes') {
                     Ext.Msg.alert ('Button Click', 'You clicked the Yes button');
                  } else {
                     Ext.Msg.alert ('Button Click', 'You clicked the No button');
                  }

                  };
               }
             }
         });
      });
   </script>
</head>
<body>
   <p> Click the button for alert box </p>
   <div id = "msgBox" ></div>
</body>
</html>
```

The above program will produce the following result –


Click the button for alert box

Prompt Box

This message box prompts for user input. It is a single line user input.

Syntax

Following is a simple syntax.

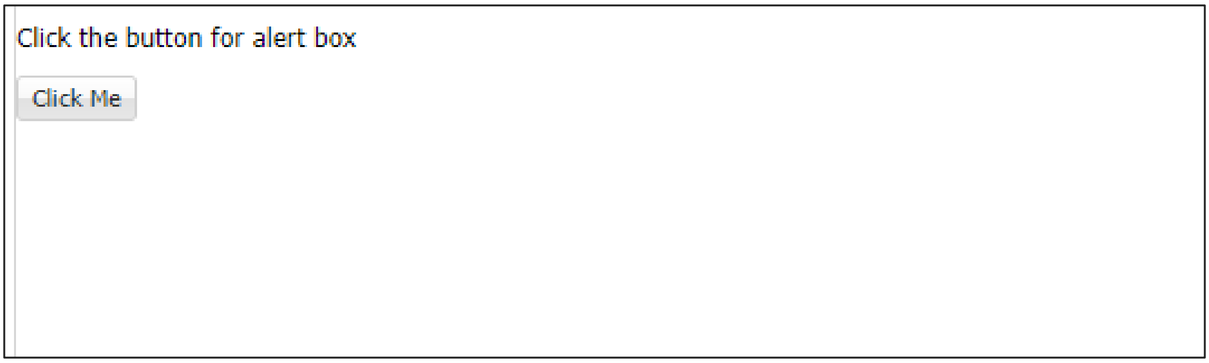
```
Ext.Msg.prompt('Name', 'Please enter your name:', function(){});
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.Button', {
            renderTo: Ext.getElementById('msgBox'),
            text: 'Click Me',
            listeners: {
               click: function() {
                  Ext.Msg.prompt('Name', 'Please enter your name:', function(btn, text){
                     if (btn == 'ok'){
                        Ext.Msg.alert('Hello', 'Hello '+text);
                     }
                  });
               }
            }
         });
      });
   </script>
</head>
<body>
   <p> Click the button for alert box </p>
   <div id = "msgBox" ></div>
</body>
</html>
```

The above program will produce the following result –



Click the button for alert box

Multiline User Input Box

This is also like the prompt box but it allows THE user to enter multiline information instead of just one line.

Syntax

Following is a simple syntax.

```
Ext.MessageBox.show({
   title: 'Details',
   msg: 'Please enter your details:',
   width:300,
   buttons: Ext.MessageBox.OKCANCEL,
   multiline: true,      // this property is for multiline user input.
   fn: callbackFunction
});
```

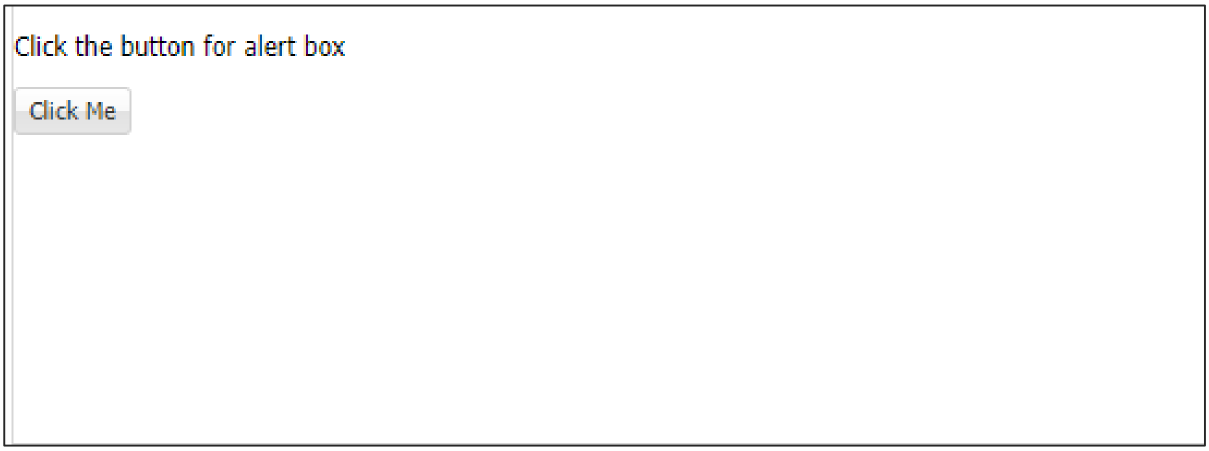
Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.Button', {
            renderTo: Ext.getElementById('msgBox'),
            text: 'Click Me',
            listeners: {
               click: function() {
                  Ext.MessageBox.show({
                     title: 'Details',
                     msg: 'Please enter your details:',
                     width:300,
                     buttons: Ext.MessageBox.OKCANCEL,
                     multiline: true,
                     fn: callbackFunction
                  });

                  function callbackFunction (){
                     Ext.Msg.alert('status', 'Details entered succesfully');
                  }
```

```
                }
             }
          });
       });
    </script>
</head>
<body>
    <p> Click the button for alert box </p>
    <div id = "msgBox" ></div>
</body>
</html>
```

The above program will produce the following result –



Click the button for alert box

Yes NO Cancel Box

This is like a confirm box which asks the user for some confirmation, such as whether the user wants to do the task or decline or cancel the task. Based on the user selection different methods get called.

Syntax

Following is a simple syntax.

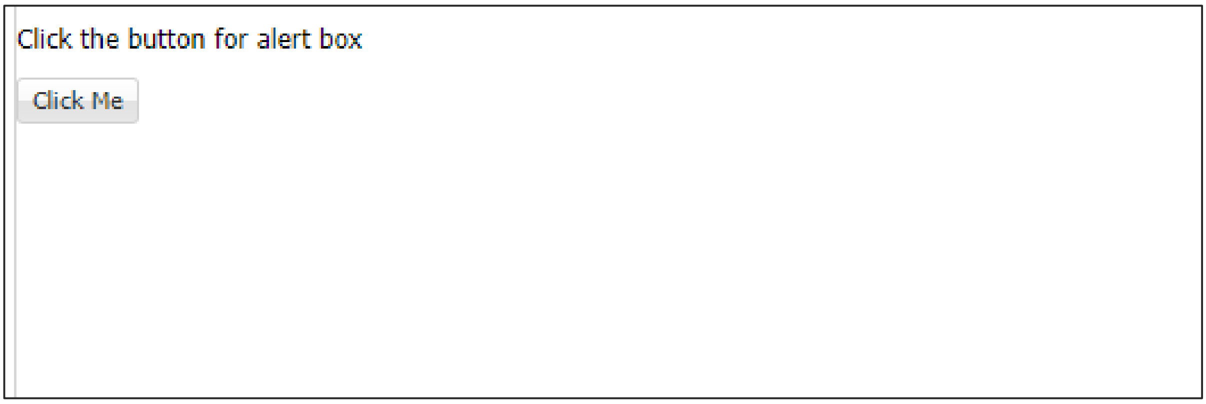
```
Ext.MessageBox.show({
    title: 'Details',
    msg: 'Please enter your details:',
    width:300,
   buttons: Ext.MessageBox.YESNOCANCEL // this button property for all three
options YES, NO, Cancel.
});
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.Button', {
            renderTo: Ext.getElementById('msgBox'),
            text: 'Click Me',
            listeners: {
               click: function() {
                  Ext.MessageBox.show({
                     title: 'Details',
                     msg: 'Please enter your details:',
                     width:300,
                     buttons: Ext.MessageBox.YESNOCANCEL,
                     multiline: true,
                     fn: callbackFunction
                  });

                  function callbackFunction (){
                     Ext.Msg.alert('status', 'Details entered succesfully');
                  }
               }
            }
         });
      });
   </script>
</head>
<body>
   <p> Click the button for alert box </p>
   <div id = "msgBox" ></div>
</body>
</html>
```

The above program will produce the following result –



Click the button for alert box

Chart

Charts are used to represent data in pictorial format. Following are the different types of charts provided by Ext JS.

Sr. No.	Charts
1	**Pie Chart** As the name indicates, this chart is used for representing data in the pie graph format.
2	**Line Chart**

	This chart is used for representing data in the line graph format.
3	**Bar Chart** This chart is used for representing data in the bar graph format.
4	**Area Chart** This chart is used for representing data in the form of area graph.

Pie Chart

This chart is used for representing data in the pie graph format. It is a polar chart.

Syntax

Following is a simple syntax.

```
Ext.create('Ext.chart.PolarChart', {
series: [{
  Type: 'pie'
  ..
  }]
  render and other properties.
});
```

Example

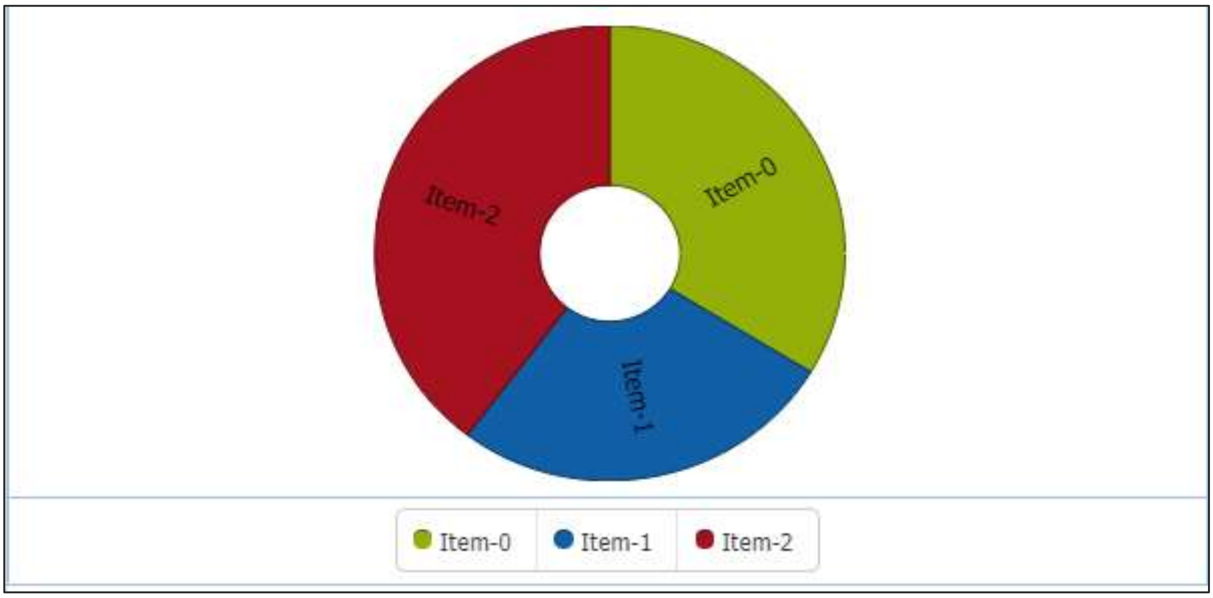
Following is a simple example showing the usage of pie chart.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
```

```
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classi
c/classic/resources/charts-all.css" rel="stylesheet" />
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.chart.PolarChart', {
             renderTo: document.body,
             width: 600,
             height: 300,
             store: {
                 fields: ['name', 'g1'],
                 data: [
                     {"name": "Item-0", "g1": 57},
                     {"name": "Item-1", "g1": 45},
                     {"name": "Item-2", "g1": 67}
                 ]
             },

             //configure the legend.
             legend: {
                 docked: 'bottom'
             },

             //describe the actual pie series.
             series: [{
                 type: 'pie',
                 xField: 'g1',
                 label: {
                     field: 'name'
                 },
                 donut: 30
                 // increase or decrease for increasing or decreasing donut area in middle.
             }]
         });
```

```
       });
    </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Line Chart

This chart is used for representing data in the Line graph format. It is a Cartesian Chart.

Syntax

Following is a simple syntax.

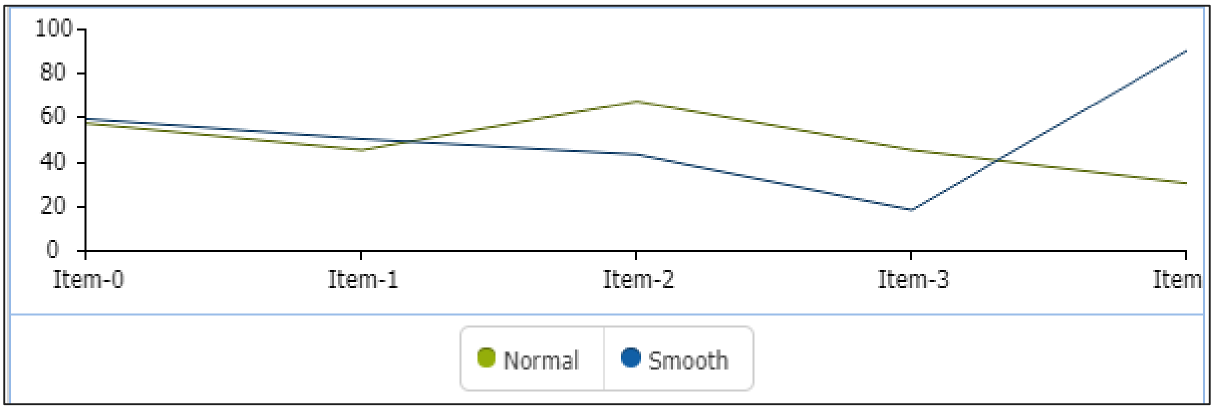
```
Ext.create('Ext.chart.CartesianChart',{
   series: [{
      type: 'line',
      xField: 'name',
      yField: 'G1'
      }]
      render, legend and other properties
});
```

Example

Following is a simple example showing the usage of line chart.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classi
c/classic/resources/charts-all.css" rel="stylesheet" />
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.chart.CartesianChart', {
             renderTo: document.body,
             width: 600,
             height: 200,
             store: {
                 fields: ['name', 'g1', 'g2'],
                 data: [
                        {"name": "Item-0", "g1": 57, "g2": 59},
                        {"name": "Item-1", "g1": 45, "g2": 50},
                        {"name": "Item-2", "g1": 67, "g2": 43},
                        {"name": "Item-3", "g1": 45, "g2": 18},
                        {"name": "Item-4", "g1": 30, "g2": 90}
                 ]
             },

             legend: {
                 docked: 'bottom'
             },

             //define x and y axis.
             axes: [{
                 type: 'numeric',
                 position: 'left'
             }, {
                 type: 'category',
                 visibleRange: [0, 1],
                 position: 'bottom'
             }],

             //define the actual series
             series: [{
                 type: 'line',
                 xField: 'name',
                 yField: 'g1',
                 title: 'Normal'
             }, {
                 type: 'line',
                 xField: 'name',
                 yField: 'g2',
                 title: 'Smooth'
             }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Bar Chart

This chart is used for representing data in the Bar graph format. It is a Cartesian Chart.

