



QUERY UI

user interface framework

tutorialspoint
SIMPLY EASY LEARNING

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About the Tutorial

JqueryUI is the most popular front end frameworks currently. It is sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript.

This tutorial will teach you basics of JQueryUI Framework, which you can use to create complex web applications GUI with ease. This Tutorial is divided into sections such as JQueryUI Basic Structure, JQueryUI CSS, JQueryUI Layout Components and JQueryUI Plugins. Each of these sections contain related topics with simple and useful examples.

Audience

This tutorial has been prepared for anyone who has a basic knowledge of HTML and CSS and has an urge to develop websites. After completing this tutorial, you will find yourself at a moderate level of expertise in developing web projects using Twitter JQueryUI.

Prerequisites

Before you start proceeding with this tutorial, I'm making an assumption that you are already aware about basics of HTML and CSS. If you are not well aware of these concepts, then I will suggest to go through our short tutorial on [HTML Tutorial](#) and [CSS Tutorial](#).

Disclaimer & Copyright

© Copyright 2015 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute, or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

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

About the Tutorial	i
Audience	i
Prerequisites	i
Disclaimer & Copyright	i
Table of Contents	ii
1. JQuery – Introduction	1
Features	1
Benefits of JQueryUI	2
2. JQueryUI – Environment Setup	3
Download UI Library from Its Official Website	3
Custom Download with Download Builder	3
Stable Download	5
Legacy Download	5
Download UI Library from CDNs	5
Example	6
UNIT I – JQUERY UI INTERACTIONS	8
3. JQueryUI – Draggable	9
\$(selector, context).draggable (options) Method	9
Default Functionality	16
Use of Disable, Distance, and Delay	17
Constrain Movement	19
Move Content By Duplicating	20
Get Current Option Value	21
\$(selector, context).draggable ("action", [params]) Method	22
Example	23
Event Management on the Moved Elements	24
Example	25
4. JQueryUI – Droppable	27
\$(selector, context).draggable (options) Method	27
Default Functionality	30
Use of Disable, Distance, and Delay	31
Constrain Movement	32
Move Content By Duplicating	33
Get Current Option Value	34
\$(selector, context).draggable ("action", [params]) Method	35
Event Management On Droppable Elements	39
Example	42
5. JQueryUI – Resizable	45
\$(selector, context).resizable (options) Method	45
Default Functionality	47
Use of Animate and Ghost	48
Use of containment, minHeight, and minWidth	50
Use of delay, distance, and autoHide	52
Use of alsoResize	54
	ii

Use of Aspectratio, Grid.....	55
\$(Selector, Context).Resizable ("Action", Params) Method	57
Example	59
Event Management on Resizable Elements	61
Example	63
6. JQueryUI – Selectable.....	66
\$(selector, context).selectable (options) Method	66
Default Functionality	68
Use of Delay and Distance	69
Use of Filter	71
\$(selector, context).selectable ("action", params) Method	73
Example	75
Event Management on Selectable Elements	77
Example	79
7. JQueryUI – Sortable	82
\$(selector, context).sortable (options) Method	82
Default Functionality	90
Use of Options Delay and Distance.....	91
Use of Placeholder.....	93
Use of Options Connectwith and Droponempty	95
\$(selector, context).sortable ("action", [params]) Method	97
Event Management on The Sortable Elements	102
Example	112
UNIT II – JQUERYUI WIDGETS.....	116
8. JQueryUI – Accordion	117
\$(selector, context).accordion (options) Method	117
Default Functionality	121
Use of collapsible	124
Use of Heightstyle.....	126
Height style-content	129
Height style-Fill	129
\$(selector, context).accordion ("action", params) Method.....	130
Example	132
Event Management on Accordion Elements	135
Example	137
9. JQueryUI –AutoComplete.....	140
Syntax.....	140
\$(selector, context).autocomplete (options) Method	140
Default Functionality	142
Use of autoFocus	143
Use of minLength and delay	145
Use of Label	146
Use of External Source.....	147
\$(selector, context).autocomplete ("action", params) Method	149
Example	151
Extension Points	152
Event Management on Autocomplete Elements	153

Example	155
10. JQueryUI – Button.....	159
\$(selector, context).button (options) Method	159
\$(selector, context).button ("action", params) Method	160
Event Management on Buttons.....	161
Example	161
11. JQueryUI – DatePicker.....	163
\$(selector, context).datepicker (options) Method	163
Default Functionality	173
Inline Datepicker.....	174
Use of appendText, dateFormat, altField and altFormat	175
Use of beforeshowday	176
Use of showon, buttonimage, and buttonimageonly	177
Use of defaultDate, dayNamesMin, and duration.....	178
Use of prevText, nextText, showOtherMonths and selectOtherMonths	179
Use of changeMonth, changeYear, and numberOfMonths	180
Use of showWeek, yearSuffix, and showAnim	181
\$(selector, context).datepicker ("action", [params]) Method	182
Use of setDate() action	184
Use of show() action	184
Event Management on datepicker elements.....	185
12. JQueryUI – Dialog.....	186
\$(selector, context).dialog (options) Method.....	186
Default Functionality	188
Use of buttons, title and position	189
Use of hide, show and height	191
Use of Modal	192
\$(selector, context).dialog ("action", [params]) Method	194
Example	195
Event Management on Dialog Box	196
Use of beforeClose Event method	198
Use of resize Event method	200
Extension Points	201
13. JQueryUI – Menu	202
\$(selector, context).menu (options) Method	202
Default Functionality	203
Use of Icons And Position	204
\$(selector, context).menu ("action", params) Method	206
Use of Disable Method	209
Use of focus and collapseAll methods	210
Event Management on menu elements	212
Example	212
14. JQueryUI – Progress Bar	215
\$(selector, context).progressbar (options) Method	215
Default Functionality	216
Use of Max and Value	217
\$(selector, context).progressbar ("action", params) Method	218

Example	219
Event Management on Progress Bar Elements	221
Example	221
15. JQueryUI – Slider	224
\$(selector, context).slider (options) Method.....	224
Default Functionality	225
Use of value, animate, and orientation	226
Use of Range, Min, Max and Values	227
\$(selector, context).slider ("action", params) Method.....	228
Example	230
Event Management On Slider Elements	231
Example	232
16. JQueryUI – Spinner.....	235
\$(selector, context).spinner (options) Method	235
Default Functionality	236
Use of Min, Max, and Step Options	237
Use of icons Option	239
Use of culture, numberFormat, and page options	240
\$(selector, context).spinner ("action", params) Method	241
Use of action stepUp, stepDown, pageUp, and pageDown.....	242
Use of action enable, and disable.....	244
Event Management on Spinner Elements	245
Example	246
Extension Points	247
17. JQueryUI – Tabs	249
\$(selector, context).tabs (options) Method	249
Default Functionality	253
Use of heightStyle, collapsible, and hide	255
Use of Event.....	257
\$(selector, context).tabs ("action", params) Method.....	259
Use of Action Disable()	262
Use of Action Disable(Index)	264
Event Management on tabs elements.....	266
Example	269
18. JQueryUI – Tooltip.....	273
\$(selector, context).tooltip (options) Method.....	273
Default Functionality	276
Use of Content, Track, and Disabled.....	277
Use of Position	279
\$(selector, context).tooltip ("action", [params]) Method	280
Examples.....	282
Event Management on Tooltip Elements	283
Examples.....	284
UNIT III – JQUERYUI EFFECTS	287
19. JQueryUI – addClass().....	288
Examples.....	289

20. JQueryUI – Color Animation	292
21. JQueryUI – Effects	294
jQueryUI Effects	294
Examples	296
22. JQueryUI – Hide	299
jQueryUI Effects	299
Examples	301
23. JQueryUI – Remove Class	305
Examples	306
24. JQueryUI – Show	308
jQueryUI Effects	308
Examples	310
Show with Blind Effect	311
25. JQueryUI – switchClass.....	314
Examples	315
26. JQueryUI – Toggle	318
jQueryUI Effects	318
Example	320
27. JQueryUI – toggleClass	323
Added in Version 1.9 of jQueryUI	323
Examples	324
UNIT IV – JQUERYUI UTILITIES.....	326
28. JQueryUI – Position.....	327
Example	329
29. JQueryUI – Widget Factory.....	332
Base Widget.....	332
Options	332
Methods	334
Events	336
jQueryUI widget factory Lifecycle.....	336
Example	337
Creating Custom Widget.....	337
Adding Options To Custom Widget	338
Adding Methods to Custom Widget	339
Adding Events To Custom Widget	341

1. JQUERY – INTRODUCTION

JqueryUI is a powerful Javascript library built on top of jQuery JavaScript library. UI stands for User interface, It is a set of plug-ins for jQuery that adds new functionalities to the jQuery core library.

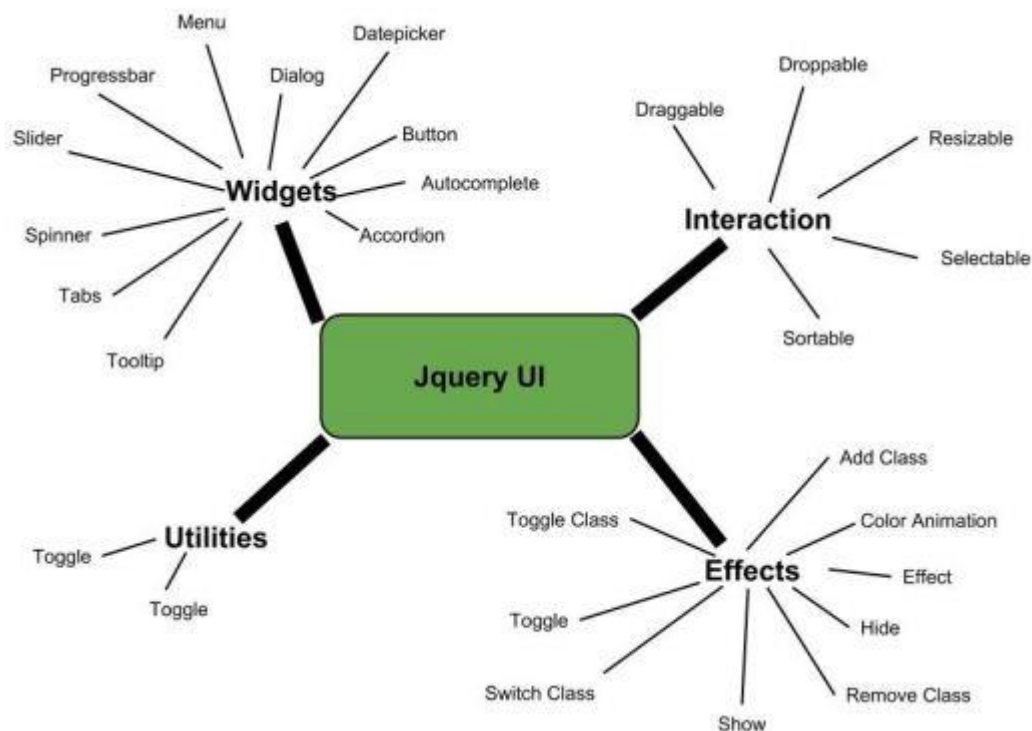
The set of plug-ins in JqueryUI includes interface interactions, effects, animations, widgets, and themes built on top of jQuery JavaScript Library.

It was released in September 2007, announced in a blog post by John Resig on jquery.com. The latest release, 1.10.4, requires jQuery 1.6 or later version. jQuery UI is a free, open source software, licensed under the MIT License.

Features

JqueryUI is categorized into four groups namely, **Interactions**, **Widgets**, **Effects**, and **Utilities**.

These will be discussed in detail in the subsequent chapters. The structure of the library is as shown in the image below:



- **Interactions** : These are the interactive plugins like drag, drop, resize and more which give the user the ability to interact with DOM elements.
- **Widgets** : Using widgets which are jQuery plugins, you can create user interface elements like accordion, datepicker, etc.

- **Effects** : These are built on the internal jQuery effects. They contain a full suite of custom animations and transitions for DOM elements.
- **Utilities** : These are a set of modular tools the JQueryUI library uses internally.

Benefits of JQueryUI

The below are some of the benefits of JQuery UI:

- Cohesive and Consistent APIs.
- Comprehensive Browser Support
- Open Source and Free to Use
- Good Documentation.
- Powerful Theming Mechanism.
- Stable and Maintenance Friendly.

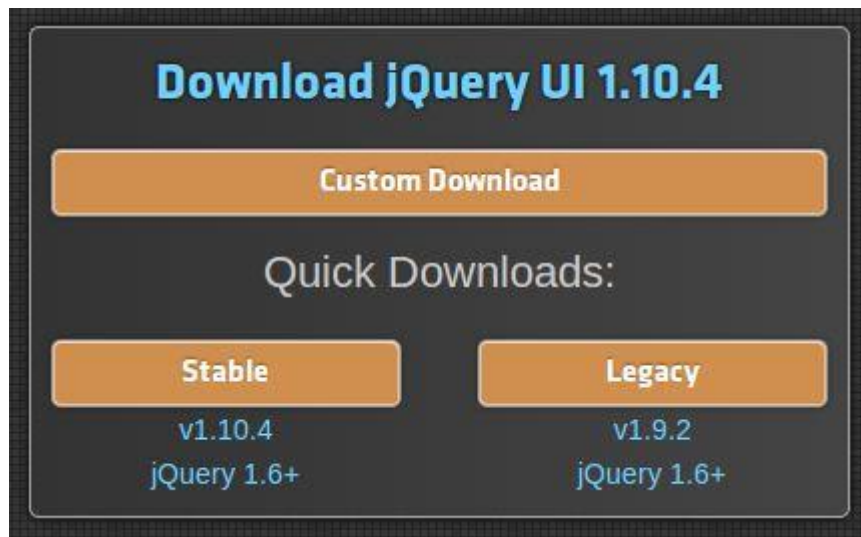
2. JQUERYUI – ENVIRONMENT SETUP

This chapter will discuss about download and set up of JQueryUI library. We will also briefly study the directory structure and its contents. JQueryUI library can be used in two ways in your web page:

- [Downloading UI Library from its official website](#)
- [Downloading UI Library from CDNs](#)

Download UI Library from Its Official Website

When you open the link <http://jqueryui.com/>, you will see there are three options to download JQueryUI library:



- **Custom Download** : Click on this button to download a customized version of library.
- **Stable** : Click on this button to get the stable and latest version of JQueryUI library.
- **Legacy** : Click on this button to get the previous major release of the JQueryUI library.

Custom Download with Download Builder

Using Download Builder, you can create a custom build to include only those portions of the library that you need. You can download this new customized version of JQueryUI, depending on the chosen theme. You will see the following screen (same page is split into two images):

[Demos](#)
[Download](#)
[API Documentation](#)
[Themes](#)
[Development](#)
[Support](#)
[Blog](#)
[About](#)

Download Builder

Quick downloads: [Stable \(Themes\) \(1.10.4 for jQuery 1.6+\)](#) | [Legacy \(Themes\) \(1.9.2 for jQuery 1.6+\)](#)
[All jQuery UI Downloads](#)

Version

☒ **1.10.4** (Stable for jQuery 1.6+)

☐ **1.9.2** (Legacy for jQuery 1.6+)

Components

☒ **Toggle All**

UI Core <input checked="" type="checkbox"/> Toggle All A required dependency, contains basic functions and initializers.	<input checked="" type="checkbox"/> Core <input checked="" type="checkbox"/> Widget <input checked="" type="checkbox"/> Mouse <input checked="" type="checkbox"/> Position	The core of jQuery UI, required for all interactions and widgets. Provides a factory for creating stateful widgets with a common API. Abstracts mouse-based interactions to assist in creating certain widgets. Positions elements relative to other elements.
Interactions <input checked="" type="checkbox"/> Toggle All These add basic behaviors to any element and are used by many components below.	<input checked="" type="checkbox"/> Draggable <input checked="" type="checkbox"/> Droppable <input checked="" type="checkbox"/> Resizable <input checked="" type="checkbox"/> Selectable <input checked="" type="checkbox"/> Sortable	Enables dragging functionality for any element. Enables drop targets for draggable elements. Enables resize functionality for any element. Allows groups of elements to be selected with the mouse. Enables items in a list to be sorted using the mouse.
Widgets <input checked="" type="checkbox"/> Toggle All Full-featured UI Controls - each has a range of options and is fully themeable.	<input checked="" type="checkbox"/> Accordion <input checked="" type="checkbox"/> Autocomplete <input checked="" type="checkbox"/> Button <input checked="" type="checkbox"/> Datepicker <input checked="" type="checkbox"/> Dialog <input checked="" type="checkbox"/> Menu <input checked="" type="checkbox"/> Progressbar <input checked="" type="checkbox"/> Slider <input checked="" type="checkbox"/> Spinner <input checked="" type="checkbox"/> Tabs <input checked="" type="checkbox"/> Tooltip	Displays collapsible content panels for presenting information in a limited amount of space. Lists suggested words as the user is typing. Enhances a form with themeable buttons. Displays a calendar from an input or inline for selecting dates. Displays customizable dialog windows. Creates nestable menus. Displays a status indicator for loading state, standard percentage, and other progress indicators. Displays a flexible slider with ranges and accessibility via keyboard. Displays buttons to easily input numbers via the keyboard or mouse. Transforms a set of container elements into a tab structure. Shows additional information for any element on hover or focus.
	<input checked="" type="checkbox"/> Fade Effect <input checked="" type="checkbox"/> Fold Effect <input checked="" type="checkbox"/> Highlight Effect <input checked="" type="checkbox"/> Pulse Effect <input checked="" type="checkbox"/> Scale Effect <input checked="" type="checkbox"/> Shake Effect <input checked="" type="checkbox"/> Slide Effect <input checked="" type="checkbox"/> Transfer Effect	Fades an element. Folds an element first horizontally and then vertically. Highlights the background of an element in a defined color for a custom duration. Pulsates an element n times by changing the opacity to zero and back. Grows or shrinks an element and its content. Restores an element to its original size. Shakes an element horizontally or vertically n times. Slides an element in and out of the viewport. Displays a transfer effect from one element to another.

Theme

Select the theme you want to include or [design a custom theme](#).

Theme Folder Name:

CSS Scope:

This is useful when you require only specific plugins or features of the JQueryUI library. The directory structure of this version is shown in the following figure:



Uncompressed files are located in the *development-bundle* directory. The uncompressed file is best used during development or debugging; the compressed file saves bandwidth and improves performance in production.

Stable Download

Click on the Stable button, which leads directly to a ZIP file containing the sources, examples, and documentation for latest version of JQueryUI library. Extract the ZIP file contents to a *jqueryui* directory.

This version contains all files including all dependencies, a large collection of demos, and even the library's unit test suite. This version is helpful to getting started.

Legacy Download

Click on the Legacy button, which leads directly to a ZIP file of previous major release of JQueryUI library. This version also contains all files including all dependencies, a large collection of demos, and even the library's unit test suite. This version is helpful to get you started.

Download UI Library from CDNs

A CDN or Content Delivery Network is a network of servers designed to serve files to users. If you use a CDN link in your web page, it moves the responsibility of hosting files from your own servers to a series of external ones. This also offers an advantage that if the visitor to your webpage has already downloaded a copy of JQueryUI from the same CDN, it won't have to be re-downloaded.

The [jQuery Foundation](#), [Google](#), and [Microsoft](#) all provide CDNs that host jQuery core as well as jQuery UI.

Because a CDN does not require you to host your own version of jQuery and jQuery UI, it is perfect for demos and experimentation.

We are using the CDN versions of the library throughout this tutorial.

Example

Now let us write a simple example using JQueryUI. Let us create an HTML file, copy the following content to the <head> tag:

```
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css"
rel="stylesheet">

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
```

Details of the above code are:

- The first line, adds jQuery UI theme (in our case *ui-lightness*) via CSS. This CSS will make our UI stylish.
- Second line, adds the jQuery library, as jQuery UI is built on top of jQuery library.
- Third line, adds the jQuery UI library. This enables jQuery UI in your page.

Now let's add some content to <head> tag:

```
<script type="text/javascript">
    $(function () {
        $('#dialogMsg').dialog();
    });
</script>
```

In the <body> add this:

```
<body>
    <form id="form1" runat="server">
        <div id="dialogMsg" title="First JQueryUI Example">
            Hello this is my first JQueryUI example.
        </div>
    </form>
</body>
```

The complete HTML code is as follows. Save it as **myfirstexample.html**

```
<!DOCTYPE html>
<head>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script type="text/javascript">
        $(function () {
```

```
        $('#dialogMsg').dialog();  
    });  
</script>  
</head>  
<body>  
    <form id="form1" runat="server">  
        <div id="dialogMsg" title="First JQueryUI Example">  
            Hello this is my first JQueryUI example.  
        </div>  
    </form>  
</body>  
<html>
```

Open the above page in your browser. It will produce the following screen.



Unit I – JQuery UI Interactions

3. JQUERYUI – DRAGGABLE

jQueryUI provides **draggable()** method to make any DOM element draggable. Once the element is draggable, you can move that element by clicking on it with the mouse and dragging it anywhere within the viewport.

Syntax

The **draggable()** method can be used in two forms:

[\\$\(selector, context\).draggable \(options\)](#) Method

[\\$\(selector, context\).draggable \("action", \[params\]\)](#) Method

`$(selector, context).draggable (options)` Method

The *draggable (options)* method declares that an HTML element can be moved in the HTML page. The *options* parameter is an object that specifies the behavior of the elements involved.

Syntax

```
$(selector, context).draggable(options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).draggable({option1: value1, option2: value2..... });
```

The following table lists the different *options* that can be used with this method:

Option	Description
addClasses	<p>If this option is set to false, it will prevent the ui-draggable class from being added in the list of selected DOM elements. By default its value is true.</p> <p>Syntax</p> <pre>\$(".selector").draggable({ addClasses: false });</pre>
appendTo	<p>Specifies the element in which the draggable helper should be appended to while dragging. By default its value is "parent".</p> <p>Syntax</p>

	<pre>\$(".selector").draggable({ appendTo: "body" });</pre>
axis	<p>This option constrains dragging to either the horizontal (x) or vertical (y) axis. Possible values: "x", "y".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ axis: "x" });</pre>
cancel	<p>You can use this option to prevent dragging from starting on specified elements. By default its value is "input,textarea, button,select,option".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ cancel: ".title" });</pre>
connectToSortable	<p>You can use this option to specify a list whose elements are interchangeable. At the end of placement, the element is part of the list. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ connectToSortable: "#my-sortable" });</pre>
<u>containment</u>	<p>Constrains dragging to within the bounds of the specified element or region. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ containment: "parent" });</pre>

<p>cursor</p>	<p>Specifies the cursor CSS property when the element moves. It represents the shape of the mouse pointer. By default its value is "auto".</p> <p>By default its value is "auto". Other possible values are:</p> <p>"crosshair" (across)</p> <p>"default" (an arrow)</p> <p>"pointer" (hand)</p> <p>"move" (two arrows cross)</p> <p>"e-resize" (expand to the right)</p> <p>"ne-resize" (expand up right)</p> <p>"nw-resize" (expand up left)</p> <p>"n-resize" (expand up)</p> <p>"se-resize" (expand down right)</p> <p>"sw-resize" (expand down left)</p> <p>"s-resize" (expand down)</p> <p>"auto" (default)</p> <p>"w-resize" (expand left)</p> <p>"text" (pointer to write text)</p>
---------------	---

	<p>"wait" (hourglass)</p> <p>"help" (help pointer)</p> <p>Syntax</p> <pre>\$(".selector").draggable({ cursor: "crosshair" });</pre>
cursorAt	<p>Sets the offset of the dragging helper relative to the mouse cursor. Coordinates can be given as a hash using a combination of one or two keys: { top, left, right, bottom }. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable(\$(".selector").draggable({ cursorAt: { left: 5 } }););</pre>
delay	<p>Delay, in milliseconds, after which the first movement of the mouse is taken into account. The displacement may begin after that time. By default its value is "0".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ delay: 300 });</pre>
disabled	<p>When set to true, disables the ability to move items. Items cannot be moved until this function is enabled (using the draggable ("enable") instruction). By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ disabled: true });</pre>

distance	<p>Number of pixels that the mouse must be moved before the displacement is taken into account. By default its value is "1".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ distance: 10 });</pre>
grid	<p>Snaps the dragging helper to a grid, every x and y pixels. The array must be of the form [x, y]. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ grid: [50, 20] });</pre>
handle	<p>If specified, restricts dragging from starting unless the mousedown occurs on the specified element(s). By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ handle: "h2" });</pre>
<u>helper</u>	<p>Allows for a helper element to be used for dragging display. By default its value is "original".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ helper: "clone" });</pre>
<u>iframeFix</u>	<p>Prevent iframes from capturing the mousemove events during a drag. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ iframeFix: true });</pre>

<u>opacity</u>	<p>Opacity of the element moved when moving. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ opacity: 0.35 });</pre>
<u>refreshPositions</u>	<p>If set to <i>true</i>, all droppable positions are calculated on every mousemove. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ refreshPositions: true });</pre>
<u>revert</u>	<p>Indicates whether the element is moved back to its original position at the end of the move. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ revert: true });</pre>
<u>revertDuration</u>	<p>Duration of displacement (in milliseconds) after which the element returns to its original position (see options.revert). By default its value is "500".</p> <p>Duration of displacement (in milliseconds) after which the element returns to its original position (see options.revert). By default its value is "500".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ revertDuration: 200 });</pre>
<u>scope</u>	<p>Used to group sets of draggable and droppable items, in addition to droppable's accept option. By default its value is "default".</p> <p>Syntax</p> <pre>\$(".selector").draggable(</pre>

	<pre>{ scope: "tasks" });</pre>
<u>scroll</u>	<p>When set to <i>true</i> (the default), the display will scroll if the item is moved outside the viewable area of the window. By default its value is "true".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ scroll: false });</pre>
scrollSensitivity	<p>Indicates how many pixels the mouse must exit the window to cause scrolling of the display. By default its value is "20".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ scrollSensitivity: 100 });</pre>
scrollSpeed	<p>Indicates the scrolling speed of the display once scrolling begins. By default its value is "20".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ scrollSpeed: 100 });</pre>
snap	<p>Adjusts the display of the item being moved on other elements (which are flown). By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ snap: true });</pre>
<u>snapMode</u>	<p>Specifies how the adjustment should be made between the moved element and those indicated in <i>options.snap</i>. By default its value is "both".</p> <p>Syntax</p>

	<pre>\$(".selector").draggable({ snapMode: "inner" });</pre>
snapTolerance	<p>Maximum number of pixels in the difference in position necessary to establish the adjustment. By default its value is "20".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ snapTolerance: 30 });</pre>
stack	<p>Controls the z-index of the set of elements that match the selector, always brings the currently dragged item to the front. Very useful in things like window managers. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ stack: ".products" });</pre>
zIndex	<p>Z-index for the helper while being dragged. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").draggable({ zIndex: 100 });</pre>

The following section will show you a few working examples of drag functionality.

Default Functionality

The following example demonstrates a simple example of draggable functionality passing no parameters to the **draggable()** method .

```
<!DOCTYPE html>
<head>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
```

```
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<style>
#draggable { width: 150px; height: 150px; padding: 0.5em; background:#eee;}
</style>
<script>
$(function() {
    $( "#draggable" ).draggable();
});
</script>
</head>
<body>
    <div id="draggable" class="ui-widget-content">
        <p>Drag me !!!</p>
    </div>
</body>
</html>
```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser that supports javascript. You should see the following output. Now, you can play with the result:

Drag me !!!

Use of Disable, Distance, and Delay

The following example shows the usage of three important options **(a) disabled** **(b) delay** and **(c) distance** in the drag function of JQueryUI.

```
<!DOCTYPE html>
<head>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>
    <div id="div1" style="border:solid 1px;background-color:gainsboro;">
        <span>You can't move me!</span><br /><br />
    </div>
```



```

<div id="div2" style="border:solid 1px;background-color:grey;">
  <span>
    Dragging will start only after you drag me for 50px
  </span>
  <br /><br />
</div>
<div id="div3" style="border:solid 1px;background-color:gainsboro;">
  <span>
    You have to wait for 500ms for dragging to start!
  </span>
  <br /><br />
</div>

<script>
  $("#div1 span").draggable (
    { disabled: true }
  );
  $("#div2 span").draggable (
    { distance: 50 }
  );
  $("#div3 span").draggable (
    { delay: 500 }
  );

</script>
</body>
</html>

```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser that supports javascript, you should see the following output. Now, you can play with the result:

You can't move me!

Dragging will start only after you drag me for 50px

You have to wait for 500ms for dragging to start!

Constrain Movement

The following example shows how to limit the movement of elements on the screen using **containment** option in the drag function of JQueryUI.

```
<!DOCTYPE html>
<head>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
  ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>
  <div id="div4" style="border:solid 1px;background-color:gainsboro;">
    <span>You can drag me only within this div.</span><br /><br />
  </div>
  <div id="div5" style="border:solid 1px;background-color:grey;">
    <span>You can drag me only along x axis.</span><br /><br />
  </div>
  <script>
    $("#div4 span").draggable ({
      containment : "#div4"
    });
    $("#div5 span").draggable ({
      axis : "x"
    });
  </script>
</body>
</html>
```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript. It should produce the following output. Now, you can play with the output:

You can drag me only within this div.

You can drag me only along x axis.

Here, `` elements are prevented from going outside a `<div>` whose ID is `div4`. You can also impose constraints on vertical or horizontal motion using options `axis` worth "x" or "y", which is also demonstrated.

Move Content By Duplicating

The following example demonstrates how to move an item that is the clone of the selected element. This is done using the option *helper* with value *clone*.

```
<!DOCTYPE html>
<head>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>
  <div id="div6" style="border:solid 1px;background:#eee; height:50px;">
    <span>You can duplicate me....</span>
  </div>
  <script>
    $("#div6 span").draggable ({
      helper : "clone"
    });
  </script>
</body>
</html>
```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

You can duplicate me....

As you can see when the first element is being dragged, only the cloned element moves, while the original item stays put. If you release the mouse, the cloned element disappears and the original item is still in its original position.

Get Current Option Value

The following example demonstrates how you can get a value of any option at any time during your script execution. Here we will read the value of **cursor** and **cursorAt** options set at the time of execution. Similar way you can get value of any other options available.

```
<!DOCTYPE html>
<head>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>
  <div id="divX" style="border:solid 1px;background:#eee; height:50px;">
    <span>Click anywhere on me to see cursor type...</span>
  </div>

  <script>
    /* First make the item draggable */
    $("#divX span").draggable();

    $("#divX span").bind('click', function( event ){
      var cursor = $( "#divX span" ).draggable( "option", "cursor" );
      var cursorAt = $( "#divX span" ).draggable( "option", "cursorAt" );
      alert("Cursor type - " + cursor + ", cursorAt - " + cursorAt);
    });
  </script>

</body>
</html>
```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Click anywhere on me to see cursor type...

\$ (selector, context).draggable ("action", [params]) Method

The *draggable (action, params)* method can perform an action on the movable elements, such as to prevent displacement. The **action** is specified as a string in the first argument and optionally, one or more **params** can be provided based on the given action.

Basically, Here actions are nothing but they are jQuery methods which we can use in the form of string.

Syntax

```
$(selector, context).draggable ("action", [params]);
```

The following table lists the actions for this method:

Action	Description
<u>destroy()</u>	Remove drag functionality completely. The elements are no longer movable. This will return the element back to its pre-init state. Syntax \$(".selector").draggable("destroy");
<u>disable()</u>	Disable drag functionality. Elements cannot be moved until the next call to the draggable("enable") method. Syntax \$(".selector").draggable("disable");
<u>enable()</u>	Reactivates drag management. The elements can be moved again. Syntax \$(".selector").draggable("enable");
<u>option(optionName)</u>	Gets the value currently associated with the specified <i>optionName</i> . Where <i>optionName</i> is name of the option to get and is of type <i>String</i> . Syntax var isDisabled = \$(".selector").draggable("option", "disabled");
<u>option()</u>	Gets an object containing key/value pairs representing the current draggable options hash. Syntax var options = \$(".selector").draggable("option");

<u><code>option(optionName, value)</code></u>	<p>Sets the <i>value</i> of the draggable option associated with the specified <i>optionName</i>. Where <i>optionName</i> is the name of the option to set and <i>value</i> is the value to set for the option.</p> <p>Syntax</p> <pre>\$(".selector").draggable("option", "disabled", true);</pre>
<u><code>option(options)</code></u>	<p>Sets one or more options for the draggable. Where <i>options</i> is a map of option-value pairs to set.</p> <p>Syntax</p> <pre>\$(".selector").draggable("option", { disabled: true });</pre>
<u><code>widget()</code></u>	<p>Returns a jQuery object containing the draggable element.</p> <p>Syntax</p> <pre>var widget = \$(".selector ").draggable("widget");</pre>

Example

Now let us see an example using the actions from the above table. The following example demonstrates the use of actions *disable* and *enable*.

```
<!DOCTYPE html>
<head>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>
  <div id="div7" style="border:solid 1px;background-color:gainsboro;">
    <span>You can't move me. Dragging is disabled.</span><br><br>
  </div>
  <div id="div8" style="border:solid 1px;background-color:grey;">
    <span>You can move me. Dragging is enabled.</span><br><br>
  </div>
  <script>
    $("#div7 span").draggable ();
    $("#div7 span").draggable ('disable');
    $("#div8 span").draggable ();
    $("#div8 span").draggable ('enable');
```

```

    </script>
  </body>
</html>

```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

```

You can't move me. Dragging is disabled.

You can move me. Dragging is enabled.