Syntax

Following is a simple syntax.

```
Ext.create('Ext.chart.CartesianChart',{
   series: [{
      type: 'bar',
      xField: 'name',
      yField: 'G1'
      }]
      render, legend and other properties
});
```

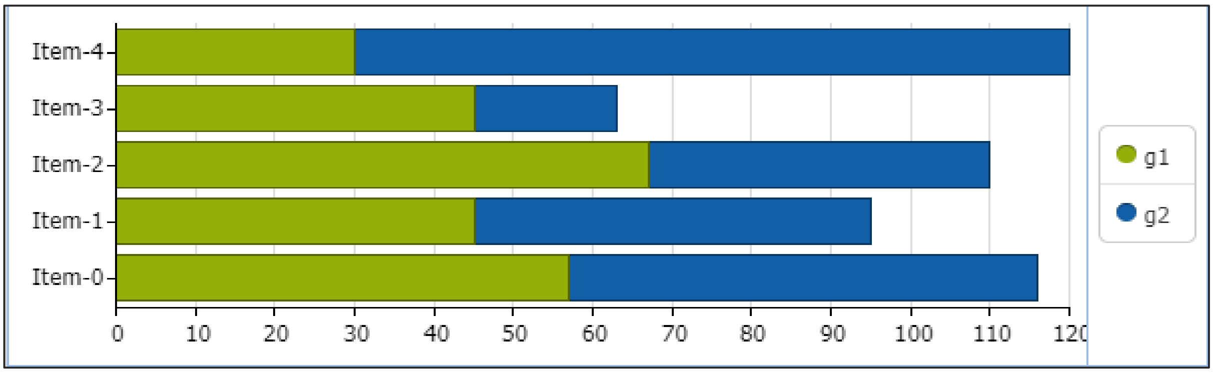
Example

Following is a simple example showing the usage of bar chart.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
```

```
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classi
c/classic/resources/charts-all.css" rel="stylesheet" />
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.chart.CartesianChart', {
            renderTo: document.body,
            width: 600,
            height: 200,
            flipXY: true,
            store: {
              fields: ['name', 'g1', 'g2'],
              data: [
                     {"name": "Item-0", "g1": 57, "g2": 59},
                     {"name": "Item-1", "g1": 45, "g2": 50},
                     {"name": "Item-2", "g1": 67, "g2": 43},
                     {"name": "Item-3", "g1": 45, "g2": 18},
                     {"name": "Item-4", "g1": 30, "g2": 90}
              ]
            },

            // set legend configuration
            legend: {
              docked: 'right'
            },

            // define the x and y-axis configuration.
            axes: [{
              type: 'numeric',
              position: 'bottom',
              grid: true,
              minimum: 0
            }, {
              type: 'category',
              position: 'left'
            }],

            // define the actual bar series.
            series: [{
              type: 'bar',
              xField: 'name',
              yField: ['g1', 'g2'],
              axis: 'left'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Area Chart

This chart is used for representing data in the Area graph format. It is a Cartesian Chart.

Syntax

Following is a simple syntax.

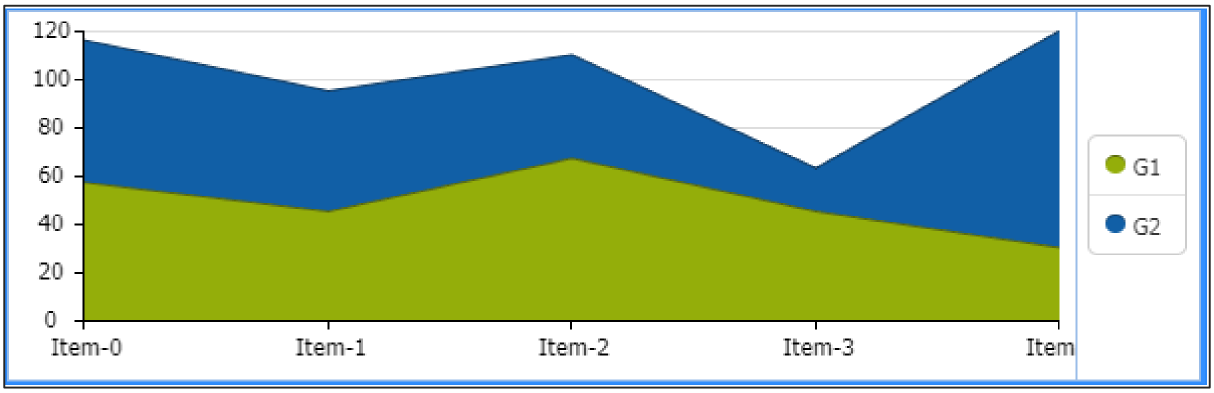
```
Ext.create('Ext.chart.CartesianChart',{
   series: [{
      type: 'area',
      xField: 'name',
      yField: 'G1'
      }]
      render, legend and other properties
});
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classi
c/classic/resources/charts-all.css" rel="stylesheet" />
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.chart.CartesianChart', {
             renderTo: document.body,
             width: 600,
             height: 200,
             store: {
                 fields: ['name', 'g1', 'g2'],
                 data: [
                     {"name": "Item-0", "g1": 57, "g2": 59},
                     {"name": "Item-1", "g1": 45, "g2": 50},
                     {"name": "Item-2", "g1": 67, "g2": 43},
                     {"name": "Item-3", "g1": 45, "g2": 18},
                     {"name": "Item-4", "g1": 30, "g2": 90}
                 ]
             },

             legend: {
                 docked: 'right'
             },
             axes: [{
                 type: 'numeric',
                 position: 'left',
                 grid: true
             }, {
                 type: 'category',
                 position: 'bottom',
                 visibleRange: [0,5]
             }],

             series: [{
                 type: 'area',
                 xField: 'name',
                 yField: ['g1', 'g2'],
                 title: ['G1', 'G2']
             }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Tooltip

A small piece of information that appears when some event occurs. Basically, it is used for hover over event.

Syntax

Following is a simple syntax to create a tooltip.

```
T1 = new 'Ext.ToolTip'({properties});
```

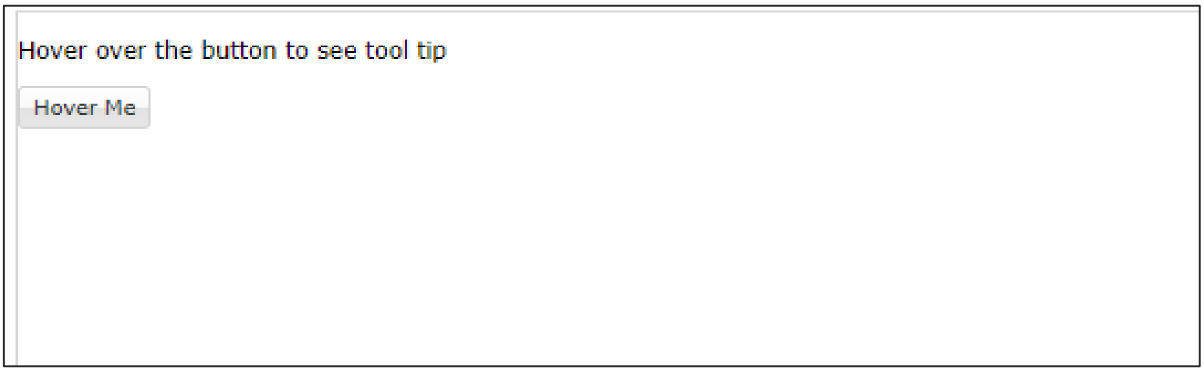
Example

Following is a simple example showing a tooltip.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         toolTip = new Ext.ToolTip({
            id : 'toolTip',
            anchor : 'bottom',
            html : 'This is a basic toolTip',
            title : 'Tool - Tip Title',
            closable : true,
            closeAction : 'hide'
         });
```

```
            Ext.create('Ext.Button', {
                renderTo: Ext.getElementById('buttonId'),
                text: 'Hover Me',
                listeners: {
                   mouseover: function() {
                      toolTip.show();
                   }
                }
             });
          });
       </script>
    </head>
    <body>
       <div id = "buttonId"></div>
    </body>
    </html>
```

The above program will produce the following result –



Hover over the button to see tool tip

Window

This UI component is to create a window, which should pop up when any event occurs. Window is basically a panel, which should appear when any event occurs as a button/link click or hover over.

Syntax

Following is a simple syntax to create a window.

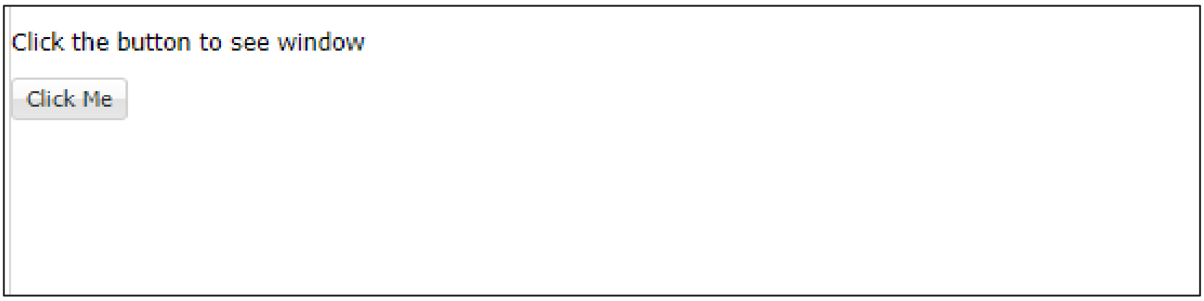
```
win = new Ext.Window({ properties });
```

Example

Following is a simple example showing an email window.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         win = new Ext.Window({
            title:'Email Window',
            layout:'form',
            width:400,
            closeAction:'close',
            target : document.getElementById('buttonId'),
            plain: true,
            items: [{
               xtype : 'textfield',
               fieldLabel: 'To'
            },{
               xtype : 'textfield',
               fieldLabel: 'CC'
            },{
               xtype : 'textfield',
               fieldLabel: 'BCC'
            },{
               xtype : 'textfield',
               fieldLabel: 'Subject'
            },{
               xtype : 'textarea',
               fieldLabel: 'Email Message'
            }],
            buttons: [{
               text: 'Save Draft',
               handler: function(){Ext.Msg.alert('Save Draft', 'Your msg is saved');}
            },{
               text: 'Send',
               handler: function(){Ext.Msg.alert('Message Sent', 'Your msg is sent');}
            },{
               text: 'Cancel',
               handler: function(){win.close(); Ext.Msg.alert('close', 'Email 
window is closed');}
            }],
            buttonAlign: 'center',
         });
         Ext.create('Ext.Button', {
            renderTo: Ext.getElementById('buttonId'),
            text: 'Click Me',
            listeners: {
               click: function() {
                  win.show();
               }
            }
         });
      });
   </script>
</head>
<body>
   <p> Click the button to see window </p>
   <div id = "buttonId"></div>
</body>
</html>
```

The above program will produce the following result –



Click the button to see window

HTML Editor

This Ext JS UI widget is to create a html editor so that the user can edit a piece of information in terms of font, color, size, etc.

Syntax

Following is a simple syntax to create HTML Editor.

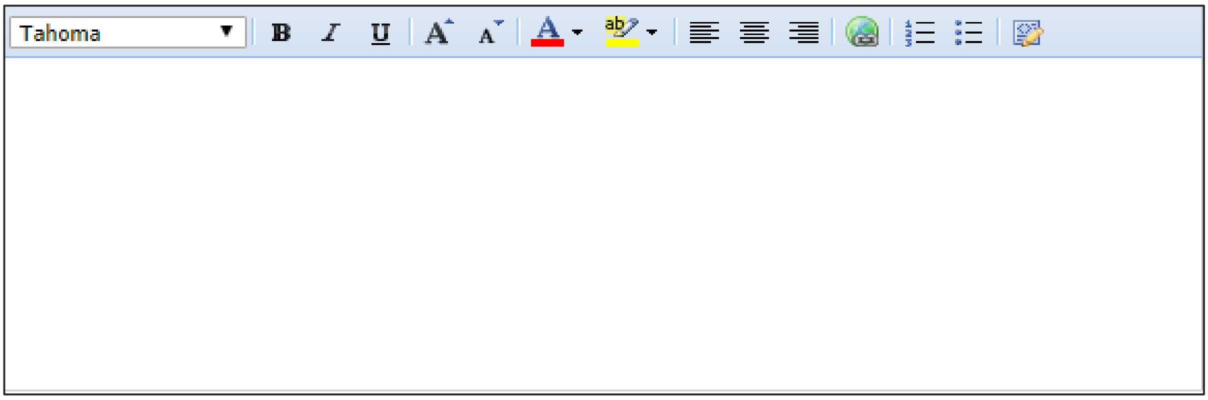
```
Ext.create('Ext.form.HtmlEditor')
```

Example

Following is a simple example showing HTML editor.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.form.HtmlEditor', {
            width: 580,
            height: 250,
            renderTo: document.getElementById('editorId')
         });
      });
   </script>
</head>
<body>
   <div id = "editorId"></div>
</body>
</html>
```

The above program will produce the following result –



Progress Bar

This is used to show the progress of the work done and to show that the backend work is still in progress, hence wait until its done.

Syntax

It is basically a message box showing the progress of the task. Following is a simple syntax to create a Progress bar.

```
 Ext.MessageBox.show({
   title: 'Please wait',
   msg: 'Loading items...',
   progressText: 'Initializing...',
   width:300,
   progress:true,
   closable:false
});
```

Example

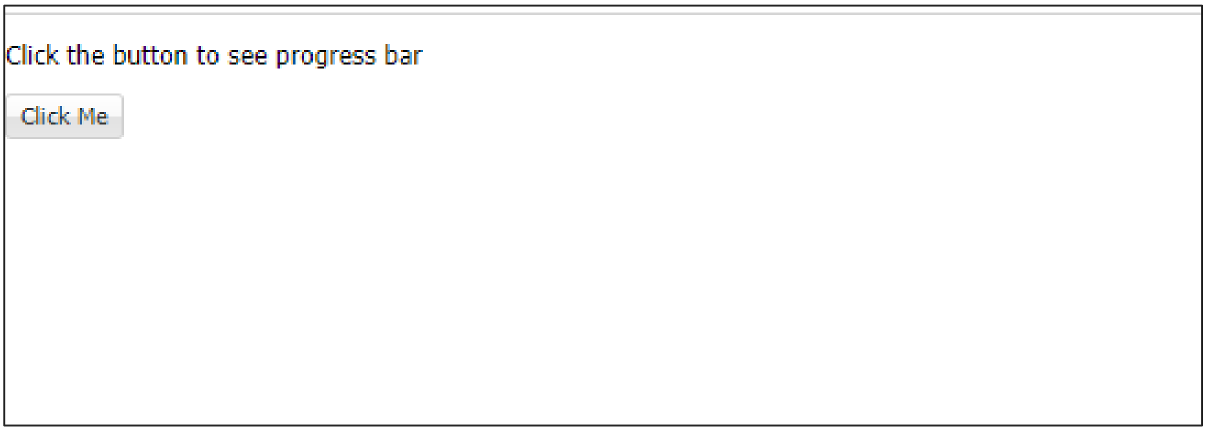
Following is a simple example showing a Progress bar.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
```

```
Ext.onReady(function() {
   function progressBar(v) {
      return function()  {
         if(v == 10) {
            Ext.MessageBox.hide();
            result();
         } else {
            var i = v/9;
            Ext.MessageBox.updateProgress(i, Math.round(100*i)+'% completed');
         }
      };
   };
   function showProgressBar() {
      for(var i = 1; i < 11; i++){
         setTimeout(progressBar(i), i*500);
      }
   }
   function result(){
      Ext.Msg.alert('status', 'Process completed succesfully');
   }
   Ext.create('Ext.Button', {
      renderTo: Ext.getElementById('buttonId'),
      text: 'Click Me',
      listeners: {
         click: function() {
            Ext.MessageBox.show({
               title: 'Please wait',
               msg: 'Loading items...',
               progressText: 'Initializing...',
               width:300,
               progress:true,
               closable:false
            });
            showProgressBar();
         }
      }
```

```
          });
       });
    </script>
</head>
<body>
    <p> Click the button to see progress bar  </p>
    <div id = "buttonId"></div>
</body>
</html>
```

The above program will produce the following result –


Click the button to see progress bar

10. Ext.js – Drag and Drop

Drag and drop feature is one of the powerful features added to make the developer's task easy. A drag operation, essentially, is a click gesture on some UI element, while the mouse button is held down and the mouse is moved. A drop operation occurs when the mouse button is released after a drag operation.

Syntax

Adding drag and drop class to the draggable targets.

```
var dd = Ext.create('Ext.dd.DD', el, 'imagesDDGroup', {
    isTarget: false
});
```

Adding drag and drop target class to drappable target.

```
var mainTarget = Ext.create('Ext.dd.DDTarget', 'mainRoom', 'imagesDDGroup',{
   ignoreSelf: false
});
```

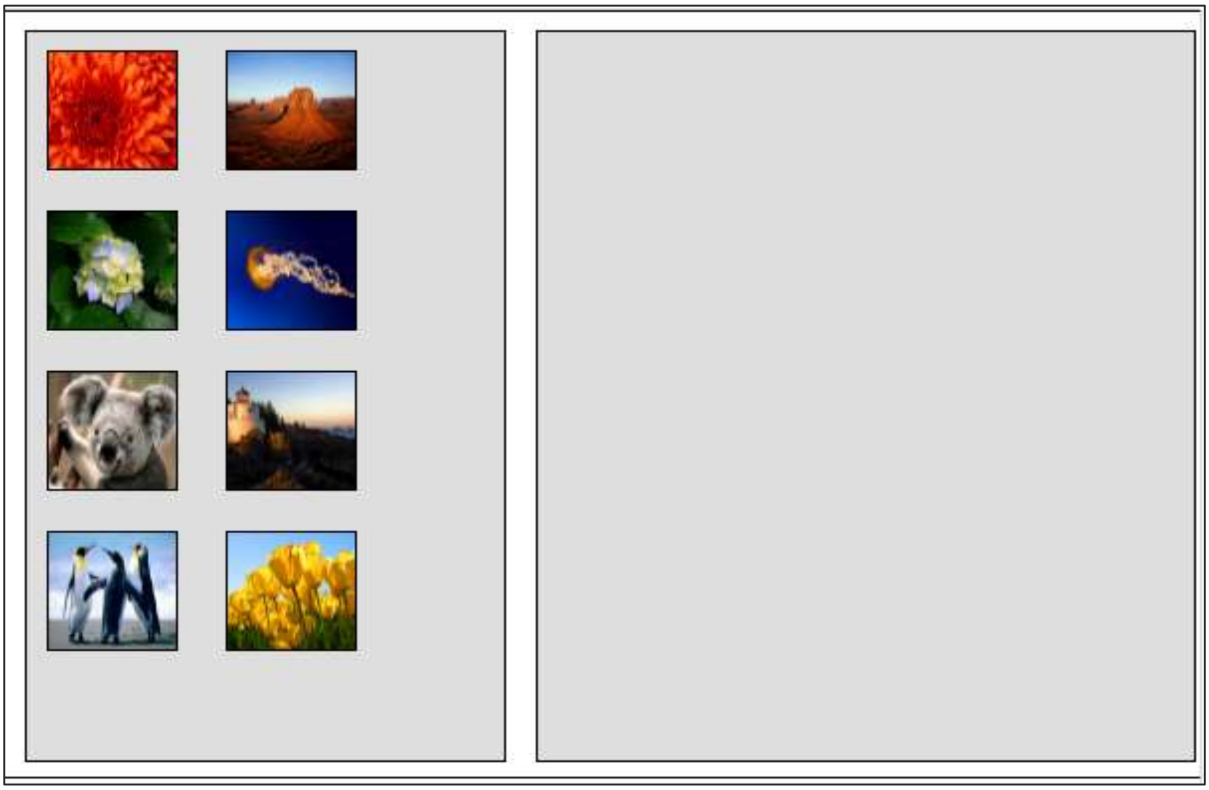
Example

Following is a simple example.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.application({
          launch: function() {
              var images = Ext.get('images').select('img');
              Ext.each(images.elements, function(el) {
                  var dd = Ext.create('Ext.dd.DD', el, 'imagesDDGroup', {
                      isTarget: false
                  });
              });
```

```
       }
   });
   var mainTarget = Ext.create('Ext.dd.DDTarget', 'mainRoom', 'imagesDDGroup', {
      ignoreSelf: false
   });
</script>
<style>
   #content{
      width:600px;
      height:400px;
      padding:10px;
      border:1px solid #000;
   }
   #images{
      float:left;
      width:40%;
      height:100%;
      border:1px solid Black;
      background-color:rgba(222, 222, 222, 1.0);
   }
   #mainRoom{
      float:left;
      width:55%;
      height:100%;
      margin-left:15px;
      border:1px solid Black;
      background-color:rgba(222, 222, 222, 1.0);
   }
   .image{
      width:64px;
      height:64px;
      margin:10px;
      cursor:pointer;
      border:1px solid Black;
      display: inline-block;
   }
   </style>
</head>
<body>
   <div id="content">
      <div id="images">
         <img src = "/extjs/images/1.jpg" class = "image" />
         <img src = "/extjs/images/2.jpg" class = "image" />
         <img src = "/extjs/images/3.jpg" class = "image" />
         <img src = "/extjs/images/4.jpg" class = "image" />
         <img src = "/extjs/images/5.jpg" class = "image" />
         <img src = "/extjs/images/6.jpg" class = "image" />
         <img src = "/extjs/images/7.jpg" class = "image" />
         <img src = "/extjs/images/8.jpg" class = "image" />
      </div>
      <div id="mainRoom"></div>
   </div>
</body>
</html>
```

The above program will produce the following result –



With the help of drag and drop in Extjs, we can move data from grid to grid and grid to form. Following are the examples of moving data between grids and forms.

Drag and drop - Grid to Grid.

drag and drop - Grid to Form

Grid to Grid Drag and Drop

With the help of drag and drop plugin, we can drag data from one grid and drop it to another grid and vice versa.

The following example shows how we can drag data from one grid and drop it to another. Here, we have a reset button to reset data in both the grids.

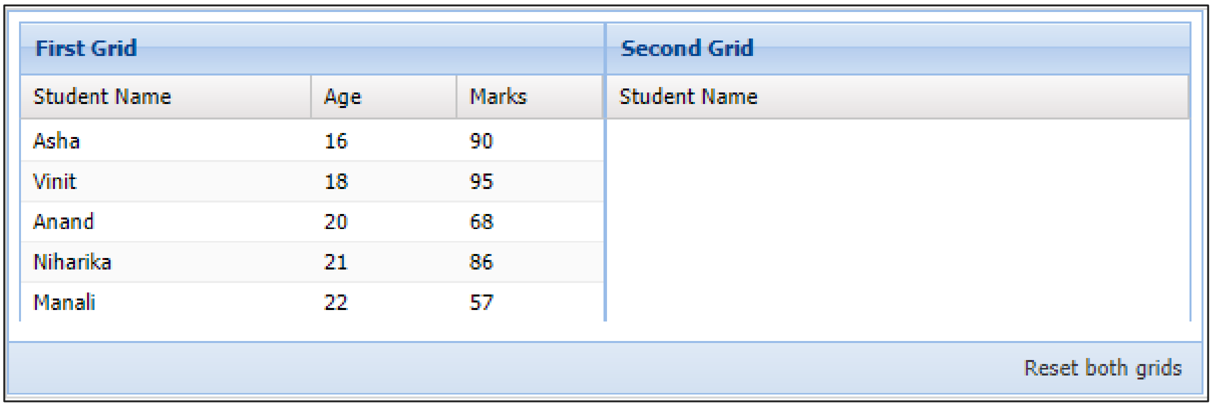
Example

Following is a simple example showing Drag and Drop between grids.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.require([
         'Ext.grid.*',
         'Ext.data.*',
         'Ext.dd.*'
      ]);
      // Creation of data model
      Ext.define('StudentDataModel', {
         extend: 'Ext.data.Model',
         fields: [
         {name: 'name', mapping : 'name'},
         {name: 'age', mapping : 'age'},
         {name: 'marks', mapping : 'marks'}
         ]
      });
      Ext.onReady(function(){
         // Store data
         var myData = [
            { name : "Asha", age : "16", marks : "90" },
            { name : "Vinit", age : "18", marks : "95" },
            { name : "Anand", age : "20", marks : "68" },
            { name : "Niharika", age : "21", marks : "86" },
            { name : "Manali", age : "22", marks : "57" }
         ];
         // Creation of first grid store
         var firstGridStore = Ext.create('Ext.data.Store', {
         model: 'StudentDataModel',
         data: myData
         });
         // Creation of first grid
         var firstGrid = Ext.create('Ext.grid.Panel', {
            multiSelect: true,
            viewConfig: {
               plugins: {
                  ptype: 'gridviewdragdrop',
                  dragGroup: 'firstGridDDGroup',
                  dropGroup: 'secondGridDDGroup'
               },
               listeners: {
                  drop: function(node, data, dropRec, dropPosition) {
                     var dropOn = dropRec ? ' ' + dropPosition + ' ' +
dropRec.get('name') : ' on empty view';
                  }
               }
            },
            store            : firstGridStore,
            columns          :
               [{
                  header: "Student Name",
                  dataIndex: 'name',
                  id : 'name',
                  flex:  1,
               sortable: true
               },{
                  header: "Age",
                  dataIndex: 'age',
                  flex: .5,
                  sortable: true
               },{
                  header: "Marks",
                  dataIndex: 'marks',
                  flex: .5,
                  sortable: true
               }],
            stripeRows       : true,
            title            : 'First Grid',
            margins          : '0 2 0 0'
         });
         // Creation of second grid store
         var secondGridStore = Ext.create('Ext.data.Store', {
            model: 'StudentDataModel'
         });
         // Creation of second grid
         var secondGrid = Ext.create('Ext.grid.Panel', {
            viewConfig: {
               plugins: {
                  ptype: 'gridviewdragdrop',
                  dragGroup: 'secondGridDDGroup',
                  dropGroup: 'firstGridDDGroup'
               },
               listeners: {
                  drop: function(node, data, dropRec, dropPosition) {
                     var dropOn = dropRec ? ' ' + dropPosition + ' ' +
dropRec.get('name') : ' on empty view';
                  }
               }
            },
            store            : secondGridStore,
            columns          :
            [{
               header: "Student Name",
               dataIndex: 'name',
               id : 'name',
               flex:  1,
               sortable: true
            },{
               header: "Age",
               dataIndex: 'age',
               flex: .5,
               sortable: true
            },{
               header: "Marks",
               dataIndex: 'marks',
               flex: .5,
               sortable: true
            }],
            stripeRows       : true,
            title            : 'Second Grid',
            margins          : '0 0 0 3'
         });
         // Creation of a panel to show both the grids.
         var displayPanel = Ext.create('Ext.Panel', {
            width        : 600,
            height       : 200,
            layout       : {
               type: 'hbox',
               align: 'stretch',
               padding: 5
            },
            renderTo     : 'panel',
            defaults     : { flex : 1 },
            items        : [
               firstGrid,
               secondGrid
```