```

As you can see first element is disabled and the second element's dragging is enabled which you can try to drag.

Event Management on the Moved Elements

In addition to the `draggable (options)` method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
<u>create(event, ui)</u>	<p>Triggered when the draggable is created. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Syntax</p> <pre> \$.selector).draggable(create: function(event, ui) {}); </pre>
<u>drag(event, ui)</u>	<p>Triggered while the mouse is moved during the dragging. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i> like helper, position, offset.</p> <p>Syntax</p> <pre> \$.selector).draggable(drag: function(event, ui) {}); </pre>
<u>start(event, ui)</u>	<p>Triggered when dragging starts. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i> like helper, position, offset.</p> <p>Syntax</p> <pre> \$.selector).draggable(</pre>

	<pre>start: function(event, ui) {});</pre>
<u>stop(event, ui)</u>	<p>Triggered when dragging stops. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i> like helper, position, offset.</p> <p>Syntax</p> <pre>\$(".selector").draggable(stop: function(event, ui) {});</pre>

Example

The following example demonstrates the use of event method during drag functionality. This example demonstrates use of *drag* event.

```
<!DOCTYPE html>
<head>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>
    <div id="div9" style="border:solid 1px;background-color:gainsboro;">
        <span>Drag me to check the event method firing</span><br /><br />
    </div>
    <script>
        $("#div9 span").draggable ({
            cursor: "move",
            axis : "x",
            drag: function( event, ui ){
                alert("hi..");
            }
        });
    </script>
</body>
</html>
```


Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

Drag me to check the event method firing

Now try to drag the written content and you will see that **start** of a drag event gets fired which results in showing a dialogue box and cursor will change to move icon and text will move in X-axis only.

4. JQUERYUI – DROPPABLE

jQueryUI provides **draggable()** method to make any DOM element draggable. Once the element is draggable, you can move that element by clicking on it with the mouse and dragging it anywhere within the viewport.

Syntax

The **draggable()** method can be used in two forms:

- `$(selector, context).draggable (options) Method`
- `$(selector, context).draggable ("action", [params]) Method`

`$(selector, context).draggable (options) Method`

The *draggable (options)* method declares that an HTML element can be moved in the HTML page. The *options* parameter is an object that specifies the behavior of the elements involved.

Syntax

```
$(selector, context).draggable(options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).draggable({option1: value1, option2: value2..... });
```

The following table lists the different *options* that can be used with this method:

Option	Description
<u>accept</u>	<p>This option is used when you need to control which draggable elements are to be accepted for dropping. By default its value is <code>*</code> meaning that every item is accepted by droppable.</p> <p>This can be of type:</p> <p>Selector: This type indicates which draggable elements are accepted.</p> <p>Function: A callback function will be called for each draggable element on page. This function should return true if the draggable element is accepted by droppable.</p> <p>Syntax</p> <pre>\$(".selector").droppable({ accept: ".special" }</pre>

);
<u>activeClass</u>	<p>This option is a String representing one or more CSS classes to be added to the droppable element when an accepted element (one of those indicated in <i>options.accept</i>) is being dragged. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").droppable({ activeClass: "ui-state-highlight" });</pre>
<u>addClasses</u>	<p>This option when set to <i>false</i> will prevent the <i>ui-droppable</i> class from being added to the droppable elements. By default its value is true.</p> <p>This may be desired as a performance optimization when calling .droppable() init on hundreds of elements.</p> <p>Syntax</p> <pre>\$(".selector").droppable({ addClasses: false });</pre>
<u>disabled</u>	<p>This option when set to <i>true</i> disables the droppable. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").droppable({ disabled: true });</pre>

<p><u>greedy</u></p>	<p>This option is used when you need to control which draggable elements are to be accepted for dropping on nested droppables. By default its value is false. If this option is set to <i>true</i>, any parent droppables will not receive the element.</p> <p>Syntax</p> <pre>\$(".selector").droppable({ greedy: true });</pre>
<p><u>hoverClass</u></p>	<p>This option is <i>String</i> representing one or more CSS classes to be added to the element of droppable when an accepted element (an element indicated in <i>options.accept</i>) moves into it. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").droppable({ hoverClass: "drop-hover" });</pre>
<p><u>scope</u></p>	<p>This option is used to restrict the droppable action of draggable elements only to items that have the same <i>options.scope</i> (defined in draggable (options)). By default its value is "default".</p> <p>Syntax</p> <pre>\$(".selector").droppable({ scope: "tasks" });</pre>
<p><u>tolerance</u></p>	<p>This option is a <i>String</i> that specifies which mode to use for testing whether a draggable is hovering over a droppable. By default its value is "intersect".</p> <p>Possible values are:</p> <p>fit: Draggable covers the droppable element in full.</p> <p>intersect: Draggable overlaps the droppable element at least 50% in both directions.</p>

	<p>pointer: Mouse pointer overlaps the droppable element.</p> <p>touch: Draggable overlaps the droppable any amount of touching.</p> <p>Syntax</p> <pre>\$(".selector").droppable({ tolerance: "fit" });</pre>
--	---

The following section will show you a few working examples of drag functionality.

Default Functionality

The following example demonstrates a simple example of draggable functionality passing no parameters to the **draggable()** method .

```
<!DOCTYPE html>
<head>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <style>
    #draggable { width: 150px; height: 150px; padding: 0.5em; background:#eee;}
  </style>
  <script>
    $(function() {
      $( "#draggable" ).draggable();
    });
  </script>
</head>

<body>

  <div id="draggable" class="ui-widget-content">
    <p>Drag me !!!</p>
  </div>
```

```
</body>
</html>
```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Drag me !!!

Use of Disable, Distance, and Delay

The following example shows the usage of three important options **(a) disabled** **(b) delay** and **(c) distance** in the drag function of JQueryUI.

```
<!DOCTYPE html>
<head>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
  ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>
  <div id="div1" style="border:solid 1px;background-color:gainsboro;">
    <span>You can't move me!</span><br /><br />
  </div>
  <div id="div2" style="border:solid 1px;background-color:grey;">
    <span>
      Dragging will start only after you drag me for 50px
    </span>
    <br /><br />
  </div>
  <div id="div3" style="border:solid 1px;background-color:gainsboro;">
    <span>
      You have to wait for 500ms for dragging to start!
    </span>
    <br /><br />
  </div>

  <script>
```

```

        $("#div1 span").draggable (
            { disabled: true }
        );
        $("#div2 span").draggable (
            { distance: 50 }
        );
        $("#div3 span").draggable (
            { delay: 500 }
        );

    </script>
</body>
</html>

```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

You can't move me!

Dragging will start only after you drag me for 50px

You have to wait for 500ms for dragging to start!

Constrain Movement

The following example shows how to limit the movement of elements on the screen using **containment** option in the drag function of JQueryUI.

```

<!DOCTYPE html>
<head>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>

    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

</head>
<body>
    <div id="div4" style="border:solid 1px;background-color:gainsboro;">

```

```

    <span>You can drag me only within this div.</span><br /><br />
</div>
<div id="div5" style="border:solid 1px;background-color:grey;">
    <span>You can drag me only along x axis.</span><br /><br />
</div>
<script>
    $("#div4 span").draggable ({
        containment : "#div4"
    });
    $("#div5 span").draggable ({
        axis : "x"
    });
</script>
</body>
</html>

```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the output:

```

You can drag me only within this div.

You can drag me only along x axis.

```

Here, elements are prevented from going outside a <div> whose ID is div4. You can also impose constraints on vertical or horizontal motion using options *axis* worth "x" or "y", which is also demonstrated.

Move Content By Duplicating

The following example demonstrates how to move an item that is the clone of the selected element. This is done using the option *helper* with value *clone*.

```

<!DOCTYPE html>
<head>

    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">

    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>

```



```
<body>
  <div id="div6" style="border:solid 1px;background:#eee; height:50px;">
    <span>You can duplicate me....</span>
  </div>
  <script>
    $("#div6 span").draggable ({
      helper : "clone"
    });
  </script>

</body>
</html>
```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

You can duplicate me....

As you can see first element is being dragged-only the cloned element is moved, while the original item remains in its original position. If you release the mouse, the cloned element disappears and the original item is still in its original position.

Get Current Option Value

Following example demonstrates how you can get a value of any option at any time during your script execution. Here we will read the value of **cursor** and **cursorAt** options set at the time of execution. Similar way you can get value of any other options available.

```
<!DOCTYPE html>
<head>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
  ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

  <div id="divX" style="border:solid 1px;background:#eee; height:50px;">

    <span>Click anywhere on me to see cursor type...</span>
  </div>

  <script>
```

```

/* First make the item draggable */
$("#divX span").draggable();

$("#divX span").bind('click', function( event ){
    var cursor = $( "#divX span" ).draggable( "option", "cursor" );
    var cursorAt = $( "#divX span" ).draggable( "option", "cursorAt" );
    alert("Cursor type - " + cursor + ", cursorAt - " + cursorAt);
});
</script>

</body>
</html>

```

Let us save the above code in an HTML file **dragexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Click anywhere on me to see cursor type...

`$(selector, context).draggable ("action", [params])` Method

The *draggable* (*action*, *params*) method can perform an action on the movable elements, such as to prevent displacement. The **action** is specified as a string in the first argument and optionally, one or more **params** can be provided based on the given action.

Basically, Here actions are nothing but they are jQuery methods which we can use in the form of string.

Syntax

```
$(selector, context).draggable ("action", [params]);
```

The following table lists the actions for this method:

Action	Description
destroy()	Remove drag functionality completely. The elements are no longer movable. This will return the element back to its pre-init state. Syntax <code>\$(".selector").draggable("destroy");</code>
disable()	Disable drag functionality. Elements cannot be moved until the next call to the draggable("enable") method.

	<p>Syntax</p> <pre>\$(".selector").droppable("disable");</pre>
enable()	<p>Reactivates drag management. The elements can be moved again.</p> <p>Syntax</p> <pre>\$(".selector").droppable("enable");</pre>
option(optionName)	<p>Gets the value currently associated with the specified <i>optionName</i>. Where <i>optionName</i> is name of the option to get and is of type <i>String</i>.</p> <p>Syntax</p> <pre>var isDisabled = \$(".selector").droppable("option", "disabled");</pre>
option()	<p>Gets an object containing key/value pairs representing the current draggable options hash.</p> <p>Syntax</p> <pre>var options = \$(".selector").droppable("option");</pre>
option(optionName, value)	<p>Sets the <i>value</i> of the draggable option associated with the specified <i>optionName</i>. Where <i>optionName</i> is the name of the option to set and <i>value</i> is the value to set for the option.</p> <p>Syntax</p> <pre>\$(".selector").droppable("option", "disabled", true);</pre>
option(options)	<p>Sets one or more options for the draggable. Where <i>options</i> is a map of option-value pairs to set.</p> <p>Syntax</p> <pre>\$(".selector").droppable("option", { disabled: true });</pre>
widget()	<p>Returns a jQuery object containing the draggable element.</p> <p>Syntax</p> <pre>var widget = \$(".selector").droppable("widget");</pre>

Example

Now let us see an example using the actions from the above table. The following example demonstrates the use of *destroy()* method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Droppable - Default functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
      .draggable-4 {
        width: 90px; height: 50px; padding: 0.5em; float: left;
        margin: 0px 5px 10px 0;
        border: 1px solid red;
        background-color:#B9CD6D;
      }
      .droppable-7 {
        width: 100px; height: 90px;padding: 0.5em; float: left;
        margin: 10px;
        border: 1px solid black;
        background-color:#A39480;
      }
      .droppable.active {
        background-color: red;
      }
    </style>
    <script>
      $(function() {
        $('.draggable-4').draggable({ revert: true });
        $('.droppable-7').droppable({
          hoverClass: 'active',
          drop: function(e, ui) {
```

```

        $(this).html(ui.draggable.remove().html());
        $(this).droppable('destroy');
        $( this )
        .addClass( "ui-state-highlight" )
        .find( "p" )
        .html( "i'm destroyed!" );
    }
});
});
</script>
</head>
<body>
    <div class="draggable-4"><p>drag 1</p></div>
    <div class="draggable-4"><p>drag 2</p></div>
    <div class="draggable-4"><p>drag 3</p></div>

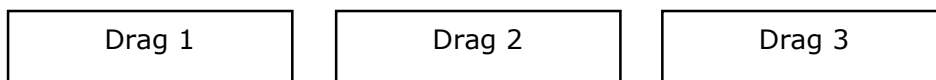
    <div style="clear: both;padding:10px"></div>

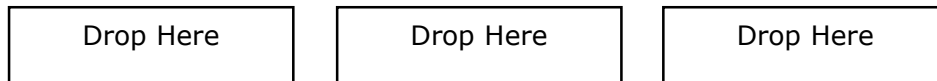
    <div class="droppable-7">drop here</div>
    <div class="droppable-7">drop here</div>
    <div class="droppable-7">drop here</div>
</body>
</html>

```

Let us save the above code in an HTML file **dropexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

If you drop "drag1" on any of the elements named "drop here", you will notice that this element gets dropped and this action destroys the droppable functionality of an element completely. You cannot drop "drag2" and "drag3" on this element again.





Event Management On Droppable Elements

In addition to the droppable (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
<u>activate(event, ui)</u>	<p>This event is triggered when the accepted draggable element starts dragging. This can be useful if you want to make the droppable "light up" when it can be dropped on.</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>draggable: A jQuery object representing the draggable element.</p> <p>helper: A jQuery object representing the helper that is being dragged.</p> <p>position: Current CSS position of the draggable helper as { top, left } object.</p> <p>offset: Current offset position of the draggable helper as { top, left } object.</p> <p>Syntax</p> <pre>\$(".selector").droppable({ activate: function(event, ui) {} });</pre>
<u>create(event, ui)</u>	<p>This event is triggered when a droppable element is created. Where event is of type Event, and ui is of type Object.</p>

	<p>Syntax</p> <pre>\$(".selector").droppable({ create: function(event, ui) {} });</pre>
<u>deactivate(event, ui)</u>	<p>This event is triggered when an accepted draggable stops dragging. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i> and possible types are:</p> <p>draggable: A jQuery object representing the draggable element.</p> <p>helper: A jQuery object representing the helper that is being dragged.</p> <p>position: Current CSS position of the draggable helper as { top, left } object.</p> <p>offset: Current offset position of the draggable helper as { top, left } object.</p> <p>Syntax</p> <pre>\$(".selector").droppable(deactivate: function(event, ui) {});</pre>
<u>drop(event, ui)</u>	<p>This action is triggered when an element is dropped on the droppable. This is based on the <i>tolerance</i> option. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>and possible types are:</p> <p>draggable: A jQuery object representing the draggable element.</p>

	<p>helper: A jQuery object representing the helper that is being dragged.</p> <p>position: Current CSS position of the draggable helper as { top, left } object.</p> <p>offset: Current offset position of the draggable helper as { top, left } object.</p> <p>Syntax</p> <pre>\$(".selector").droppable(drop: function(event, ui) {});</pre>
<u>out(event, ui)</u>	<p>This event is triggered when an accepted draggable element is dragged out of the droppable. This is based on the <i>tolerance</i> option. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Syntax</p> <pre>\$(".selector").droppable(out: function(event, ui) {});</pre>
<u>over(event, ui)</u>	<p>This event is triggered when an accepted draggable element is dragged over the droppable. This is based on the <i>tolerance</i> option. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>and possible types are:</p> <p>draggable: A jQuery object representing the draggable element.</p> <p>helper: A jQuery object representing the helper that is being dragged.</p>

position: Current CSS position of the draggable helper as { top, left } object.

offset: Current offset position of the draggable helper as { top, left } object.

Syntax

```
$(".selector").draggable(  
    over: function(event, ui) {}  
);
```

Example

The following example demonstrates the event method usage during drop functionality. This example demonstrates the use of events *drop*, *over* and *out*.

```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Droppable - Default functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <style>
            #draggable-5 {
                width: 100px; height: 50px; padding: 0.5em; float: left;
                margin: 22px 5px 10px 0;
            }
            #droppable-8 {
                width: 120px; height: 90px;padding: 0.5em; float: left;
                margin: 10px;
            }
        </style>
    </head>
    <body>
```

```

</style>
<script>
    $(function() {
        $( "#draggable-5" ).draggable();
        $("#droppable-8").droppable({
            drop: function (event, ui) {
                $( this )
                    .addClass( "ui-state-highlight" )
                    .find( "p" )
                    .html( "Dropped!" );
            },
            over: function (event, ui) {
                $( this )
                    .addClass( "ui-state-highlight" )
                    .find( "p" )
                    .html( "moving in!" );
            },
            out: function (event, ui) {
                $( this )
                    .addClass( "ui-state-highlight" )
                    .find( "p" )
                    .html( "moving out!" );
            }
        });
    });
</script>
</head>
<body>

<div id="draggable-5" class="ui-widget-content">
    <p>Drag me to my target</p>
</div>

<div id="droppable-8" class="ui-widget-header">
    <p>Drop here</p>
</div>

```

```
</body>  
</html>
```

Let us save the above code in an HTML file **dropexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

```
Drag me to my target  
Drop here
```

Here you will notice how the message in the droppable element changes as you drag the element.

5. JQUERYUI – RESIZABLE

jQueryUI provides `resizable()` method to resize any DOM element. This method simplifies the resizing of element which otherwise takes time and lot of coding in HTML. The `resizable()` method displays an icon in the bottom right of the item to resize.

Syntax

The **resizable()** method can be used in two forms:

- `$(selector, context).resizable (options) Method`
- `$(selector, context).resizable ("action", params) Method`

`$(selector, context).resizable (options) Method`

The *resizable (options)* method declares that an HTML element can be resized. The *options* parameter is an object that specifies the behavior of the elements involved when resizing.

Syntax

```
$(selector, context).resizable (options);
```

You can provide one or more options at a time of using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).resizable({option1: value1, option2: value2..... });
```

Following table lists the different *options* that can be used with this method:

Option	Description
alsoResize	This option is of type <i>Selector, jQuery</i> , or any <i>DOM Element</i> . It represents elements that also resize when resizing the original object. By default its value is false.
animate	This option when set to true is used to enable a visual effect during resizing when the mouse button is released. The default value is false (no effect).
animateDuration	This option is used to set the duration (in milliseconds) of the resizing effect. This option is used only when <i>animate</i> option is <i>true</i> . By default its value is "slow".
animateEasing	This option is used to specify which <i>easing</i> to apply when using the <i>animate</i> option. By default its value is "swing".

aspectRatio	This option is used to indicate whether to keep the aspect (height and width) ratio for the item. By default its value is false.
autoHide	This option is used to hide the magnification icon or handles, except when the mouse is over the item. By default its value is false.
cancel	This option is used to prevent resizing on specified elements. By default its value is input, textarea, button, select, option.
containment	This option is used restrict the resizing of elements within a specified element or region. By default its value is false.
delay	This option is used to set tolerance or delay in milliseconds. Resizing or displacement will begin thereafter. By default its value is 0.
disabled	This option disables the resizing mechanism when set to <i>true</i> . The mouse no longer resizes elements, until the mechanism is enabled, using the resizable ("enable"). By default its value is false.
distance	With this option, the resizing starts only when the mouse moves a distance(pixels). By default its value is 1 pixel. This can help prevent unintended resizing when clicking on an element.
ghost	This option when set to <i>true</i> , a semi-transparent helper element is shown for resizing. This ghost item will be deleted when the mouse is released. By default its value is false.
grid	This option is of type Array [x, y], indicating the number of pixels that the element expands horizontally and vertically during movement of the mouse. By default its value is false.
handles	This option is a character string indicating which sides or angles of the element can be resized. By default its values are e, s, se.
helper	This option is used to add a CSS class to style the element to be resized. When the element is resized a new <div> element is created, which is the one that is scaled (ui-resizable-helper class). Once the resize is complete, the original element is sized and the <div> element disappears. By default its value is false.
maxHeight	This option is used to set the maximum height the resizable should be allowed to resize to. By default its value is null.

maxWidth	This option is used to set the maximum width the resizable should be allowed to resize to. By default its value is null.
minHeight	This option is used to set the minimum height the resizable should be allowed to resize to. By default its value is 10.
minWidth	This option is used to set the minimum width the resizable should be allowed to resize to. By default its value is 10.

The following section will show you few a working examples of resize functionality.

Default Functionality

The following example demonstrates a simple example of resizable functionality, passing no parameters to the **resizable()** method.

```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Resizable functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <!-- CSS -->
        <style>
            .ui-widget-header {
                background:#b9cd6d;
                border: 1px solid #b9cd6d;
                color: #FFFFFFF;
                font-weight: bold;
            }
            .ui-widget-content {
                background: #cedc98;
                border: 1px solid #DDDDDD;
                color: #333333;
            }

```

```

        #resizable { width: 150px; height: 150px; padding: 0.5em;
            text-align: center; margin: 0; }
    </style>
    <!-- Javascript -->
    <script>
        $(function() {
            $( "#resizable" ).resizable();
        });
    </script>
</head>

<body>
    <!-- HTML -->
    <div id="resizable" class="ui-widget-content">
        <h3 class="ui-widget-header">Pull my edges to resize me!!</h3>
    </div>
</body>
</html>

```

Let us save the above code in an HTML file **resizeexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

Pull my edges to resize me!!

Drag the square border to resize.

Use of Animate and Ghost

The following example demonstrates the usage of two options **animate** and **ghost** in the resize function of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">

        <title>jQuery UI Resizable functionality</title>

        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
        ui.css" rel="stylesheet">

        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>

```

```

<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<!-- CSS -->
<style>
    .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
    }
    .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
    }
    #resizable-2,#resizable-3 {
        width: 150px; height: 150px; padding: 0.5em;
        text-align: center; margin: 0; }
</style>
<!-- Javascript -->
<script>
    $(function() {
        $( "#resizable-2" ).resizable({
            animate: true
        });
        $( "#resizable-3" ).resizable({
            ghost: true
        });
    });
</script>
</head>
<body>
<!-- HTML -->

<div id="resizable-2" class="ui-widget-content">

    <h3 class="ui-widget-header">
        Pull my edges and Check the animation!!
    </h3>

```



```

    </div><br>
    <div id="resizable-3" class="ui-widget-content">
        <h3 class="ui-widget-header">I'm ghost!!</h3>
    </div>
</body>
</html>

```

Let us save the above code in an HTML file `resizeexample.htm` and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Pull my edges and Check the animation!!

I'm ghost!!

Drag the square border to resize and see the effect of `animate` and `ghost` options.

Use of containment, minHeight, and minWidth

The following example demonstrates the usage of three options **containment**, **minHeight** and **minWidth** in the `resize` function of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Resizable functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
        ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <style>
            .ui-widget-header {
                background:#b9cd6d;
                border: 1px solid #b9cd6d;
                color: #FFFFFF;
                font-weight: bold;
            }

            .ui-widget-content {
                background: #cedc98;
                border: 1px solid #DDDDDD;

```

```

        color: #333333;
    }
    #container-1 { width: 300px; height: 300px; }
    #resizable-4 {background-position: top left;
        width: 150px; height: 150px; }
    #resizable-4, #container { padding: 0.5em; }
</style>
<script>
    $(function() {
    $( "#resizable-4" ).resizable({
        containment: "#container",
        minHeight: 70,
        minWidth: 100
    });
    });
</script>
</head>
<body>
    <div id="container" class="ui-widget-content">
        <div id="resizable-4" class="ui-state-active">
            <h3 class="ui-widget-header">
                Resize contained to this container
            </h3>
        </div>
    </div>
</body>
</html>

```

Let us save the above code in an HTML file **resizeexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

Resize contained to this container

Drag the square border to resize, you cannot resize beyond the main container.

Use of delay, distance, and autoHide

The following example demonstrates the usage of three options **delay**, **distance** and **autoHide** in the **resize** function of jQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Resizable functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
      .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
      }
      .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
      }
      .square {
        width: 150px;
        height: 150px;
        border: 1px solid black;
        text-align: center;
        float: left;
        margin-left: 20px;
        -right: 20px;
      }
    </style>
    <script>
      $(function() {
```

```

$( "#resizable-5" ).resizable({
    delay: 1000
});

$( "#resizable-6" ).resizable({
    distance: 40
});
$( "#resizable-7" ).resizable({
    autoHide: true
});
});
</script>
</head>
<body>
    <div id="resizable-5" class="square ui-widget-content">
        <h3 class="ui-widget-header">
            Resize starts after delay of 1000ms
        </h3>
    </div><br>
    <div id="resizable-6" class="square ui-widget-content">
        <h3 class="ui-widget-header">
            Resize starts at distance of 40px
        </h3>
    </div><br>
    <div id="resizable-7" class="square ui-widget-content">
        <h3 class="ui-widget-header">
            Hover over me to see the magnification icon!
        </h3>
    </div>
</body>
</html>

```

Let us save the above code in an HTML file **resizeexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

Resize starts after delay of 1000ms

Resize starts at distance of 40px

Hover over me to see the magnification icon!

Drag the square border to resize and you can notice that:

- The first square box resizes after a delay of 1000ms,
- Second square box starts resizing after the mouse moves 40px .
- Hover the mouse on the third square box, and the magnification icon appears.

Use of alsoResize

The following example demonstrates the usage of option **alsoResize** in the resize function of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Resizable functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
      }
      .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
      }
      #resizable-8,#resizable-9{ width: 150px; height: 150px;
        padding: 0.5em;text-align: center; margin: 0; }

    </style>
```

```

<!-- Javascript -->
<script>
    $(function() {
        $( "#resizable-8" ).resizable({
            alsoResize: "#resizable-9"
        });
        $( "#resizable-9" ).resizable();
    });
</script>
</head>
<body>
    <!-- HTML -->
    <div id="resizable-8" class="ui-widget-content">
        <h3 class="ui-widget-header">Resize!!</h3>
    </div><br>
    <div id="resizable-9" class="ui-widget-content">
        <h3 class="ui-widget-header">I also get resized!!</h3>
    </div>
</body>
</html>

```

Let us save the above code in an HTML file **resizeexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

Resize!!

I also get resized!!

Drag the square border to resize and you can notice that the second square box also resizes with the first square box.

Use of Aspectratio, Grid

The following example demonstrates the usage of option **aspectRatio** and **grid** in the resize function of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>

        <meta charset="utf-8">

```

```

<title>jQuery UI Resizable functionality</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<style>
    .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFF;
        font-weight: bold;
    }
    .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
    }
    .square {
        width: 150px;
        height: 150px;
        border: 1px solid black;
        text-align: center;
        float: left;
        margin-left: 20px;
        margin-right: 20px;
    }
</style>
<script>
    $(function() {
        $( "#resizable-10" ).resizable({
            aspectRatio: 10 / 3
        });

        $( "#resizable-11" ).resizable({
            grid: [50,20]
        });
    });

```

```

    });
</script>
</head>
<body>
    <div id="resizable-10" class="square ui-widget-content">
        <h3 class="ui-widget-header">
            Resize with aspectRatio of 10/3
        </h3>
    </div>
    <div id="resizable-11" class="square ui-widget-content">
        <h3 class="ui-widget-header">
            Snap me to the grid of [50,20]
        </h3>
    </div>
</body>
</html>

```

Let us save the above code in an HTML file **resizeexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

Resize with aspectRatio of 10/3

Snap me to the grid of [50,20]

Drag the square border to resize, the first square box resizes with the aspect ratio of 10 / 3 and the second one resizes with grid of [50,20].

\$(Selector, Context).Resizable ("Action", Params) Method

The *resizable* ("action", params) method can perform an action on resizable elements, such as allowing or preventing resizing functionality. The action is specified as a string in the first argument (e.g., "disable" to prevent the resize). Check out the actions that can be passed, in the following table.

Syntax

```
$(selector, context).resizable ("action", params);;
```


The following table lists the different *actions* that can be used with this method:

Action	Description
destroy	<p>This action destroys the resizable functionality of an element completely. This will return the element back to its pre-init state.</p> <p>Syntax</p> <pre>\$(".selector").resizable("destroy");</pre>
disable	<p>This action disables the resizing functionality of an element. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").resizable("disable");</pre>
enable	<p>This action enables the resizing functionality of an element. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").resizable("enable");</pre>
option(optionName)	<p>This action retrieves the value of the specified <i>optionName</i>. This option corresponds to one of those used with resizable (options).</p> <p>Syntax</p> <pre>var isDisabled = \$(".selector").resizable("option", "disabled");</pre>
option()	<p>Gets an object containing key/value pairs representing the current resizable options hash. This does not accept any arguments.</p> <p>Syntax</p> <pre>var options = \$(".selector").resizable("option");</pre>
option(optionName, value)	<p>This action sets the value of the resizable option with the specified <i>optionName</i>. This option corresponds to one of those used with resizable (options).</p> <p>Syntax</p>

	<code>\$(".selector").resizable("option", "disabled", true);</code>
option(options)	<p>This action sets one or more options for the resizable.</p> <p>Syntax</p> <pre>\$(".selector").resizable("option", { disabled: true });</pre>
widget()	<p>This action returns a <i>jQuery</i> object containing the resizable element. This method does not accept any arguments.</p> <p>Syntax</p> <pre>var widget = \$(".selector").resizable("widget");</pre>

Example

Now let us see an example using the actions from the above table. The following example demonstrates the use of *destroy()* and *disable()* methods.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Resizable functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
      }
      .ui-widget-content {
        background: #cedc98;
```

```

        border: 1px solid #DDDDDD;
        color: #333333;
    }
    #resizable-12,#resizable-13 { width: 150px; height: 150px;
        padding: 0.5em;text-align: center; margin: 0; }
</style>
<!-- Javascript -->
<script>
    $(function() {
        $( "#resizable-12" ).resizable();
        $( "#resizable-12" ).resizable('disable');
        $( "#resizable-13" ).resizable();
        $( "#resizable-13" ).resizable('destroy');
    });
</script>
</head>

<body>
    <!-- HTML -->
    <div id="resizable-12" class="ui-widget-content">
        <h3 class="ui-widget-header">I'm disable!!</h3>
    </div><br>
    <div id="resizable-13" class="ui-widget-content">
        <h3 class="ui-widget-header">I'm Destroyed!!</h3>
    </div>
</body>
</html>

```

Let us save the above code in an HTML file **resizeexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

```

I'm disable!!
I'm Destroyed!!

```

You cannot resize the first square box as its disabled and the second square box is destroyed.

Event Management on Resizable Elements

In addition to the resizable (options) method we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
create(event, ui)	<p>This event is triggered when the resizable element is created.</p> <p>Syntax</p> <pre>\$(".selector").resizable({ create: function(event, ui) {} });</pre>
resize(event, ui)	<p>This event is triggered when the handler of resizable element is dragged.</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>element: A jQuery object representing the resizable element.</p> <p>helper: A jQuery object representing the helper that is being resized.</p> <p>originalElement: The jQuery object representing the original element before it is wrapped.</p> <p>originalPosition: The position represented as {left, top } before the resizable is resized.</p> <p>originalSize: The size represented as { width, height } before the resizable is resized.</p> <p>position: The current position represented as {left, top }.</p> <p>size: The current size represented as { width, height }.</p> <p>Syntax</p>

	<pre>\$(".selector").resizable({ resize: function(event, ui) {} });</pre>
start(event, ui)	<p>This event is triggered at the start of a resize operation.</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>element: A jQuery object representing the resizable element.</p> <p>helper: A jQuery object representing the helper that is being resized.</p> <p>originalElement: The jQuery object representing the original element before it is wrapped.</p> <p>originalPosition: The position represented as {left, top } before the resizable is resized.</p> <p>originalSize: The size represented as { width, height } before the resizable is resized.</p> <p>position: The current position represented as {left, top }.</p> <p>size: The current size represented as { width, height }.</p> <p>Syntax</p> <pre>\$(".selector").resizable({ start: function(event, ui) {} });</pre>

<p>stop(event, ui)</p>	<p>This event is triggered at the end of a resize operation.</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>element: A jQuery object representing the resizable element.</p> <p>helper: A jQuery object representing the helper that is being resized.</p> <p>originalElement: The jQuery object representing the original element before it is wrapped.</p> <p>originalPosition: The position represented as {left, top } before the resizable is resized.</p> <p>originalSize: The size represented as { width, height } before the resizable is resized.</p> <p>position: The current position represented as {left, top }.</p> <p>size: The current size represented as { width,height }.</p> <p>Syntax</p> <pre>\$(".selector").resizable({ stop: function(event, ui) {} });</pre>
------------------------	--

Example

The following example demonstrates the event method usage during resize functionality. This example demonstrates the use of events *create*, and *resize*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
```

```

<title>jQuery UI Resizable functionality</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<!-- CSS -->
<style>
    .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFF;
        font-weight: bold;
    }
    .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
    }
    #resizable-14{ width: 150px; height: 150px;
        padding: 0.5em;text-align: center; margin: 0; }
</style>
<!-- Javascript -->
<script>
    $(function() {
        $( "#resizable-14" ).resizable({
            create: function( event, ui ) {
                $( "#resizable-15" ).text ( "I'm Created!!" );
            },
            resize: function (event, ui)
            {
                $( "#resizable-16" ).text ( "top = " + ui.position.top +
                    ", left = " + ui.position.left +
                    ", width = " + ui.size.width +
                    ", height = " + ui.size.height );
            }
        });
    });

```

```
});  
</script>  
</head>  
<body>  
  <!-- HTML -->  
  <div id="resizable-14" class="ui-widget-content">  
    <h3 class="ui-widget-header">Resize !!</h3>  
  </div><br>  
  <span id="resizable-15"></span><br>  
  <span id="resizable-16"></span>  
</body>  
</html>
```

Let us save the above code in an HTML file **resizeexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

Resize !!

Drag the square box and you will see the output getting displayed on resize event.

6. JQUERYUI – SELECTABLE

jQueryUI provides `selectable()` method to select DOM element individually or in a group. With this method elements can be selected by dragging a box (sometimes called a lasso) with the mouse over the elements. Also, elements can be selected by clicking or dragging while holding the Ctrl/Meta key, allowing for multiple (non-contiguous) selections.

Syntax

The **`selectable()`** method can be used in two forms:

- `$(selector, context).selectable (options) Method`
- `$(selector, context).selectable ("action", params) Method`

`$(selector, context).selectable (options) Method`

The *selectable (options)* method declares that an HTML element contains selectable items. The *options* parameter is an object that specifies the behavior of the elements involved when selecting.

Syntax

```
$(selector, context).selectable (options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided, then you will separate them using a comma as follows:

```
$(selector, context).selectable({option1: value1, option2: value2..... });
```

Following table lists the different *options* that can be used with this method:

Option	Description
appendTo	<p>This option is tells which element the selection helper (the lasso) should be appended to. By default its value is body.</p> <p>Syntax</p> <p><code>\$(".selector").selectable({ appendTo: "#identifier" });</code></p>
autoRefresh	<p>This option if set to <i>true</i>, the position and size of each selectable item is computed at the beginning of a select operation. By default its value is <i>true</i>.</p> <p>If you have many items, you may want to set this to false and call the <code>refresh()</code> method manually.</p> <p>Syntax</p>

	<pre>\$(".selector").selectable({ autoRefresh: false });</pre>
cancel	<p>This option forbids selecting if you start selection of elements. By default its value is input,textarea,button,select,option.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ cancel: "a,.cancel" });</pre>
delay	<p>This option is used to set time in milliseconds and defines when the selecting should start. This can be used to prevent unwanted selections. By default its value is 0.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ delay: 150 });</pre>
disabled	<p>This option when set to true, disables the selection mechanism. Users cannot select the elements until the mechanism is restored using the selectable ("enable") instruction. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ disabled: true });</pre>
distance	<p>This option is the distance (in pixels) the mouse must move to consider the selection in progress. This is useful, for example, to prevent simple clicks from being interpreted as a group selection. By default its value is 0.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ distance: 30 });</pre>
filter	<p>This option is a selector indicating which elements can be part of the selection. By default its value is *.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ filter: "li" });</pre>
tolerance	<p>This option specifies which mode to use for testing whether the selection helper (the lasso) should select an item. By default its value is touch.</p> <p>This can be of type:</p> <p>fit: This type indicates a drag selection must completely encompass an element for it to be selected.</p>

	<p>touch: This type indicates the drag rectangle only needs to intersect any portion of the selectable item.</p> <p>Syntax</p> <p><code>\$(".selector").selectable({ tolerance: "fit" });</code></p>
--	---

The following section will show you a few working examples of selectable functionality.

Default Functionality

The following example demonstrates a simple example of selectable functionality, passing no parameters to the **selectable()** method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI selectable-1</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
      #selectable-1 .ui-selecting { background: #707070 ; }
      #selectable-1 .ui-selected { background: #EEEEEE; color: #000000; }
      #selectable-1 { list-style-type: none; margin: 0;
        padding: 0; width: 20%; }
      #selectable-1 li { margin: 3px; padding: 0.4em;
        font-size: 16px; height: 18px; }
      .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
      }
    </style>
    <script>
```

```

        $(function() {
            $( "#selectable-1" ).selectable();
        });
    </script>
</head>
<body>
    <ol id="selectable-1">
        <li class="ui-widget-content">Product 1</li>
        <li class="ui-widget-content">Product 2</li>
        <li class="ui-widget-content">Product 3</li>
        <li class="ui-widget-content">Product 4</li>
        <li class="ui-widget-content">Product 5</li>
        <li class="ui-widget-content">Product 6</li>
        <li class="ui-widget-content">Product 7</li>
    </ol>
</body>
</html>

```

Let's save the above code in an HTML file **selectableexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

```

Product 1
Product 2
Product 3
Product 4
Product 5
Product 6
Product 7

```

Try to click on products, use CTRLS key to select multiple products.

Use of Delay and Distance

The following example demonstrates the usage of two options **delay** and **distance** in the selectable function of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">

```

```

<title>jQuery UI Selectable</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

<style>
    #selectable-2 .ui-selecting,#selectable-3 .ui-selecting {
        background: #707070 ; }
    #selectable-2 .ui-selected,#selectable-3 .ui-selected {
        background: #EEEEEE; color: #000000; }
    #selectable-2,#selectable-3 { list-style-type: none; margin: 0;
        padding: 0; width: 20%; }
    #selectable-2 li,#selectable-3 li { margin: 3px; padding: 0.4em;
        font-size: 16px; height: 18px; }
    .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
    }
</style>
<script>
    $(function() {
        $( "#selectable-2" ).selectable({
            delay : 1000
        });
        $( "#selectable-3" ).selectable({
            distance : 100
        });
    });
</script>
</head>

<body>

    <h3>Starts after delay of 1000ms</h3>
    <ol id="selectable-2">
        <li class="ui-widget-content">Product 1</li>

```

```

        <li class="ui-widget-content">Product 2</li>
        <li class="ui-widget-content">Product 3</li>
    </ol>
    <h3>Starts after mouse moves distance of 100px</h3>
    <ol id="selectable-3">
        <li class="ui-widget-content">Product 4</li>
        <li class="ui-widget-content">Product 5</li>
        <li class="ui-widget-content">Product 6</li>
        <li class="ui-widget-content">Product 7</li>
    </ol>
</body>
</html>

```

Let's save the above code in an HTML file **selectableexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

Starts after delay of 1000ms

```

Product 1
Product 2
Product 3

```

Starts after mouse moves distance of 100px

```

Product 4
Product 5
Product 6
Product 7

```

Try to click on products, use CTRL key to select multiple products. You will notice that selection of the Product 1, Product 2 and Product 3 start after a delay of 1000ms. Selection of the Product 4, Product 5, Product 6 and Product 7 can't be done individually. The selection starts only after the mouse moves a distance of 100px.

Use of Filter

The following example demonstrates the usage of two options **delay** and **distance** in the selectable function of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">

```

```

<title>jQuery UI selectable-4</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<style>
    #selectable-4 .ui-selecting { background: #707070 ; }
    #selectable-4 .ui-selected { background: #EEEEEE; color: #000000; }
    #selectable-4 { list-style-type: none; margin: 0;
        padding: 0; width: 20%; }
    #selectable-4 li { margin: 3px; padding: 0.4em;
        font-size: 16px; height: 18px; }
    .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
    }
</style>
<script>
    $(function() {
        $( "#selectable-4" ).selectable({
            filter : "li:first-child"
        });
    });
</script>
</head>
<body>
    <ol id="selectable-4">
        <li class="ui-widget-content">Product 1</li>
        <li class="ui-widget-content">Product 2</li>
        <li class="ui-widget-content">Product 3</li>
        <li class="ui-widget-content">Product 4</li>

        <li class="ui-widget-content">Product 5</li>

        <li class="ui-widget-content">Product 6</li>
        <li class="ui-widget-content">Product 7</li>
    </ol>

```

```
</body>
</html>
```

Let's save the above code in an HTML file **selectableexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

```
Product 1
Product 2
Product 3
Product 4
Product 5
Product 6
Product 7
```

Try to click on products. You will notice that only first product can be selected.

`$(selector, context).selectable("action", params)` Method

The *selectable* ("action", params) method can perform an action on selectable elements, such as preventing selectable functionality. The action is specified as a string in the first argument (e.g., "disable" to stop the selection). Check out the actions that can be passed, in the following table.

Syntax

```
$(selector, context).selectable ("action", params);;
```

Following table lists the different *actions* that can be used with this method:

Action	Description
destroy	<p>This action removes the selectable functionality of an element completely. The elements return to their pre-init state.</p> <p>Syntax</p> <pre>\$(".selector").selectable("destroy");</pre>
disable	<p>This action deactivates the selectable functionality of an element. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").selectable("disable");</pre>

enable	<p>This action enables the selectable functionality of an element. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").selectable("enable");</pre>
option(optionName)	<p>This action gets the value currently associated with the specified <i>optionName</i>.</p> <p>Syntax</p> <pre>var isDisabled = \$(".selector").selectable("option", "disabled");</pre>
option()	<p>This action gets an object containing key/value pairs representing the current selectable options hash.</p> <p>Syntax</p> <pre>var options = \$(".selector").selectable("option");</pre>
option(optionName, value)	<p>This action sets the value of the selectable option associated with the specified <i>optionName</i>. The argument <i>optionName</i> is name of the option to be set and <i>value</i> is the value to be set for the option.</p> <p>Syntax</p> <pre>\$(".selector").selectable("option", "disabled", true);</pre>
option(options)	<p>This action is sets one or more options for the selectable. The argument <i>options</i> is a map of option-value pairs to be set.</p> <p>Syntax</p> <pre>\$(".selector").selectable("option", { disabled: true });</pre>
refresh	<p>This action causes the size and position of the selectable elements to be refreshed. Used mostly when the <i>autoRefresh</i> option is disabled (set to <i>false</i>). This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").selectable("refresh");</pre>

widget	<p>This action returns a jQuery object containing the selectable element. This method does not accept any arguments.</p> <p>Syntax</p> <pre>var widget = \$(".selector").selectable("widget");</pre>

Example

Now let us see an example using the actions from the above table. The following example demonstrates the use of `disable()` and `option(optionName, value)` methods.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Selectable</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
      #selectable-5 .ui-selecting,#selectable-6 .ui-selecting {
        background: #707070 ; }
      #selectable-5 .ui-selected,#selectable-6 .ui-selected {
        background: #EEEEEE; color: #000000; }
      #selectable-5,#selectable-6 {
        list-style-type: none; margin: 0; padding: 0; width: 20%; }
      #selectable-5 li,#selectable-6 li {
        margin: 3px; padding: 0.4em; font-size: 16px; height: 18px; }
      .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
      }
    </style>
  </script>
```

```

$(function() {
    $( "#selectable-5" ).selectable();
    $( "#selectable-5" ).selectable('disable');
    $( "#selectable-6" ).selectable();
    $( "#selectable-6" ).selectable( "option", "distance", 1 );
});
</script>
</head>
<body>
    <h3>Disabled using disable() method</h3>
    <ol id="selectable-5">
        <li class="ui-widget-content">Product 1</li>
        <li class="ui-widget-content">Product 2</li>
        <li class="ui-widget-content">Product 3</li>
    </ol>
    <h3>Select using method option( optionName, value )</h3>
    <ol id="selectable-6">
        <li class="ui-widget-content">Product 4</li>
        <li class="ui-widget-content">Product 5</li>
        <li class="ui-widget-content">Product 6</li>
        <li class="ui-widget-content">Product 7</li>
    </ol>
</body>
</html>

```

Let's save the above code in an HTML file **selectableexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

Disabled using disable() method

Product 1
Product 2
Product 3

Select using method option(optionName, value)

Product 4
Product 5

Product 6

Product 7

Try to click on products, use CTRL key to select multiple products. You will notice that Product 1, Product 2, and Product 3 are disabled. Selection of Product 4, Product 5, Product 6 and Product 7 happens after the mouse moves distance of 1px.

Event Management on Selectable Elements

In addition to the selectable (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
create(event, ui)	<p>This event is triggered when the selectable element is created. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ create: function(event, ui) {} });</pre>
selected(event, ui)	<p>This event is triggered for each element that becomes selected. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>. Possible values of <i>ui</i> are:</p> <p>selected: This specifies the selectable item that has been selected..</p> <p>Syntax</p> <pre>\$(".selector").selectable({ selected: function(event, ui) {} });</pre>
selecting(event, ui)	<p>This event is triggered for each selectable element that's about to get selected. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Possible values of <i>ui</i> are:</p> <p>selecting: This specifies a reference to the element that's about to become selected.</p> <p>Syntax</p>

	<pre>\$(".selector").selectable({ selecting: function(event, ui) {} });</pre>
start(event, ui)	<p>This event is triggered at the beginning of the select operation. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ start: function(event, ui) {} });</pre>
stop(event, ui)	<p>This event is triggered at the end of the select operation. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ stop: function(event, ui) {} });</pre>
unselected(event, ui)	<p>This event is triggered at the end of the select operation for each element that becomes unselected. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Possible values of ui are:</p> <p>unselected: An element that contains a reference to the element that has become unselected.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ unselected: function(event, ui) {} });</pre>

unselecting(event, ui)	<p>This event is triggered during select operation for each selected element that's about to become unselected. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Possible values of ui are:</p> <p>unselecting: An element that contains a reference to the element that's about to become unselected.</p> <p>Syntax</p> <pre>\$(".selector").selectable({ unselecting: function(event, ui) {} });</pre>
------------------------	---

Example

The following example demonstrates the event method usage during selectable functionality. This example demonstrates the use of event *selected*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI selectable-7</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
      #selectable-7 .ui-selecting { background: #707070 ; }
      #selectable-7 .ui-selected { background: #EEEEEE; color: #000000; }
      #selectable-7 { list-style-type: none; margin: 0;
        padding: 0; width: 20%; }
      #selectable-7 li { margin: 3px; padding: 0.4em;

        font-size: 16px; height: 18px; }

      .ui-widget-content {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
```

```

    }
    .resultarea {
        background: #cedc98;
        border-top: 1px solid #000000;
        border-bottom: 1px solid #000000;
        color: #333333;
        font-size:14px;
    }
</style>
<script>
    $(function() {
        $( "#selectable-7" ).selectable({
            selected: function() {
                var result = $( "#result" ).empty();
                $( ".ui-selected", this ).each(function() {
                    var index = $( "#selectable-7 li" ).index( this );
                    result.append( " #" + ( index + 1 ) );
                });
            }
        });
    });
</script>
</head>
<body>
    <h3>Events</h3>
    <ol id="selectable-7">
        <li class="ui-widget-content">Product 1</li>
        <li class="ui-widget-content">Product 2</li>
        <li class="ui-widget-content">Product 3</li>
        <li class="ui-widget-content">Product 4</li>
        <li class="ui-widget-content">Product 5</li>
        <li class="ui-widget-content">Product 6</li>
        <li class="ui-widget-content">Product 7</li>
    </ol>
    <span class="resultarea">Selected Product</span>>
    <span id=result class="resultarea"></span>

```

```
</body>  
</html>
```

Let's save the above code in an HTML file **selectableexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

Events

```
Product 1  
Product 2  
Product 3  
Product 4  
Product 5  
Product 6  
Product 7
```

Selected Product

Try to click on products, use CTRL key to select multiple products. You will notice that the product number selected is printed at the bottom.

7. JQUERYUI – SORTABLE

jQueryUI provides **sortable()** method to reorder elements in list or grid using the mouse. This method performs sortability action based upon an operation string passed as the first parameter.

Syntax

The **sortable ()** method can be used in two forms:

- `$(selector, context).sortable (options) Method`
- `$(selector, context).sortable ("action", [params]) Method`

\$(selector, context).sortable (options) Method

The *sortable (options)* method declares that an HTML element contains interchangeable elements. The *options* parameter is an object that specifies the behavior of the elements involved during reordering.

Syntax

```
$(selector, context).sortable(options);
```

Following table lists the different *options* that can be used with this method:

Option	Description
appendTo	<p>This option specifies the element in which the new element created with <i>options.helper</i> will be inserted during the time of the move/drag. By default its value is parent.</p> <p>This can be of type:</p> <p>Selector: This indicates a selector specifying which element to append the helper to..</p> <p>jQuery: This indicates a jQuery object containing the element to append the helper to.</p> <p>Element: An element in the Document Object Model (DOM) to append the helper to.</p>

	<p>String: The string "parent" will cause the helper to be a sibling of the sortable item.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ appendTo: document.body });</pre>
axis	<p>This option indicates an axis of movement ("x" is horizontal, "y" is vertical). By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ axis: "x" });</pre>
cancel	<p>This option is used to prevent sorting of elements by clicking on any of the selector elements. By default its value is "input,textarea,button,select,option".</p> <p>Syntax</p> <pre>\$(".selector").sortable({ cancel: "a,button" });</pre>
connectWith	<p>This option is a Selector that identifies another sortable element that can accept items from this sortable. This allows items from one list to be moved to other lists, a frequent and useful user interaction. If omitted, no other element is connected. This is a one-way relationship. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").sortable(</pre>

	<pre>{ connectWith: "#identifier" } };</pre>
containment	<p>This option indicates an element within which the displacement takes place. The element will be represented by a selector (only the first item in the list will be considered), a DOM element, or the string "parent" (parent element) or "window" (HTML page).</p> <p>Syntax</p> <pre>\$(".selector").sortable({ containment: "parent" });</pre>
cursor	<p>Specifies the cursor CSS property when the element moves. It represents the shape of the mouse pointer. By default its value is "auto".</p> <p>Possible values are:</p> <p>"crosshair" (across)</p> <p>"default" (an arrow)</p> <p>"pointer" (hand)</p> <p>"move" (two arrows cross)</p> <p>"e-resize" (expand to the right)</p> <p>"ne-resize" (expand up right)</p> <p>"nw-resize" (expand up left)</p> <p>"n-resize" (expand up)</p>

	<p>"se-resize" (expand down right)</p> <p>"sw-resize" (expand down left)</p> <p>"s-resize" (expand down)</p> <p>"auto" (default)</p> <p>"w-resize" (expand left)</p> <p>"text" (pointer to write text)</p> <p>"wait" (hourglass)</p> <p>"help" (help pointer)</p> <p>Syntax</p> <pre>\$(".selector").sortable({ cursor: "move" });</pre>
cursorAt	<p>Sets the offset of the dragging helper relative to the mouse cursor. Coordinates can be given as a hash using a combination of one or two keys: { top, left, right, bottom }. By default its value is "false".</p> <p>Syntax</p> <pre>\$(".selector").sortable({ cursorAt: { left: 5 } });</pre>
delay	<p>Delay, in milliseconds, after which the first movement of the mouse is taken into account. The displacement may begin after that time. By default its value is "0".</p>

	<p>Syntax</p> <pre>\$(".selector").sortable({ delay: 150 });</pre>
disabled	<p>This option if set to <i>true</i>, disables the sortable functionality. By default its value is false.</p> <p>This option if set to <i>true</i>, disables the sortable functionality. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ disabled: true });</pre>
distance	<p>Number of pixels that the mouse must be moved before the sorting starts. If specified, sorting will not start until after mouse is dragged beyond distance. By default its value is "1".</p> <p>Syntax</p> <pre>\$(".selector").sortable({ distance: 5 });</pre>
dropOnEmpty	<p>This option if set to <i>false</i>, then items from this sortable can't be dropped on an empty connect sortable. By default its value is true.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ dropOnEmpty: false });</pre>
forceHelperSize	<p>If this option if set to <i>true</i> forces the helper to have a size. By default its value is false.</p> <p>Syntax</p>

	<pre>\$(".selector").sortable({ forceHelperSize: true });</pre>
forcePlaceholderSize	<p>This option when set to <i>true</i>, takes into account the size of the placeholder when an item is moved. This option is only useful if <i>options.placeholder</i> is initialized. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ forcePlaceholderSize: true });</pre>
grid	<p>This option is an Array [x, y] indicating the number of pixels that the sorting element moves horizontally and vertically during displacement of the mouse. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ grid: [20, 10] });</pre>
handle	<p>If specified, restricts sort from starting unless the mousedown occurs on the specified element(s). By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ handle: ".handle" });</pre>
helper	<p>Allows for a helper element to be used for dragging display. By default its value is original.</p> <p>Possible values are:</p> <p>String: If set to "clone", then the element will be cloned and the clone will be dragged.</p>

	<p>Function: A function that will return a DOMElement to use while dragging.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ helper: "clone" });</pre>
items	<p>This option specifies which items inside the DOM element to be sorted. By default its value is > *.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ items: "> li" });</pre>
opacity	<p>This option is used to define the opacity of the helper while sorting. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ opacity: 0.5 });</pre>
placeholder	<p>This option is used to class name that gets applied to the otherwise white space. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ addClasses: false });</pre>
revert	<p>This option decides whether the sortable items should revert to their new positions using a smooth animation. By default its value is false.</p>

	<p>Syntax</p> <pre>\$(".selector").sortable({ revert: true });</pre>
scroll	<p>This option is used to enable scrolling. If set to <i>true</i> the page scrolls when coming to an edge. By default its value is true.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ scroll: false });</pre>
scrollSensitivity	<p>This option indicates how many pixels the mouse must exit the visible area to cause scrolling. By default its value is 20. This option is used only with options.scroll set to true.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ scrollSensitivity: 10 });</pre>
scrollSpeed	<p>This option indicates the scrolling speed of the display once the scrolling begins. By default its value is 20.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ scrollSpeed: 40 });</pre>
tolerance	<p>This option is a <i>String</i> that specifies which mode to use for testing whether the item being moved is hovering over another item. By default its value is "intersect".</p> <p>intersect: The item overlaps the other item by at least 50%.</p> <p>pointer: The mouse pointer overlaps the other item.</p> <p>Syntax</p>

	<pre>\$(".selector").sortable({ tolerance: "pointer" });</pre>
zIndex	<p>This option represents z-index for element/helper while being sorted. By default its value is 1000.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ zIndex: 9999 });</pre>

The following section will show you a few working examples of drag functionality.

Default Functionality

The following example demonstrates a simple example of sortable functionality, passing no parameters to the **sortable()** method .

```
<!DOCTYPE html>
<head>
    <title>jQuery UI Sortable - Example</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
        #sortable-1 { list-style-type: none; margin: 0;
            padding: 0; width: 25%; }
        #sortable-1 li { margin: 0 3px 3px 3px; padding: 0.4em;
            padding-left: 1.5em; font-size: 17px; height: 16px; }
        .default {
            background: #cedc98;
            border: 1px solid #DDDDDD;
            color: #333333;
        }
    </style>
```

```

<script>
    $(function() {
        $( "#sortable-1" ).sortable();
    });
</script>
</head>
<body>
    <ul id="sortable-1">
        <li class="default">Product 1</li>
        <li class="default">Product 2</li>
        <li class="default">Product 3</li>
        <li class="default">Product 4</li>
        <li class="default">Product 5</li>
        <li class="default">Product 6</li>
        <li class="default">Product 7</li>
    </ul>
</body>
</html>

```

Let's save the above code in an HTML file **sortexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

```

Product 1
Product 2
Product 3
Product 4
Product 5
Product 6
Product 7

```

Re-arrange the products above, use mouse to drag items.

Use of Options Delay and Distance

The following example demonstrates the usage of three options **(a) delay** and **(b) distance** in the sort function of JQueryUI.

```

<!DOCTYPE html>
<head>

```

```

<title>jQuery UI Sortable - Example</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<style>
    #sortable-2, #sortable-3 { list-style-type: none; margin: 0;
        padding: 0; width: 25%; }
    #sortable-2 li, #sortable-3 li { margin: 0 3px 3px 3px; padding: 0.4em;
        padding-left: 1.5em; font-size: 17px; height: 16px; }
    .default {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
    }
</style>
<script>
    $(function() {
        $( "#sortable-2" ).sortable({
            delay:500
        });
        $( "#sortable-3" ).sortable({
            distance:30
        });
    });
</script>
</head>
<body>
    <h3>Delay by 500ms</h3>
    <ul id="sortable-2">
        <li class="default">Product 1</li>
        <li class="default">Product 2</li>
        <li class="default">Product 3</li>
        <li class="default">Product 4</li>
    </ul>
    <h3>Distance Delay by 30px</h3>

```

```

<ul id="sortable-3">
  <li class="default">Product 1</li>
  <li class="default">Product 2</li>
  <li class="default">Product 3</li>
  <li class="default">Product 4</li>
</ul>
</body>
</html>

```

Let's save the above code in an HTML file **sortexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

Delay by 500ms

```

Product 1
Product 2
Product 3
Product 4

```

Distance Delay by 30px

```

Product 1
Product 2
Product 3
Product 4

```

Re-arrange the products above, use mouse to drag items. To prevent accidental sorting either by delay (time) or distance, we have set a number of milliseconds the element needs to be dragged before sorting starts with the *delay* option. We have also set a distance in pixels the element needs to be dragged before sorting starts with the *distance* option.

Use of Placeholder

The following example demonstrates the usage of three option **placeholder** in the sort function of JQueryUI.

```

<!DOCTYPE html>
<head>
  <title>jQuery UI Sortable - Example</title>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
  ui.css" rel="stylesheet">

```

```

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<style>
    #sortable-4 { list-style-type: none; margin: 0;
        padding: 0; width: 25%; }
    #sortable-4 li { margin: 0 3px 3px 3px; padding: 0.4em;
        padding-left: 1.5em; font-size: 17px; height: 16px; }
    .highlight {
        border: 1px solid red;
        font-weight: bold;
        font-size: 45px;
        background-color: #333333;
    }
    .default {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
    }
</style>
<script>
    $(function() {
        $( "#sortable-4" ).sortable({
            placeholder: "highlight"
        });
    });
</script>
</head>
<body>
    <ul id="sortable-4">
        <li class="default">Product 1</li>
        <li class="default">Product 2</li>
        <li class="default">Product 3</li>
        <li class="default">Product 4</li>
        <li class="default">Product 5</li>
        <li class="default">Product 6</li>
        <li class="default">Product 7</li>
    </ul>

```

```

    </ul>
</body>
</html>

```

Let's save the above code in an HTML file **sortexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

```

Product 1
Product 2
Product 3
Product 4
Product 5
Product 6
Product 7

```

Try to drag items to rearrange them, while you're dragging items, the placeholder (we have used *highlight* class to style this space) will show up on an available place.

Use of Options **connectWith** and **DropOnempty**

The following example demonstrates the usage of three options **(a) connectWith** and **(b) dropOnEmpty** in the sort function of JQueryUI.

```

<!DOCTYPE html>
<head>
    <title>jQuery UI Sortable - Example</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
        #sortable-5, #sortable-6, #sortable-7 {
            list-style-type: none; margin: 0; padding: 0;
            width: 20%; float: left }
        #sortable-5 li, #sortable-6 li, #sortable-7 li {
            margin: 0 3px 3px 3px; padding: 0.4em;
            padding-left: 1.5em; font-size: 17px; height: 16px; }
        .default {
            background: #cedc98;

            border: 1px solid #DDDDDD;

```

```

        color: #333333;
    }
</style>
<script>
    $(function() {
        $( "#sortable-5, #sortable-6" ).sortable({
            connectWith: "#sortable-5, #sortable-6"
        });
        $( "#sortable-7").sortable({
            connectWith: "#sortable-5",
            dropOnEmpty: false
        });

    });
</script>
</head>
<body>
    <ul id="sortable-5"><h3>List 1</h3>
        <li class="default">A</li>
        <li class="default">B</li>
        <li class="default">C</li>
        <li class="default">D</li>
    </ul>
    <ul id="sortable-6"><h3>List 2</h3>
        <li class="default">a</li>
        <li class="default">b</li>
        <li class="default">c</li>
        <li class="default">d</li>
    </ul>
    <ul id="sortable-7"><h3>List 3</h3>
        <li class="default">e</li>
        <li class="default">f</li>
        <li class="default">g</li>
        <li class="default">h</li>
    </ul>
</body>

```

```
</html>
```

Let's save the above code in an HTML file **sortexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

List 1

```
A
B
C
D
```

List 2

```
a
b
c
d
```

List 3

```
e
f
g
h
```

Sort items from one List1 into another (List2) and vice versa, by passing a selector into the *connectWith* option. This is done by grouping all related lists with a CSS class, and then pass that class into the sortable function (i.e., *connectWith*: '#sortable-5, #sortable-6').

Try to drag the items under List 3 to the List 2 or List 1. As we have set *dropOnEmpty* option to *false*, it won't be possible to drop these items.

\$(selector, context).sortable ("action", [params]) Method

The *sortable (action, params)* method can perform an action on the sortable elements, such as to prevent displacement. The **action** is specified as a string in the first argument and optionally, one or more **params** can be provided based on the given action.

Basically, here actions are nothing but they are jQuery methods which we can use in the form of string.

Syntax

```
$(selector, context).sortable ("action", [params]);
```


The following table lists the actions for this method:

Action	Description
cancel()	<p>This action cancels the current sort operation. This is most useful within handlers for the sort receive and sort stop events. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").sortable("cancel");</pre>
destroy()	<p>This action removes the sortability functionality completely. This will return the element back to its pre-init state. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").sortable("disable");</pre>
disable()	<p>This action disables the sortability of any sortable elements in the wrapped set. The sortability of the elements isn't removed and can be restored by calling the enable variant of this method. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").sortable("disable");</pre>
enable()	<p>Re-enables sortability on any sortable elements in the wrapped set whose sortability has been disabled. Note that this method won't add sortability to any non-sortable elements. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").sortable("enable");</pre>
option(optionName)	<p>This action gets the value currently associated with the specified <i>optionName</i>. Where <i>optionName</i> is the name of the option to get.</p> <p>Syntax</p>

	<pre>var isDisabled = \$(".selector").sortable("option", "disabled");</pre>
option()	<p>Gets an object containing key/value pairs representing the current sortable options hash.. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").sortable("option");</pre>
option(optionName, value)	<p>This action sets the value of the sortable option associated with the specified optionName. Where <i>optionName</i> is the name of the option to set and <i>value</i> is the value to set for the option.</p> <p>Syntax</p> <pre>\$(".selector").sortable("option");</pre>
option(options)	<p>Sets one or more options for the sortable. Where <i>options</i> is a map of option-value pairs to set.</p> <p>Syntax</p> <pre>\$(".selector").sortable("option");</pre>
refresh()	<p>This action refreshes the list of items if necessary. This method does not accept any arguments. Calling this method will cause new items added to the sortable to be recognized.</p> <p>Syntax</p> <pre>\$(".selector").sortable("refresh");</pre>
toArray(options)	<p>This method returns an array of the <i>id</i> values of the sortable elements in sorted order. This method takes <i>Options</i> as parameter, to customize the serialization or sorted order.</p> <p>Syntax</p> <pre>var sortedIDs = \$(".selector").sortable("toArray");</pre>

serialize(options)	<p>This method returns a serialized query string (submittable via Ajax) formed from the sortable.</p> <p>Syntax</p> <pre>var sorted = \$(".selector").sortable("serialize", { key: "sort" });</pre>
refreshPositions()	<p>This method is used mostly internally to refresh the cached information of the sortable. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").sortable("refreshPositions");</pre>
widget()	<p>This method returns a jQuery object containing the sortable element. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").sortable("widget");</pre>

Example

Now let us see an example using the actions from the above table. The following example demonstrates the use of *toArray(options)* method.