```
            ],
            dockedItems: {
               xtype: 'toolbar',
               dock: 'bottom',
               items: ['->',
               {
                  text: 'Reset both grids',
                  handler: function(){
                     firstGridStore.loadData(myData);
                     secondGridStore.removeAll();
                  }
               }]
            }
         });
      });
   </script>
</head>
<body>
<div id = "panel" > </div>
</body>
</html>
```

The above program will produce the following result –



First Grid			Second Grid
Student Name	Age	Marks	Student Name
Asha	16	90	
Vinit	18	95	
Anand	20	68	
Niharika	21	86	
Manali	22	57	

Grid to Form Drag and Drop

With the help of drag and drop plugin, we can drag data from one grid and drop it to form fields.

The following example shows that we can drag data from a grid and drop it to a form. Here, we have a reset button to reset data in both the form and the grid.

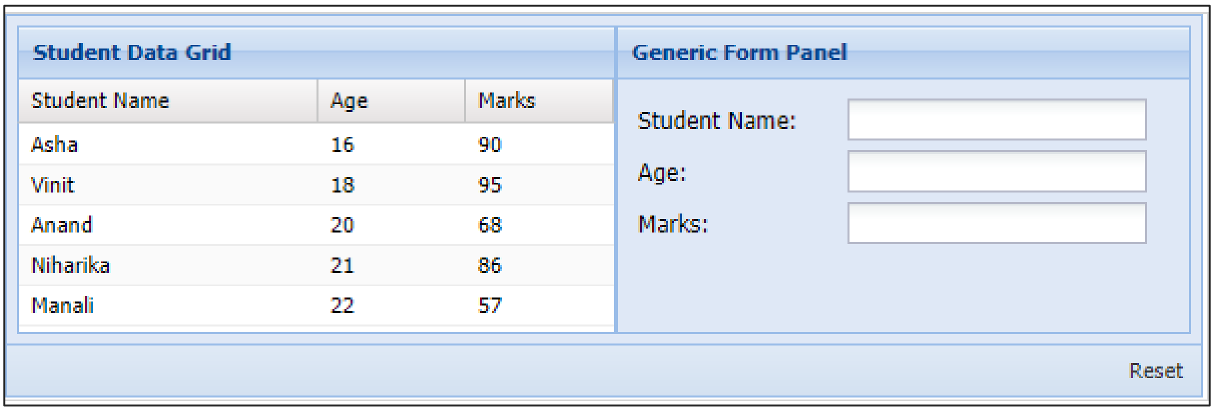
Example

Following is a simple example showing Drag and Drop grid to form.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.require(['*']);
      Ext.onReady(function(){
         // Store data
         var myData = [
            { name : "Asha", age : "16", marks : "90" },
            { name : "Vinit", age : "18", marks : "95" },
            { name : "Anand", age : "20", marks : "68" },
            { name : "Niharika", age : "21", marks : "86" },
            { name : "Manali", age : "22", marks : "57" }
         ];
         // Creation of data model
         Ext.define('StudentDataModel', {
            extend: 'Ext.data.Model',
            fields: [
               {name: 'name', mapping : 'name'},
               {name: 'age', mapping : 'age'},
               {name: 'marks', mapping : 'marks'}
            ]
         });
          // Creation of grid store
         var gridStore = Ext.create('Ext.data.Store', {
            model  : 'StudentDataModel',
            data   : myData
         });
```

```
// Creation of first grid
var grid = Ext.create('Ext.grid.Panel', {
   viewConfig: {
      plugins: {
         ddGroup: 'GridExample',
         ptype: 'gridviewdragdrop',
         enableDrop: false
      }
   },
   enableDragDrop   : true,
   stripeRows       : true,
   width            : 300,
   margins          : '0 2 0 0',
   region           : 'west',
   title            : 'Student Data Grid',
   selModel         : Ext.create('Ext.selection.RowModel',{
      singleSelect : true
   }),
   store            : gridStore,
   columns          :
      [{
         header: "Student Name",
         dataIndex: 'name',
         id : 'name',
         flex:  1,
         sortable: true
      },{
         header: "Age",
         dataIndex: 'age',
         flex: .5,
         sortable: true
      },{
         header: "Marks",
         dataIndex: 'marks',
         flex: .5,
         sortable: true
      }]
});
// Creation of form panel
var formPanel = Ext.create('Ext.form.Panel', {
   region     : 'center',
   title      : 'Generic Form Panel',
   bodyStyle  : 'padding: 10px; background-color: #DFE8F6',
   labelWidth : 100,
   width      : 300,
   margins    : '0 0 0 3',
   items      : [{
      xtype : 'textfield',
      fieldLabel : 'Student Name',
      name       : 'name'
   },{
      xtype : 'textfield',
      fieldLabel : 'Age',
      name       : 'age'
   },{
      xtype : 'textfield',
      fieldLabel : 'Marks',
      name       : 'marks'
   }]
});
// Creation of display panel for showing both grid and form
var displayPanel = Ext.create('Ext.Panel', {
   width    : 600,
   height   : 200,
   layout   : 'border',
   renderTo : 'panel',
   bodyPadding: '5',
   items    : [grid, formPanel],
   bbar    : [
      '->',
      {
      text    : 'Reset',
               handler : function() {
                  gridStore.loadData(myData);
                  formPanel.getForm().reset();
               }
            }]
         });

         var formPanelDropTargetEl =  formPanel.body.dom;
        //Creation of tager variable for drop.
         var formPanelDropTarget = Ext.create('Ext.dd.DropTarget',
formPanelDropTargetEl, {
            ddGroup: 'GridExample',
            notifyEnter: function(ddSource, e, data) {
               formPanel.body.stopAnimation();
               formPanel.body.highlight();
            },
            notifyDrop  : function(ddSource, e, data){
               var selectedRecord = ddSource.dragData.records[0];
               formPanel.getForm().loadRecord(selectedRecord);
               ddSource.view.store.remove(selectedRecord);
               return true;
            }
         });
      });
   </script>
</head>
<body>
   <div id = "panel" > </div>
</body>
</html>
```

The above program will produce the following result –



Student Data Grid

Student Name	Age	Marks
Asha	16	90
Vinit	18	95
Anand	20	68
Niharika	21	86
Manali	22	57

Generic Form Panel

Student Name:

Age:

Marks:

11. Ext.js – Themes

Ext.js provides a number of themes to be used in your applications. You can add a different theme in place of a classic theme and see the difference in the output. This is done simply by replacing the theme CSS file as explained ahead.

Neptune Theme

Consider your very first Hello World application. Remove the following CSS from the application.

```
https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css
```

Add the following CSS to use the Neptune theme.

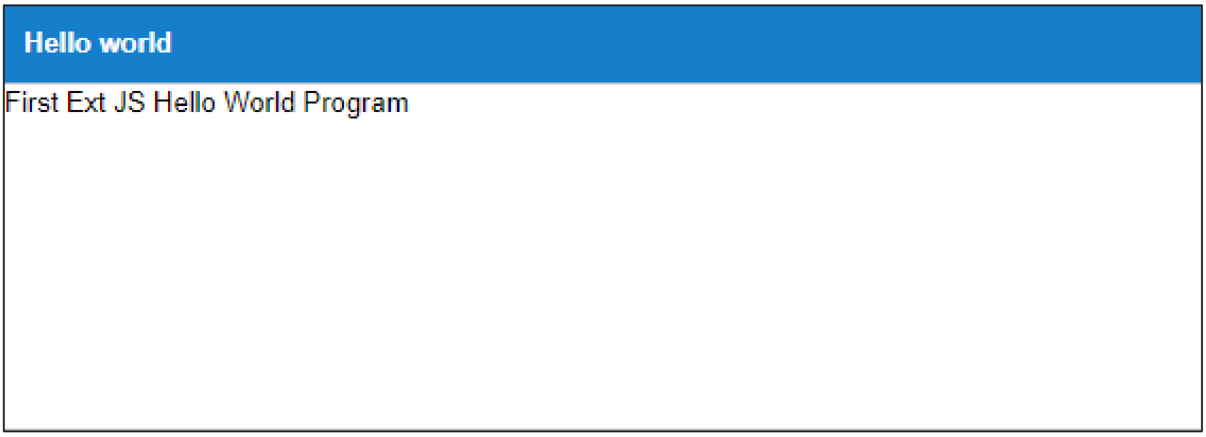
```
https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
neptune/resources/theme-neptune-all.css
```

To see the effect, try the following program.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
neptune/resources/theme-neptune-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function() {
         Ext.create('Ext.Panel', {
            renderTo: 'helloWorldPanel',
            height: 200,
            width: 600,
            title: 'Hello world',
            html: 'First Ext JS Hello World Program'
            });
         });
      </script>
   </head>
   <body>
```

```
      <div id="helloWorldPanel" />
   </body>
</html>
```

The above program will produce the following result -



Hello world

First Ext JS Hello World Program

Crisp Theme

Consider your very first Hello World application. Remove the following CSS from the application.

```
https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-classic/resources/theme-classic-all.css
```

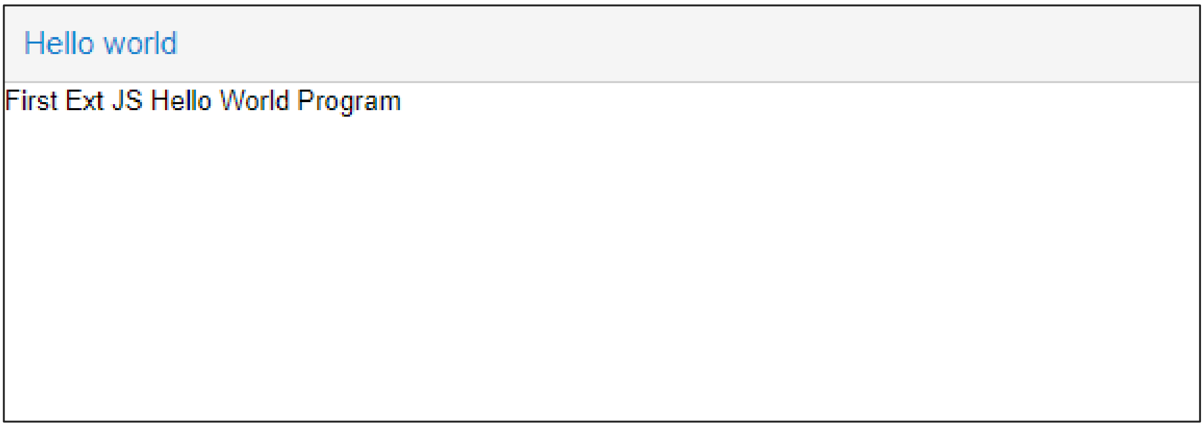
Add the following CSS to use the Neptune theme.

```
https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-crisp/resources/theme-crisp-all.css
```

To see the effect, try the following program.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-crisp/resources/theme-crisp-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function() {
         Ext.create('Ext.Panel', {
            renderTo: 'helloWorldPanel',
                height: 200,
                width: 600,
                title: 'Hello world',
                html: 'First Ext JS Hello World Program'
                });
            });
        </script>
    </head>
    <body>
        <div id="helloWorldPanel" />
    </body>
</html>
```

The above program will produce the following result -


Hello world

First Ext JS Hello World Program

Triton Theme

Consider your very first Hello World application. Remove the following CSS from the application.

```
https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css
```

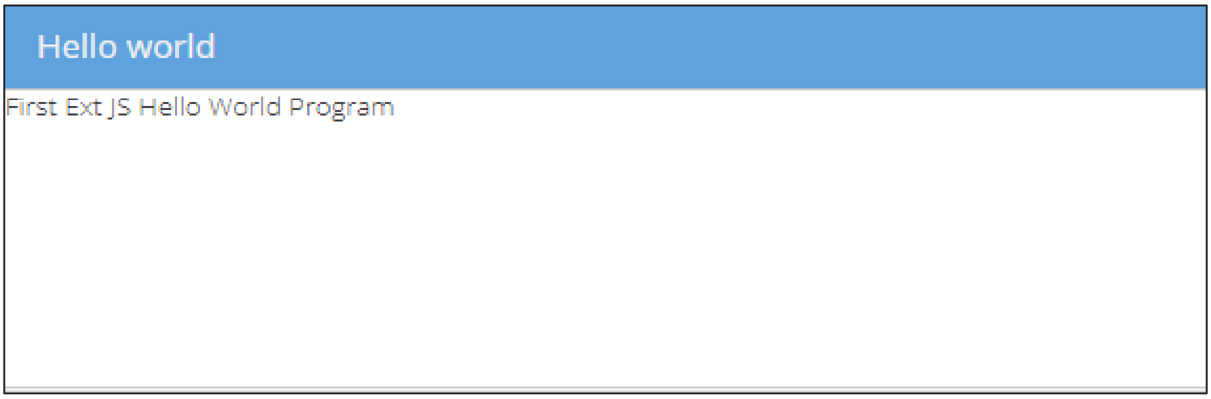
Add the following CSS to use the Triton theme.

```
https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
triton/resources/theme-triton-all.css
```

To see the effect, try the following program.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
triton/resources/theme-triton-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function() {
         Ext.create('Ext.Panel', {
            renderTo: 'helloWorldPanel',
            height: 200,
            width: 600,
            title: 'Hello world',
            html: 'First Ext JS Hello World Program'
            });
         });
      </script>
   </head>
   <body>
      <div id="helloWorldPanel" />
   </body>
</html>
```

The above program will produce the following result -



Hello world

First Ext JS Hello World Program

Gray Theme

Consider your very first Hello World application. Remove the following CSS from the application.

```
https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css
```

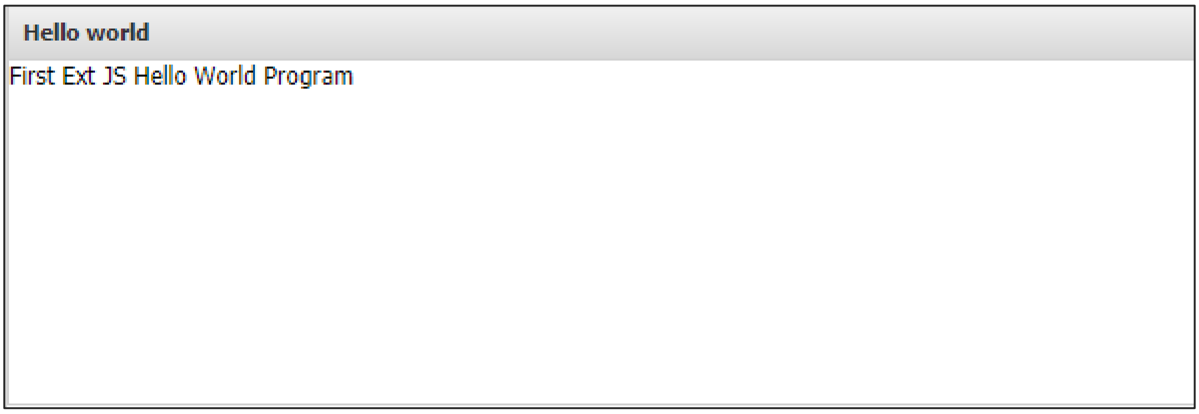
Add the following CSS to use the Gray theme.

```
https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
gray/resources/theme-gray-all.css
```

To see the effect, try the following program.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
gray/resources/theme-gray-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function() {
         Ext.create('Ext.Panel', {
            renderTo: 'helloWorldPanel',
            height: 200,
            width: 600,
            title: 'Hello world',
            html: 'First Ext JS Hello World Program'
            });
         });
      </script>
   </head>
   <body>
      <div id="helloWorldPanel" />
   </body>
</html>
```

The above program will produce the following result -



Hello world

First Ext JS Hello World Program

12. Ext.js – Custom Events and Listeners

Events are something which get fired when something happens to the class. For example, when a button is getting clicked or before/after the element is rendered.

Methods of Writing Events

- Built-in events using listeners
- Attaching events later
- Custom events

Built-in Events Using Listeners

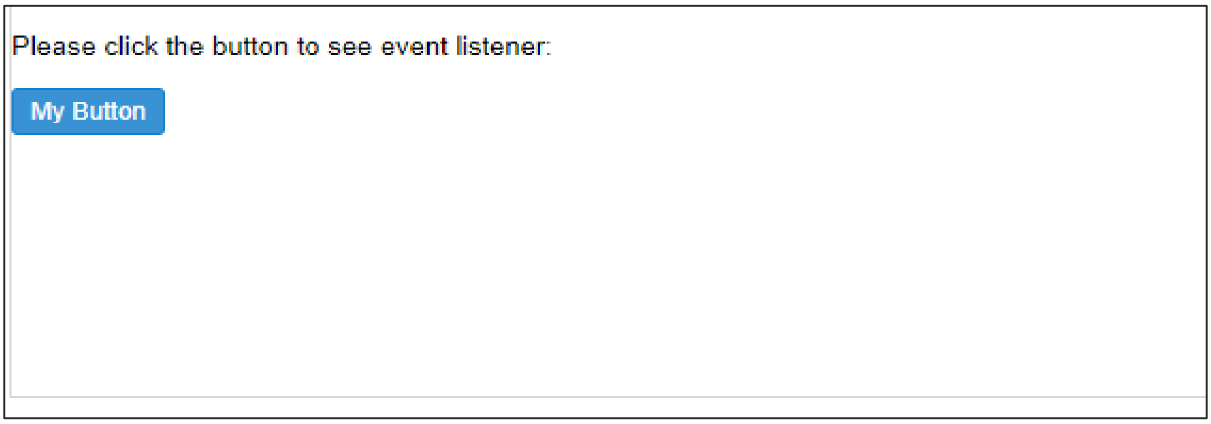
Ext JS provides listener property for writing events and custom events in Ext JS files.