```
<!DOCTYPE html>
<head>
  <title>jQuery UI Sortable - Example</title>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <style>

    #sortable-8{ list-style-type: none; margin: 0;
      padding: 0; width: 25%; float:left;}
    #sortable-8 li{ margin: 0 3px 3px 3px; padding: 0.4em;
      padding-left: 1.5em; font-size: 17px; height: 16px; }
    .default {
      background: #cedc98;
```

```

        border: 1px solid #DDDDDD;
        color: #333333;
    }
</style>
<script>
    $(function() {
        $('#sortable-8').sortable({
            update: function(event, ui) {
                var productOrder = $(this).sortable('toArray').toString();
                $("#sortable-9").text (productOrder);
            }
        });
    });
</script>
</head>
<body>
    <ul id="sortable-8">
        <li id="1" class="default">Product 1</li>
        <li id="2" class="default">Product 2</li>
        <li id="3" class="default">Product 3</li>
        <li id="4" class="default">Product 4</li>
    </ul>
    <br>
    <h3><span id="sortable-9"></span></h3>
</body>
</html>

```

Let's save the above code in an HTML file **sortexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

```

Product 1
Product 2
Product 3
Product 4

```

Try sorting the items, the order of items is displayed at the bottom. Here we are calling `$(this).sortable('toArray').toString()`, which will give a string list of all the item id's, it might look like **1,2,3,4**.

Event Management on The Sortable Elements

In addition to the sortable (options) method which we saw in the previous sections, JQueryUI provides event methods as which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
activate(event, ui)	<p>This event is triggered on the sortable when a sort operation starts on connected sortable. Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ activate: function(event, ui) {} });</pre>

<p>beforeStop(event, ui)</p>	<p>This event is triggered when the sort operation is about to end, with the helper and placeholder element reference still valid.</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ beforeStop: function(event, ui) {} });</pre>
<p>change(event, ui)</p>	<p>This event is triggered when the sorted element changes position within the DOM.</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p>

	<p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ beforeStop: function(event, ui) {} });</pre>
create(event, ui)	<p>This event is triggered when the sortable is created. Syntax</p> <pre>\$(".selector").sortable({ create: function(event, ui) {} });</pre>
deactivate(event, ui)	<p>This event is triggered when a connected sort stops, propagated to the connected sortable. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p>

	<p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ deactivate: function(event, ui) {} });</pre>
out(event, ui)	<p>This event is triggered when the sort item is moved away from a connected list. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p>

	<p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ out: function(event, ui) {} });</pre>
over(event, ui)	<p>This event is triggered when a sort item moves into a connected list. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p>

	<p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ over: function(event, ui) {} });</pre>
receive(event, ui)	<p>This event is triggered when a connected list has received a sort item from another list. Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p>

	<pre>\$(".selector").sortable({ receive: function(event, ui) {} });</pre>
remove(event, ui)	<p>This event is triggered when the sort item is removed from a connected list and is dragged into another. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ remove: function(event, ui) {} });</pre>

<p>sort(event, ui)</p>	<p>This event is repeatedly triggered for mousemove events during a sort operation. Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ sort: function(event, ui) {} });</pre>
<p>start(event, ui)</p>	<p>This event is triggered when a sort operation starts. Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p>

	<p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ start: function(event, ui) {} });</pre>
stop(event, ui)	<p>This event is triggered when a sort operation has concluded. Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p>

	<p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p> <p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ stop: function(event, ui) {} });</pre>
update(event, ui)	<p>This event is triggered when a sort operation stops and the position of the item has been changed. Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>helper: A jQuery object representing the helper being sorted.</p> <p>item: A jQuery object representing the current dragged element.</p> <p>offset: The current absolute position of the helper represented as { top, left }..</p> <p>position: Current CSS position of the helper as { top, left } object.</p> <p>originalPosition: The original position of the element represented as { top, left }.</p> <p>sender: The sortable that the item comes from if moving from one sortable to another.</p>

	<p>placeholder: The jQuery object representing the element being used as a placeholder.</p> <p>Syntax</p> <pre>\$(".selector").sortable({ update: function(event, ui) {} });</pre>
--	---

Example

The following example demonstrates the event method usage during drop functionality. This example demonstrates the use of events *receive*, *start* and *stop*.

```
<!DOCTYPE html>
<head>
  <title>jQuery UI Sortable - Example</title>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <style>
    #sortable-10, #sortable-11 { list-style-type: none;
      margin: 0; padding: 0; width: 80%; }
    #sortable-10 li, #sortable-11 li { margin: 0 3px 3px 3px;
      padding: 0.4em; padding-left: 1.5em;
      font-size: 17px; height: 16px; }
    .highlight {
      border: 1px solid #000000;

      font-weight: bold;

      font-size: 45px;
      background-color: #cedc98;
    }
    .default {
      background: #cedc98;
```

```

        border: 1px solid #DDDDDD;
        color: #333333;
    }
    .wrap{
        display: table-row-group;
    }
    .wrap1{
        float:left;
        width: 100px;
    }
</style>
<script>
    $(function() {
        $( "#sortable-10" ).sortable({
            start: function (event, ui) {
                $("span#result").html ($("span#result").html ()
                    + "<b>start</b><br>");
            },
            receive : function (event, ui)
            {
                $("span#result").html ($("span#result").html ()
                    + ", receive");
            },
            stop: function (event, ui) {
                $("span#result").html ($("span#result").html ()
                    + "<b>stop</b><br>");
            }
        });
        $( "#sortable-11" ).sortable({
            connectWith : "#sortable-10, #sortable-11"

        });

    });
</script>
</head>
<body>
    <div class="wrap">

```



```

<div class="wrap1">
  <h3>List 1</h3>
  <ul id="sortable-10">
    <li class="default">A</li>
    <li class="default">B</li>
    <li class="default">C</li>
    <li class="default">D</li>
  </ul>
</div>
<div class="wrap1">
  <h3>List 2</h3>
  <ul id="sortable-11">
    <li class="default">a</li>
    <li class="default">b</li>
    <li class="default">c</li>
    <li class="default">d</li>
  </ul>
</div>
</div>
<hr />
<span id=result></span>
</body>
</html>

```

Let's save the above code in an HTML file **sortexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

List 1

A

B

C

D

List 2

a

b

c

d

Try sorting the items in List 1, you will see the message getting displayed at the *start* and *stop* of event. Now drop items from List 2 to List 1, again a message gets displayed on the *receive* event.

Unit II – JQueryUI Widgets

8. JQUERYUI – ACCORDION

Accordion Widget in jQueryUI is a jQuery based expandable and collapsible content holder that is broken into sections and probably looks like tabs. jQueryUI provides `accordion()` method to achieve this.

Syntax

The **`accordion()`** method can be used in two forms:

- `$(selector, context).accordion (options) Method`
- `$(selector, context).accordion ("action", params) Method`

`$(selector, context).accordion (options) Method`

The *accordion (options)* method declares that an HTML element and its contents should be treated and managed as accordion menus. The *options* parameter is an object that specifies the appearance and behavior of the menu involved.

Syntax

```
$(selector, context).accordion (options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).accordion({option1: value1, option2: value2..... });
```

Following table lists the different *options* that can be used with this method:

Option	Description
<u>active</u>	Indicates the index of the menu that is open when the page is first accessed. By default its value is 0, i.e the first menu.
	Option - active
	Indicates the index of the menu that is open when the page is first accessed. By default its value is 0, i.e the first menu.
	This can be of type: Boolean: If set to false will collapse all panels. This requires the collapsible option to be true.

	<p>Integer: The zero-based index of the panel that is active (open). A negative value selects panels going backward from the last panel.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ active: 2 });</pre>
<u>animate</u>	<p>This option is used to set how to animate changing panels. By default its value is {}.</p> <p>Option - animate</p> <p>This option is used to set how to animate changing panels. By default its value is {}.</p> <p>This can be of type:</p> <p>Boolean: A value of false will disable animations.</p> <p>Number: This is a duration in milliseconds</p> <p>String: Name of easing to use with default duration.</p> <p>Object: Animation settings with easing and duration properties.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ animate: "bounceslide" });</pre>

<u>collapsible</u>	<p>This option when set to <i>true</i>, it allows users to close a menu by clicking on it. By default, clicks on the open panel's header have no effect. By default its value is false.</p> <p>Option - collapsible</p> <p>This option when set to <i>true</i>, it allows users to close a menu by clicking on it. By default, clicks on the open panel's header have no effect. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ collapsible: true });</pre>
<u>disabled</u>	<p>This option when set to <i>true</i> disables the accordion. By default its value is false.</p> <p>Option - disabled</p> <p>This option when set to <i>true</i> disables the accordion. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ disabled: true });</pre>
<u>event</u>	<p>This option specifies the event used to select an accordion header. By default its value is click.</p> <p>Option - event</p>

	<p>This option specifies the event used to select an accordion header. By default its value is click.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ event: "mouseover" });</pre>
<u>header</u>	<p>This option specifies a selector or element to override the default pattern for identifying the header elements. By default its value is > li > :first-child,> :not(li):even.</p> <p>Option - header</p> <p>This option specifies a selector or element to override the default pattern for identifying the header elements. By default its value is > li > :first-child,> :not(li):even.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ header: "h3" });</pre>
<u>heightStyle</u>	<p>This option is used to control the height of accordion and panels. By default its value is auto.</p> <p>Option - heightStyle</p> <p>This option is used to control the height of accordion and panels. By default its value is auto.</p> <p>Possible values are:</p> <p>auto: All panels will be set to the height of the tallest panel.</p>

	<p>fill: Expand to the available height based on the accordion's parent height.</p> <p>content: Each panel will be only as tall as its content.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ heightStyle: "fill" });</pre>
<u>icons</u>	<p>This option is an object that defines the icons to use to the left of the header text for opened and closed panels. The icon to use for closed panels is specified as a property named <i>header</i>, whereas the icon to use for open panels is specified as a property named <i>headerSelected</i>. By default its value is { "header": "ui-icon-triangle-1-e", "activeHeader": "ui-icon-triangle-1-s" }.</p> <p>Option - icons</p> <p>This option is an object that defines the icons to use to the left of the header text for opened and closed panels. The icon to use for closed panels is specified as a property named <i>header</i>, whereas the icon to use for open panels is specified as a property named <i>headerSelected</i>. By default its value is { "header": "ui-icon-triangle-1-e", "activeHeader": "ui-icon-triangle-1-s" }.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ icons: { "header": "ui-icon-plus", "activeHeader": "ui-icon-minus" } });</pre>

The following section will show you a few working examples of accordion widget functionality.

Default Functionality

The following example demonstrates a simple example of accordion widget functionality, passing no parameters to the **accordion ()** method.


```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Accordion Example </title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script>
      $(function() {
        $( "#accordion-1" ).accordion();
      });
    </script>
    <style>
      #accordion-1{font-size: 14px;}
    </style>
  </head>
  <body>
    <div id="accordion-1">
      <h3>Tab 1</h3>
      <div>
        <p>
          Lorem ipsum dolor sit amet, consectetur adipisicing elit,
          sed do eiusmod tempor incididunt ut labore et dolore magna
          aliqua.

          Ut enim ad minim veniam, quis nostrud exercitation ullamco
          laboris nisi ut aliquip ex ea commodo consequat.
        </p>
      </div>
      <h3>Tab 2</h3>
      <div>
        <p>
          Lorem ipsum dolor sit amet, consectetur adipisicing elit,
          sed do eiusmod tempor incididunt ut labore et dolore magna
          aliqua.

```

```

        Ut enim ad minim veniam, quis nostrud exercitation ullamco
        laboris nisi ut aliquip ex ea commodo consequat.

    </p>
</div>
<h3>Tab 3</h3>
<div>
    <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
        sed do eiusmod tempor incididunt ut labore et dolore magna
        aliqua.
        Ut enim ad minim veniam, quis nostrud exercitation ullamco
        laboris nisi ut aliquip ex ea commodo consequat.
    </p>

</div>
</div>
</body>
</html>

```

Let us save the above code in an HTML file **accordionexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Tab 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Tab 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Tab 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Click headers(Tab 1,Tab 2,Tab 3) to expand/collapse content that is broken into logical sections, much like tabs.

Use of collapsible

The following example demonstrates the usage of three options **collapsible** in the accordion widget of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Accordion Example </title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script>
      $(function() {
        $( "#accordion-2" ).accordion({
          collapsible: true
        });
      });
    </script>
    <style>
      #accordion-2{font-size: 14px;}
    </style>
  </head>
  <body>
    <div id="accordion-2">
      <h3>Tab 1</h3>
      <div>
        <p>
          Lorem ipsum dolor sit amet, consectetur adipisicing elit,
          sed do eiusmod tempor incididunt ut labore et dolore magna
          aliqua. Ut enim ad minim veniam, quis nostrud exercitation
          ullamco laboris nisi ut aliquip ex ea commodo consequat.
        </p>
      </div>
    </div>

    <h3>Tab 2</h3>

    <div>
```

```

    <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
        sed do eiusmod tempor incididunt ut labore et dolore magna
        aliqua. Ut enim ad minim veniam, quis nostrud exercitation
        ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </p>
</div>
<h3>Tab 3</h3>
<div>
    <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
        sed do eiusmod tempor incididunt ut labore et dolore magna
        aliqua. Ut enim ad minim veniam, quis nostrud exercitation
        ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </p>
    <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
    </ul>
</div>
</div>
</body>
</html>

```

Let us save the above code in an HTML file **accordionexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Tab 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Tab 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Tab 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

- List item one
- List item two
- List item three

Here we have set collapsible to *true*. Click on an accordion header, this allows collapsing the active section.

Use of Heightstyle

The following example demonstrates the usage of three options **heightStyle** in the accordion widget of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Accordion Example </title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script>
      $(function() {
        $( "#accordion-3" ).accordion({
          heightStyle: "content"
        });
        $( "#accordion-4" ).accordion({
          heightStyle: "fill"
        });
      });
    </script>
    <style>
      #accordion-3, #accordion-4{font-size: 14px;}
    </style>
  </head>

  <body>
```

```

<h3>Height style-content</h3>
<div style="height:250px">
  <div id="accordion-3">
    <h3>Tab 1</h3>
    <div>
      <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing
        elit, sed do eiusmod tempor incididunt ut labore et
        dolore magna aliqua.
      </p>
      <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
        <li>List item four</li>
        <li>List item five</li>
      </ul>
    </div>
    <h3>Tab 2</h3>
    <div>
      <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing
        elit, sed do eiusmod tempor incididunt ut labore et
        dolore magna aliqua.
      </p>
    </div>
    <h3>Tab 3</h3>
    <div>
      <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing
        elit,
        sed do eiusmod tempor incididunt ut labore et dolore
        magna aliqua.
      </p>
    </div>
  </div>
</div>

```

```

</div><br><br>
<h3>Height style-Fill</h3>
  <div style="height:250px">
    <div id="accordion-4">
      <h3>Tab 1</h3>
      <div>
        <p>
          Lorem ipsum dolor sit amet, consectetur adipisicing
          elit, sed do eiusmod tempor incididunt ut labore
          et dolore magna aliqua.
        </p>
        <ul>
          <li>List item one</li>
          <li>List item two</li>
          <li>List item three</li>
          <li>List item four</li>
          <li>List item five</li>
        </ul>
      </div>
      <h3>Tab 2</h3>
      <div>
        <p>
          Lorem ipsum dolor sit amet, consectetur adipisicing
          elit, sed do eiusmod tempor incididunt ut labore
          et dolore magna aliqua.
        </p>
      </div>
      <h3>Tab 3</h3>
      <div>
        <p>
          Lorem ipsum dolor sit amet, consectetur adipisicing
          elit, sed do eiusmod tempor incididunt ut labore
          et dolore magna aliqua.
        </p>
      </div>
    </div>
  </div>

```

```

    </div>
  </body>
</html>

```

Let us save the above code in an HTML file **accordionexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Height style-content

Tab 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- List item one
- List item two
- List item three
- List item four
- List item five

Tab 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Tab 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Height style-Fill

Tab 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- List item one
- List item two
- List item three
- List item four
- List item five

Tab 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Tab 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Here we have two accordions, the first one has *heightStyle* option set to *content*, which allows the accordion panels to keep their native height. Second accordion has *heightStyle* option set to *fill*, the script will automatically set the dimensions of the accordion to the height of its parent container.

\$(selector, context).accordion ("action", params) Method

The *accordion* ("action", params) method can perform an action on accordion elements, such as selecting/de-selecting the accordion menu. The action is specified as a string in the first argument (e.g., "disable" disables all menus). Check out the actions that can be passed, in the following table.

Syntax

```
$(selector, context).accordion ("action", params);;
```

Following table lists the different *actions* that can be used with this method:

Action	Description
<u>destroy</u>	<p>This action destroys the accordion functionality of an element completely. The elements return to their pre-init state.</p> <p>Action - destroy</p> <p>This action destroys the accordion functionality of an element completely. The elements return to their pre-init state.</p> <p>Syntax</p> <p><code>\$(".selector").accordion("destroy");</code></p>
<u>disable</u>	<p>This action disable all menus. No click will be taken into account. This method does not accept any arguments.</p> <p>Action - disable</p> <p>This action disable all menus. No click will be taken into account. This method does not accept any arguments.</p> <p>Syntax</p> <p><code>\$(".selector").accordion("disable");</code></p>
<u>enable</u>	<p>This action reactivate all menus. The clicks are again considered. This method does not accept any arguments.</p> <p>Action - enable</p>

	<p>This action reactivate all menus. The clicks are again considered. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").accordion("enable");</pre>
<u>option(optionName)</u>	<p>This action gets the value of currently associated accordion element with the specified <i>optionName</i>. This takes a String value as argument.</p> <p>Action - option(optionName)</p> <p>This action gets the value of currently associated accordion element with the specified <i>optionName</i>. This takes a String value as argument.</p> <p>Syntax</p> <pre>var isDisabled = \$(".selector").accordion("option", "disabled");</pre>
<u>option</u>	<p>This action gets an object containing key/value pairs representing the current accordion options hash.</p> <p>Action - option</p> <p>This action gets an object containing key/value pairs representing the current accordion options hash.</p> <p>Syntax</p> <pre>var options = \$(".selector").accordion("option");</pre>
<u>option(optionName, value)</u>	<p>This action sets the value of the accordion option associated with the specified optionName.</p> <p>Action - option(optionName, value)</p> <p>This action sets the value of the accordion option associated with the specified optionName.</p> <p>Syntax</p> <pre>\$(".selector").accordion("option", "disabled", true);</pre>
<u>option(options)</u>	<p>This action sets one or more options for the accordion.</p> <p>Action - option(options)</p>

	<p>This action sets one or more options for the accordion. Where <i>options</i> is a map of option-value pairs to set.</p> <p>Syntax</p> <pre>\$(".selector").accordion("option", { disabled: true });</pre>
<u>refresh</u>	<p>This action processes any headers and panels that were added or removed directly in the DOM. It then recomputes the height of the accordion panels. Results depend on the content and the heightStyle option. This method does not accept any arguments.</p> <p>Action - refresh</p> <p>This action processes any headers and panels that were added or removed directly in the DOM. It then recomputes the height of the accordion panels. Results depend on the content and the heightStyle option. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").accordion("refresh");</pre>
<u>widget</u>	<p>This action returns the accordion widget element; the one annotated with the <i>ui-accordion</i> class name.</p> <p>Action - widget</p> <p>This action returns the accordion widget element; the one annotated with the <i>ui-accordion</i> class name.</p> <p>Syntax</p> <pre>var widget = \$(".selector").accordion("widget");</pre>

Example

Now let us see an example using the actions from the above table. The following example demonstrates the use of *option(optionName, value)* method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Accordion Example </title>
```

```

<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<script>
    $(function() {
        $( "#accordion-5" ).accordion({
            disabled: false
        });
        $("input").each(function () {
            $(this).change(function () {
                if ($(this).attr("id") == "disableaccordion") {
                    $("#accordion-5").accordion("option", "disabled", true);
                } else {
                    $("#accordion-5").accordion("option", "disabled", false);
                }
            });
        });
    });
</script>
<style>
    #accordion-5{font-size: 14px;}
</style>
</head>
<body>
    <div id="accordion-5">
        <h3>Tab 1</h3>
        <div>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipisicing
elit,

                sed do eiusmod tempor incididunt ut labore et dolore magna

                aliqua. Ut enim ad minim veniam, quis nostrud

                exercitation ullamco laboris nisi ut aliquip ex ea

                commodo consequat.

            </p>

```

```

</div>
<h3>Tab 2</h3>
<div>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing
    elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua. Ut enim ad minim veniam, quis
    nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea
    commodo consequat.
  </p>
</div>
<h3>Tab 3</h3>
<div>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing
    elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua. Ut enim ad minim veniam, quis
    nostrud exercitation ullamco laboris nisi ut aliquip
    ex ea commodo consequat.
  </p>
  <ul>
    <li>List item one</li>
    <li>List item two</li>
    <li>List item three</li>
  </ul>
</div>
</div>
<div style="margin-top:30px">
  <input type="radio" name="disable" id="disableaccordion"
    value="disable">Disable accordion
  <input type="radio" name="disable" id="enableaccordion"
    checked
    value="enable">Enable accordion
</div>
</body>

```

```
</html>
```

Let us save the above code in an HTML file **accordionexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Tab 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Tab 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Tab 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

- List item one
- List item two
- List item three

Here we demonstrate enabling and disabling of the accordions. Select the respective radio buttons to check each action.

Event Management on Accordion Elements

In addition to the accordion (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
<u>activate(event, ui)</u>	<p>This event is triggered when a menu is activated. This event is only fired on panel activation, it is not fired for the initial panel when the accordion widget is created.</p> <p>Event - activate(event, ui)</p> <p>This event is triggered when a menu is activated. This event is only fired on panel activation, it is not fired for the initial panel when the accordion widget is created. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>. Possible values of <i>ui</i> are:</p> <p>newHeader: A jQuery object representing the header that was just activated.</p>

	<p>oldHeader: A jQuery object representing the header that was just deactivated.</p> <p>newPanel: A jQuery object representing the panel that was just activated.</p> <p>oldPanel: A jQuery object representing the panel that was just deactivated.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ activate: function(event, ui) {} });</pre>
<u>beforeActivate(event, ui)</u>	<p>This event is triggered before a panel is activated. This event can be canceled to prevent the panel from activating.</p> <p>Event - beforeActivate(event, ui)</p> <p>This event is triggered before a panel is activated. This event can be canceled to prevent the panel from activating. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>. Possible values of <i>ui</i> are:</p> <p>newHeader: A jQuery object representing the header that is about to be activated.</p> <p>oldHeader: A jQuery object representing the header that is about to be deactivated.</p> <p>newPanel: A jQuery object representing the panel that is about to be activated.</p> <p>oldPanel: A jQuery object representing the panel that is about to be deactivated.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ beforeActivate: function(event, ui) {} });</pre>
<u>create(event, ui)</u>	<p>This event is triggered when the accordion is created.</p> <p>Event - create(event, ui)</p>

	<p>This event is triggered when the accordion is created. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>. Possible values of <i>ui</i> are:</p> <p>header: A jQuery object representing the active header.</p> <p>panel: A jQuery object representing the active panel.</p> <p>Syntax</p> <pre>\$(".selector").accordion({ create: function(event, ui) {} });</pre>
--	--

Example

The following example demonstrates the event method usage in accordion widgets. This example demonstrates the use of events *create*, *beforeActive* and *active*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Accordion Example </title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script>
      $(function() {
        $( "#accordion-6" ).accordion({
          create: function (event, ui) {
            $("span#result").html ($("#span#result").html () +
"<b>Created</b><br>");
          },
          beforeActivate : function (event, ui)
          {
            $("span#result").html ($("#span#result").html () + "
<b>beforeActivate</b><br>");
          },
          activate: function (event, ui) {
```



```

        $("span#result").html ($("span#result").html () +
"<b>activate</b><br>");
    }
    });
});
</script>
<style>
    #accordion-6{font-size: 14px;}
</style>
</head>
<body>
    <div id="accordion-6">
        <h3>Tab 1</h3>
        <div>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit,
                sed do eiusmod tempor incididunt ut labore et dolore
                magna aliqua. Ut enim ad minim veniam, quis nostrud
                exercitation ullamco laboris nisi ut aliquip ex ea
                commodo consequat.
            </p>
        </div>
        <h3>Tab 2</h3>
        <div>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit,
                sed do eiusmod tempor incididunt ut labore et dolore
                magna aliqua. Ut enim ad minim veniam, quis nostrud
                exercitation ullamco laboris nisi ut aliquip ex ea
                commodo consequat.
            </p>
        </div>
        <h3>Tab 3</h3>
        <div>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit,

```

```

        sed do eiusmod tempor incididunt ut labore et dolore
        magna aliqua. Ut enim ad minim veniam, quis nostrud
        exercitation ullamco laboris nisi ut aliquip ex ea
        commodo consequat.

    </p>
    <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
    </ul>
</div>
</div>
<hr />
<span id=result></span>
</body>
</html>

```

Let us save the above code in an HTML file **accordionexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Tab 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Tab 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Tab 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

- List item one
- List item two
- List item three

Here we are displaying a text at the bottom, based on events.

9. JQUERYUI –AUTOCOMPLETE

Auto completion is a mechanism frequently used in modern websites to provide the user with a list of suggestions for the beginning of the word, which he/she has typed in a text box. The user can then select an item from the list, which will be displayed in the input field. This feature prevents the user from having to enter an entire word or a set of words.

jQueryUI provides an autocomplete widget - a control that acts a lot like a `<select>` dropdown, but filters the choices to present only those that match what the user is typing into a control. jQueryUI provides the **autocomplete()** method to create a list of suggestions below the input field and adds new CSS classes to the elements concerned to give them the appropriate style.

Any field that can receive input can be converted into an Autocomplete, namely, `<input>` elements, `<textarea>` elements, and elements with the *contenteditable* attribute.

Syntax

The **autocomplete()** method can be used in two forms:

- `$(selector, context).autocomplete (options) Method`
- `$(selector, context).autocomplete ("action", params) Method`

\$ (selector, context).autocomplete (options) Method

The *autocomplete (options)* method declares that an HTML `<input>` element must be managed as an input field that will be displayed above a list of suggestions. The *options* parameter is an object that specifies the behavior of the list of suggestions when the user is typing in the input field.

Syntax

```
$(selector, context).autocomplete (options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).autocomplete({option1: value1, option2: value2..... });
```

Following table lists the different *options* that can be used with this method:

Option	Description
<u>appendTo</u>	This option is used append an element to the menu. By default its value is null. Syntax

	<pre>\$(".selector").autocomplete({ appendTo: "#identifier" });</pre>
<u>autoFocus</u>	<p>This option when set to <i>true</i>, the first item of the menu will automatically be focused when the menu is shown. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ autoFocus: true });</pre>
<u>delay</u>	<p>This option is an Integer representing number of milliseconds to wait before trying to obtain the matching values (as specified by the <i>source</i> option). This can help reduce thrashing when non-local data is being obtained by giving the user time to enter more characters before the search is initiated. By default its value is 300.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ delay: 500 });</pre>
<u>disabled</u>	<p>This option if specified and <i>true</i>, the autocomplete widget is initially disabled. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ disabled: true });</pre>
<u>minLength</u>	<p>The number of characters that must be entered before trying to obtain the matching values (as specified by the <i>source</i> option). This can prevent too large a value set from being presented when a few characters isn't enough to whittle the set down to a reasonable level. By default its value is 1.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ minLength: 0 });</pre>
<u>position</u>	<p>This option identifies the position of the suggestions menu in relation to the associated input element. The <i>of</i> option defaults to the input element, but you can specify another element to position against. By default its value is { my: "left top", at: "left bottom", collision: "none" }.</p> <p>Syntax</p>

	<pre>\$(".selector").autocomplete({ position: { my : "right top", at: "right bottom" } });</pre>
<p><u>source</u></p>	<p>This option specifies the manner in which the data that matches the input data is obtained. A value must be provided or the autocomplete widget won't be created.. This value can be a :</p> <p>String representing the URL of a server resource that will return matching data,</p> <p>An array of local data from which the value will be matched, or a function that serves as a general callback from providing the matching values.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ source: ["c++", "java", "php", "coldfusion", "javascript", "asp", "ruby"] });</pre>

The following section will show you a few working examples of autocomplete widget functionality.

Default Functionality

The following example demonstrates a simple example of autocomplete widget functionality, passing no parameters to the **autocomplete()** method .

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Autocomplete functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>

      $(function() {

        var availableTutorials = [
```

```

        "ActionScript",
        "Bootstrap",
        "C",
        "C++",
    ];
    $( "#autocomplete-1" ).autocomplete({
        source: availableTutorials
    });
});
</script>
</head>
<body>
    <!-- HTML -->
    <div class="ui-widget">
        <p>Type "a" or "s"</p>
        <label for="autocomplete-1">Tags: </label>
        <input id="autocomplete-1">
    </div>
</body>
</html>

```

Let's save the above code in an HTML file **autocompleteexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

Type "a" or "s"

Tags:

Use of autoFocus

The following example demonstrates the usage of option **autoFocus** in the autocomplete widget of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">

        <title>jQuery UI Autocomplete functionality</title>
    </head>
    <body>

```

```

    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">

    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
        $(function() {
            var availableTutorials = [
                "ActionScript",
                "Bootstrap",
                "C",
                "C++",
            ];
            $( "#autocomplete-2" ).autocomplete({
                source: availableTutorials,
                autoFocus:true
            });
        });
    </script>
</head>
<body>
    <!-- HTML -->
    <div class="ui-widget">
        <p>Type "a" or "s"</p>
        <label for="autocomplete-2">Tags: </label>
        <input id="autocomplete-2">
    </div>
</body>
</html>

```

Let's save the above code in an HTML file **autocompleteexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Type "a" or "s"

Tags:

Use of minLength and delay

The following example demonstrates the usage of two options **minLength** and **delay** in the autocomplete widget of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Autocomplete functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        var availableTutorials = [
          "ActionScript",
          "Bootstrap",
          "C",
          "C++",
          "Ecommerce",
          "Jquery",
          "Groovy",
          "Java",
          "JavaScript",
          "Lua",
          "Perl",
          "Ruby",
          "Scala",
          "Swing",
          "XHTML"
        ];
        $( "#autocomplete-3" ).autocomplete({
          minLength:2,
          delay:500,
          source: availableTutorials
```



```

        });
    });
</script>
</head>
<body>
    <!-- HTML -->
    <div class="ui-widget">
        <p>Type two letter for e.g:ja,sc etc</p>
        <label for="autocomplete-3">Tags: </label>
        <input id="autocomplete-3">
    </div>
</body>
</html>

```

Let's save the above code in an HTML file **autocompleteexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Type two letter for e.g:ja,sc etc

Tags:

Use of Label

The following example demonstrates the usage of option **label** in the autocomplete widget of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Autocomplete functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <!-- Javascript -->
        <script>
            $(function() {
                $( "#autocomplete-4" ).autocomplete({

```

```

        source: [
            { label: "India", value: "IND" },
            { label: "Australia", value: "AUS" }
        ]
    });
});
</script>
</head>
<body>
    <!-- HTML -->
    <div class="ui-widget">
        <p>Type I OR A</p>
        <input id="autocomplete-4">
    </div>
</body>
</html>

```

Let's save the above code in an HTML file **autocompleteexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Type I OR A

Use of External Source

The following example demonstrates the use of external file for **source** option in the autocomplete widget of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Autocomplete functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <!-- Javascript -->

        <script>

```

```

        $( "#autocomplete-5" ).autocomplete({
            source: "/jqueryui/search.php",
            minLength: 2
        });
    </script>
</head>
<body>
    <input id="autocomplete-5">
</body>
</html>

```

The file *search.php* is placed at the same location as the above file (autocompleteexample.html). Contents of search.php are as below:

```

<?
$term = $_GET[ "term" ];
$companies = array(
    array( "label" => "JAVA", "value" => "1" ),
    array( "label" => "DATA IMAGE PROCESSING", "value" => "2" ),
    array( "label" => "JAVASCRIPT", "value" => "3" ),
    array( "label" => "DATA MANAGEMENT SYSTEM", "value" => "4" ),
    array( "label" => "COMPUTER PROGRAMMING", "value" => "5" ),
    array( "label" => "SOFTWARE DEVELOPMENT LIFE CYCLE", "value" => "6" ),
    array( "label" => "LEARN COMPUTER FUNDAMENTALS", "value" => "7" ),
    array( "label" => "IMAGE PROCESSING USING JAVA", "value" => "8" ),
    array( "label" => "CLOUD COMPUTING", "value" => "9" ),
    array( "label" => "DATA MINING", "value" => "10" ),
    array( "label" => "DATA WAREHOUSE", "value" => "11" ),
    array( "label" => "E-COMMERCE", "value" => "12" ),
    array( "label" => "DBMS", "value" => "13" ),
    array( "label" => "HTTP", "value" => "14" )
);

$result = array();
foreach ($companies as $company) {

    $companyLabel = $company[ "label" ];

```

```

        if ( strpos( strtoupper($companyLabel), strtoupper($term) )
            !== false ) {
            array_push( $result, $company );
        }
    }

    echo json_encode( $result );
?>

```

Let's save the above code in an HTML file **autocompleteexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Type two letter words for e.g: ja, sc etc

`$(selector, context).autocomplete ("action", params) Method`

The *autocomplete ("action", params)* method can perform an action on the list of suggestions, such as show or hide. The action is specified as a String in the first argument (e.g., "close" to hide the list). Check out the actions that can be passed, in the following table.

Syntax

```
$(selector, context).autocomplete ("action", params);;
```

Following table lists the different actions that can be used with this method:

Action	Description
<u>close</u>	<p>This action hides the list of suggestions in the Autocomplete menu. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete("close");</pre>
<u>destroy</u>	<p>This action removes the autocomplete functionality. Lists of suggestions are deleted. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete("destroy");</pre>

<u>disable</u>	<p>This action disables the autocomplete mechanism. The list of suggestions no longer appears. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete("disable");</pre>
<u>enable</u>	<p>This action reactivates the autocomplete mechanism. The list of suggestions will again be displayed. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete("enable");</pre>
<u>option(optionName)</u>	<p>This action retrieves the value of the specified <i>optionName</i>. This option corresponds to one of those used with autocomplete (options).</p> <p>Syntax</p> <pre>var isDisabled = \$(".selector").autocomplete("option", "disabled");</pre>
<u>option</u>	<p>This action gets an object containing key/value pairs representing the current autocomplete options hash.</p> <p>Syntax</p> <pre>var options = \$(".selector").autocomplete("option");</pre>
<u>option(optionName, value)</u>	<p>This action sets the value of the autocomplete option associated with the specified <i>optionName</i>. The argument <i>optionName</i> is name of the option to be set and <i>value</i> is the value to be set for the option.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete("option", "disabled", true);</pre>
<u>option(options)</u>	<p>This action is sets one or more options for the autocomplete. The argument <i>options</i> is a map of option-value pairs to be set.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete("option", { disabled: true });</pre>
<u>search([value])</u>	<p>This action searches for correspondence between the string value and the data source (specified in <i>options.source</i>). The minimum number of characters (indicated in <i>options.minLength</i>) must be reached in value, otherwise the search is not performed.</p>

	Syntax <code>\$(".selector").autocomplete("search", "");</code>
<u>widget</u>	<p>Retrieve the DOM element corresponding to the list of suggestions. This is an object of jQuery class that allows easy access to the list without using jQuery selectors.</p> Syntax <code>\$(".selector").autocomplete("widget");</code>

Example

Now let us see an example using the actions from the above table. The following example demonstrates the use of *option(optionName, value)* method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Autocomplete functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        var availableTutorials = [
          "ActionScript",
          "Boostrap",
          "C",
          "C++",
          "Ecommerce",
          "Jquery",
          "Groovy",
          "Java",
          "JavaScript",
          "Lua",
          "Perl",
```

```

        "Ruby",
        "Scala",
        "Swing",
        "XHTML"
    ];
    $( "#autocomplete-6" ).autocomplete({
        source: availableTutorials
    });
    $( "#autocomplete-6" ).autocomplete("option", "position", { my :
"right-10 top+10", at: "right top" })
    });
</script>
</head>
<body>
    <!-- HTML -->
    <div class="ui-widget">
        <p>Type "a" or "s"</p>
        <label for="autocomplete-6">Tags: </label>
        <input id="autocomplete-6">
    </div>
</body>
</html>

```

Let's save the above code in an HTML file **autocompleteexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Type "a" or "s"

Tags:

Extension Points

The autocomplete widget can be extended as its built with the widget factory. When extending widgets, you have the ability to override or add to the behavior of existing methods. The following table lists methods that act as extension points with the same API stability as the plugin methods listed [above](#).

Method	Description
<u>renderItem(ul, item)</u>	This method controls the creation of each option in the widget's menu. This method creates a new element, appends it to the menu and return it.

<u><code>renderMenu(ul, items)</code></u>	This method controls building the widget's menu.
<u><code>resizeMenu()</code></u>	This method controls sizing the menu before it is displayed. The menu element is available at <i>this.menu.element</i> . This method does not accept any arguments.

Event Management on Autocomplete Elements

In addition to the autocomplete (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
<u><code>change(event, ui)</code></u>	<p>This event is triggered when the value of the <input> element is changed based upon a selection. When triggered, this event will always come after the <i>close</i> event is triggered.</p> <p>Possible values of ui are:</p> <p>item: The item selected from the menu, if any. Otherwise the property is null.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ change: function(event, ui) {} });</pre>
<u><code>close(event, ui)</code></u>	<p>This event is triggered whenever the autocomplete menu closes.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ close: function(event, ui) {} });</pre>
<u><code>create(event, ui)</code></u>	<p>This event is triggered when the autocomplete is created.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ create: function(event, ui) {} });</pre>
<u><code>focus(event, ui)</code></u>	<p>This event is triggered whenever one of the menu choices receives focus. Unless canceled (for example, by returning false), the</p>

	<p>focused value is set into the <input> element. Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>item: The focused item.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ focus: function(event, ui) {} });</pre>
<u>open(event, ui)</u>	<p>This event is triggered after the data has been readied and the menu is about to open.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ open: function(event, ui) {} });</pre>
<u>response(event, ui)</u>	<p>This event is triggered after a search completes, before the menu is shown. This event is always triggered when a search completes, even if the menu will not be shown because there are no results or the Autocomplete is disabled. Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>content: Contains the response data and can be modified to change the results that will be shown.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ response: function(event, ui) {} });</pre>

<p><u>search(event, ui)</u></p>	<p>This event is triggered after any delay and minLength criteria have been met, just before the mechanism specified by source is activated. If canceled, the search operation is aborted. Where event is of type Event, and ui is of type Object.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ search: function(event, ui) {} });</pre>
<p><u>select(event, ui)</u></p>	<p>This event is triggered when a value is selected from the autocomplete menu. Cancelling this event prevents the value from being set into the <input> element (but doesn't prevent the menu from closing). Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>item: An Object with label and value properties for the selected option.</p> <p>Syntax</p> <pre>\$(".selector").autocomplete({ select: function(event, ui) {} });</pre>

Example

The following example demonstrates the event method usage in autocomplete widget. This example demonstrates the use of events *focus*, and *select*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Autocomplete functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
```

```
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<style>
    #project-label {
        display: block;
        font-weight: bold;
        margin-bottom: 1em;
    }
    #project-icon {
        float: left;
        height: 32px;
        width: 32px;
    }
    #project-description {
        margin: 0;
        padding: 0;
    }
</style>
<!-- Javascript -->

<script>

    $(function() {
        var projects = [
            {
                value: "java",
                label: "Java",
                desc: "write once run anywhere",
            },
            {
                value: "jquery-ui",
                label: "jQuery UI",
                desc: "the official user interface library for jQuery",
            },
            {
                value: "Bootstrap",
                label: "Twitter Bootstrap",
                desc: "popular front end frameworks ",
            }
        ]
    })

```

```

    ];
    $( "#project" ).autocomplete({
        minLength: 0,
        source: projects,
        focus: function( event, ui ) {
            $( "#project" ).val( ui.item.label );
            return false;
        },
        select: function( event, ui ) {
            $( "#project" ).val( ui.item.label );
            $( "#project-id" ).val( ui.item.value );
            $( "#project-description" ).html( ui.item.desc );
            return false;
        }
    })
    .data( "ui-autocomplete" )._renderItem = function( ul, item ) {
        return $( "<li>" )
            .append( "<a>" + item.label + "<br>" + item.desc + "</a>" )
            .appendTo( ul );
    };
});
</script>
</head>
<body>
    <div id="project-label">Select a project (type "a" for a start):</div>
    <input id="project">
    <input type="hidden" id="project-id">
    <p id="project-description"></p>
</body>
</html>

```

Let's save the above code in a HTML file **autocompleteexample.htm** and open it in a standard browser which supports javascript. you must also see the following output:

Select a project (type "a" for a start):

10. JQUERYUI – BUTTON

jQueryUI provides `button()` method to transform the HTML elements (like buttons, inputs and anchors) into themeable buttons, with automatic management of mouse movements on them, all managed transparently by jQuery UI.

In order to group radio buttons, Button also provides an additional widget, called *Buttonset*. *Buttonset* is used by selecting a container element (which contains the radio buttons) and calling `.buttonset()`.

Syntax

The **button()** method can be used in two forms:

- `$(selector, context).button (options) Method`
- `$(selector, context).button ("action", params) Method`

`$(selector, context).button (options) Method`

The *button (options)* method declares that an HTML element should be treated as button. The *options* parameter is an object that specifies the behavior and appearance of the button.

Syntax

```
$(selector, context).button (options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).button({option1: value1, option2: value2..... });
```

Following table lists the different *options* that can be used with this method:

Option	Description
<u>disabled</u>	<p>This option deactivates the button is set to <i>true</i>. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").button({ disabled: true });</pre>
<u>icons</u>	<p>This option specifies that one or two icons are to be displayed in the <i>button</i>: <i>primary</i> icons to the left, secondary icons to the right. The primary icon is identified by the <i>primary</i> property of the object, and the <i>secondary</i> icon is identified by the <i>secondary</i> property. By default its value is <i>primary: null, secondary: null</i>.</p>

	Syntax <code>\$(".selector").button({ icons: { primary: "ui-icon-gear", secondary: "ui-icon-triangle-1-s" } });</code>
<u>label</u>	<p>This option specifies text to display on the button that overrides the natural label. If omitted, the natural label for the element is displayed. In the case of radio buttons and checkboxes, the natural label is the <code><label></code> element associated with the control. By default its value is null.</p> Syntax <code>\$(".selector").button({ label: "custom label" });</code>
<u>text</u>	<p>This option specifies whether text is to be displayed on the button. If specified as <i>false</i>, text is suppressed if (and only if) the icons option specifies at least one icon. By default its value is true.</p> Syntax <code>\$(".selector").button({ text: false });</code>

`$(selector, context).button ("action", params) Method`

The *button ("action", params)* method can perform an action on buttons, such as disabling the button. The action is specified as a string in the first argument (e.g., "disable" to disable button). Check out the actions that can be passed, in the following table.

Syntax

```
$(selector, context).button ("action", params);
```

The following table lists the different *actions* that can be used with this method:

Action	Description
<u>destroy</u>	This action removes the button functionality of an element completely. The elements return to their pre-init state. This method does not accept any arguments.
<u>disable</u>	This action disables the button functionality of an element. This method does not accept any arguments.
<u>enable</u>	This action enables the button functionality of an element. This method does not accept any arguments.
<u>option(optionName)</u>	This action retrieves the value of the option specified in <i>optionName</i> . Where <i>optionName</i> is a String.

<u>option</u>	This action retrieves an object containing key/value pairs representing the current button options hash.
<u>option(optionName, value)</u>	This action sets the value of the button option associated with the specified <i>optionName</i> .
<u>option(options)</u>	This action sets one or more options for the button. Where <i>options</i> is map of option-value pairs to set.
<u>refresh</u>	This action refreshes the display of the button. This is useful when the buttons are handled by the program and the display does not necessarily correspond to the internal state. This method does not accept any arguments.
<u>widget</u>	This action returns a jQuery object containing the button element. This method does not accept any arguments.

Event Management on Buttons

In addition to the button (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
<u>create(event, ui)</u>	<p>This event is triggered when the button is created. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Syntax</p> <pre>\$(".selector").button({ create: function(event, ui) {} });</pre>

Example

The following example demonstrates the event method usage for button widget functionality. This example demonstrates the use of event create.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Buttons functionality</title>
```



```

    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">

    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
        .resultarea {
            background: #cedc98;
            border-top: 1px solid #000000;
            border-bottom: 1px solid #000000;
            color: #333333;
            font-size:14px;
        }
    </style>
    <script>
        $(function() {
            $( "#button-12" ).button({
                create: function() {
                    $("p#result").html ($("p#result")
                    .html () + "<b>created</b><br>");
                }
            });
        });
    </script>
</head>
<body>

    <button id="button-12">
        A button element
    </button>

    <p class="resultarea" id=result></p>

</body>
</html>

```

Let us save the above code in an HTML file **buttonexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

A Button Element

11. JQUERYUI – DATEPICKER

Datepickers in jQueryUI allow users to enter dates easily and visually. You can customize the date format and language, restrict the selectable date ranges and add in buttons and other navigation options easily.

jQueryUI provides **datepicker()** method that creates a datepicker and changes the appearance of HTML elements on a page by adding new CSS classes. Transforms the <input>, <div>, and elements in the wrapped set into a datepicker control.

By default, for <input> elements, the datepicker calendar opens in a small overlay when the associated text field gains focus. For an inline calendar, simply attach the datepicker to a <div>, or element.

Syntax

The **datepicker()** method can be used in two forms:

- \$(selector, context).datepicker (options) Method
- \$(selector, context).datepicker ("action", [params]) Method

`$(selector, context).datepicker (options) Method`

The *datepicker (options)* method declares that an <input> element (or <div>, or , depending on how you choose to display the calendar) should be managed as a datepicker. The *options* parameter is an object that specifies the behavior and appearance of the datepicker elements.

Syntax

```
$(selector, context).datepicker(options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).datepicker({option1: value1, option2: value2..... });
```

The following table lists the different *options* that can be used with this method:

Option	Description
<u>altField</u>	This option specifies a jQuery selector for a field that is also updated with any date selections. The <i>altFormat</i> option can be used to set the format for this value. This is quite useful for setting date values into a hidden input element to be submitted to the server, while displaying a more user-friendly format to the user. By default its value is "".
	Syntax

	<pre>\$(".selector").datepicker({ altField: "#actualDate" });</pre>
<u>altFormat</u>	<p>This option is used when an <i>altField</i> option is specified. It provides the format for the value to be written to the alternate element. By default its value is "".</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ altFormat: "yy-mm-dd" });</pre>
<u>appendText</u>	<p>This option is a String value to be placed after the <input> element, intended to show instructions to the user. By default its value is "".</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ appendText: "(yyyy-mm-dd)" });</pre>
<u>autoSize</u>	<p>This option when set to <i>true</i> resizes the <input> element to accommodate the datepicker's date format as set with the <i>dateFormat</i> option. By default its value is <i>false</i>.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ autoSize: true });</pre>
<u>beforeShow</u>	<p>This option is a callback function that's invoked just before a datepicker is displayed, with the <input> element and datepicker instance passed as parameters. This function can return an options hash used to modify the datepicker. By default its value is "".</p>
<u>beforeShowDay</u>	<p>This option is a callback function which takes a date as parameter, that's invoked for each day in the datepicker just before it's displayed, with the date passed as the only parameter. This can be used to override some of the default</p>

	behavior of the day elements. This function must return a three-element array. By default its value is null.
<u>buttonImage</u>	This option specifies the path to an image to be displayed on the button enabled by setting the <i>showOn</i> option to one of buttons or both. If <i>buttonText</i> is also provided, the button text becomes the <i>alt</i> attribute of the button. By default its value is "".
<u>buttonImageOnly</u>	This option if set to <i>true</i> , specifies that the image specified by <i>buttonImage</i> is to appear standalone (not on a button). The <i>showOn</i> option must still be set to one of button or both for the image to appear. By default its value is false.
<u>buttonText</u>	This option specifies the caption for the button enabled by setting the <i>showOn</i> option to one of <i>button</i> or <i>both</i> . By default its value is "...".
<u>calculateWeek</u>	This option is a custom function to calculate and return the week number for a date passed as the lone parameter. The default implementation is that provided by the <i>\$.datepicker.iso8601Week()</i> utility function.
<u>changeMonth</u>	This option if set to <i>true</i> , a month dropdown is displayed, allowing the user to directly change the month without using the arrow buttons to step through them. By default its value is false.
<u>changeYear</u>	This option if set to <i>true</i> , a year dropdown is displayed, allowing the user to directly change the year without using the arrow buttons to step through them. Option <i>yearRange</i> can be used to control which years are made available for selection. By default its value is false.
<u>closeText</u>	This option specifies the text to replace the default caption of Done for the close button. It is used when the button panel is displayed via the <i>showButtonPanel</i> option. By default its value is "Done".
<u>constrainInput</u>	This option if set <i>true</i> (the default), text entry into the <code><input></code> element is constrained to characters allowed by the <i>currentdateformat</i> option. By default its value is true.
<u>currentText</u>	This option specifies the text to replace the default caption of Today for the current button. This is used if the button panel is displayed via the <i>showButtonPanel</i> option. By default its value is Today.

<u>dateFormat</u>	This option specifies the date format to be used. By default its value is mm/dd/yy.
<u>dayNames</u>	This option is a 7-element array providing the full day names with the 0th element representing Sunday. Can be used to localize the control. By default its value is ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"].
<u>dayNamesMin</u>	This option is a 7-element array providing the minimal day names with the 0th element representing Sunday, used as column headers. Can be used to localize the control. By default its value is ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"].
<u>dayNamesShort</u>	This option specifies a 7-element array providing the short day names with the 0th element representing Sunday. Can be used to localize the control. By default its value is ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"].
<u>defaultDate</u>	This option sets the initial date for the control, overriding the default value of today, if the <input> element has no value. This can be a <i>Date</i> instance, the <i>number</i> of days from today, or a <i>string</i> specifying an absolute or relative date. By default its value is null.
<u>duration</u>	<p>This option specifies the speed of the animation that makes the datepicker appear. Can be one of <i>slow</i>, <i>normal</i>, or <i>fast</i>, or the number of milliseconds for the animation to span. By default its value is normal.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ duration: "slow" });</pre>
<u>firstDay</u>	<p>This option specifies which day is considered the first day of the week, and will be displayed as the left-most column. By default its value is 0.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ firstDay: 1 });</pre>

<u>gotoCurrent</u>	<p>This option when set to <i>true</i>, the current day link is set to the selected date, overriding the default of today. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ gotoCurrent: true });</pre>
<u>hideIfNoPrevNext</u>	<p>This option if set to <i>true</i>, hides the next and previous links (as opposed to merely disabling them) when they aren't applicable, as determined by the settings of the <i>minDate</i> and <i>maxDate</i> options. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ hideIfNoPrevNext: true });</pre>
<u>isRTL</u>	<p>This option when set to <i>true</i>, specifies the localizations as a right-to-left language. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ isRTL: true });</pre>
<u>maxDate</u>	<p>This option sets the maximum selectable date for the control. This can be a <i>Date</i> instance, the number of days from today, or a string specifying an absolute or relative date. By default its value is null.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ maxDate: "+1m +1w" });</pre>
<u>minDate</u>	<p>This option sets the minimum selectable date for the control. This can be a <i>Date</i> instance, the <i>number</i> of days from today, or a <i>string</i> specifying an absolute or relative date. By default its value is null.</p>

	<p>Syntax</p> <pre>\$(".selector").datepicker({ minDate: new Date(2007, 1 - 1, 1) });</pre>
<u>monthNames</u>	<p>This option is a 12-element array providing the full month names with the 0th element representing January. Can be used to localize the control. By default its value is ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"].</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ monthNames: ["janvier", "février", "mars", "avril", "mai", "juin", "juillet", "août", "septembre", "octobre", "novembre", "décembre"] });</pre>
<u>monthNamesShort</u>	<p>This option specifies a 12-element array providing the short month names with the 0th element representing January. Can be used to localize the control. By default its value is ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"].</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ monthNamesShort: ["jan", "fév", "mar", "avr", "mai", "Jui", "Jul", "aoû", "sep", "Oot", "nov", "déc"] });</pre>
<u>navigationAsDateFormat</u>	<p>This option if set to <i>true</i>, the navigation links for <i>nextText</i>, <i>prevText</i>, and <i>currentText</i> are passed through the <i>\$.datepicker.formatDate()</i> function prior to display. This allows date formats to be supplied for those options that get replaced with the relevant values. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").datepicker(</pre>

	<pre>{ navigationAsDateFormat: true });</pre>
<u>nextText</u>	<p>This option specifies the text to replace the default caption of Next for the next month navigation link. ThemeRoller replaces this text with an icon. By default its value is Next.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ nextText: "Later" });</pre>
<u>numberOfMonths</u>	<p>This option specifies the number of months to show in the datepicker. By default its value is 1.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ numberOfMonths: [2, 3] });</pre>
<u>onChangeMonthYear</u>	<p>This option is a callback that's invoked when the datepicker moves to a new month or year, with the selected year, month (1-based), and datepicker instance passed as parameters, and the function context is set to the input field element. By default its value is null.</p>
<u>onClose</u>	<p>This option is a callback invoked whenever a datepicker is closed, passed the selected date as text (the empty string if there is no selection), and the datepicker instance, and the function context is set to the input field element. By default its value is null.</p>
<u>onSelect</u>	<p>This option is a callback invoked whenever a date is selected, passed the selected date as text (the empty string if there is no selection), and the datepicker instance, and the function context is set to the input field element. By default its value is null.</p>
<u>prevText</u>	<p>This option specifies the text to replace the default caption of <i>Prev</i> for the previous month navigation link. (Note that the ThemeRoller replaces this text with an icon). By default its value is PrevdefaultDatedayNamesMin.</p>

<u>selectOtherMonths</u>	<p>This option if set to <i>true</i>, days shown before or after the displayed month(s) are selectable. Such days aren't displayed unless the <i>showOtherMonths</i> option is true. By default its value is false.</p>
<u>shortYearCutoff</u>	<p>This option if its a number, specifies a value between 0 and 99 years before which any 2-digit year values will be considered to belong to the previous century. If this option is a string, the value undergoes a numeric conversion and is added to the current year. The default is +10 which represents 10 years from the current year.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ shortYearCutoff: 50 });</pre>
<u>showAnim</u>	<p>This option specifies sets the name of the animation to be used to show and hide the datepicker. If specified, must be one of <i>show</i> (the default), <i>fadeIn</i>, <i>slideDown</i>, or any of the jQuery UI show/hide animations. By default its value is show.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ showAnim: "fold" });</pre>
<u>showButtonPanel</u>	<p>This option if set to <i>true</i>, a button panel at the bottom of the datepicker is displayed, containing current and close buttons. The caption of these buttons can be provided via the <i>currentText</i> and <i>closeText</i> options. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ showButtonPanel: true });</pre>
<u>showCurrentAtPos</u>	<p>This option specifies the 0-based index, starting at the upper left, of where the month containing the current date should be placed within a multi-month display. By default its value is 0.</p>

	<p>Syntax</p> <pre>\$(".selector").datepicker({ showCurrentAtPos: 3 });</pre>
<u>showMonthAfterYear</u>	<p>This option specifies if set to <i>true</i>, the positions of the month and year are reversed in the header of the datepicker. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ showMonthAfterYear: true });</pre>
<u>showOn</u>	<p>This option specifies when the datepicker should appear. The possible values are <i>focus</i>, <i>button</i> or <i>both</i>. By default its value is <i>focus</i>.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ showOn: "both" });</pre>
<u>showOptions</u>	<p>This option provides a hash to be passed to the animation when a jQuery UI animation is specified for the <i>showAnim</i> option. By default its value is <i>{}</i>.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ showOptions: { direction: "up" } });</pre>
<u>showOtherMonths</u>	<p>This option if set to <i>true</i>, dates before or after the first and last days of the current month are displayed. These dates aren't selectable unless the <i>selectOtherMonths</i> option is also set to <i>true</i>. By default its value is false.</p> <p>Syntax</p>

	<pre>\$(".selector").datepicker({ showOtherMonths: true });</pre>
<u>showWeek</u>	<p>This option if set to <i>true</i>, the week number is displayed in a column to the left of the month display. The <i>calculateWeek</i> option can be used to alter the manner in which this value is determined. By default its value is <i>false</i>.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ showWeek: true });</pre>
<u>stepMonths</u>	<p>This option specifies specifies how many months to move when one of the month navigation controls is clicked. By default its value is 1.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ stepMonths: 3 });</pre>
<u>weekHeader</u>	<p>This option specifies the text to display for the week number column, overriding the default value of <i>Wk</i>, when <i>showWeek</i> is <i>true</i>. By default its value is <i>Wk</i>.</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ weekHeader: "W" });</pre>
<u>yearRange</u>	<p>This option specifies limits on which years are displayed in the dropdown in the form <i>from:to</i> when <i>changeYear</i> is <i>true</i>. The values can be absolute or relative (for example: 2005:+2, for 2005 through 2 years from now). The prefix <i>c</i> can be used to make relative values offset from the selected year rather than the current year (example: c-2:c+3). By default its value is c-10:c+10.</p> <p>Syntax</p> <pre>\$(".selector").datepicker(</pre>

	<pre>{ yearRange: "2002:2012" });</pre>
<u>yearSuffix</u>	<p>This option displays additional text after the year in the datepicker header. By default its value is "".</p> <p>Syntax</p> <pre>\$(".selector").datepicker({ yearSuffix: "CE" });</pre>

The following section will show you a few working examples of datepicker functionality.

Default Functionality

The following example demonstrates a simple example of datepicker functionality passing no parameters to the **datepicker()** method .

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-1" ).datepicker();
      });
    </script>
  </head>
  <body>
    <!-- HTML -->
    <p>Enter Date: <input type="text" id="datepicker-1"></p>
  </body>
```

```
</html>
```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Date:

Inline Datepicker

The following example demonstrates a simple example of inline datepicker functionality.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-2" ).datepicker();
      });
    </script>
  </head>
  <body>
    <!-- HTML -->
    Enter Date: <div id="datepicker-2"></div>
  </body>
</html>
```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Start Date:

In the above example we use <div> element to get the inline date picker.

Use of appendText, dateFormat, altField and altFormat

The following example shows the usage of three important options **(a) appendText** **(b) dateFormat** **(c) altField** and **(d) altFormat** in the datepicker function of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-3" ).datepicker({
          appendText:"(yy-mm-dd)",
          dateFormat:"yy-mm-dd",
          altField: "#datepicker-4",
          altFormat: "DD, d MM, yy"
        });
      });
    </script>
  </head>
  <body>
    <!-- HTML -->
    <p>Enter Date: <input type="text" id="datepicker-3"></p>
    <p>Alternate Date: <input type="text" id="datepicker-4"></p>
  </body>
</html>
```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Date:

Alternate Date:

In the above example, you can see that, the date format for first input is set as *yy-mm-dd*. If you select some date from datepicker, the same date is reflected in the second input field whose date format is set as "DD, MM, yy".

Use of beforeShowDay

The following example shows the usage of option **beforeShowDay** in the datepicker function of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-5" ).datepicker({
          beforeShowDay : function (date)
          {
            var dayOfWeek = date.getDay ();
            // 0 : Sunday, 1 : Monday, ...
            if (dayOfWeek == 0 || dayOfWeek == 6) return [false];
            else return [true];
          }
        });
      });
    </script>
  </head>
  <body>
    <!-- HTML -->
    <p>Enter Date: <input type="text" id="datepicker-5"></p>
  </body>
</html>
```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Date:

In the above example sunday and saturday are disabled.

Use of showon, buttonimage, and buttonimageonly

The following example shows the usage of three important options **(a) showOn** **(b) buttonImage** and **(c) buttonImageOnly** in the datepicker function of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-6" ).datepicker({
          showOn:"button",
          buttonImage: "/jqueryui/images/calendar-icon.png",
          buttonImageOnly: true
        });
      });
    </script>
  </head>
  <body>
    <!-- HTML -->
    <p>Enter Date: <input type="text" id="datepicker-6"></p>
  </body>
</html>
```


Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Date:

In the above example an icon is displayed which needs to be clicked to open the datepicker.

Use of defaultDate, dayNamesMin, and duration

The following example shows the usage of three important options **(a) dayNamesMin** **(b) dayNamesMin** and **(c) duration** in the datepicker function of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-7" ).datepicker({
          defaultDate:+9,
          dayNamesMin: [ "So", "Mo", "Di", "Mi", "Do", "Fr", "Sa" ],
          duration: "slow"
        });
      });
    </script>
  </head>
  <body>
    <!-- HTML -->
    <p>Enter Date: <input type="text" id="datepicker-7"></p>
  </body>
</html>
```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Date:

In the above example the names of the days are changed using *dayNamesMin*. You can also see a default date is set.

Use of prevText, nextText, showOtherMonths and selectOtherMonths

The following example shows the usage of three important options **(a) prevText** **(b) nextText** (c) showOtherMonths and **(d) selectOtherMonths** in the datepicker function of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-8" ).datepicker({
          prevText:"click for previous months",
          nextText:"click for next months",
          showOtherMonths:true,
          selectOtherMonths: false
        });
        $( "#datepicker-9" ).datepicker({
          prevText:"click for previous months",
          nextText:"click for next months",
          showOtherMonths:true,
          selectOtherMonths: true
        });
      });
    </script>
  </head>
  <body>
```

```
<!-- HTML -->
<p>Enter Start Date: <input type="text" id="datepicker-8"></p>
<p>Enter End Date: <input type="text" id="datepicker-9"></p>
</body>
</html>
```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Start Date:

Enter End Date:

In the above example the prev and next links have captions. If you click on the element, the datepicker opens. Now in the first datepicker, the other months dates are disabled as `selectOtherMonths` is set *false*. In the second date picker for second input type, the `selectOtherMonths` is set to *true*.

Use of `changeMonth`, `changeYear`, and `numberOfMonths`

The following example shows the usage of three important options **(a) `changeMonth`** **(b) `changeYear`** and **(c) `numberOfMonths`** in the datepicker function of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-10" ).datepicker({
          changeMonth:true,
          changeYear:true,
          numberOfMonths:[2,2]
        });
      });
    </script>
```

```

    </script>
</head>
<body>
    <!-- HTML -->
    <p>Enter Date: <input type="text" id="datepicker-10"></p>
</body>
</html>

```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Date:

In the above example you can see dropdown menus for Month and Year fields. And we are displaying 4 months in an array structure of [2,2].

Use of showWeek, yearSuffix, and showAnim

The following example shows the usage of three important options **(a) showWeek (b) yearSuffix** and **(c) showAnim** in the datepicker function of JQueryUI.

```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-11" ).datepicker({
          showWeek:true,
          yearSuffix:"-CE",
          showAnim: "slide"
        });
      });
    </script>

```

```

    </script>
</head>
<body>
    <!-- HTML -->
    <p>Enter Date: <input type="text" id="datepicker-11"></p>
</body>
</html>

```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Date:

In the above example, you can see that, week numbers are displayed on the left side of datepicker as *showWeek* is set to *true*. The year is have a suffix of "-CE".

`$(selector, context).datepicker ("action", [params])` Method

The *datepicker (action, params)* method can perform an action on the calendar, such as such as selecting a new date. The **action** is specified as a string in the first argument and optionally, one or more **params** can be provided based on the given action.

Basically, here actions are nothing but they are jQuery methods which we can use in the form of string.

Syntax

```
$(selector, context).datepicker ("action", [params]);
```

The following table lists the actions for this method:

Action	Description
<u>destroy()</u>	This action removes the datepicker functionality completely. This will return the element back to its pre-init state. This method does not accept any arguments.
<u>dialog(date [, onSelect] [, settings] [, pos])</u>	This action displays datepicker in a jQuery UI dialog box .
<u>getDate()</u>	This action returns the Date corresponding to the selected date. This method does not accept any arguments.
<u>hide()</u>	This action closes the previously opened date picker. This method does not accept any arguments.

<u>isDisabled()</u>	This action checks if the date picker functionality is disabled. This method does not accept any arguments.
<u>option(optionName)</u>	This action retrieves the value currently associated with the specified <i>optionName</i> .
<u>option()</u>	This action gets an object containing key/value pairs representing the current datepicker options hash. This method does not accept any arguments.
<u>option(____optionName, value)</u>	This action sets the value of the datepicker option associated with the specified optionName.
<u>option(options)</u>	This action sets one or more options for the datepicker.
<u>refresh()</u>	This action redraws the date picker, after having made some external modifications. This method does not accept any arguments.
<u>setDate(date)</u>	This action sets the specified date as the current date of the datepicker.
<u>show()</u>	This action opens the date picker. If the datepicker is attached to an input, the input must be visible for the datepicker to be shown. This method does not accept any arguments.
<u>widget()</u>	This action returns a jQuery object containing the datepicker.

The following examples show the use of some of the actions listed in the above table.

Use of setDate() action

Now let us see an example using the actions from the above table. The following example demonstrates the use of actions *setDate*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#datepicker-12" ).datepicker();
        $( "#datepicker-12" ).datepicker("setDate", "10w+1");
      });
    </script>
  </head>
  <body>
    <!-- HTML -->
    <p>Enter Date: <input type="text" id="datepicker-12"></p>
  </body>
</html>
```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Enter Date:

Use of show() action

The following example demonstrates the use of action *show*.

```
<!doctype html>
<html lang="en">

  <head>

    <meta charset="utf-8">
```

```

<title>jQuery UI Datepicker functionality</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<!-- Javascript -->
<script>
    $(function() {
        $( "#datepicker-13" ).datepicker();
        $( "#datepicker-13" ).datepicker("show");
    });
</script>
</head>
<body>
    <!-- HTML -->
    <p>Enter Date: <input type="text" id="datepicker-13"></p>
</body>
</html>

```

Let us save the above code in an HTML file **datepickerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Enter Date:

Event Management on datepicker elements

There are no datepicker event methods as of now!

12. JQUERYUI – DIALOG

Dialog boxes are one of the nice ways of presenting information on an HTML page. A dialog box is a floating window with a title and content area. This window can be moved, resized, and of course, closed using "X" icon by default.

jQueryUI provides **dialog()** method that transforms the HTML code written on the page into HTML code to display a dialog box.

Syntax

The **dialog()** method can be used in two forms:

- `$(selector, context).dialog (options) Method`
- `$(selector, context).dialog ("action", [params]) Method`

`$(selector, context).dialog (options) Method`

The *dialog (options)* method declares that an HTML element can be administered in the form of a dialog box. The *options* parameter is an object that specifies the appearance and behavior of that window.

Syntax

```
$(selector, context).dialog(options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).dialog({option1: value1, option2: value2..... });
```

Following table lists the different *options* that can be used with this method:

Option	Description
appendTo	If this option is set to <i>false</i> , it will prevent the ui-draggable class from being added in the list of selected DOM elements. By default its value is <i>true</i> .
autoOpen	This option unless set to <i>false</i> , the dialog box is opened upon creation. When <i>false</i> , the dialog box will be opened upon a call to <code>dialog('open')</code> . By default its value is <i>true</i> .
buttons	This option adds buttons in the dialog box. These are listed as objects, and each property is the text on the button. The value is a callback

	function called when the user clicks the button. By default its value is {}.
closeOnEscape	Unless this option set to <i>false</i> , the dialog box will be closed when the user presses the Escape key while the dialog box has focus. By default its value is true.
closeText	This option contains text to replace the default of Close for the close button. By default its value is "close".
dialogClass	If this option is set to false, it will prevent the ui-draggable class from being added in the list of selected DOM elements. By default its value is "".
draggable	Unless you this option to false, dialog box will be draggable by clicking and dragging the title bar. By default its value is true.
height	This option sets the height of the dialog box. By default its value is "auto".
hide	This option is used to set the effect to be used when the dialog box is closed. By default its value is null.
maxHeight	This option sets maximum height, in pixels, to which the dialog box can be resized. By default its value is false.
maxWidth	This option sets the maximum width to which the dialog can be resized, in pixels. By default its value is false.
minHeight	This option is the minimum height, in pixels, to which the dialog box can be resized. By default its value is 150.
minWidth	This option is the minimum width, in pixels, to which the dialog box can be resized. By default its value is 150.
modal	If this option is set to true, the dialog will have modal behavior; other items on the page will be disabled, i.e., cannot be interacted with. Modal dialogs create an overlay below the dialog but above other page elements. By default its value is false.
position	This option specifies the initial position of the dialog box. Can be one of the predefined positions: <i>center (the default), left, right, top, or bottom</i> . Can also be a 2-element array with the left and top values (in

	pixels) as [left,top], or text positions such as ['right','top']. By default its value is { my: "center", at: "center", of: window }.
resizable	This option unless set to false, the dialog box is resizable in all directions. By default its value is true.
show	This option is an effect to be used when the dialog box is being opened. By default its value is null.
title	This option specifies the text to appear in the title bar of the dialog box. By default its value is null.
width	This option specifies the width of the dialog box in pixels. By default its value is 300.