Writing listener in Ext JS

We will add the listener in the previous program itself by adding a listen property to the panel.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-neptune/resources/theme-neptune-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function(){
            Ext.create('Ext.Button', {
               renderTo: Ext.getElementById('helloWorldPanel'),
               text: 'My Button',
               listeners: {
                  click: function() {
                     Ext.MessageBox.alert('Alert box', 'Button is clicked');
                  }
               }
            });
         });
      </script>
   </head>
   <body>
      <p> Please click the button to see event listener </p>
      <div id = 'helloWorldPanel' />   <!--  panel will be rendered here-- >
   </body>
</html>
```

The above program will produce the following result -



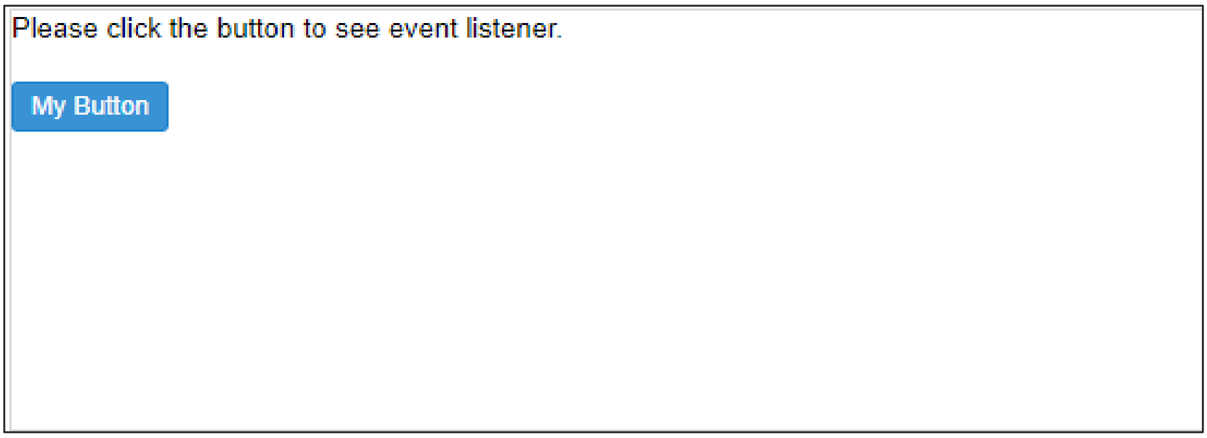
Please click the button to see event listener:

This way we can also write multiple events in listeners property.

Multiple Events in the Same Listener

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
neptune/resources/theme-neptune-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function(){
            Ext.get('tag2').hide()
            Ext.create('Ext.Button', {
               renderTo: Ext.getElementById('helloWorldPanel'),
               text: 'My Button',
               listeners: {
                  click: function() {
                     this.hide();
                  },
                  hide: function() {
                     Ext.get('tag1').hide();
                     Ext.get('tag2').show();
                  }
               }
            });
         });
      </script>
   </head>
   <body>
   <div id = "tag1">Please click the button to see event listener.</div>
   <div id = "tag2">The button was clicked and now it is hidden.</div>
   <div id = 'helloWorldPanel' />   <!--  panel will be rendered here-- >
   </body>
</html>
```

The above program will produce the following result -



Please click the button to see event listener.

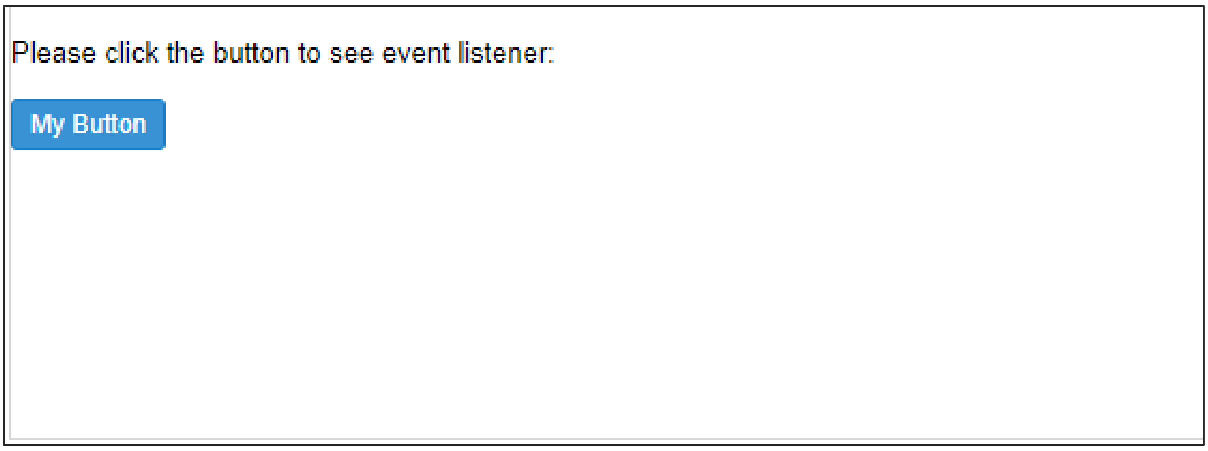
Attaching an Event Later

In the previous method of writing events, we have written events in listeners at the time of creating elements. The other way is to attach events.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
neptune/resources/theme-neptune-all.css" rel="stylesheet" />
```

```
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function(){
            var button = Ext.create('Ext.Button', {
               renderTo: Ext.getElementById('helloWorldPanel'),
            text: 'My Button'
            });

            // This way we can attach event to the button after the button is created.
            button.on('click', function() {
               Ext.MessageBox.alert('Alert box', 'Button is clicked');
            });
         });
      </script>
   </head>
   <body>
      <p> Please click the button to see event listener </p>
      <div id = 'helloWorldPanel' />   <!--  panel will be rendered here-- >
   </body>
</html>
```

The above program will produce the following result -



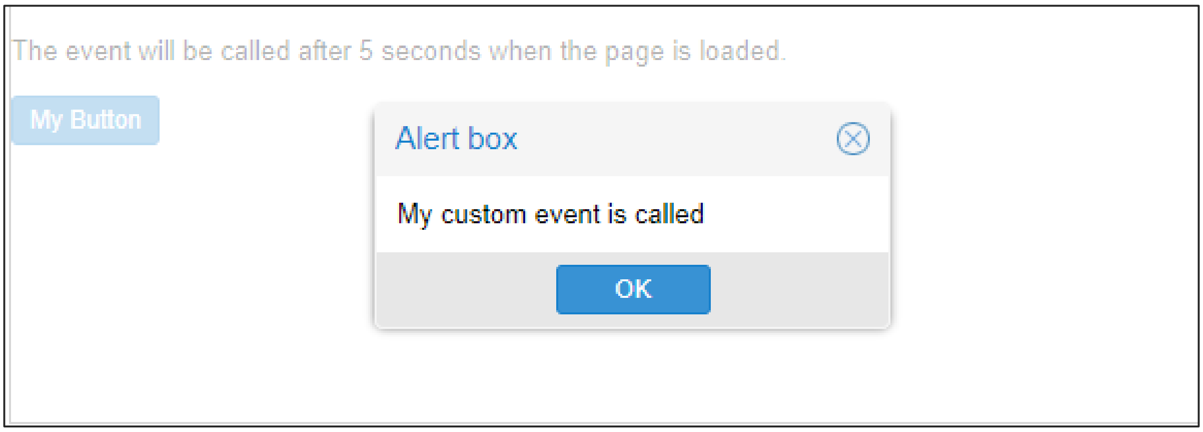
Please click the button to see event listener:

Custom Events

We can write custom events in Ext JS and fire the events with fireEvent method. Following example explains how to write custom events.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
neptune/resources/theme-neptune-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function(){
            var button = Ext.create('Ext.Button', {
               renderTo: Ext.getElementById('helloWorldPanel'),
               text: 'My Button',
               listeners: {
                  myEvent: function(button) {
                     Ext.MessageBox.alert('Alert box', 'My custom event is called');
                  }
               }
            });
            Ext.defer(function() {
               button.fireEvent('myEvent');
            }, 5000);
         });
      </script>
   </head>
   <body>
      <p> The event will be called after 5 seconds when the page is loaded. </p>
      <div id = 'helloWorldPanel' />   <!--  panel will be rendered here-- >
   </body>
</html>
```

Once the page is loaded and the document is ready, the UI page with a button will appear and as we are firing an event after 5 secs, the document is ready. The alert box will appear after 5 seconds.



The event will be called after 5 seconds when the page is loaded.

My custom event is called

Here, we have written the custom event 'myEvent' and we are firing events as button.fireEvent(eventName);

13. Ext.js – Data

Data package is used for loading and saving all the data in the application.

Data package has numerous number of classes but the most important classes are:

- Model
- Store
- Proxy

Model

The base class for model is **Ext.data.Model**. It represents an entity in an application. It binds the store data to view. It has mapping of backend data objects to the view dataIndex. The data is fetched with the help of store.

Creating a Model

For creating a model, we need to extend Ext.data.Model class and we need to define the fields, their name, and mapping.

```
Ext.define('StudentDataModel', {
   extend: 'Ext.data.Model',
   fields: [
   {name: 'name', mapping : 'name'},
   {name: 'age', mapping : 'age'},
   {name: 'marks', mapping : 'marks'}
   ]
});
```

Here, the name should be the same as the dataIndex, which we declare in the view and the mapping should match the data, either static or dynamic from the database, which is to be fetched using store.

Store

The base class for store is **Ext.data.Store**. It contains the data locally cached, which is to be rendered on view with the help of model objects. Store fetches the data using proxies, which has the path defined for services to fetch the backend data.

Store data can be fetched in two ways - static or dynamic.

Static Store

For static store, we will have all the data present in the store as shown in the following code.

```
   Ext.create('Ext.data.Store', {
      model: 'StudentDataModel',
      data: [
         { name : "Asha", age : "16", marks : "90" },
         { name : "Vinit", age : "18", marks : "95" },
         { name : "Anand", age : "20", marks : "68" },
         { name : "Niharika", age : "21", marks : "86" },
         { name : "Manali", age : "22", marks : "57" }
      ];
   });
```

Dynamic Store

Dynamic data can be fetched using proxy. We can have proxy which can fetch data from Ajax, Rest. and Json.

Proxy

The base class for proxy is Ext.data.proxy.Proxy. Proxy is used by Models and Stores to handle the loading and saving of Model data.

There are two types of proxies:

- Client Proxy
- Server Proxy

Client Proxy

Client proxies include Memory and Local Storage using HTML5 local storage.

Server Proxy

Server proxies handle data from the remote server using Ajax, Json data, and Rest service.

DEFINING PROXIES IN THE SERVER

```
Ext.create('Ext.data.Store', {
   model: 'StudentDataModel',
   proxy : {
      type : 'rest',
      actionMethods : {
         read : 'POST'  // Get or Post type based on requirement
      },
      url : 'restUrlPathOrJsonFilePath', // here we have to include the rest
URL path which fetches data from database or Json file path where the data is
stored

      reader: {

         type : 'json',  // the type of data which is fetched is of JSON type

         root : 'data'

      },

   }

});
```

14. Ext.js – Fonts

Ext.js provides the facility to use different font packages. Font packages are used to add different classes for icons available in the package.

- Font-Awesome
- Font-Pictos

Font-Awesome

ExtJS new theme, Triton, has an inbuilt font family font-awesome included in the framework itself, hence we do not need any explicit requirement for the font-awesome stylesheet.

Following is an example of using Font-Awesome classes in the Triton theme.

Font-Awesome with Triton theme

Font-Awesome Normal Theme

With the Triton theme, we do not need any explicit requirement of font-awesome stylesheet. We can directly use font-awesome classes as font-awesome is included in the Triton theme inbuilt.

Syntax

Just add font-awesome class to iconCls as:

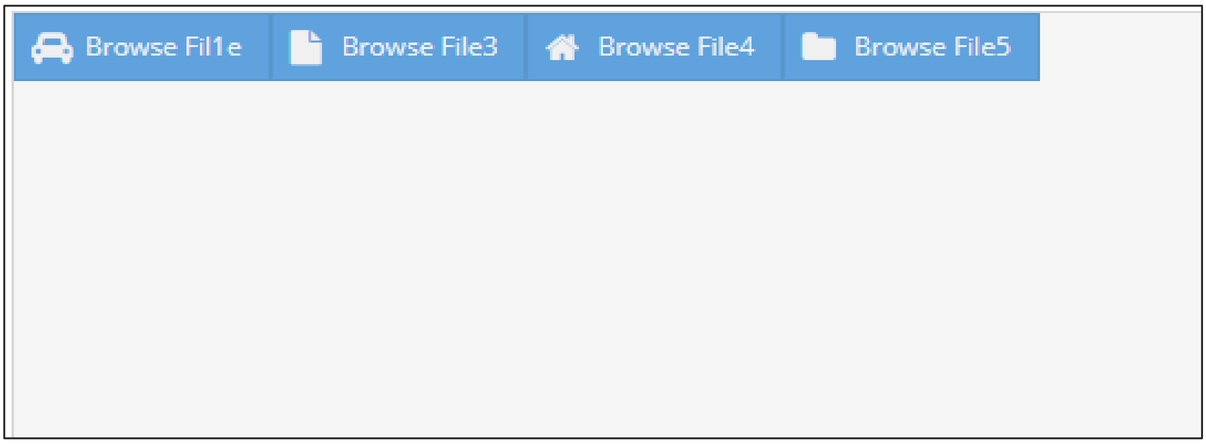
```
iconCls : 'fa fa-car'
```

Example

Following is a simple example to add font-awesome class.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
triton/resources/theme-triton-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.container.Container', {
            renderTo : Ext.getBody(),
            layout : 'auto' ,
            width : 600,
            items : [{
               xtype     : 'button',
               iconCls   : 'fa fa-car',
               text      : 'Browse Fil1e'
            },{
               xtype     : 'button',
               iconCls   : 'fa fa-file',
               text      : 'Browse File3'
            },{
               xtype     : 'button',
               iconCls   : 'fa fa-home',
               text      : 'Browse File4'
            },{
               xtype     : 'button',
               iconCls   : 'fa fa-folder',
               text      : 'Browse File5'
            }]
         });
      });
   </script>
</head>
<body>
   <div id = "panel" > </div>
</body>
</html>
```

The above program will produce the following result –



When we are using any other theme apart from Triton, we need or require to add a stylesheet for font-awesome explicitly.

Following is an example of using Font-Awesome classes without the Triton theme.

Font-Awesome with normal theme(Any theme other then Triton theme)

Font-Awesome Normal Theme

When we are using any theme which is not the Triton theme, then we need to add font-awesome stylesheet explicitly in our project.

Syntax

Add CDN file for font-awesome style in your html page.