The following section will show you a few working examples of dialog functionality.

Default Functionality

The following example demonstrates a simple example of dialog functionality passing no parameters to the **dialog()** method .

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Dialog functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-widget-header,.ui-state-default, ui-button{
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFF;
        font-weight: bold;
      }
    </style>
```

```

<!-- Javascript -->
<script>
    $(function() {
        $( "#dialog-1" ).dialog({
            autoOpen: false,
        });
        $( "#opener" ).click(function() {
            $( "#dialog-1" ).dialog( "open" );
        });
    });
</script>
</head>
<body>
    <!-- HTML -->
    <div id="dialog-1" title="Dialog Title goes here...">This my first jQuery
    UI Dialog!</div>
    <button id="opener">Open Dialog</button>
</body>
</html>

```

Let us save the above code in an HTML file **dialogexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

This my first jQuery UI Dialog!

Open Dialog

Use of buttons, title and position

The following example demonstrates the usage of three options **buttons**, **title** and **position** in the dialog widget of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Dialog functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
        ui.css" rel="stylesheet">

```

```

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<!-- CSS -->
<style>
    .ui-widget-header,.ui-state-default, ui-button{
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
    }
</style>
<!-- Javascript -->
<script>
    $(function() {
        $( "#dialog-2" ).dialog({
            autoOpen: false,
            buttons: {
                OK: function() {$(this).dialog("close");}
            },
            title: "Success",
            position: {
                my: "left center",
                at: "left center"
            }
        });
        $( "#opener-2" ).click(function() {
            $( "#dialog-2" ).dialog( "open" );
        });
    });
</script>
</head>
<body>
    <!-- HTML -->
    <div id="dialog-2" title="Dialog Title goes here...">This my first jQuery
    UI Dialog!</div>
    <button id="opener-2">Open Dialog</button>

```

```
</body>
</html>
```

Let us save the above code in an HTML file **dialogexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

This my first jQuery UI Dialog!

Open Dialog

Use of hide, show and height

The following example demonstrates the usage of three options **hide**, **show** and **height** in the dialog widget of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Dialog functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-widget-header,.ui-state-default, ui-button{
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFF;
        font-weight: bold;
      }
    </style>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#dialog-3" ).dialog({
          autoOpen: false,
          hide: "puff",
```

```

        show : "slide",
        height: 200
    });
    $( "#opener-3" ).click(function() {
        $( "#dialog-3" ).dialog( "open" );
    });
});
</script>
</head>
<body>
    <!-- HTML -->
    <div id="dialog-3" title="Dialog Title goes here...">This my first jQuery
    UI Dialog!</div>
    <button id="opener-3">Open Dialog</button>
</body>
</html>

```

Let us save the above code in an HTML file **dialogexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

This my first jQuery UI Dialog!

Open Dialog

Use of Modal

The following example demonstrates the usage of three options **buttons**, **title** and **position** in the dialog widget of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Dialog functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
        ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <!-- CSS -->

```

```

<style>
    .ui-widget-header,.ui-state-default, ui-button{
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
    }
</style>
<!-- Javascript -->
<script>
    $(function() {
        $( "#dialog-4" ).dialog({
            autoOpen: false,
            modal: true,
            buttons: {
                OK: function() {$(this).dialog("close");}
            },
        });
        $( "#opener-4" ).click(function() {
            $( "#dialog-4" ).dialog( "open" );
        });
    });
</script>
</head>
<body>
    <!-- HTML -->
    <div id="dialog-4" title="Dialog Title goes here...">This my first jQuery
    UI Dialog!</div>
    <button id="opener-4">Open Dialog</button>
    <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
        eiusmod tempor incididunt
        ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
        nostrud exercitation ullamco
        laboris nisi ut aliquip ex ea commodo consequat.    </p>
    <label for="textbox">Enter Name: </label>
    <input type="text">

```



```
</body>
</html>
```

Let us save the above code in an HTML file **dialogexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

This my first jQuery UI Dialog!

Open Dialog

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Enter Name:

`$(selector, context).dialog ("action", [params])` Method

The *dialog* (*action*, *params*) method can perform an action on the dialog box, such as closing the box. The **action** is specified as a string in the first argument and optionally, one or more **params** can be provided based on the given action.

Basically, here actions are nothing but they are jQuery methods which we can use in the form of string.

Syntax

```
$(selector, context).dialog ("action", [params]);
```

The following table lists the actions for this method:

Action	Description
close()	This action closes the dialog box. This method does not accept any arguments.
destroy()	This action removes the dialog box completely. This will return the element back to its pre-init state. This method does not accept any arguments.
isOpen()	This action checks if the dialog box is open. This method does not accept any arguments.
moveToTop()	This action positions the corresponding dialog boxes to the foreground (on top of the others). This method does not accept any arguments.

open()	This action opens the dialog box. This method does not accept any arguments.
option(optionName)	This action gets the value currently associated with the specified optionName. Where <i>optionName</i> is the name of the option to get.
option()	This action gets an object containing key/value pairs representing the current dialog options hash. This method does not accept any arguments.
option(optionName, value)	This action sets the value of the dialog option associated with the specified optionName.
option(options)	This action sets one or more options for the dialog.
widget()	This action returns the dialog box's widget element; the element annotated with the ui-dialog class name. This method does not accept any arguments.

Example

Now let us see an example using the actions from the above table. The following example demonstrates the use of *isOpen()*, *open()* and *close()* methods.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Dialog functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-widget-header,.ui-state-default, ui-button{
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFF;
        font-weight: bold;
```

```

    }
</style>
<!-- Javascript -->
<script type="text/javascript">
    $(function(){
        $("#toggle").click(function() {

            ($("#dialog-5").dialog("isOpen") == false) ? ($("#dialog-5").dialog("open") : ($("#dialog-5").dialog("close") ;

        });
        ($("#dialog-5").dialog({autoOpen: false});
    });
</script>
</head>
<body>
    <button id="toggle">Toggle dialog!</button>
    <div id="dialog-5" title="Dialog Title!">
        Click on the Toggle button to open and cose this dialog box.
    </div>
</body>
</html>

```

Let us save the above code in an HTML file **dialogexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Toggle dialog!

Click on the Toggle button to open and cose this dialog box.

Event Management on Dialog Box

In addition to the dialog (options) method which we saw in the previous sections, JQueryUI provides event methods as which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
beforeClose(event, ui)	This event is triggered when the dialog box is about to close. Returning <i>false</i> prevents the dialog box from closing. It is

	handy for dialog boxes with forms that fail validation. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
close(event, ui)	This event is triggered when the dialog box is closed. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
create(event, ui)	This event is triggered when the dialog box is created. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
drag(event, ui)	This event is triggered repeatedly as a dialog box is moved about during a drag. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
dragStart(event, ui)	This event is triggered when a repositioning of the dialog box commences by dragging its title bar. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
dragStop(event, ui)	This event is triggered when a drag operation terminates. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
focus(event, ui)	This event is triggered when the dialog gains focus. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
open(event, ui)	This event is triggered when the dialog box is opened. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
resize(event, ui)	This event is triggered repeatedly as a dialog box is resized. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
resizeStart(event, ui)	This event is triggered when a resize of the dialog box commences. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
resizeStop(event, ui)	This event is triggered when a resize of the dialog box terminates. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .

The following examples demonstrate the use of the events listed in the above table.

Use of beforeClose Event method

The following example demonstrates the use of **beforeClose** event method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Dialog functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-widget-header,.ui-state-default, ui-button{
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
      }
      .invalid { color: red; }
      textarea {
        display: inline-block;
        width: 100%;
        margin-bottom: 10px;
      }
    </style>
    <!-- Javascript -->
    <script type="text/javascript">
      $(function(){
        $( "#dialog-6" ).dialog({
          autoOpen: false,
          buttons: {
            OK: function() {
              $( this ).dialog( "close" );
            }
          },
        },
```

```

        beforeClose: function( event, ui ) {
            if ( !$( "#terms" ).prop( "checked" ) ) {
                event.preventDefault();
                $( "[for=terms]" ).addClass( "invalid" );
            }
        },
        width: 600
    });
    $( "#opener-5" ).click(function() {
        $( "#dialog-6" ).dialog( "open" );
    });
});
</script>
</head>
<body>
    <div id="dialog-6">
        <p>You must accept these terms before continuing.</p>
        <textarea>This Agreement and the Request constitute the entire
agreement of the
        parties with respect to the subject matter of the Request. This
Agreement shall be
        governed by and construed in accordance with the laws of the State,
without giving
        effect to principles of conflicts of law.</textarea>
        <div>
            <label for="terms">I accept the terms</label>
            <input type="checkbox" id="terms">
        </div>
    </div>
    <button id="opener-5">Open Dialog</button>
</body>
</html>

```

Let us save the above code in an HTML file **dialogexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

You must accept these terms before continuing.



I accept the terms

Open Dialog

Use of resize Event method

The following example demonstrates the use of **resize** event method.

[illegible]

```

        }
    });
    $( "#opener-6" ).click(function() {
        $( "#dialog-7" ).dialog( "open" );
    });
});
</script>
</head>
<body>
    <div id="dialog-7" title="Dialog Title goes here...">Resize this dialog
    to see the dialog co-ordinates in the title!</div>
    <button id="opener-6">Open Dialog</button>
</body>
</html>

```

Let us save the above code in an HTML file **dialogexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Open Dialog
Resize this dialog to see the dialog co-ordinates in the title!

Extension Points

The dialog widget is built with the widget factory and can be extended. To extend widgets, we can either override or add to the behavior of existing methods. Following method provides as extension point with the same API stability as the dialog methods.

Method	Description
<code>_allowInteraction(event)</code>	This method allows the user to interact with a given target element by whitelisting elements that are not children of the dialog but allow the users to be able to use. Where <i>event</i> is of type <i>Event</i> .

The jQuery UI Widget Factory is an extensible base on which all of jQuery UI's widgets are built. Using the widget factory to build a plugin provides conveniences for state management, as well as conventions for common tasks like exposing plugin methods and changing options after instantiation.

13. JQUERYUI – MENU

A menu widget usually consists of a main menu bar with pop up menus. Items in pop up menus often have sub pop up menus. A menu can be created using the markup elements as long as the parent-child relation is maintained (using `` or ``). Each menu item has an anchor element.

The Menu Widget in jQueryUI can be used for inline and popup menus, or as a base for building more complex menu systems. For example, you can create nested menus with custom positioning.

jQueryUI provides `menu()` methods to create a menu.

Syntax

The **`menu()`** method can be used in two forms

- `$(selector, context).menu (options) Method`
- `$(selector, context).menu ("action", params) Method`

`$(selector, context).menu (options) Method`

The *menu (options)* method declares that an HTML element and its contents should be treated and managed as menus. The *options* parameter is an object that specifies the appearance and behavior of the menu items involved.

Syntax

```
$(selector, context).menu (options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).menu({option1: value1, option2: value2..... });
```

The following table lists the different *options* that can be used with this method:

Option	Description
disabled	This option if set to <i>true</i> disables the menu. By default its value is false .
icons	This option sets the icons for submenus. By default its value is { submenu: "ui-icon-carat-1-e" } .

menus	This option is a selector for the elements that serve as the menu container, including sub-menus. By default its value is ul .
position	This option sets the position of submenus in relation to the associated parent menu item. By default its value is { my: "left top", at: "right top" } .
role	This option is used to customize the ARIA roles used for the menu and menu items. By default its value is menu .

Default Functionality

The following example demonstrates a simple example of menu widget functionality, passing no parameters to the **menu()** method .

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Menu functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-menu {
        width: 200px;
      }
    </style>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#menu-1" ).menu();
      });
    </script>
  </head>
```

```

<body>
  <!-- HTML -->
  <ul id="menu-1">
    <li><a href="#">Spring</a></li>
    <li><a href="#">Hibernate</a></li>
    <li><a href="#">Java</a>
      <ul>
        <li><a href="#">Java IO</a></li>
        <li><a href="#">Swing</a></li>
        <li><a href="#">Jaspr Reports</a></li>
      </ul>
    </li>
    <li><a href="#">JSF</a></li>
    <li><a href="#">HTML5</a></li>
  </ul>
</body>
</html>

```

Let us save the above code in an HTML file **menuexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

- **Spring**
- **Hibernate**
- **Java**
 - **Java IO**
 - **Swing**
 - **Jaspr Reports**
- **JSF**
- **HTML5**

In the above example you can see a themeable menu with mouse and keyboard interactions for navigation.

Use of Icons And Position

The following example demonstrates the usage of two options **icons**, and **position** in the menu function of JQueryUI.

```
<!doctype html>
```

```

<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Menu functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-menu {
        width: 200px;
      }
    </style>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#menu-2" ).menu({
          icons: { submenu: "ui-icon-circle-triangle-e"},
          position: { my: "right top", at: "right-10 top+5" }
        });
      });
    </script>
  </head>
  <body>
    <!-- HTML -->
    <ul id="menu-2">
      <li><a href="#">Spring</a></li>
      <li><a href="#">Hibernate</a></li>
      <li><a href="#">Java</a>
        <ul>
          <li><a href="#">Java IO</a></li>
          <li><a href="#">Swing</a></li>
          <li><a href="#">Jaspr Reports</a></li>
        </ul>
      </li>
    </ul>
  </body>
</html>

```

```

        <li><a href="#">JSF</a></li>
        <li><a href="#">HTML5</a></li>
    </ul>
</body>
</html>

```

Let us save the above code in an HTML file **menuexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

- **Spring**
- **Hibernate**
- **Java**
 - **Java IO**
 - **Swing**
 - **Jaspr Reports**
- **JSF**
- **HTML5**

In the above example you can see we have applied an icon image for the submenu list and also changed the submenu position.

`$(selector, context).menu ("action", params) Method`

The *menu ("action", params)* method can perform an action on menu elements, such as enabling/disabling the menu. The action is specified as a string in the first argument (e.g., "disable" disables the menu). Check out the actions that can be passed, in the following table:

Syntax

```
$(selector, context).menu ("action", params);;
```

The following table lists the different *actions* that can be used with this method:

Action	Description
<code>blur([event])</code>	This action removes the focus from a menu. It triggers the menu's <i>blur</i> event by resetting any active element style. Where <i>event</i> is of type Event and represents what triggered the menu to blur.

collapse([event])	This action closes the current active sub-menu. Where <i>event</i> is of type Event and represents what triggered the menu to collapse.
collapseAll([event] [, all])	This action closes all the open submenus.
destroy()	This action removes menu functionality completely. This will return the element back to its pre-init state. This method does not accept any arguments.
disable()	This action disables the menu. This method does not accept any arguments.
enable()	This action enables the the menu. This method does not accept any arguments.
expand([event])	This action opens the sub-menu below the currently active item, if one exists. Where <i>event</i> is of type Event and represents what triggered the menu to expand.
focus([event], item)	This action activates a particular menu item, begins opening any sub-menu if present and triggers the menu's focus event. Where <i>event</i> is of type Event and represents what triggered the menu to gain focus. and <i>item</i> is a jQuery object representing the menu item to focus/activate.
isFirstItem()	This action returns a <i>boolean</i> value, which states if the current active menu item is the first menu item. This method does not accept any arguments.
isLastItem()	This action returns a <i>boolean</i> value, which states if the current active menu item is the last menu item. This method does not accept any arguments.
next([event])	This action delegates the active state to the next menu item. Where <i>event</i> is of type Event and represents what triggered the focus to move.
nextPage([event])	This action moves active state to first menu item below the bottom of a scrollable menu or the last item if not scrollable. Where <i>event</i> is of type Event and represents what triggered the focus to move.

<code>option(optionName)</code>	This action gets the value currently associated with the specified <i>optionName</i> . Where <i>optionName</i> is of type String and represents the name of the option to get.
<code>option()</code>	This action gets an object containing key/value pairs representing the current menu options hash.
<code>option(optionName, value)</code>	This action sets the value of the menu option associated with the specified <i>optionName</i> . Where <i>optionName</i> is of type String and represents name of option to set and <i>value</i> is of type <i>Object</i> and represents value to set for the option.
<code>option(options)</code>	This action sets one or more options for the menu. Where <i>options</i> is of type Object and represents a map of option-value pairs to set.
<code>previous([event])</code>	This action moves active state to previous menu item. Where <i>event</i> is of type Event and represents what triggered the focus to move.
<code>previousPage([event])</code>	This action moves active state to first menu item above the top of a scrollable menu or the first item if not scrollable. Where <i>event</i> is of type Event and represents what triggered the focus to move.
<code>refresh()</code>	This action initializes sub-menus and menu items that have not already been initialized. This method does not accept any arguments.
<code>select([event])</code>	This action selects the currently active menu item, collapses all sub-menus and triggers the menu's select event. Where <i>event</i> is of type Event and represents what triggered the selection.
<code>widget()</code>	This action returns a jQuery object containing the menu. This method does not accept any arguments.

The following examples demonstrate how to use the actions given in the above table.

Use of Disable Method

The following example demonstrates the use of *disable()* method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Menu functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-menu {
        width: 200px;
      }
    </style>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#menu-3" ).menu();
        $( "#menu-3" ).menu("disable");
      });
    </script>
  </head>
  <body>
    <!-- HTML -->
    <ul id="menu-3">
      <li><a href="#">Spring</a></li>
      <li><a href="#">Hibernate</a></li>
      <li><a href="#">Java</a>
        <ul>
          <li><a href="#">Java IO</a></li>
          <li><a href="#">Swing</a></li>
          <li><a href="#">Jaspr Reports</a></li>
        </ul>
      </li>
    </ul>
```



```

        </li>
        <li><a href="#">JSF</a></li>
        <li><a href="#">HTML5</a></li>
    </ul>
</body>
</html>

```

Let us save the above code in an HTML file **menuexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

- **Spring**
- **Hibernate**
- **Java**
 - **Java IO**
 - **Swing**
 - **Jaspr Reports**
- **JSF**
- **HTML5**

In the above example, you can see that, the menu is disabled.

Use of focus and collapseAll methods

The following example demonstrates the use of *focus()* and *collapseAll* methods.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Menu functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <!-- CSS -->
        <style>
            .ui-menu {
                width: 200px;
            }
        </style>

```

```

<!-- Javascript -->
<script>
    $(function() {
        var menu = $("#menu-4").menu();
        $( "#menu-4" ).menu("focus", null, $( "#menu-4" ).menu().find(
            ".ui-menu-item:last" ));
        $(menu).mouseleave(function () {
            menu.menu('collapseAll');
        });
    });
</script>
</head>
<body>
<!-- HTML -->
<ul id="menu-4">
    <li><a href="#">Spring</a></li>
    <li><a href="#">Hibernate</a></li>
    <li><a href="#">JSF</a></li>
    <li><a href="#">HTML5</a></li>
    <li><a href="#">Java</a>
        <ul>
            <li><a href="#">Java IO</a></li>
            <li><a href="#">Swing</a></li>
            <li><a href="#">Jaspr Reports</a></li>
        </ul>
    </li>
</ul>
</body>
</html>

```

Let us save the above code in an HTML file **menuexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

- **Spring**
- **Hibernate**
- **JSF**
- **HTML5**
- **Java**

- **Java IO**
- **Swing**
- **Jaspr Reports**

In the above example you can see the focus is on the last menu item. Now expand the submenu and when the mouse leaves the submenu, the submenu is closed.

Event Management on menu elements

In addition to the menu (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
blur(event, ui)	This event is triggered when a menu loses focus.
create(event, ui)	This event is triggered when a menu is created.
focus(event, ui)	This event is triggered when a menu gains focus or when any menu item is activated.
select(event, ui)	This event is triggered when a menu item is selected.

Example

The following example demonstrates the event method usage for menu widget functionality. This example demonstrates the use of event *create*, *blur* and *focus*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Menu functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .ui-menu {
```

```

        width: 200px;
    }
</style>
<!-- Javascript -->
<script>
    $(function() {
        $( "#menu-5" ).menu({
            create: function( event, ui ) {
                var result = $( "#result" );
                result.append( "Create event<br>" );
            },
            blur: function( event, ui ) {
                var result = $( "#result" );
                result.append( "Blur event<br>" );
            },
            focus: function( event, ui ) {
                var result = $( "#result" );
                result.append( "focus event<br>" );
            }
        });
    });
</script>
</head>
<body>
    <!-- HTML -->
    <ul id="menu-5">
        <li><a href="#">Spring</a></li>
        <li><a href="#">Hibernate</a></li>
        <li><a href="#">JSF</a></li>
        <li><a href="#">HTML5</a></li>
        <li><a href="#">Java</a>
            <ul>
                <li><a href="#">Java IO</a></li>
                <li><a href="#">Swing</a></li>
                <li><a href="#">Jaspr Reports</a></li>
            </ul>
        </li>
    </ul>

```

```
        </li>
    </ul>
    <span id="result"></span>
</body>
</html>
```

Let us save the above code in an HTML file **menuexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

- **Spring**
- **Hibernate**
- **JSF**
- **HTML5**
- **Java**
 - **Java IO**
 - **Swing**
 - **Jaspr Reports**

In the above example, we are printing the messages based on the events triggered.

14. JQUERYUI – PROGRESS BAR

Progress bars indicate the completion percentage of an operation or process. We can categorize progress bar as **determinate progress bar** and **indeterminate progress bar**.

Determinate progress bar should only be used in situations where the system can accurately update the current status. A determinate progress bar should never fill from left to right, then loop back to empty for a single process.

If the actual status cannot be calculated, an **indeterminate progress bar** should be used to provide user feedback.

jQueryUI provides an easy-to-use progress bar widget that we can use to let users know that our application is hard at work performing the requested operation. jQueryUI provides `progressbar()` method to create progress bars.

Syntax

The **`progressbar()`** method can be used in two forms:

- **`$(selector, context).progressbar (options)`** Method
- **`$(selector, context).progressbar ("action", params)`** Method

`$(selector, context).progressbar (options)` Method

The *progressbar (options)* method declares that an HTML element can be managed in the form of a progress bar. The *options* parameter is an object that specifies the appearance and behavior of progress bars.

Syntax

```
$(selector, context).progressbar (options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).progressbar({option1: value1, option2: value2..... });
```

The following table lists the different *options* that can be used with this method:

Option	Description
disabled	This option when set to <i>true</i> disables the progress bars. By default its value is <i>false</i> .

max	This option sets the maximum value for a progress bar. By default its value is 100.
value	This option specifies the initial value of the progress bar. By default its value is 0.

The following section will show you a few working examples of progressbar functionality.

Default Functionality

The following example demonstrates a simple example of progressbar functionality, passing no parameters to the **progressbar()** method .

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI ProgressBar functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
      .ui-widget-header {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
        font-weight: bold;
      }
    </style>
    <script>
      $(function() {
        $( "#progressbar-1" ).progressbar({
          value: 30
        });
      });
    </script>
  </head>
```

```

<body>
    <div id="progressbar-1"></div>
</body>
</html>

```

Let us save the above code in an HTML file **progressbarexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

This example shows the creation of progress bar using of *progressbar()* method. This is a default determinate progress bar.

Use of Max and Value

The following example demonstrates the usage of two options **values** and **max** in the progressbar function of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI ProgressBar functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
        ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <style>
            .ui-widget-header {
                background: #cedc98;
                border: 1px solid #DDDDDD;
                color: #333333;
                font-weight: bold;
            }
        </style>
        <script>
            $(function() {
                var progressbar = $( "#progressbar-2" );
                $( "#progressbar-2" ).progressbar({
                    value: 30,

                    max:300

```



```

        });
        function progress() {
        var val = progressbar.progressbar( "value" ) || 0;
        progressbar.progressbar( "value", val + 1 );
        if ( val < 99 ) {
            setTimeout( progress, 100 );
        }
        }
        setTimeout( progress, 3000 );
    });
</script>
</head>
<body>
    <div id="progressbar-2"></div>
</body>
</html>

```

Let us save the above code in an HTML file **progressbarexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Here you can see that the progress bar fills from right to left and stops when the value reaches 300.

`$(selector, context).progressbar ("action", params) Method`

The *progressbar ("action", params)* method can perform an action on progress bar, such as changing the percentage filled. The action is specified as a string in the first argument (e.g., "value" to change the percentage filled). Check out the actions that can be passed, in the following table.

Syntax

```
$(selector, context).progressbar ("action", params);;
```

The following table lists the different *actions* that can be used with this method:

Action	Description
destroy	This action removes the progress bar functionality of an element completely. The elements return to their pre-init state. This method does not accept any arguments.

destroy	This action removes the progress bar functionality of an element completely. The elements return to their pre-init state. This method does not accept any arguments.
disable	This action disables the progress bar functionality of an element. This method does not accept any arguments.
enable	This action enables the progress bar functionality. This method does not accept any arguments.
option(optionName)	This action retrieves the value currently associated with specified <i>optionName</i> . Where <i>optionName</i> is a String.
option	This action gets an object containing key/value pairs representing the current progressbar options hash. This method does not accept any arguments.
option(optionName, value)	This action sets the value of the progressbar option associated with the specified <i>optionName</i> .
option(options)	This action sets one or more options for the progress bars. The argument <i>options</i> is a map of option-value pairs to be set.
value	This action retrieves the current value of <i>options.value</i> , that is, the percentage of fill in the progress bar.
value(value)	This action specifies a new value to the percentage filled in the progress bar. The argument <i>value</i> can be a Number or Boolean.
widget	This action returns a jQuery object containing the progressbar. This method does not accept any arguments.

Example

Now let us see an example using the actions from the above table. The following example demonstrates the use of *disable()* and *option(optionName, value)* methods.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
```

```

<title>jQuery UI ProgressBar functionality</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<style>
    .ui-widget-header {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
        font-weight: bold;
    }
</style>
<script>
    $(function() {
        $( "#progressbar-3" ).progressbar({
            value: 30
        });
        $( "#progressbar-3" ).progressbar('disable');
        $( "#progressbar-4" ).progressbar({
            value: 30
        });
        var progressbar = $( "#progressbar-4" );
        $( "#progressbar-4" ).progressbar( "option", "max", 1024 );
        function progress() {
            var val = progressbar.progressbar( "value" ) || 0;
            progressbar.progressbar( "value", val + 1 );
            if ( val < 99 ) {
                setTimeout( progress, 100 );
            }
        }
        setTimeout( progress, 3000 );
    });
</script>
</head>
<body>

```

```

<h3>Disabled Progressbar</h3>
<div id="progressbar-3"></div><br>
<h3>Progressbar with max value set</h3>
<div id="progressbar-4"></div>
</body>
</html>

```

Let us save the above code in an HTML file **progressbarexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

DISABLED PROGRESS BAR

PROGRESS BAR WITH MAX VALUE SET

Event Management on Progress Bar Elements

In addition to the progressbar (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
change(event, ui)	This event is triggered whenever the value of progress bar changes. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
complete(event, ui)	This event is triggered when the progressbar reaches the maximumm value. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .
create(event, ui)	This event is triggered whenever progressbar is created. Where <i>event</i> is of type <i>Event</i> , and <i>ui</i> is of type <i>Object</i> .

Example

The following example demonstrates the event method usage during progressbar functionality. This example demonstrates the use of events *change* and *complete*.

```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI ProgressBar functionality</title>

```

```

<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<style>
    .ui-widget-header {
        background: #cedc98;
        border: 1px solid #DDDDDD;
        color: #333333;
        font-weight: bold;
    }
    .progress-label {
        position: absolute;
        left: 50%;
        top: 13px;
        font-weight: bold;
        text-shadow: 1px 1px 0 #fff;
    }
</style>
<script>
$(function() {
    var progressbar = $( "#progressbar-5" );
    progressLabel = $( ".progress-label" );
    $( "#progressbar-5" ).progressbar({
        value: false,
        change: function() {
            progressLabel.text(
                progressbar.progressbar( "value" ) + "%" );
        },
        complete: function() {
            progressLabel.text( "Loading Completed!" );
        }
    });
    function progress() {
        var val = progressbar.progressbar( "value" ) || 0;
        progressbar.progressbar( "value", val + 1 );
    }
});

```

```
        if ( val < 99 ) {  
            setTimeout( progress, 100 );  
        }  
    }  
    setTimeout( progress, 3000 );  
});  
</script>  
</head>  
<body>  
    <div id="progressbar-5">  
        <div class="progress-label">  
            Loading...  
        </div>  
    </div>  
</body>  
</html>
```

Let us save the above code in an HTML file **progressbarexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Here you can see as the progressbar changes its changed value is printed and upon complete event the "Loading Completed!" message displays.

15. JQUERYUI – SLIDER

A slider is used whenever a numeric value within a certain range is to be obtained. The advantage of a slider over text input is that it becomes impossible for the user to enter a bad value. Any value that they can pick with the slider is valid.

jQueryUI provides us a slider control through slider widget. jQueryUI provides slider() method changes the appearance of HTML elements in the page, adding new CSS classes that give them the appropriate style.

Syntax

The **slider ()** method can be used in two forms:

- **\$(selector, context).slider (options)** Method
- **\$(selector, context).slider ("action", params)** Method

\$(selector, context).slider (options) Method

The *slider (options)* method declares that an HTML element should be managed as a slider. The *options* parameter is an object that specifies the appearance and behavior of slider.

Syntax

```
$(selector, context).slider (options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).slider({option1: value1, option2: value2..... });
```

The following table lists the different *options* that can be used with this method:

Option	Description
animate	This option when set to <i>true</i> , creates an animated effect when users click directly on the axis. By default its value is false.
disabled	This option when set to <i>true</i> , disables the slider. By default its value is false.
max	This option specifies the upper value of the range that the slider can attain—the value represented when the handle is moved to the far right (for horizontal sliders) or top (for vertical sliders). By default its value is 100.

min	This option specifies the lower value of the range that the slider can attain—the value represented when the handle is moved to the far left (for horizontal sliders) or bottom (for vertical sliders). By default its value is 0.
orientation	This option indicates the horizontal or vertical orientation of the slider. By default its value is horizontal.
range	This option specifies whether the slider represents a range. By default its value is false.
step	This option specifies discrete intervals between the minimum and maximum values that the slider is allowed to represent. By default its value is 1.
value	This option specifies the initial value of a single-handle slider. If there are multiple handles (see the values options), specifies the value for the first handle. By default its value is 1.
values	This option is of type Array and causes multiple handles to be created and specifies the initial values for those handles. This option should be an array of possible values, one for each handle. By default its value is null.

The following section will show you a few working examples of slider functionality.

Default Functionality

The following example demonstrates a simple example of slider functionality, passing no parameters to the **slider()** method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Slider functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

    <!-- Javascript -->
    <script>
      $(function() {
```



```

        $( "#slider-1" ).slider();
    });
</script>
</head>
<body>
    <!-- HTML -->
    <div id="slider-1"></div>
</body>
</html>

```

Let us save the above code in an HTML file **sliderexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

In the above example, it is a basic horizontal slider and has a single handle that can be moved with the mouse or by using the arrow keys.

Use of value, animate, and orientation

The following example demonstrates the usage of three options **(a) value** **(b)animate**, and **(c) orientation** in the slider function of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Slider functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <!-- Javascript -->
        <script>
            $(function() {
                $( "#slider-2" ).slider({
                    value: 60,
                    animate:"slow",
                    orientation: "horizontal"
                });
            });
        </script>

```

```

</head>
<body>
  <!-- HTML -->
  <div id="slider-2"></div>
</body>
</html>

```

Let us save the above code in an HTML file **sliderexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

In the above example the *value* of slider i.e the initial value is set as 60, hence you see the handle at initial value of 60. Now just click directly on the axis and see the animation effect.

Use of Range, Min, Max and Values

The following example demonstrates the usage of three options **(a) range**, **(b) min**, **(c) max**, and **(d) values** in the slider function of JQueryUI.