```
   <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css" rel="stylesheet" />
```

Now add the class as:

```
iconCls : 'fa fa-car'
```

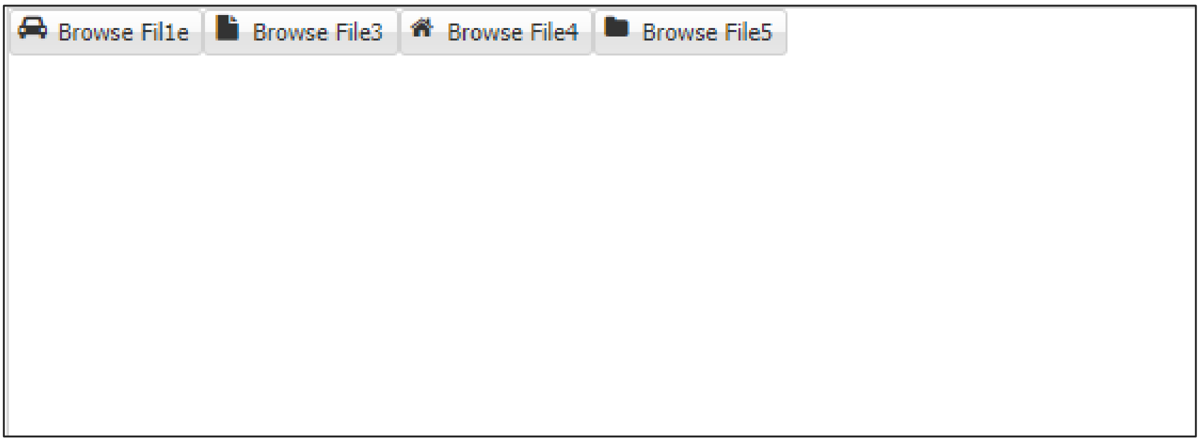
Example

Following is a simple example to add font-awesome class.

```
<!DOCTYPE html>

<html>

<head>

   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-classic/resources/theme-classic-all.css" rel="stylesheet" />

   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
```

```
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-
awesome.min.css" rel="stylesheet"  />
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.container.Container', {
            renderTo : Ext.getBody(),
            layout : 'auto' ,
            width : 600,
            items : [{
               xtype     : 'button',
               iconCls   : 'fa fa-car',
               text      : 'Browse Fil1e'
            },{
               xtype     : 'button',
               iconCls   : 'fa fa-file',
               text      : 'Browse File3'
            },{
               xtype     : 'button',
               iconCls   : 'fa fa-home',
               text      : 'Browse File4'
            },{
               xtype     : 'button',
               iconCls   : 'fa fa-folder',
               text      : 'Browse File5'
            }]
         });
      });
   </script>
</head>
<body>
   <div id = "panel" > </div>
</body>
</html>
```

The above program will produce the following result –



Font-Pictos

Font-pictos is not included in the framework for ExtJS, hence we have to require the same. Only licenced users of Sencha will have the benefit to use font-pictos.

Steps to Add Font-Pictos

Step 1: Require font-pictos class using the following command.

```
"requires": ["font-pictos"]
```

Step 2: Now add pictos classes as:

```
iconCls: 'pictos pictos-home'
```

15. Ext.js – Style

Application Styling refers to user adjustment of the look and feel of the components. These adjustments may include: color, color gradients, font, margins/padding, etc. Ext JS 6 has a new way of styling the application.

It uses SCSS for styling. SCSS is a more dynamic way of writing the CSS code. We can write the variables in our stylesheet with the help of this. However, a browser cannot understand SCSS. It can only understand CSS, hence all SCSS files should get compiled into CSS to a production-ready code.

Thus, SCSS file is called preprocessor files. In Ext.js, compilation is done through Sencha CMD tool. Sencha CMD compiles it manually only once using the following command.

```
sencha app build [development]
```

Global_CSS is the main CSS file, which has all SCSS variables associated with it in ExtJS that can be used in our application for customizing our theme by providing different value based on our need.

Following are some of the CSS variables available in the Global_CSS in Ext.js.

Sr. No.	Variable & Description
1	**$base-color** $base-color: color (e.g. $base-color : #808080) This base color is to be used throughout the theme
2	**$base-gradient** $base-gradient: string (e.g. $base-gradient : 'matte') This base gradient is to be used throughout the theme
3	**$body-background-color** $body-background-color: color (e.g. $body-background-color : #808080) Background color to apply to the body element. If set to transparent or 'none', no background-color style will be set on the body element
4	**$color** $color: color (e.g. $color : #808080) This default text color is to be used throughout the theme
5	**$font-family** $font-family: string (e.g. $font-family : arial) This default font-family is to be used throughout the theme
6	**$font-size** $font-size: number (e.g. $font-size : 9px) This default font-size is to be used throughout the theme
7	**$font-weight** $font-weight: string/number (e.g. $font-weight : normal) This default font-weight is to be used throughout the theme

8	**$font-weight-bold** $font-weight-bold: string/number (e.g. $font-weight-bold : bold) This default font-weight for bold font is to be used throughout the theme
9	**$include-chrome** $include-chrome: boolean (e.g. $include-chrome : true) True to include Chrome specific rules
10	**$include-ff** $include-ff: boolean (e.g. $include-ff : true) True to include Firefox specific rules
11	**$include-ie** $include-ie: boolean (e.g. $include-ie : true) True to include Internet Explorer specific rules for IE9 and lower
12	**$include-opera** $include-opera: boolean (e.g. $include-opera : true) True to include Opera specific rules
13	**$include-safari** $include-safari: boolean (e.g. $include-safari : true) True to include Opera specific rules
14	**$include-webkit** $include-webkit: boolean (e.g. $include-webkit : true) True to include Webkit specific rules

16. Ext.js – Drawing

Drawing package in ExtJS enables you to draw general purpose graphics. This can be used for graphics that work on all browsers and mobile devices.

Sr. No.	Drawing
1	**Circle** This graphics is used to create a circulare shape
2	**Rectangle** This graphics is used to create a rectanglar shape
3	**Arc** This graphics is used to create an arc shape
4	**Ellipse** This graphics is used to create an ellipse shape
5	**EllipticalArc** This graphics is used to create an elliptical arc shape
6	**Image** This graphics is used to add an image to your application
7	**Path** This graphics is used to create a free path
8	**Text** This graphics is used to add any text to your application.
9	**Translate after render** This property is used to move the starting point in your container, after the graphic is rendered. It can be used with any graphics.
10	**Rotation** This property is used to add a rotation to the drawing added. It can be used with any graphics.
11	**Square** This graphic is used to create a square.

Circle Drawing

This is used to create a circular shape.

Syntax

Following is a simple syntax to add a circle.

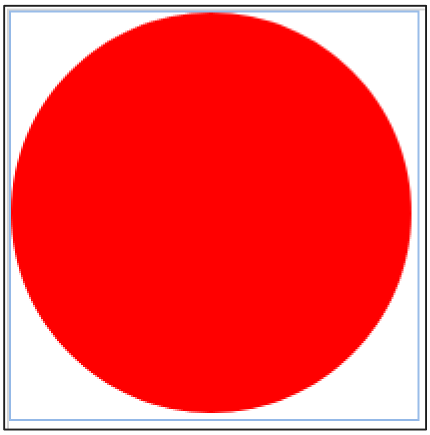
```
xtype: 'draw',
type: 'circle'
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create({
            xtype: 'draw',
            sprites: [{
               type: 'circle',
               fillStyle: 'red',
               r: 100,
               cx: 100,
               cy: 100
            }],
            height:205,
            width:205,
            renderTo:Ext.getBody()
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Rectangle Drawing

This type is used for creating a rectanglar shape of drawing.

Syntax

Following is a simple syntax to add a rectangle.

```
xtype: 'draw',
type: 'rect'
```

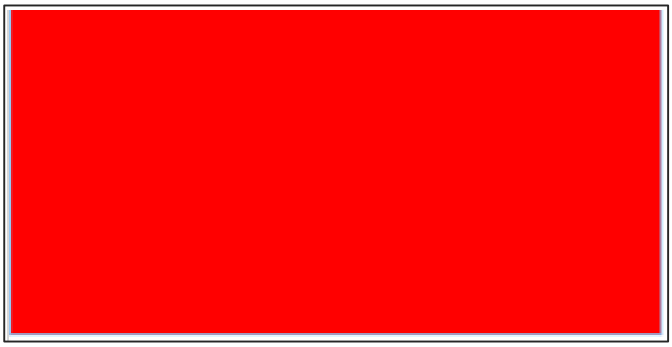
Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
```

```
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create('Ext.draw.Component', {
            sprites: [{
               type: 'rect',
               height : 200,
               width: 600,
               fill: 'red'
            }],
            height:150,
            width:300,
            renderTo:Ext.getBody()
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Arc Drawing

This is used to create an arc shape.

Syntax

Following is a simple syntax to add an arc.

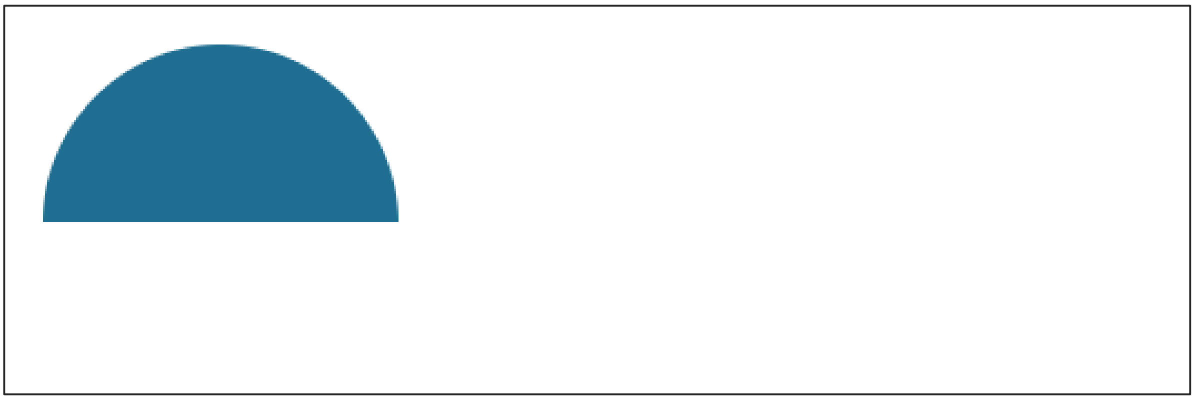
```
xtype: 'draw',
type: 'arc'
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create({
            xtype: 'draw',
            renderTo: Ext.getBody(),
            width: 600,
            height: 400,
            sprites: [{
               type: 'arc',
               cx: 100,
               cy: 100,
               r: 80,
               fillStyle: '#1F6D91',
               startAngle: 0,
               endAngle: Math.PI,
               anticlockwise: true
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Ellipse Drawing

This is used to create an elliptical shape.

Syntax

Following is a simple syntax to add an ellipse.

```
xtype: 'draw',
type: 'ellipse',
```

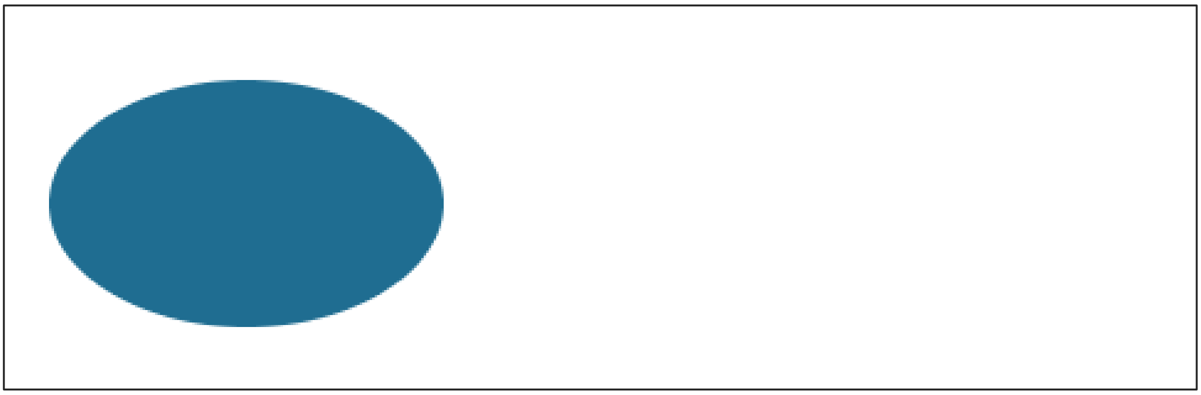
Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
```

```
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create({
            xtype: 'draw',
            renderTo: Ext.getBody(),
            width: 600,
            height: 200,
            sprites: [{
               type: 'ellipse',
               cx: 100,
               cy: 100,
               rx: 80,
               ry: 50,
               fillStyle: '#1F6D91'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Elliptical Arc Drawing

This is used to create an elliptical Arc shape.

Syntax

Following is a simple syntax to add an elliptical arc.

```
xtype: 'draw',
type: 'ellipticalArc',
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create({
            xtype: 'draw',
            renderTo: Ext.getBody(),
            width: 600,
            height: 200,
            sprites: [{
               type: 'ellipticalArc',
               cx: 100,
               cy: 100,
               rx: 80,
               ry: 50,
               fillStyle: '#1F6D91',
               startAngle: 0,
               endAngle: Math.PI,
                anticlockwise: true
             }]
          });
       });
    </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –

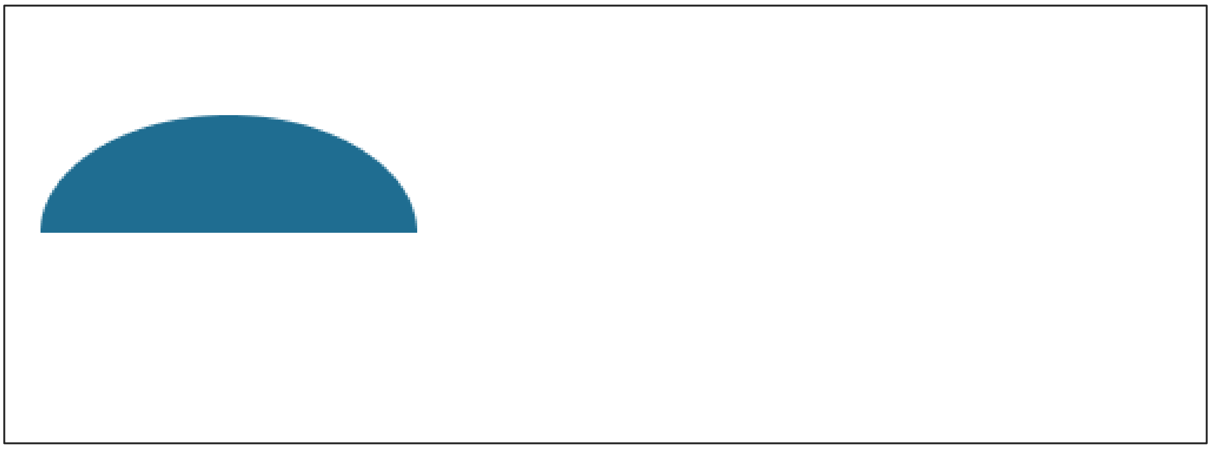


Image Drawing

This is used to add an image to the drawing.

Syntax

Following is a simple syntax to add an image.

```
xtype: 'draw',
type: 'image'
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
```

```
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create({
            xtype: 'draw',
            renderTo: Ext.getBody(),
            width: 600,
            height: 200,
            sprites: [{
               type: 'image',
               src : '/extjs/images/1.jpg'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –

Path Drawing

This is used to create a freepath.

Syntax

Following is a simple syntax to add a path.

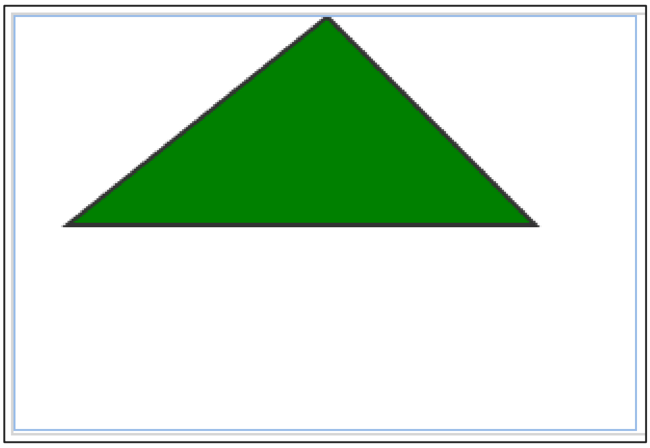
```
xtype: 'draw',
type: 'path'
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create({
            xtype: 'draw',
            renderTo: Ext.getBody(),
            width: 300,
            height:200,
            sprites: [{
               type: 'path',
               path: 'M150 0 L25 100 L250 100 Z',
               strokeStyle: '#333',
               fill: 'green',
               lineWidth: 2
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Text Drawing

This type of draw element is used to draw any text.

Syntax

Following is a simple syntax to add text.

```
xtype: 'draw',
type: 'text'
```

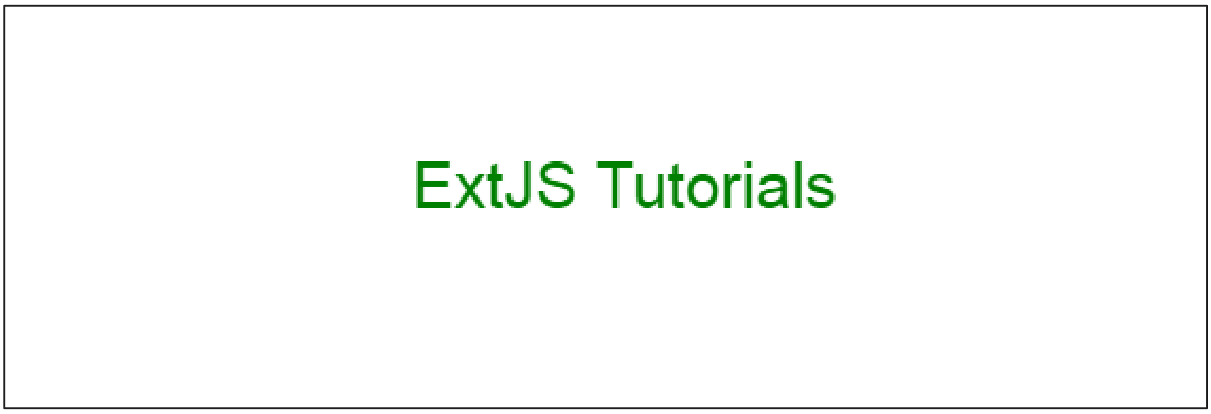
Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
```

```
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create({
            xtype: 'draw',
            renderTo: Ext.getBody(),
            width: 600,
            height: 200,
            sprites: [{
                type: 'text',
                x: 200,
                y: 100,
                text: 'ExtJS Tutorials',
                fontSize: 30,
                fillStyle: 'green'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



ExtJS Tutorials

Translate Drawing

Translate property is used for translation in x and y, once the drawing is rendered.

Syntax

Following is a simple syntax to add translate.

```
xtype: 'draw',

translate: {x:10, y:10}
```

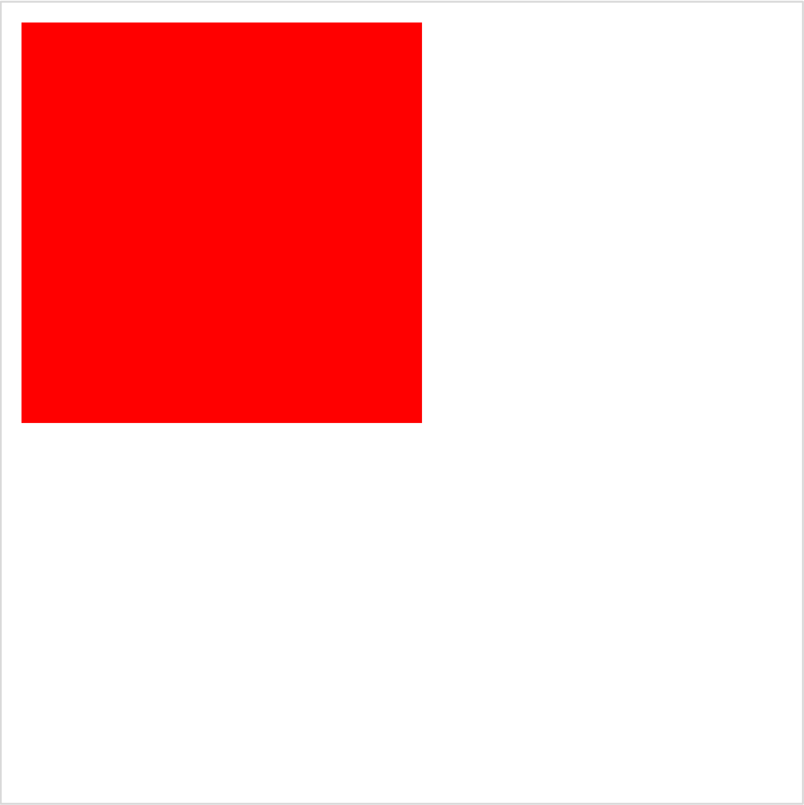
Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>

<html>

<head>

   <link 
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />

   <script type="text/javascript" 
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>

   <script type="text/javascript" 
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>

   <script type="text/javascript">

      Ext.onReady(function() {

         Ext.create({

            xtype: 'draw',

            renderTo: Ext.getBody(),

            width: 400,

            height: 400,

            sprites: [{

               type: 'rect',

               height : 200,

               width: 200,

               fill: 'red',

               translate: {

                  x:10,

                  y:10

               }
```

```
             }],
          });
       });
    </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Rotate Drawing

This property is used to rotate the drawing by providing an angle in degree.

Syntax

Following is a simple syntax to add a degree.

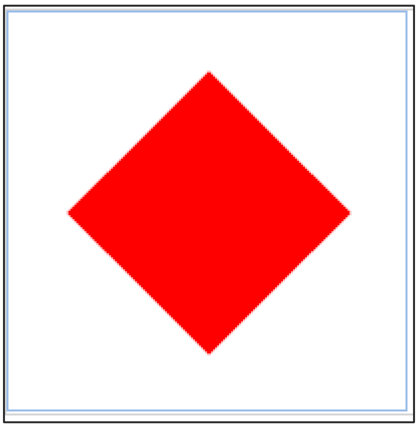
```
xtype: 'draw',
rotate: {degrees: 45}
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create({
            xtype: 'draw',
            renderTo: Ext.getBody(),
            width: 200,
            height: 200,
            sprites: [{
               type: 'rect',
               height: 100,
               width: 100,
               x:50,
               y:50,
               fill: 'red',
               rotate: {
                  degrees: 45
               }
            }],
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



Square Drawing

This is used to create a square.

Syntax

Following is a simple syntax to add a square.

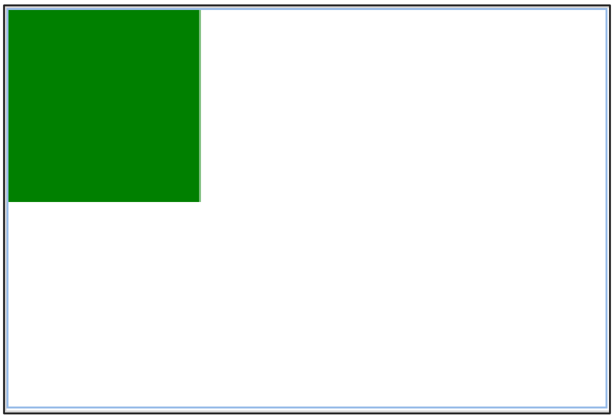
```
xtype: 'draw',
type: 'square'
```

Example

Following is a simple example showing the usage.

```
<!DOCTYPE html>
<html>
<head>
   <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/packages/charts/classic
/charts.js"></script>
   <script type="text/javascript">
      Ext.onReady(function() {
         Ext.create({
            xtype: 'draw',
            renderTo: Ext.getBody(),
            width: 300,
            height:200,
            sprites: [{
               type: 'square',
               height: 100,
               width: 100,
               fill: 'green'
            }]
         });
      });
   </script>
</head>
<body>
</body>
</html>
```

The above program will produce the following result –



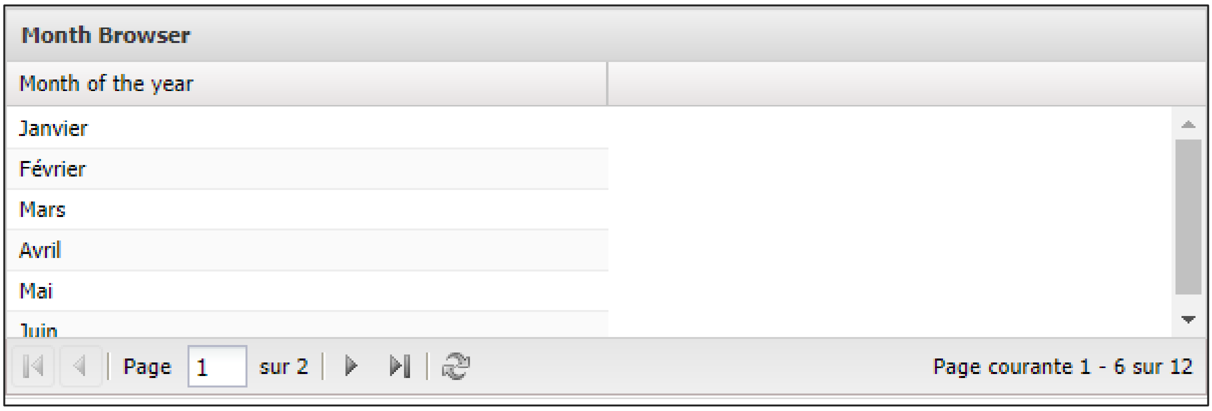
17. Ext.js – Localization

It is always best to communicate with the users in the language they understand and prefer. Extjs localization package supports over 40 languages such as German, French, Korean, Chinese, etc. It is very simple to implement the locale in ExtJs. You'll find all of the bundled locale files in the override folder of the ext-locale package. Locale files just overrides that tells Ext JS to replace the default English values of certain components.

The following program is to show the month in different locale to see the effect. Try the following program.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-
fr.js"></script>
      <script type="text/javascript">
          Ext.onReady(function() {
         var monthArray = Ext.Array.map(Ext.Date.monthNames, function (e)
{ return [e]; });
         var ds = Ext.create('Ext.data.Store', {
            fields: ['month'],
            remoteSort: true,
            pageSize: 6,
            proxy: {
               type: 'memory',
               enablePaging: true,
               data: monthArray,
               reader: {type: 'array'}
            }
         });
         Ext.create('Ext.grid.Panel', {
            renderTo: 'grid',
            id : 'gridId',
            width: 600,
            height: 200,
            title:'Month Browser',
            columns:[{
               text: 'Month of the year',
               dataIndex: 'month',
               width: 300
            }],
            store: ds,
            bbar: Ext.create('Ext.toolbar.Paging', {
               pageSize: 6,
               store: ds,
               displayInfo: true
            })
         });
         Ext.getCmp('gridId').getStore().load();
      });
      </script>
   </head>
   <body>
      <div id="grid" />
   </body>
</html>
```

The above program will produce the following result -



Month Browser

Month of the year
Janvier
Février
Mars
Avril
Mai
Juin

Page 1 sur 2

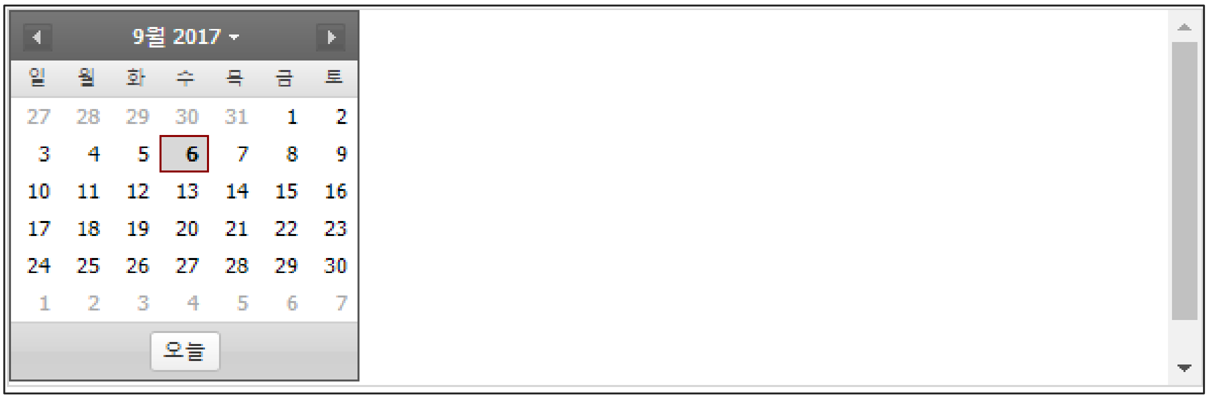
Page courante 1 - 6 sur 12

For using different locale other than English, we would need to add the locale specific file in our program. Here we are using https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-fr.js for French. You can use different locale for different languages such as https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-ko.js for korean, etc.