```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Slider functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#slider-3" ).slider({
          range:true,
          min: 0,
          max: 500,
          values: [ 35, 200 ],
          slide: function( event, ui ) {
            $( "#price" ).val( "$" + ui.values[ 0 ] + " - $" + ui.values[
1 ] );
          }

```

```

    });
    $( "#price" ).val( "$" + $( "#slider-3" ).slider( "values", 0 ) +
        " - $" + $( "#slider-3" ).slider( "values", 1 ) );
    });
</script>
</head>
<body>
    <!-- HTML -->
    <p>
        <label for="price">Price range:</label>
        <input type="text" id="price"
            style="border:0; color:#b9cd6d; font-weight:bold;">
    </p>
    <div id="slider-3"></div>
</body>
</html>

```

Let us save the above code in an HTML file **sliderexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Price range:

In the above example we have set the range option to true to capture a range of values with two drag handles. The space between the handles is filled with a different background color to indicate those values are selected.

\$(selector, context).slider ("action", params) Method

The *slider ("action", params)* method allows an action on the slider, such as moving the cursor to a new location. The action is specified as a string in the first argument (e.g., "value" to indicate a new value of the cursor). Check out the actions that can be passed, in the following table.

Syntax

```
$(selector, context).slider ("action", params);;
```

The following table lists the different *actions* that can be used with this method:

Action	Description
destroy	This action destroys the slider functionality of an element completely. The elements return to their pre-init state. This method does not accept any arguments.
disable	This action disables the slider functionality. This method does not accept any arguments.
enable	This action enables the slider functionality. This method does not accept any arguments.
option(optionName)	This action retrieves the value of the specified param option. This option corresponds to one of those used with <i>slider (options)</i> . Where <i>optionName</i> is the name of the option to get.
option()	This action gets an object containing key/value pairs representing the current slider options hash.
option(optionName, value)	This action sets the value of the slider option associated with the specified <i>optionName</i> . The argument <i>optionName</i> is name of the option to be set and <i>value</i> is the value to be set for the option.
option(options)	This action sets one or more options for the slider. The argument <i>options</i> is a map of option-value pairs to be set.
value	This action retrieves the current value of <i>options.value (the slider)</i> . Use only if the slider is unique (if not, use <i>slider ("values")</i>). This signature does not accept any arguments.
value(value)	This action sets the value of the slider.
values	This action retrieves the current value of <i>options.values</i> (the value of the sliders in an array). This signature does not accept any arguments.
values(index)	This action gets the value for the specified handle. Where <i>index</i> is of type Integer and is a zero-based index of the handle.


```

    </script>
</head>
<body>
    <!-- HTML -->
    <div id="slider-4"></div>
    <p>
        <label for="minval">Minumum value:</label>
        <input type="text" id="minval"
            style="border:0; color:#b9cd6d; font-weight:bold;">
    </p>
    <div id="slider-5"></div>
</body>
</html>

```

Let us save the above code in an HTML file **sliderexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Minumum value:

In the above example, the first slider is disabled and the second slider the value is set to 50.

Event Management On Slider Elements

In addition to the slider (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
change(event, ui)	This event is triggered handle's value changes, either through user action or programmatically.
create(event, ui)	This event is triggered when the slider is created.
slide(event, ui)	This event is triggered for mouse move events whenever the handle is being dragged through the slider. Returning false cancels the slide.
start(event, ui)	This event is triggered when the user starts sliding.
stop(event, ui)	This event is triggered when a slide stops.

Example

The following example demonstrates the event method usage during slider functionality. This example demonstrates the use of events *start*, *stop*, *change* and *slide*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Slider functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#slider-6" ).slider({
          range:true,
          min: 0,
          max: 500,
          values: [ 35, 200 ],
          start: function( event, ui ) {
            $( "#startvalue" )
              .val( "$" + ui.values[ 0 ] + " - $" + ui.values[ 1 ] );
          },
          stop: function( event, ui ) {
            $( "#stopvalue" )
              .val( "$" + ui.values[ 0 ] + " - $" + ui.values[ 1 ] );
          },
          change: function( event, ui ) {
            $( "#changevalue" )
              .val( "$" + ui.values[ 0 ] + " - $" + ui.values[ 1 ] );
          },
          slide: function( event, ui ) {
            $( "#slidevalue" )
              .val( "$" + ui.values[ 0 ] + " - $" + ui.values[ 1 ] );
          }
        });
      });
    </script>
  </head>
  <body>
```

```

        });
    });
</script>
</head>
<body>
    <!-- HTML -->
    <div id="slider-6"></div>
    <p>
        <label for="startvalue">Start:</label>
        <input type="text" id="startvalue"
            style="border:0; color:#b9cd6d; font-weight:bold;">
    </p>
    <p>
        <label for="stopvalue">Stop:</label>
        <input type="text" id="stopvalue"
            style="border:0; color:#b9cd6d; font-weight:bold;">
    </p>
    <p>
        <label for="changevalue">Change:</label>
        <input type="text" id="changevalue"
            style="border:0; color:#b9cd6d; font-weight:bold;">
    </p>
    <p>
        <label for="slidevalue">Slide:</label>
        <input type="text" id="slidevalue"
            style="border:0; color:#b9cd6d; font-weight:bold;">
    </p>
</body>
</html>

```

Let us save the above code in an HTML file **sliderexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Start:

Stop:

Change:

Slide:

16. JQUERYUI – SPINNER

Spinner widget adds a up/down arrow to the left of a input box thus allowing a user to increment/decrement a value in the input box. It allows users to type a value directly, or modify an existing value by spinning with the keyboard, mouse or scrollwheel. It also has a step feature to skip values. In addition to the basic numeric features, it also enables globalized formatting options (ie currency, thousand separator, decimals, etc.) thus providing a convenient internationalized masked entry box.

The following example depends on *Globalize*. You can get the Globalize files from <https://github.com/jquery/globalize>. Click the *releases* link, select the version you want, and download the *.zip* or *tar.gz* file. Extract the files and copy the following files to the folder where your example is located.

- `lib/globalize.js` : This file contains the Javascript code for dealing with localizations
- `lib/globalize.culture.js` : This file contains a complete set of the locales that the Globalize library comes with.

These files are also present in the *external* folder of your jquery-ui library.

jQueryUI provides `spinner()` method which creates a spinner.

Syntax

The **`spinner()`** method can be used in two forms:

- `$(selector, context).spinner (options) Method`
- `$(selector, context).spinner ("action", params) Method`

`$(selector, context).spinner (options) Method`

The *spinner (options)* method declares that an HTML element and its contents should be treated and managed as spinner. The *options* parameter is an object that specifies the appearance and behavior of the spinner elements involved.

Syntax

```
$(selector, context).spinner (options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).spinner({option1: value1, option2: value2..... });
```

The following table lists the different *options* that can be used with this method:

Option	Description
<u>culture</u>	This option sets the culture to use for parsing and formatting the value. By default its value is null which means the currently set culture in Globalize is used.
<u>disabled</u>	This option if set to <i>true</i> disables spinner. By default its value is false.
<u>icons</u>	This option sets icons to use for buttons, matching an icon provided by the jQuery UI CSS Framework . By default its value is { down: "ui-icon-triangle-1-s", up: "ui-icon-triangle-1-n" } .
<u>incremental</u>	This option controls the number of steps taken when holding down a spin button. By default its value is true.
<u>max</u>	This option indicates the maximum allowed value. By default its value is null which means there is no maximum enforced.
<u>min</u>	This option indicates the minimum allowed value. By default its value is null which means there is no minimum enforced.
<u>numberFormat</u>	This option indicates format of numbers passed to <i>Globalize</i> , if available. Most common are "n" for a decimal number and "C" for a currency value. By default its value is null.
<u>page</u>	This option indicates the number of steps to take when paging via the pageUp/pageDown methods. By default its value is 10.
<u>step</u>	This option indicates size of the step to take when spinning via buttons or via the <i>stepUp()/stepDown()</i> methods. The element's <i>step</i> attribute is used if it exists and the option is not explicitly set.

The following section will show you a few working examples of spinner widget functionality.

Default Functionality

The following example demonstrates a simple example of spinner widget functionality, passing no parameters to the **spinner()** method .

```
<!doctype html>
<html lang="en">

<head>

<meta charset="utf-8">
```

```

<title>jQuery UI Spinner functionality</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<!-- CSS -->
<style type="text/css">
    #spinner-1 input {width: 100px}
</style>
<!-- Javascript -->
<script>
    $(function() {
        $( "#spinner-1" ).spinner();
    });
</script>
</head>
<body>
    <!-- HTML -->
    <div id="example">
        <input type="text" id="spinner-1" value="0" />
    </div>
</body>
</html>

```

Let's save the above code in an HTML file **spinnerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Use of Min, Max, and Step Options

The following example demonstrates the usage of three options **min**, **max** and **step** in the spinner widget of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>

        <meta charset="utf-8">

        <title>jQuery UI Spinner functionality</title>

```

```

    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">

    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style type="text/css">
        #spinner-2,#spinner-3 input {width: 100px}
    </style>
    <!-- Javascript -->
    <script>
        $(function() {
            $( "#spinner-2" ).spinner({
                min: -10,
                max: 10
            });
            $( '#spinner-3' ).spinner({
                step: 100,
                min: -1000000,
                max: 1000000
            });
        });
    </script>
</head>
<body>
    <!-- HTML -->
    <div id="example">
        Spinner Min, Max value set:
        <input type="text" id="spinner-2" value="0" /><br><br>
        Spinner increments/decrements in step of of 100:
        <input type="text" id="spinner-3" value="0" />
    </div>
</body>
</html>

```

Let us save the above code in an HTML file **spinnerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Spinner Min, Max value set:

Spinner increments/decrements in step of 100:

In the above example, you can see in the first spinner the max and min values are set to 10 and -10 respectively. Hence crossing these values, the spinner will stop incrementing/decrementing. In the second spinner the spinner value increments/decrements in steps of 100.

Use of icons Option

The following example demonstrates the usage of option **icons** in the spinner widget of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Spinner functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style type="text/css">
      #spinner-5 input {width: 100px}
    </style>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#spinner-5" ).spinner({
          icons: {
            down: "custom-down-icon", up: "custom-up-icon"
          }
        });
      });
    </script>

  </head>
```

```
<body>
  <!-- HTML -->
  <div id="example">
    <input type="text" id="spinner-5" value="0" />
  </div>
</body>
</html>
```

Let's save the above code in an HTML file **spinnerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

In the above example, you can notice the images spinner are changed.

Use of culture, numberFormat, and page options

The following example demonstrates the usage of three options **culture**, **numberFormat** and **page** in the spinner widget of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Spinner functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script src="/jqueryui/jquery-ui-
1.10.4/external/jquery.mousewheel.js"></script>
    <script src="/jqueryui/jquery-ui-1.10.4/external/globalize.js"></script>
    <script src="/jqueryui/jquery-ui-1.10.4/external/globalize.culture.de-
DE.js"></script>
    <script>
      $(function() {
        $( "#spinner-4" ).spinner({
          culture:"de-DE",
          numberFormat:"C",
          step:2,
```

```

        page:10
    });
});
</script>
</head>
<body>
    <p>
        <label for="spinner-4">Amount to donate:</label>
        <input id="spinner-4" name="spinner" value="5">
    </p>

</body>
</html>

```

Let's save the above code in an HTML file **spinnerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Amount to donate:

In the above example you can see the spinner displays the number in currency format as the *numberFormat* is set to "C" and *culture* is set to "de-DE". Here we have used the Globalize files from the jquery-ui library.

\$(selector, context).spinner ("action", params) Method

The *spinner ("action", params)* method can perform an action on spinner elements, such as enabling/disabling the spinner. The action is specified as a string in the first argument (e.g., "disable" disables the spinner). Check out the actions that can be passed, in the following table.

Syntax

```
$(selector, context).spinner ("action", params);;
```

The following table lists the different *actions* that can be used with this method:

Action	Description
<u>destroy</u>	This action destroys the spinner functionality of an element completely. The elements return to their pre-init state. This method does not accept any arguments.
<u>disable</u>	This action disables the spinner functionality. This method does not accept any arguments.

<u>enable</u>	This action enables the spinner functionality. This method does not accept any arguments.
<u>option(optionName)</u>	This action gets the value currently associated with the specified <i>optionName</i> . Where <i>optionName</i> is the name of the option to get.
<u>option</u>	This action gets an object containing key/value pairs representing the current spinner options hash. This method does not accept any arguments.
<u>option(optionName, value)</u>	This action sets the value of the spinner option associated with the specified <i>optionName</i> .
<u>option(options)</u>	This action sets one or more options for the spinner.
<u>pageDown([pages])</u>	This action decrements the value by the specified number of pages, as defined by the page option.
<u>pageUp([pages])</u>	This action increments the value by the specified number of pages, as defined by the page option.
<u>stepDown([steps])</u>	This action decrements the value by the specified number of steps.
<u>stepUp([steps])</u>	This action increments the value by the specified number of steps.
<u>value</u>	This action gets the current value as a number. The value is parsed based on the numberFormat and culture options. This method does not accept any arguments.
<u>value(value)</u>	This action sets the value. If value is passed value is parsed based on the numberFormat and culture options.
<u>widget</u>	This action returns the spinner widget element; the one annotated with the <i>ui-spinner</i> class name.

The following examples demonstrate how to use the actions given in the above table.

Use of action stepUp, stepDown, pageUp, and pageDown

The following example demonstrates the use of *stepUp*, *stepDown*, *pageUp* and *pageDown* methods.

```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Spinner functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style type="text/css">
      #spinner-6 input {width: 100px}
    </style>
    <!-- Javascript -->
    <script>
      $(function() {
        $("#spinner-6").spinner();
        $('button').button();

        $('#stepUp-2').click(function () {
          $("#spinner-6").spinner("stepUp");
        });

        $('#stepDown-2').click(function () {
          $("#spinner-6").spinner("stepDown");
        });

        $('#pageUp-2').click(function () {
          $("#spinner-6").spinner("pageUp");
        });

        $('#pageDown-2').click(function () {
          $("#spinner-6").spinner("pageDown");
        });
      });
    </script>

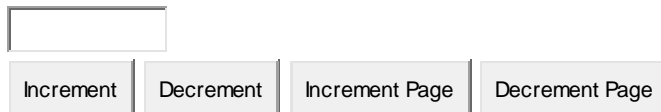
```

```

</head>
<body>
  <!-- HTML -->
  <input id="spinner-6" />
  <br/>
  <button id="stepUp-2">Increment</button>
  <button id="stepDown-2">Decrement</button>
  <button id="pageUp-2">Increment Page</button>
  <button id="pageDown-2">Decrement Page</button>
</body>
</html>

```

Let us save the above code in an HTML file **spinnerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:



In the above example, use the respective buttons to increment/decrement the spinner.

Use of action enable, and disable

The following example demonstrates the use of *enable* and *disable* methods.

```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Spinner functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style type="text/css">

      #spinner-7 input {width: 100px}

    </style>
    <!-- Javascript -->
    <script>

```

```

$(function() {
    $("#spinner-7").spinner();
    $('button').button();
    $('#stepUp-3').click(function () {
        $("#spinner-7").spinner("enable");
    });
    $('#stepDown-3').click(function () {
        $("#spinner-7").spinner("disable");
    });
});
</script>
</head>
<body>
    <!-- HTML -->
    <input id="spinner-7" />
    <br/>
    <button id="stepUp-3">Enable</button>
    <button id="stepDown-3">Disable</button>
</body>
</html>

```

Let us save the above code in an HTML file **spinnerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Enable

Disable

In the above example, use the Enable/Disable buttons to enable or disable the spinner.

Event Management on Spinner Elements

In addition to the spinner (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
<u>change(event, ui)</u>	This event is triggered when the value of the spinner has changed and the input is no longer focused.
<u>create(event, ui)</u>	This event is triggered when the spinner is created.

<u>spin(event, ui)</u>	This event is triggered during increment/decrement.
<u>start(event, ui)</u>	This event is triggered before a spin. Can be canceled, preventing the spin from occurring.
<u>stop(event, ui)</u>	This event is triggered after a spin.

Example

The following example demonstrates the event method usage in spinner widgets. This example demonstrates the use of events *spin*, *change* and *stop*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Spinner functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style type="text/css">
      #spinner-8 input {width: 100px}
    </style>
    <!-- Javascript -->
    <script>
      $(function() {
        $( "#spinner-8" ).spinner({
          spin: function( event, ui ) {
            var result = $( "#result-2" );
            result.append( "Spin Value: "+$( "#spinner-8"
).spinner("value") );
          },

          change: function( event, ui ) {
            var result = $( "#result-2" );
            result.append( "Change value: "+$( "#spinner-8"
).spinner("value") );
          },
```

```

        stop: function( event, ui ) {
            var result = $( "#result-2" );
            result.append( "Stop value: "+$( "#spinner-8"
).spinner("value") );
        }
    });
});
</script>
</head>
<body>
    <!-- HTML -->
    <div id="example">
        <input type="text" id="spinner-8" value="0" />
    </div>
    <span id="result-2"></span>
</body>
</html>

```

Let's save the above code in an HTML file **spinnerexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

In the above example change the value of the spinner and see, the messages being displayed below for spin and stop events. Now change the focus of the spinner and you see a message being displayed on change event.

Extension Points

The spinner widget is built with the widget factory and can be extended. To extend widgets, we can either override or add to the behavior of existing methods. Following method provides as extension point with the same API stability as the spinner methods.

Method	Description
<u>_buttonHtml(event)</u>	This method return a String which is an HTML. This HTML can be used for the spinner's increment and decrement buttons. Each button must be given a ui-spinner-button class name for the associated events to work. This method does not accept any arguments.

<u>_uiSpinnerHtml(event)</u>	This method determine the HTML to use to wrap the spinner's <input> element. This method does not accept any arguments.
------------------------------	---

The jQuery UI Widget Factory is an extensible base on which all of jQuery UI's widgets are built. Using the widget factory to build a plugin provides conveniences for state management, as well as conventions for common tasks like exposing plugin methods and changing options after instantiation.

17. JQUERYUI – TABS

Tabs are sets of logically grouped content that allow us to quickly flip between them to. Tabs allow us to save space like accordions. For tabs to work properly following set of markups needs to use:

- Tabs must be in a list either ordered() or unordered().
- Each tab title must be within each and wrapped by anchor (<a>) tag with *anhref* attribute.
- Each tab panel may be any valid element but it must have an *id* , which corresponds to the hash in the anchor of the associated tab.

jQueryUI provides us tabs () method drastically changes the appearance of HTML elements inside the page. This method traverses (internally in jQuery UI) HTML code and adds new CSS classes to the elements concerned (here, the tabs) to give them the appropriate style.

Syntax

The **tabs ()** method can be used in two forms:

- \$(selector, context).tabs (options) Method
- \$(selector, context).tabs ("action", params) Method

\$ (selector, context).tabs (options) Method

The *tabs (options)* method declares that an HTML element and its content should be managed as tabs. The *options* parameter is an object that specifies the appearance and behavior of tabs.

Syntax

```
$(selector, context).tabs (options);
```

You can provide one or more options at a time using JavaScript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).tabs({option1: value1, option2: value2..... });
```

The following table lists the different *options* that can be used with this method:

Option	Description
<u>active</u>	This option specifies the current active tab/panel. By default its value is 0. This can be of type:

	<p>Boolean: When set to false, will collapse all the panels. This requires the collapsible option to be true.</p> <p>Integer:</p> <p>The zero-based index of the panel that is active (open). A negative value selects panels going backward from the last panel.</p> <p>Syntax</p> <pre>\$(".selector").tabs ({ active: 1 });</pre>
<u>collapsible</u>	<p>This option set to <i>true</i>, it allows tabs to be deselected. When set to false (the default), clicking on a selected tab does not deselect (it remains selected). By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").tabs ({ collapsible: true });</pre>
<u>disabled</u>	<p>This option uses an array to indicate index tabs that are disabled (and therefore cannot be selected). For example, use [0, 1] to disable the first two tabs. By default its value is false.</p> <p>This can be of type:</p> <p>Boolean: Enable or disable all tabs.</p> <p>Array:</p> <p>An array containing the zero-based indexes of the tabs that should be disabled, e.g., [0, 2] would disable the first and third tab.</p>

	<p>Syntax</p> <pre>\$(".selector").tabs ({ disabled: [0, 2] });</pre>
<u>event</u>	<p>This option is a name of the event that lets users select a new tab. If, for example, this option is set to "mouseover", passing the mouse over a tab will select it. By default its value is "click".</p> <p>Syntax</p> <pre>\$(".selector").tabs ({ event: "mouseover" });</pre>
<u>heightStyle</u>	<p>This option controls the height of the tabs widget and each panel. By default its value is "content".</p> <p>Possible values are:</p> <p>auto: All panels will be set to the height of the tallest panel.</p> <p>fill: Expand to the available height based on the tabs' parent height.</p> <p>content: Each panel will be only as tall as its content.</p> <p>Syntax</p> <pre>\$(".selector").tabs ({ heightStyle: "fill" });</pre>
<u>hide</u>	<p>This option specifies how to animate hiding of panel. By default its value is null.</p>

	<p>This can be of type:</p> <p>Boolean: When set to false, no animation will be used and the panel will be hidden immediately.</p> <p>Number:</p> <p>The panel will fade out with the specified duration and the default easing.</p> <p>String:</p> <p>The panel will be hidden using the specified effect. Value can be slideUp or fold</p> <p>Object:</p> <p>For this type, properties effect, delay, duration and easing may be provided.</p> <p>Syntax</p> <pre>\$(".selector").tabs ({ { hide: { effect: "explode", duration: 1000 } } });</pre>
<u>show</u>	<p>This option specifies how to animate showing of panel. By default its value is null.</p> <p>This can be of type:</p> <p>Boolean: When set to false, no animation will be used and the panel will be shown immediately.</p> <p>Number:</p> <p>The panel will fade out with the specified duration and the default easing.</p> <p>String:</p>

The panel will be shown using the specified effect. Value can be slideUp or fold.

Object:

For this type, properties effect, delay, duration and easing may be provided.

Syntax

```
$( ".selector" ).tabs (
    { show: { effect: "blind", duration: 800 } }
);
```

The following section will show you a few working examples of tabs functionality.

Default Functionality

The following example demonstrates a simple example of tabs functionality, passing no parameters to the **tabs()** method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Tabs functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script>
      $(function() {
        $( "#tabs-1" ).tabs();
      });
    </script>
    <style>
      #tabs-1{font-size: 14px;}
      .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
```

```

        color: #FFFFFF;
        font-weight: bold;
    }
</style>
</head>
<body>
    <div id="tabs-1">
        <ul>
            <li><a href="#tabs-2">Tab 1</a></li>
            <li><a href="#tabs-3">Tab 2</a></li>
            <li><a href="#tabs-4">Tab 3</a></li>
        </ul>
        <div id="tabs-2">
            <p>Neque porro quisquam est qui dolorem ipsum quia dolor sit
            amet, consectetur, adipisci velit... </p>
        </div>
        <div id="tabs-3">
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
            sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
            Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
            nisi ut aliquip ex ea commodo consequat. </p>
        </div>
        <div id="tabs-4">
            <p>ed ut perspiciatis unde omnis iste natus error sit
            voluptatem accusantium doloremque laudantium, totam rem aperiam,
            eaque ipsa quae ab illo inventore veritatis et quasi architecto
            beatae vitae dicta sunt explicabo. </p>
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
            sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
            Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
            nisi ut aliquip ex ea commodo consequat. </p>
        </div>
    </div>
</body>
</html>

```

Let's save the above code in an HTML file **tabsexample.htm** and open it in a standard browser which supports javascript, you should see the following output. Now, you can play with the result:

- **Tab 1**
- **Tab 2**
- **Tab 3**

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

ed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

In the above example, click on tabs to swap between content.

Use of heightStyle, collapsible, and hide

The following example demonstrates the usage of three options **(a) heightStyle** **(b) collapsible**, and **(c) hide** in the tabs function of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Tabs functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

    <script>

      $(function() {
        $( "#tabs-5" ).tabs({
          heightStyle:"fill",
          collapsible:true,
          hide:"slideUp"
        });
      });
```

```

    });
</script>
<style>
    #tabs-5{font-size: 14px;}
    .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
    }
</style>
</head>
<body>
    <div id="tabs-5">
        <ul>
            <li><a href="#tabs-6">Tab 1</a></li>
            <li><a href="#tabs-7">Tab 2</a></li>
            <li><a href="#tabs-8">Tab 3</a></li>
        </ul>
        <div id="tabs-6">
            <p>Neque porro quisquam est qui dolorem ipsum quia dolor
            sit amet, consectetur, adipisci velit... </p>
        </div>
        <div id="tabs-7">
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
            sed do eiusmod tempor incididunt ut labore et dolore magna
            aliqua. Ut enim ad minim veniam, quis nostrud exercitation
            ullamco laboris nisi ut aliquip ex ea commodo consequat. </p>
        </div>

        <div id="tabs-8">
            <p>ed ut perspiciatis unde omnis iste natus error sit voluptatem
            accusantium doloremque laudantium, totam rem aperiam, eaque ipsa
            quae
            ab illo inventore veritatis et quasi architecto beatae vitae dicta
            sunt explicabo. </p>
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,

```

```

        sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
        Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
        nisi ut aliquip ex ea commodo consequat. </p>
    </div>
</div>
</body>
</html>

```

Let's save the above code in an HTML file **tabsexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

- **Tab 1**
- **Tab 2**
- **Tab 3**

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

ed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Click the selected tab to toggle its content closed/open. You can also see the animation effect "slideUp" when the tab is closed.

Use of Event

The following example demonstrates the usage of three options **event** in the tabs function of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Tabs functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
        ui.css" rel="stylesheet">

```



```

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<script>
    $(function() {
        $( "#tabs-9" ).tabs({
            event:"mouseover"
        });
    });
</script>
<style>
    #tabs-9{font-size: 14px;}
    .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
    }
</style>
</head>
<body>
    <div id="tabs-9">
        <ul>
            <li><a href="#tabs-10">Tab 1</a></li>
            <li><a href="#tabs-11">Tab 2</a></li>
            <li><a href="#tabs-12">Tab 3</a></li>
        </ul>
        <div id="tabs-10">
            <p>Neque porro quisquam est qui dolorem ipsum quia dolor
            sit amet, consectetur, adipisci velit... </p>
        </div>
        <div id="tabs-11">
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
            sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
            Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
            nisi ut aliquip ex ea commodo consequat. </p>
        </div>
    </div>

```

```

<div id="tabs-12">
  <p>ed ut perspiciatis unde omnis iste natus error sit
  voluptatem accusantium doloremque laudantium, totam rem aperiam,
  eaque ipsa quae ab illo inventore veritatis et quasi architecto
  beatae vitae dicta sunt explicabo. </p>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
  sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
  Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
  nisi ut aliquip ex ea commodo consequat. </p>
</div>
</div>
</body>
</html>

```

Let's save the above code in an HTML file **tabsexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

- **Tab 1**
- **Tab 2**
- **Tab 3**

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

ed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

In the above example toggle the sections open/closed with mouseover the tabs.

`$(selector, context).tabs ("action", params)` Method

The *tabs* ("action", params) method allows an action on the tabs (through a JavaScript program), selecting, disabling, adding, or removing a tab. The action is specified as a string in the first argument (e.g., "add" to add a new tab). Check out the actions that can be passed, in the following table.

Syntax

```
$(selector, context).tabs ("action", params);;
```

Following table lists the different *actions* that can be used with this method:

Action	Description
<u>destroy</u>	<p>This action destroys the tabs functionality of an element completely. The elements return to their pre-init state. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").tabs("destroy");</pre>
<u>disable</u>	<p>This action disables all tabs. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").tabs("disable");</pre>
<u>disable(index)</u>	<p>This action disables the specified tab. Where <i>index</i> is the tab to be disabled.</p> <p>Syntax</p> <pre>\$(".selector").tabs("disable", 1);</pre>
<u>enable</u>	<p>This action activates all the tabs. This signature does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").tabs("enable");</pre>
<u>enable(index)</u>	<p>This action activates a specified tab. Where <i>index</i> is the tab to be enabled.</p> <p>Syntax</p> <pre>\$(".selector").tabs("enable", 1);</pre>
<u>load(index)</u>	<p>This action forces a reload of the indexed tab, ignoring the cache. Where <i>index</i> is the tab to load.</p> <p>Syntax</p> <pre>\$(".selector").tabs("load", 1);</pre>
<u>option(optionName)</u>	<p>This action gets the value currently associated with the specified <i>optionName</i>.</p>

	<p>Syntax</p> <pre>var isDisabled = \$(".selector").tabs("option", "disabled");</pre>
<u>option</u>	<p>This action gets an object containing key/value pairs representing the current tabs options hash.</p> <p>Syntax</p> <pre>\$(".selector").tabs("option");</pre>
<u>option(optionName, value)</u>	<p>This action sets the value of the tabs option associated with the specified <i>optionName</i>. The argument <i>optionName</i> is name of the option to be set and <i>value</i> is the value to be set for the option.</p> <p>Syntax</p> <pre>var isDisabled = \$(".selector").tabs("option", "disabled");</pre>
<u>option(options)</u>	<p>This action sets one or more options to the tabs.</p> <p>Syntax</p> <pre>\$(".selector").tabs("option", { disabled: true });</pre>
<u>refresh</u>	<p>This action recomputes the height of the tab panels when any tabs that were added or removed directly in the DOM. Results depend on the content and the <i>heightStyle</i> option.</p> <p>Syntax</p> <pre>\$(".selector").tabs("refresh");</pre>
<u>widget</u>	<p>This action returns the element serving as the tabs widget, annotated with the <i>ui-tabs</i> class name.</p> <p>Syntax</p> <pre>var widget = \$(".selector").tabs("widget");</pre>

Use of Action Disable()

Now let us see an example using the actions from the above table. The following example demonstrates the use of *disable()* method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Tabs functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-
lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-
ui.js"></script>
    <script>
      $(function() {
        $( "#tabs-13" ).tabs();
        $( "#tabs-13" ).tabs("disable");
      });
    </script>
    <style>
      #tabs-13{font-size: 14px;}
      .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;
        color: #FFFFFFF;
        font-weight: bold;
      }
    </style>
  </head>

  <body>

    <div id="tabs-13">
      <ul>
        <li><a href="#tabs-14">Tab 1</a></li>
        <li><a href="#tabs-15">Tab 2</a></li>
```

```

        <li><a href="#tabs-16">Tab 3</a></li>
    </ul>
    <div id="tabs-14">
        <p>Neque porro quisquam est qui dolorem ipsum quia dolor
        sit amet, consectetur, adipisci velit... </p>
    </div>
    <div id="tabs-15">
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
        sed do eiusmod tempor incididunt ut labore et dolore magna
        aliqua.
        Ut enim ad minim veniam, quis nostrud exercitation ullamco
        laboris nisi ut aliquip ex ea commodo consequat. </p>
    </div>
    <div id="tabs-16">
        <p>ed ut perspiciatis unde omnis iste natus error sit
        voluptatem accusantium doloremque laudantium, totam rem
        aperiam,
        eaque ipsa quae ab illo inventore veritatis et quasi
        architecto
        beatae vitae dicta sunt explicabo. </p>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
        sed do eiusmod tempor incididunt ut labore et dolore magna
        aliqua.
        Ut enim ad minim veniam, quis nostrud exercitation ullamco
        laboris
        nisi ut aliquip ex ea commodo consequat. </p>
    </div>
</div>
</body>
</html>

```

Let's save the above code in an HTML file **tabsexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

- **Tab 1**
- **Tab 2**
- **Tab 3**

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

ed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Here you can see all the tabs are disabled.

Use of Action Disable(Index)

Now let us see an example using the actions from the above table. The following example demonstrates the use of *disable(index)* method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Tabs functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-
lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-
ui.js"></script>
    <script>
      $(function() {
        $( "#tabs-17" ).tabs();
        $( "#tabs-17" ).tabs("disable",2);
      });
    </script>

    <style>
      #tabs-17{font-size: 14px;}
      .ui-widget-header {
        background:#b9cd6d;
        border: 1px solid #b9cd6d;

```

```

        color: #FFFFFF;
        font-weight: bold;
    }
</style>
</head>
<body>
    <div id="tabs-17">
        <ul>
            <li><a href="#tabs-18">Tab 1</a></li>
            <li><a href="#tabs-19">Tab 2</a></li>
            <li><a href="#tabs-20">Tab 3</a></li>
        </ul>
        <div id="tabs-18">
            <p>Neque porro quisquam est qui dolorem ipsum quia dolor
            sit amet, consectetur, adipisci velit... </p>
        </div>
        <div id="tabs-19">
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
            sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua.
laboris
            Ut enim ad minim veniam, quis nostrud exercitation ullamco
            nisi ut aliquip ex ea commodo consequat. </p>
        </div>
        <div id="tabs-20">
            <p>ed ut perspiciatis unde omnis iste natus error sit
voluptatem
            accusantium doloremque laudantium, totam rem aperiam, eaque
ipsa quae
            ab illo inventore veritatis et quasi architecto beatae vitae
dicta
            sunt explicabo. </p>
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
            sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua.

```



```

        Ut enim ad minim veniam, quis nostrud exercitation ullamco
laboris nisi
        ut aliquip ex ea commodo consequat. </p>
    </div>
</div>
</body>
</html>

```

Let's save the above code in an HTML file **tabsexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

- **Tab 1**
- **Tab 2**
- **Tab 3**

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

ed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

In the above example, you notice that Tab 3 is disabled.