The following program is to show the date picker in Korean locale to see the effect. Try the following program.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
classic/resources/theme-classic-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-
ko.js"></script>
      <script type="text/javascript">
          Ext.onReady(function() {
         Ext.create('Ext.picker.Date', {
            renderTo: 'datePicker'
         });
      });
      </script>
   </head>
   <body>
      <div id="datePicker" />
   </body>
</html>
```

The above program will produce the following result -



일	월	화	수	목	금	토
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Following table lists the few locales available in ExtJS and the main file locale URL to be changed.

Locale	Language	Locale URL
ko	Korean	https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-ko.js
fr	French	https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-fa.js
es	Spanish	https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-es.js
ja	Japanese	https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-ja.js
it	Italian	https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-it.js
ru	Russian	https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-ru.js
zh_CN	Simplifies Chinese	https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/locale/locale-zh_CN.js

18. Ext.js – Accessibility

In general accessibility means availability, the content is accessible means the content is available.

In software terms, the application is accessible means the application is available for all. Here, all means the persons with disabilities, the visually impaired or those who use screen readers to use a computer or those who prefer to navigate using the keyboard instead of using a mouse.

Applications which are accessible are called ARIA (Accessible Rich Internet Applications).

Accessibility in Ext JS

Ext JS is designed to keep this in mind that it should work with all keyboard navigations. It has built-in tab indexing and focus-ability, and it is always on by default so we do not need to add any property to enable this functionality.

This functionality allows all keyboard-enabled components to interact with the user when tabbed into. For example, we can use tab for moving on to the next component instead of a mouse. Same way, we can use shift+tab for moving backward and use enter on the keyboard to click, etc.

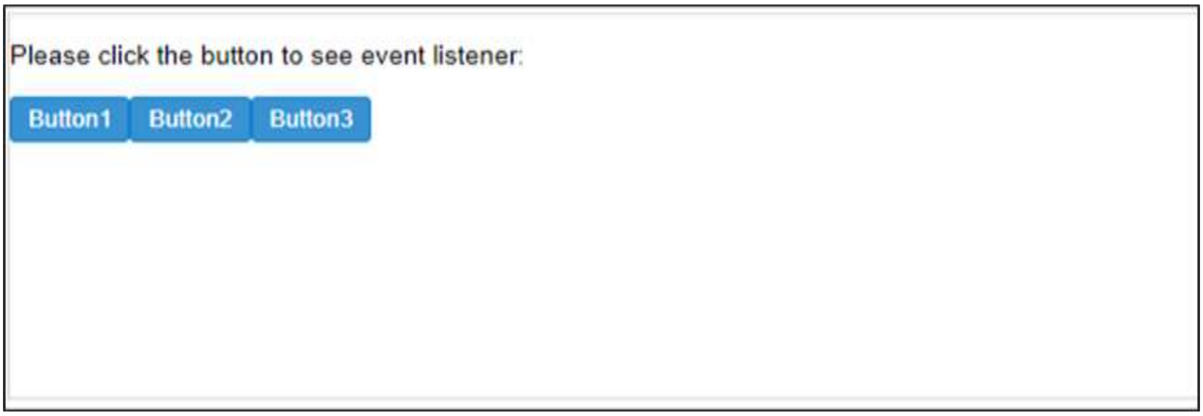
Focus Styling and Tabs

Focus is inbuilt in Extjs when using keystroke for tabbing.

Following example shows how to the style changes, when the focus changes with the tabs.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
crisp/resources/theme-crisp-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.onReady(function(){
            Ext.create('Ext.Button', {
               renderTo: Ext.getElementById('button1'),
               text: 'Button1',
               listeners: {
                  click: function() {
                     Ext.MessageBox.alert('Alert box', 'Button 1 is clicked');
                  }
               }
            });
             Ext.create('Ext.Button', {
               renderTo: Ext.getElementById('button2'),
               text: 'Button2',
               listeners: {
                  click: function() {
                     Ext.MessageBox.alert('Alert box', 'Button 2 is clicked');

                  }
               }
            });
             Ext.create('Ext.Button', {
               renderTo: Ext.getElementById('button3'),
               text: 'Button3',
               listeners: {
                  click: function() {
                     Ext.MessageBox.alert('Alert box', 'Button 3 is clicked');

                  }
               }
            });
         });
      </script>
   </head>
   <body> <p>Please click the button to see event listener:</p>
      <span id="button3"/>
      <span id="button2"/>
      <span id="button1"/>
   </body>
</html>
```

To see the effect, use tab for moving from the next button and shft+tab for focusing backward. Use enter and see how the focused button's related alert would pop up.



Please click the button to see event listener:

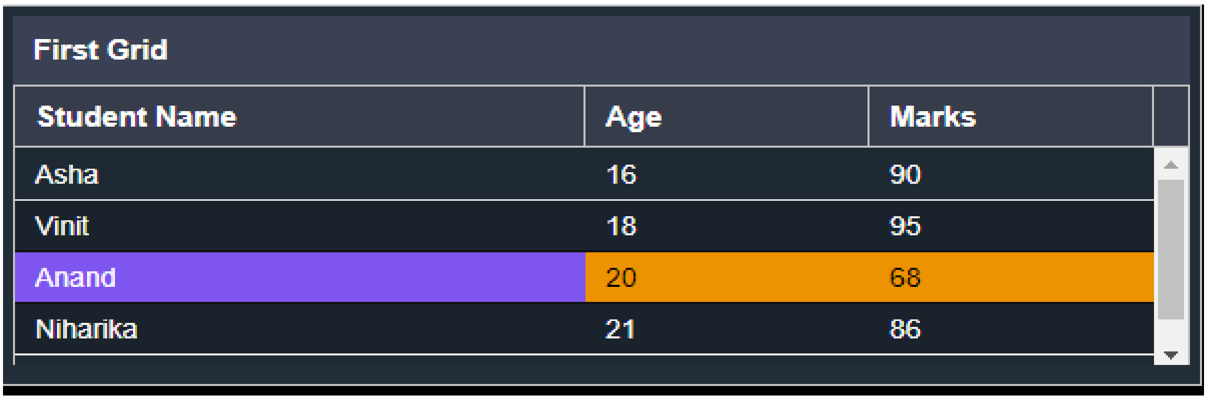
ARIA Theme

ExtJS provides the theme aria for the visually impaired.

Following example shows the aria theme which is easily accessible for the visually impaired.

```
<!DOCTYPE html>
<html>
   <head>
      <link
href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-
aria/resources/theme-aria-all.css" rel="stylesheet" />
      <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
      <script type="text/javascript">
         Ext.require([
            'Ext.grid.*',
            'Ext.data.*'
         ]);
         // Creation of data model
         Ext.define('StudentDataModel', {
            extend: 'Ext.data.Model',
            fields: [
            {name: 'name', mapping : 'name'},
            {name: 'age', mapping : 'age'},
            {name: 'marks', mapping : 'marks'}
            ]
         });

         Ext.onReady(function(){
            // Store data
            var myData = [
               { name : "Asha", age : "16", marks : "90" },
               { name : "Vinit", age : "18", marks : "95" },
               { name : "Anand", age : "20", marks : "68" },
               { name : "Niharika", age : "21", marks : "86" },
               { name : "Manali", age : "22", marks : "57" }
            ];
            // Creation of first grid store
            var firstGridStore = Ext.create('Ext.data.Store', {
            model: 'StudentDataModel',
            data: myData
            });
            // Creation of first grid
            var firstGrid = Ext.create('Ext.grid.Panel', {
               store            : firstGridStore,
               columns          :
                  [{
                     header: "Student Name",
                     dataIndex: 'name',
                     id : 'name',
                     flex:  1,
                  sortable: true
                  },{
                     header: "Age",
                     dataIndex: 'age',
                     flex: .5,
                     sortable: true
                  },{
                     header: "Marks",
                     dataIndex: 'marks',
                     flex: .5,
                     sortable: true
                  }],
               stripeRows       : true,
               title            : 'First Grid',
               margins          : '0 2 0 0'
            });
            // Creation of a panel to show both the grids.
            var displayPanel = Ext.create('Ext.Panel', {
               width        : 600,
               height       : 200,
               layout       : {
                  type: 'hbox',
                  align: 'stretch',
                  padding: 5
               },
               renderTo     : 'panel',
               defaults     : { flex : 1 },
               items        : [
                  firstGrid
               ]
            });
         });
      </script>
   </head>
   <body>
      <div id = "panel" > </div>
   </body>
</html>
```

The above program will produce the following result. You can use tab and mouse up and down keys for moving the focus across the grid and the theme is basically for the visually impaired people.



First Grid

Student Name	Age	Marks
Asha	16	90
Vinit	18	95
Anand	20	68
Niharika	21	86

19. Ext.js – Debugging Ext JS code

Any JavaScript code can be debugged using **alert()** box or **console.log()** or with the debug pointer in a debugger.

Alert Box

Place an alert box in the code where you want to check the flow or any variable value. For example, alert('message to show' + variable);

Development/Debugging Tool

Debugger is the most important tool for any developer to check the issue and error in the code while developing.

Ext JS is a JavaScript framework, hence it can be easily debugged using developer tools provided by or specific to different browsers. All the major browsers have their developer tools available to test and debug JavaScript code.

Popular debuggers are IE development tool for IE, firebug for firefox, and chrome development tool for Chrome browser.

Chrome debugger comes with Chrome browser, however, firebug has to be installed specifically as it doesn't come as a package with firefox.

Here is a link to install firebug for firefox browser http://getfirebug.com

The shortcut to open the development tool in Windows OS is F12 keyboard key.

Debugging JS Code in Debugger

There are two ways to debug JavaScript code.

- The first way, is to place **console.log()** in the code and see the value of the log, which will be printed in the console of the development tool.

- The second way is by using breakpoints in the development tool. Following is the process.

    - Open the file in all the available scripts under script tag.

    - Now place a breakpoint to the line you want to debug.

    - Run the application in the browser.

    - Now, whenever the code flow will reach this line, it will break the code and stay there until the user runs the code by keys F6 (go to the next line of the code), F7 (go inside the function) or F8 (go to the next breakpoint or run the code if there is no more breakpoints) based on the flow you want to debug.

    - You can select the variable or the function you want to see the value of.

    - You can use the console to check the value or to check some changes in the browser itself.

20. Ext.js – Methods

Following are a few inbuilt functions, which are heavily used in Ext JS.

Ext.is Class

This class checks the platform you are using, whether it is a phone or a desktop, a mac or Windows operating system. These are the following methods related to Ext.is class.

Sr. No.	Methods & Description

1	**Ext.is.Platforms**
	This function returns the platform available for this version.
	For example, when you run the following function, it returns something like this:
	`[Object { property="platform", regex=RegExp, identity="iPhone"}, Object`
	`{ property="platform", regex=RegExp, identity="iPod"}, Object`
	`{ property="userAgent", regex=RegExp, identity="iPad"}, Object`
	`{ property="userAgent", regex=RegExp, identity="Blackberry"}, Object`
	`{ property="userAgent", regex=RegExp, identity="Android"}, Object`
	`{ property="platform", regex=RegExp, identity="Mac"}, Object`
	`{ property="platform", regex=RegExp, identity="Windows"}, Object`
	`{ property="platform", regex=RegExp, identity="Linux"}]`

2	**Ext.is.Android** This function will return true, if you are using Android operating system, else it returns false.
3	**Ext.is.Desktop** This function will return true, if you are using a desktop for the application, else it returns false.
4	**Ext.is.Phone** This function will return true, if you are using a mobile, else it returns false.
5	**Ext.is.iPhone** This function will return true if you are using iPhone, else it returns false.
6	**Ext.is.iPod** This function will return true, if you are using iPod, else it returns false.
7	**Ext.is.iPad** This function will return true, if you are using an iPad, else it returns false.
8	**Ext.is.Windows** This function will return true, if you are using Windows operating system, else it returns false.

9	**Ext.is.Linux** This function will return true, if you are using Linux operating system, else it returns false.
10	**Ext.is.Blackberry** This function will return true, if you are using Blackberry, else it returns false.
11	**Ext.is.Mac** This function will return true, if you are using Mac operating system, else it returns false.

Ext.supports Class

As the name indicates, this class provides information if the feature is supported by the current environment of the browser/device or not.

Sr. No.	Methods & Description
1	**Ext.supports.History** It checks if the device supports HTML 5 history as window.history or not. If the device supports history, then it returns **true**, else false.
2	**Ext.supports.GeoLocation** It checks if the device supports geolocation method or not. Internally it checks for navigator.geolocation method.
3	**Ext.supports.Svg** It checks if the device supports HTML 5 feature scalable vector graphics (svg) method or not. Internally it checks for doc.createElementNS && !!doc.createElementNS("http:/" + "/www.w3.org/2000/svg", "svg").createSVGRect.
4	**Ext.supports.Canvas** It checks if the device supports HTML 5 feature canvas to draw method or not. Internally it checks for doc.createElement('canvas').getContext and returns a value based on the output of this method.
5	**Ext.supports.Range** It checks if the browser supports document.createRange method or not.

Ext.String Class

Ext.String class has various methods to work with string data. The most used methods are encoding decoding, trim, toggle, urlAppend, etc.

Encoding Decoding function: These are the functions available in Ext.String class to encode and decode HTML values.

Sr. No.	Methods & Description

1	**Ext.String.htmlEncode**
	This function is used to encode html value to make it parsable.
	Example:
	`Ext.String.htmlEncode("< p > Hello World < /p >");`
	`Output - "&lt; p &gt; Hello World &lt; /p &gt;".`

2	**Ext.String.htmlDecode**
	This function is used to decode the encoded html value.
	Example:
	`Ext.String.htmlDecode("&lt; p &gt; Hello World &lt; /p &gt;");`
	`Output -  "< p > Hello World < /p >"`

3	**Ext.String.trim**
	This function is to trim the unwanted space in the string.
	Example:
	`Ext.String.trim(' hello ');`
	`Output – "hello"`

4	**Ext.String.urlAppend**
	This method is used to append a value in the URL string.
	Example:
	`Ext.String.urlAppend('https://www.google.com' , 'hello');`
	`Output - "https://www.google.com?hello"`
	`Ext.String.urlAppend('https://www.google.com?index=1' , 'hello');`
	`Output – "https://www.google.com?index=1&hello"`

5	**Ext.String.toggle**
	This function is to toggle the values between two different values.
	Example:
	`var toggleString = 'ASC'`
	`toggleString = Ext.String.toggle(a, 'ASC', 'DESC');`

	Output – DESC as toggleString had value ASC. Now again, if we print the same we will get toggleString = “ASC” this time, as it had value 'DESC'. It is similar to ternary operator toggleString = ((toggleString =='ASC')? 'DESC' : 'ASC');

Miscellaneous Methods

Sr.No.	Methods & Description
1	**Ext.userAgent()** This function gives information about browser userAgent. UserAgent is to identify the browser and the operating system to the web server. Example: If you are working in Mozilla, it returns something like: "mozilla/5.0 (windows nt 6.1; wow64; rv:43.0) gecko/20100101 firefox/43.0"
2	**Version related function** This function returns the version of the browser currently in use, if the function is called related to IE. In Firefox browser, it returns 0. These functions are Ext.firefoxVersion, Ext.ieVersion, etc. Example: If you are using Firefox browser and you call the method Ext.ieVersion for fetching the version of IE, then it returns 0. If you are using the same method in IE browser, then it will return the version you are using such as 8, 9, etc.
3	**Ext.getVersion()** This function returns the current Ext JS version in use. Example: If you call Ext.getVersion(), it returns an array of values such as version, short version, etc. Ext.getVersion().version returns the current version of Ext JS used in the program, such as “4.2.2".
4	**Browser related functions** These functions return Boolean values based on the browser in use. These methods are Ext.isIE, Ext.isIE6, Ext.isFF06, and Ext.isChrome. Example: If you are using Chrome browser, then the function Ext.isChrome will return true all, otherwise it will return false.
5	**Ext.typeOf()** This function returns the datatype of the variable. Example:

```
var a = 5;
var b  = 'hello';
Ext.typeOf(a);
Output – Number
Ext.typeOf(b);
```

Output - String

6

DataType related methods: These functions return boolean value based on the datatype of the variable.

Example:

```
var a = ['a', 'bc'];
var b = 'hello';
var c = 123;
var emptyVariable;
var definedVariable;
function extraFunction(){return true;}
```

Ext.isArray(a);	//returns true
Ext.isString(b);	//return true
Ext.isNumber(c);	//return true
Ext.isEmpty(emptyVariable);	//return true
Ext.isEmpty(b);	//return false
Ext.isDefined(definedVariable);	//return true
Ext.isfunction(extraFunction);	//return true