Event Management on tabs elements

In addition to the tabs (options) method which we saw in the previous sections, JQueryUI provides event methods which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
<u>activate(event, ui)</u>	<p>This event is triggered after the tab has been activated (after the completion of animation).</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>newTab: The tab that was just activated.</p>

	<p>oldTab: The tab that was just deactivated.</p> <p>newPanel: The panel that was just activated.</p> <p>oldPanel: The panel that was just deactivated.</p> <p>Syntax</p> <pre>\$(".selector").slider({ activate: function(event, ui) {} });</pre>
<p><u>beforeActivate(event, ui)</u></p>	<p>This event is triggered before a the tab has been activated.</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>newTab: The tab that is about to be activated.</p> <p>oldTab: The tab that is about to be deactivated.</p> <p>newPanel: The panel is about to be activated.</p> <p>oldPanel: The panel is about to be deactivated.</p> <p>Syntax</p> <pre>\$(".selector").slider({ beforeActivate: function(event, ui) {} });</pre>

<p><u>beforeLoad(event, ui)</u></p>	<p>This event is triggered when a remote tab is about to be loaded, after the <i>beforeActivate</i> event. This event is triggered just before the Ajax request is made.</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>tab: The tab that is being loaded.</p> <p>panel: The panel which will be populated by the Ajax response.</p> <p>jqXHR: The jqXHR object that is requesting the content.</p> <p>ajaxSettings: The settings that will be used by jQuery.ajax to request the content.</p> <p>Syntax</p> <pre>\$(".selector").slider({ beforeLoad: function(event, ui) {} });</pre>
<p><u>create(event, ui)</u></p>	<p>This event is triggered when tabs are created.</p> <p>Where event is of type Event, and ui is of type Object. Possible values of ui are:</p> <p>tab: The active tab.</p> <p>panel: The active panel.</p> <p>Syntax</p> <pre>\$(".selector").slider({ create: function(event, ui) {}</pre>

	});
<u>load(event, ui)</u>	<p>This event is triggered after a remote tab has been loaded.</p> <p>Possible values of ui are:</p> <p>tab: The tab that was just loaded.</p> <p>panel: The panel which was just populated by the Ajax response.</p> <p>Syntax</p> <pre>\$(".selector").slider({ load: function(event, ui) {} });</pre>

Example

The following example demonstrates the event method usage in tabs widgets. This example demonstrates the use of events *activate* and *create*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Tabs functionality</title>

    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-
lightness/jquery-ui.css" rel="stylesheet">

    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-
ui.js"></script>
    <script>
      $(function() {
        $( "#tabs-21" ).tabs({
          activate: function( event, ui ) {
```

```

        var result = $( "#result-2" ).empty();
        result.append( "activated" );
    },
    create: function( event, ui ) {
        var result = $( "#result-1" ).empty();
        result.append( "created" );
    }
});
});
</script>
<style>
#tabs-21{font-size: 14px;}
.ui-widget-header {
    background:#b9cd6d;
    border: 1px solid #b9cd6d;
    color: #FFFFFFF;
    font-weight: bold;
}
.resultarea {
    background: #cedc98;
    border-top: 1px solid #000000;
    border-bottom: 1px solid #000000;
    color: #333333;
    font-size:14px;
}
</style>
</head>
<body>
<div id="tabs-21">
    <ul>
        <li><a href="#tabs-22">Tab 1</a></li>
        <li><a href="#tabs-23">Tab 2</a></li>
        <li><a href="#tabs-24">Tab 3</a></li>

```

```

    </ul>
    <div id="tabs-22">
        <p>Neque porro quisquam est qui dolorem ipsum quia dolor
        sit amet, consectetur, adipisci velit... </p>
    </div>
    <div id="tabs-23">
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
        sed do eiusmod tempor incididunt ut labore et dolore magna
        aliqua.
        Ut enim ad minim veniam, quis nostrud exercitation ullamco
        laboris
        nisi ut aliquip ex ea commodo consequat. </p>
    </div>
    <div id="tabs-24">
        <p>ed ut perspiciatis unde omnis iste natus error sit
        voluptatem
        accusantium doloremque laudantium, totam rem aperiam, eaque
        ipsa quae
        ab illo inventore veritatis et quasi architecto beatae vitae
        dicta
        sunt explicabo. </p>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
        sed do eiusmod tempor incididunt ut labore et dolore magna
        aliqua. Ut
        enim ad minim veniam, quis nostrud exercitation ullamco
        laboris nisi
        ut aliquip ex ea commodo consequat. </p>
    </div>
</div><br>
<span class="resultarea" id=result-1></span><br>
<span class="resultarea" id=result-2></span>
</body>
</html>

```

Let's save the above code in an HTML file **tabsexample.htm** and open it in a standard browser which supports javascript, you should see the following output:

- **Tab 1**
- **Tab 2**
- **Tab 3**

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

ed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Click on the tabs to see a message getting printed below on specific to events.

18. JQUERYUI – TOOLTIP

Tooltip widget of jQueryUI replaces the native tooltips. This widget adds new themes and allows for customization. First let us understand what tooltips are? Tooltips can be attached to any element. To display tooltips, just add *title* attribute to input elements and title attribute value will be used as tooltip. When you hover the element with your mouse, the title attribute is displayed in a little box next to the element.

jQueryUI provides **tooltip()** method to add tooltip to any element on which you want to display tooltip. This gives a fade animation by default to show and hide the tooltip, compared to just toggling the visibility.

Syntax

The **tooltip()** method can be used in two forms:

- `$(selector, context).tooltip (options) Method`
- `$(selector, context).tooltip ("action", [params]) Method`

\$ (selector, context).tooltip (options) Method

The *tooltip (options)* method declares that a tooltip can be added to an HTML element. The *options* parameter is an object that specifies the behavior and appearance of the tooltip.

Syntax

```
$(selector, context).tooltip(options);
```

You can provide one or more options at a time using Javascript object. If there are more than one options to be provided then you will separate them using a comma as follows:

```
$(selector, context).tooltip({option1: value1, option2: value2..... });
```

The following table lists the different *options* that can be used with this method:

Option	Description
<u>content</u>	<p>This option represents content of a tooltip. By default its value is function returning the title attribute.</p> <p>This can be of type:</p> <p>Function: The callback can either return the content directly, or call the first argument, passing in the content, eg. for ajax content.</p>

	<p>String: A string of HTML to use for the tooltip content.</p> <p>Syntax</p> <pre>\$(".selector").tooltip({ content: "Some content!" });</pre>
<u>disabled</u>	<p>This option when set to <i>true</i> disables the tooltip. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").tooltip({ disabled: true });</pre>
<u>hide</u>	<p>This option represents the animation effect when hiding the tooltip. By default its value is true.</p> <p>This can be of type:</p> <p>Boolean: This can be true or false. When set to true, the tooltip will fade out with the default duration and the default easing.</p> <p>Number: The tooltip will fade out with the specified duration and the default easing.</p> <p>String: The tooltip will be hidden using the specified effect such as "slideUp", "fold".</p> <p>Object: Possible values are effect, delay, duration, and easing.</p> <p>Syntax</p> <pre>\$(".selector").tooltip(</pre>

	<pre>{ hide: { effect: "explode", duration: 1000 } });</pre>
<u>items</u>	<p>This option indicates which items can show tooltips. By default its value is [title].</p> <p>Syntax</p> <pre>\$(".selector").tooltip({ items: "img[alt]" });</pre>
<u>position</u>	<p>This option decides the position of the tooltip w.r.t the associated target element. By default its value is function returning the title attribute. Possible values are: <i>my</i>, <i>at</i>, <i>of</i>, <i>collision</i>, <i>using</i>, <i>within</i>.</p> <p>Syntax</p> <pre>\$(".selector").tooltip({ { my: "left top+15", at: "left bottom", collision: "flipfit" } });</pre>
<u>show</u>	<p>This option represents how to animate the showing of tooltip. By default its value is true.</p> <p>This can be of type:</p> <p>Boolean: This can be true or false. When set to true, the tooltip will fade out with the default duration and the default easing.</p> <p>Number: The tooltip will fade out with the specified duration and the default easing.</p> <p>String: The tooltip will be hidden using the specified effect such as "slideUp", "fold".</p> <p>Object: Possible values are effect, delay, duration, and easing.</p> <p>Syntax</p>

	<pre>\$(".selector").tooltip({ show: { effect: "blind", duration: 800 } });</pre>
<u>tooltipClass</u>	<p>This option is a class which can be added to the tooltip widget for tooltips such as warning or errors. By default its value is null.</p> <p>Syntax</p> <pre>\$(".selector").tooltip({ tooltipClass: "custom-tooltip-styling" } });</pre>
<u>track</u>	<p>This option when set to <i>true</i>, the tooltip follows/tracks the mouse. By default its value is false.</p> <p>Syntax</p> <pre>\$(".selector").tooltip({ track: true });</pre>

The following section will show you a few working examples of tooltip functionality.

Default Functionality

The following example demonstrates a simple example of tooltip functionality passing no parameters to the **tooltip()** method .

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Tooltip functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

    <!-- Javascript -->
```

```

<script>
    $(function() {
        $("#tooltip-1").tooltip();
        $("#tooltip-2").tooltip();
    });
</script>
</head>
<body>
    <!-- HTML -->
    <label for="name">Name:</label>
    <input id="tooltip-1" title="Enter You name">
    <p><a id="tooltip-2" href="#" title="Nice tooltip">
        I also have a tooltip
    </a></p>
</body>
</html>

```

Let's save the above code in an HTML file **tooltipexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Name:

[I also have a tooltip](#)

In the above example, hover over the links above or use the tab key to cycle the focus on each element.

Use of Content, Track, and Disabled

The following example shows the usage of three important options **(a) content (b) track** and **(c) disabled** in the tooltip function of JQueryUI.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Tooltip functionality</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
        ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>

```

```

<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

<!-- Javascript -->
<script>
    $(function() {
        $( "#tooltip-3" ).tooltip({
            content: "<strong>Hi!</strong>",
            track:true
        }),
        $( "#tooltip-4" ).tooltip({
            disabled: true
        });
    });
</script>
</head>
<body>
    <!-- HTML -->
    <label for="name">Message:</label>
    <input id="tooltip-3" title="tooltip"><br><br><br>
    <label for="name">Tooltip disabled for me:</label>
    <input id="tooltip-4" title="tooltip">
</body>
</html>

```

Let's save the above code in an HTML file **tooltipexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Message:

Tooltip disabled for me:

In the above example, the content of tooltip of first box is set using *content* option. You can also notice the tooltip follows the mouse. The tooltip for second input box is disabled.

Use of Position

The following example shows the usage of option **position** in the tooltip function of JQueryUI.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Tooltip functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      body {
        margin-top: 100px;
      }

      .ui-tooltip-content::after, .ui-tooltip-content::before {
        content: "";
        position: absolute;
        border-style: solid;
        display: block;
        left: 90px;
      }
      .ui-tooltip-content::before {
        bottom: -10px;
        border-color: #AAA transparent;
        border-width: 10px 10px 0;
      }
      .ui-tooltip-content::after {
        bottom: -7px;
        border-color: white transparent;
        border-width: 10px 10px 0;
      }
    </style>
```

```

<!-- Javascript -->
<script>
    $(function() {
        $( "#tooltip-7" ).tooltip({
            position: {
                my: "center bottom",
                at: "center top-10",
                collision: "none"
            }
        });
    });
</script>
</head>
<body>
    <!-- HTML -->
    <label for="name">Enter Date of Birth:</label>
    <input id="tooltip-7" title="Please use MM.DD.YY format.">
</body>
</html>

```

Let's save the above code in an HTML file **tooltipexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Enter Date of Birth:

In the above example, the tooltip position is set on top of the input box.

`$(selector, context).tooltip ("action", [params])` Method

The *tooltip (action, params)* method can perform an action on the tooltip elements, such as disabling the tooltip. The **action** is specified as a string in the first argument and optionally, one or more **params** can be provided based on the given action.

Basically, here actions are nothing but they are jQuery methods which we can use in the form of string.

Syntax

```
$(selector, context).tooltip ("action", [params]);
```

The following table lists the actions for this method:

Action	Description
<u>close()</u>	<p>This action closes the tooltip. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").tooltip("close");</pre>
<u>destroy()</u>	<p>This action removes the tooltip functionality completely. This will return the element back to its pre-init state. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").tooltip("destroy");</pre>
<u>disable()</u>	<p>This action deactivates the tooltip. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").tooltip("disable");</pre>
<u>enable()</u>	<p>This action activates the tooltip. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").tooltip("enable");</pre>
<u>open()</u>	<p>This action programmatically opens the tooltip. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").tooltip("open");</pre>
<u>option(optionName)</u>	<p>This action gets the value associated with <i>optionName</i> for the tooltip. This method does not accept any arguments.</p> <p>Syntax</p> <pre>var isDisabled = \$(".selector").tooltip("option", "disabled");</pre>
<u>option()</u>	<p>This action gets an object containing key/value pairs representing the current tooltip options hash. This method does not accept any arguments.</p> <p>Syntax</p> <pre>\$(".selector").tooltip("option");</pre>

<u><code>option(optionName, value)</code></u>	<p>This action sets the value of the tooltip option associated with the specified <i>optionName</i>.</p> <p>Syntax</p> <p><code>\$(".selector").tooltip("option", "disabled", true);</code></p>
<u><code>option(options)</code></u>	<p>This action sets one or more options for tooltip.</p> <p>Syntax</p> <p><code>\$(".selector").tooltip("option", { disabled: true });</code></p>
<u><code>widget()</code></u>	<p>This action returns a jQuery object containing the original element. This method does not accept any arguments.</p> <p>Syntax</p> <p><code>\$(".selector").tooltip("widget");</code></p>

Examples

Now let us see an example using the actions from the above table. The following example demonstrates the use of actions *disable* and *enable*.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Tooltip functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- Javascript -->
    <script>

      $(function() {

        $("#tooltip-8").tooltip({
          //use 'of' to link the tooltip to your specified input
          position: { of: '#myInput', my: 'left center', at: 'left center'
        },

        }),

        $('#myBtn').click(function () {
```

```

        $('#tooltip-8').tooltip("open");
    });
});
</script>
</head>
<body style="padding:100px;">
    <!-- HTML -->
    <a id="tooltip-8" title="Message" href="#"></a>
    <input id="myInput" type="text" name="myInput" value="0" size="7" />
    <input id="myBtn" type="submit" name="myBtn" value="myBtn" class="myBtn"
/>
</body>
</html>

```

Let's save the above code in an HTML file **tooltipexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

In the above example, click on *myBtn* button and a tooltip pops up.

Event Management on Tooltip Elements

In addition to the tooltip (options) method which we saw in the previous sections, JQueryUI provides event methods as which gets triggered for a particular event. These event methods are listed below:

Event Method	Description
<u>create(event, ui)</u>	<p>Triggered when the tooltip is created. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Syntax</p> <pre>\$(".selector").tooltip({ create: function(event, ui) {} });</pre>
<u>close(event, ui)</u>	<p>Triggered when the tooltip is closed. Usually triggers on <i>focusout</i> or <i>mouseleave</i>. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Possible values of <i>ui</i> are:</p> <p>tooltip: A generated tooltip element.</p>

	<p>Syntax</p> <pre>\$(".selector").tooltip(close: function(event, ui) {});</pre>
<u>open(event, ui)</u>	<p>Triggered when the tooltip is displayed or shown. Usually triggered <i>onfocusin</i> or <i>mouseover</i>. Where <i>event</i> is of type <i>Event</i>, and <i>ui</i> is of type <i>Object</i>.</p> <p>Possible values of <i>ui</i> are:</p> <p>tooltip: A generated tooltip element.</p> <p>Syntax</p> <pre>\$(".selector").tooltip(open: function(event, ui) {});</pre>

Examples

The following example demonstrates the use of event method during tooltip functionality. This example demonstrates use of *open* and *close* events.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Tooltip functionality</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>

    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

    <!-- Javascript -->
    <script>
      $(function() {
        $('.tooltip-9').tooltip({
          items: 'a.tooltip-9',
          content: 'Hello welcome...',
          show: "slideDown", // show immediately
          open: function(event, ui)
```

```

        {
            ui.tooltip.hover(
                function () {
                    $(this).fadeTo("slow", 0.5);
                });
        }
    });
});
$(function() {
    $('.tooltip-10').tooltip({
        items: 'a.tooltip-10',
        content: 'Welcome to TutorialsPoint...',
        show: "fold",
        close: function(event, ui)
        {
            ui.tooltip.hover(function()
            {
                $(this).stop(true).fadeTo(500, 1);
            },
            function()
            {
                $(this).fadeOut('500', function()
                {
                    $(this).remove();
                });
            });
        }
    });
});
});
</script>
</head>
<body style="padding:100px;">
    <!-- HTML -->
    <div id="target">
        <a href="#" class="tooltip-9">Hover over me!</a>
        <a href="#" class="tooltip-10">Hover over me too!</a>
    </div>

```

```
</div>  
</body>  
</html>
```

Let's save the above code in an HTML file **tooltipexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Hover over me! Hover over me too!

In the above example, the tooltip for *Hover over me!* disappear immediately whereas the tooltip for *Hover over me too!* fades out after duration of 1000ms.

Unit III – JQueryUI Effects

19. JQUERYUI – ADDCLASS()

This chapter will discuss the **addClass()** method, which is one of the methods used to manage jQueryUI visual effects. *addClass()* method allow animating the changes to the CSS properties.

addClass() method add specified classes to the matched elements while animating all style changes.

Syntax

Added In Version 1.0 of jQueryUI

The **addClass()** method has its basic syntax as follows:

```
.addClass( className [, duration ] [, easing ] [, complete ] )
```

Parameter	Description
className	This is a String containing one or more CSS classes (separated by spaces).
duration	This is of type Number or String, and indicates the number of milliseconds of the effect. A value of 0 takes the element directly in the new style, without progress. Its default value is <i>400</i> .
easing	This is of type String and indicates the way to progress in the effect. Its default value is <i>swing</i> . Possible values are here .
complete	This is a callback method called for each element when the effect is complete for this element.

Added In Version 1.9 of jQueryUI

With version 1.9, this method now supports a *children* option, which will also animate descendant elements.

```
.addClass( className [, options ] )
```

Parameter	Description
className	This is a String containing one or more CSS classes (separated by spaces).
options	<p>This represents all animation settings. All properties are optional. Possible values are:</p> <p>duration: This is of type Number or String, and indicates the number of milliseconds of the effect. A value of 0 takes the element directly in the new style, without progress. Its default value is <i>400</i>.</p> <p>easing: This is of type String and indicates the way to progress in the effect. Its default value is <i>swing</i>.</p> <p>complete: This is a callback method called for each element when the effect is complete for this element.</p> <p>children: This is of type Boolean and represents whether the animation should additionally be applied to all descendants of the matched elements. Its default value is <i>false</i>.</p> <p>queue: This is of type Boolean or String and represents whether to place the animation in the effects queue. Its default value is <i>true</i>.</p>

Examples

The following example demonstrates the use of *addClass()* methods.

Passing single class

```
<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title>jQuery UI addClass Example</title>

        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">

        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>

        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

        <style>

            .elemClass {

                width: 200px;

                height: 50px;

                background-color: #b9cd6d;

            }

        </style>

    </head>

    <body>
```



```

        .myClass {
            font-size: 40px; background-color: #ccc; color: white;
        }
    </style>
    <script type="text/javascript">
        $(document).ready(function() {
            $('.button').click(function() {
                if (this.id == "add") {
                    $('#animTarget').addClass("myClass", "fast")
                } else {
                    $('#animTarget').removeClass("myClass", "fast")
                }
            })
        });
    </script>
</head>
<body>
    <div id=animTarget class="elemClass">
        Hello!
    </div>
    <button class="button" id="add">Add Class</button>
    <button class="button" id="remove">Remove Class</button>
</body>
</html>

```

Let's save the above code in an HTML file **addclassexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Hello!

Add Class Remove Class

Click on the *Add Class* and *Remove Class* buttons to see the effect of classes on the box.

Passing Multiple Classes

This example shows how to pass multiple classes to the *addClass* method.

```

<!doctype html>
<html lang="en">

```

```

<head>
  <meta charset="utf-8">
  <title>jQuery UI addClass Example</title>
  <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <!-- CSS -->
  <style>
    .red { color: red; }
    .big { font-size: 5em; }
    .spaced { padding: 1em; }
  </style>
  <script>
    $(document).ready(function() {
      $('.button-1').click(function() {
        $( "#welcome" ).addClass( "red big spaced", 3000 );
      });
    });
  </script>
</head>
<body>
  <p id="welcome">Welcome to Tutorials Point!</p>
  <button class="button-1">Click me</button>
</body>
</html>

```

Let's save the above code in an HTML file **addclassexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Tutorials Point!

Click Me

20. JQUERYUI – COLOR ANIMATION

jQueryUI extends some core jQuery methods so that you can animate different transitions for an element. One of them being *animate* method. jQueryUI extends the jQuery *animate* method, to add support for animating colors. You can animate one of several CSS properties that define an element's colors. Following are the CSS properties that the *animate* method supports.

- **backgroundColor** :Sets the background color of the element.
- **borderTopColor** :Sets the color for top side of the element border.
- **borderBottomColor** :Sets the color for bottom side of the element border.
- **borderLeftColor** :Sets the color for left side of the element border.
- **borderRightColor** :Sets the color for right side of the element border.
- **color** :Sets the text color of the element.
- **outlineColor** :Sets the color for the outline; used to emphasize the element.

Syntax

Following is the syntax of the jQueryUI *animate* method:

```
$( "#selector" ).animate(  
    { backgroundColor: "black",  
      color: "white"  
    }  
);
```

You can set any number of properties in this method separated by , (comma).

Example

The following example demonstrates the use of *addClass()* methods.

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>jQuery UI addClass Example</title>  
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">  
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>  
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>  
    <style>
```

```

        .elemClass {
            width: 200px;
            height: 50px;
            background-color: #b9cd6d;
        }
        .myClass {
            font-size: 40px; background-color: #ccc; color: white;
        }
    </style>
    <script type="text/javascript">
        $(document).ready(function() {
            $('#button-1').click(function() {
                $('#animTarget').animate({
                    backgroundColor: "black",
                    color: "white"
                })
            })
        });
    </script>
</head>
<body>
    <div id=animTarget class="elemClass">
        Hello!
    </div>
    <button id="button-1">Click Me</button>
</body>
</html>

```

Let's save the above code in an HTML file **animateexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Hello!

Click Me

Click on the button and see how the animation changes of the box.

21. JQUERYUI – EFFECTS

This chapter will discuss the **effect()** method, which is one of the methods used to manage jQueryUI visual effects. *effect()* method applies an animation effect to the elements without having to show or hide it.

Syntax

The **effect()** method has the following syntax:

```
.effect( effect [, options ] [, duration ] [, complete ] )
```

Parameter	Description
effect	This is a String indicating which effect to use for the transition.
options	This is of type Object and indicates effect-specific settings and <u>easing</u> . Additionally, each effect has its own set of options that can be specified common across multiple effects described in the table <i>jQueryUI Effects</i> .
duration	This is of type Number or String, and indicates the number of milliseconds of the effect. Its default value is <i>400</i> .
complete	This is a callback method called for each element when the effect is complete for this element.

jQueryUI Effects

The following table describes the various effects that can be used with the *effects()* method:

Effect	Description
blind	Shows or hides the element in the manner of a window blind: by moving the bottom edge down or up, or the right edge to the right or left, depending upon the specified <i>direction</i> and <i>mode</i> .

bounce	Causes the element to appear to bounce in the vertical or horizontal direction, optionally showing or hiding the element.
clip	Shows or hides the element by moving opposite borders of the element together until they meet in the middle, or vice versa.
drop	Shows or hides the element by making it appear to drop onto, or drop off of, the page.
explode	Shows or hides the element by splitting it into multiple pieces that move in radial directions as if imploding into, or exploding from, the page.
fade	Shows or hides the element by adjusting its opacity. This is the same as the core fade effects, but without options.
fold	Shows or hides the element by adjusting opposite borders in or out, and then doing the same for the other set of borders.
highlight	Calls attention to the element by momentarily changing its background color while showing or hiding the element.
puff	Expands or contracts the element in place while adjusting its opacity.
pulsate	Adjusts the opacity of the element on and off before ensuring that the element is shown or hidden as specified.
scale	Expands or contracts the element by a specified percentage.
shake	Shakes the element back and forth, either vertically or horizontally.
size	Resizes the element to a specified width and height. Similar to scale except for how the target size is specified.
slide	Moves the element such that it appears to slide onto or off of the page.

transfer	Animates a transient outline element that makes the element appear to transfer to another element. The appearance of the outline element must be defined via CSS rules for the ui-effects-transfer class, or the class specified as an option.
----------	--

Examples

The following examples demonstrates the use of *effect()* method with different effect listed in the above table.

Effect – Shake

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI effect Example</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      #box-1 {
        height: 100px;
        width: 100px;
        background: #b9cd6d;
      }
    </style>
    <script>
      $(document).ready(function() {
        $('#box-1').click(function() {
          $( "#box-1" ).effect( "shake", {
            times: 10,
            distance: 100
          }, 3000, function() {
            $( this ).css( "background", "#cccccc" );
          });
        });
      });
    </script>
```

```

    });
</script>
</head>
<body>
    <div id="box-1">Click On Me</div>
</body>
</html>

```

Let us save the above code in an HTML file **effectexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Click on Me

Effect – explode

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI effect Example</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <!-- CSS -->
        <style>
            #box-2 {
                height: 100px;
                width: 100px;
                background: #b9cd6d;
            }
        </style>
        <script>
            $(document).ready(function() {
                $('#box-2').click(function() {
                    $( "#box-2" ).effect({
                        effect: "explode",
                        easing: "easeInExpo",

```



```
        pieces: 4,  
        duration: 5000  
    });  
});  
});  
</script>  
</head>  
<body>  
    <div id="box-2"></div>  
</body>  
</html>
```

Let us save the above code in an HTML file **effectexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Click on Me

22. JQUERYUI – HIDE

This chapter will discuss the **hide()** method, which is one of the methods used to manage jQueryUI visual effects. *effect()* method applies an animation effect to hide elements.

Syntax

The **hide()** method has the following syntax:

```
.hide( effect [, options ] [, duration ] [, complete ] )
```

Parameter	Description
effect	This is a String indicating which effect to use for the transition.
options	This is of type Object and indicates effect-specific settings and <u>easing</u> . Additionally, each effect has its own set of options that can be specified common across multiple effects described in the table <i>jQueryUI Effects</i> .
duration	This is of type Number or String, and indicates the number of milliseconds of the effect. Its default value is <i>400</i> .
complete	This is a callback method called for each element when the effect is complete for this element.

jQueryUI Effects

The following table describes the various effects that can be used with the *hide()* method:

Effect	Description
blind	Shows or hides the element in the manner of a window blind: by moving the bottom edge down or up, or the right edge to the right or left, depending upon the specified <i>direction</i> and <i>mode</i> .
bounce	Causes the element to appear to bounce in the vertical or horizontal direction, optionally showing or hiding the element.

clip	Shows or hides the element by moving opposite borders of the element together until they meet in the middle, or vice versa.
drop	Shows or hides the element by making it appear to drop onto, or drop off of, the page.
explode	Shows or hides the element by splitting it into multiple pieces that move in radial directions as if imploding into, or exploding from, the page.
fade	Shows or hides the element by adjusting its opacity. This is the same as the core fade effects, but without options.
fold	Shows or hides the element by adjusting opposite borders in or out, and then doing the same for the other set of borders.
highlight	Calls attention to the element by momentarily changing its background color while showing or hiding the element.
puff	Expands or contracts the element in place while adjusting its opacity.
pulsate	Adjusts the opacity of the element on and off before ensuring that the element is shown or hidden as specified.
scale	Expands or contracts the element by a specified percentage.
shake	Shakes the element back and forth, either vertically or horizontally.
size	Resizes the element to a specified width and height. Similar to scale except for how the target size is specified.
slide	Moves the element such that it appears to slide onto or off of the page.
transfer	Animates a transient outline element that makes the element appear to transfer to another element. The appearance of the outline element must be defined via CSS rules for the ui-effects-transfer class, or the class specified as an option.

Examples

The following examples demonstrates the use of *hide()* method with different effect listed in the above table.

Effect - Blind

The following example shows the use of *hide()* method with *blind* effect. Click on the button *Blind Effect Hide* to see the blind effect before the element hides.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI hide Example</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .toggler { width: 500px; height: 200px; }
      #button { padding: .5em 1em; text-decoration: none; }
      #effect { width: 240px; height: 135px; padding: 0.4em; position:
relative; }
      #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
    </style>
    <script>
      $(function() {
        function runEffect() {
          $( "#effect" ).hide( "blind", {times: 10, distance: 100}, 1000,
callback );
        };
        // callback function to bring a hidden box back
        function callback() {
          setTimeout(function() {
            $( "#effect" ).removeAttr( "style" ).hide().fadeIn();
          }, 1000 );
        };
      });
    </script>
  </head>
  <body>
    <div class="toggler">
      <div id="effect">
        <h3>jQuery UI hide Example</h3>
        <button id="button">Blind Effect Hide</button>
      </div>
    </div>
  </body>
</html>
```

```

        $( "#button" ).click(function() {
            runEffect();
            return false;
        });
    });
</script>
</head>
<body>
    <div class="toggler">
        <div id="effect" class="ui-widget-content ui-corner-all">
            <h3 class="ui-widget-header ui-corner-all">Hide</h3>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
                eiusmod tempor incididunt ut labore.
            </p>
        </div>
    </div>

    <a href="#" id="button" class="ui-state-default ui-corner-all">Blind
    Effect Hide</a>
</body>
</html>

```

Let us save the above code in an HTML file **hideexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Hide

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore.

Blind Effect Hide

Effect - Shake

The following example shows the use of *shake()* method with *blind* effect. Click on the button *Shake Effect Hide* to see the shake effect before the element hides.

```

<!doctype html>
<html lang="en">
    <head>

        <meta charset="utf-8">

```

```

<title>jQuery UI hide Example</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<!-- CSS -->
<style>
    .toggler-1 { width: 500px; height: 200px; }
    #button-1 { padding: .5em 1em; text-decoration: none; }
    #effect-1 { width: 240px; height: 135px; padding: 0.4em; position:
relative; }
    #effect-1 h3 { margin: 0; padding: 0.4em; text-align: center; }
</style>
<script>
    $(function() {
        function runEffect() {
            $( "#effect-1" ).hide( "shake", {times: 10, distance: 100},
1000, callback );
        };
        // callback function to bring a hidden box back
        function callback() {
            setTimeout(function() {
                $( "#effect-1" ).removeAttr( "style" ).hide().fadeIn();
            }, 1000 );
        };
        // set effect from select menu value
        $( "#button-1" ).click(function() {
            runEffect();
            return false;
        });
    });
</script>
</head>
<body>
    <div class="toggler-1">
        <div id="effect-1" class="ui-widget-content ui-corner-all">

            <h3 class="ui-widget-header ui-corner-all">Hide</h3>

```

```
<p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
    eiusmod tempor incididunt ut labore.
</p>
</div>
</div>
<a href="#" id="button-1" class="ui-state-default ui-corner-all">Shake
Effect Hide</a>
</body>
</html>
```

Let us save the above code in an HTML file **hideexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Hide

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore.

Shake Effect Hide

23. JQUERYUI – REMOVE CLASS

This chapter will discuss the **removeClass()** method, which is one of the methods used to manage jQueryUI visual effects. *removeClass()* method removes the applied classes from the elements.

removeClass() method removes the specified classes to the matched elements while animating all style changes.

Syntax

Added In Version 1.0 of jQueryUI

The **removeClass()** method has its basic syntax as follows:

```
.removeClass( className [, duration ] [, easing ] [, complete ] )
```

Parameter	Description
className	This is a String containing one or more CSS classes (separated by spaces) to be removed.
duration	This is of type Number or String, and indicates the number of milliseconds of the effect. A value of 0 takes the element directly in the new style, without progress. Its default value is 400.
easing	This is of type String and indicates the way to progress in the effect. Its default value is <i>swing</i> . Possible values are here .
complete	This is a callback method called for each element when the effect is complete for this element.

Added In Version 1.9 of jQueryUI

With version 1.9, this method now supports a *children* option, which will also animate descendant elements.

```
.removeClass( className [, options ] )
```

Parameter	Description
className	This is a String containing one or more CSS classes (separated by spaces).
options	This represents all animation settings. All properties are optional. Possible values are:

	<p>duration: This is of type Number or String, and indicates the number of milliseconds of the effect. A value of 0 takes the element directly in the new style, without progress. Its default value is <i>400</i>.</p> <p>easing: This is of type String and indicates the way to progress in the effect. Its default value is <i>swing</i>.</p> <p>complete: This is a callback method called for each element when the effect is complete for this element.</p> <p>children: This is of type Boolean and represents whether the animation should additionally be applied to all descendants of the matched elements. Its default value is <i>false</i>.</p> <p>queue: This is of type Boolean or String and represents whether to place the animation in the effects queue. Its default value is <i>true</i>.</p>
--	--

Examples

The following example demonstrates the use of *removeClass()* methods.

Passing single class

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>jQuery UI removeClass Example</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <style>
      .elemClass {
        width: 200px;
        height: 50px;
        background-color: #b9cd6d;
      }
      .myClass {
        font-size: 40px; background-color: #ccc; color: white;
      }
    </style>
    <script type="text/javascript">
```

```

$(document).ready(function() {
    $('.button').click(function() {
        if (this.id == "add") {
            $('#animTarget').addClass("myClass", "fast")
        } else {
            $('#animTarget').removeClass("myClass", "fast")
        }
    })
});
</script>
</head>
<body>
    <div id=animTarget class="elemClass">
        Hello!
    </div>
    <button class="button" id="add">Add Class</button>
    <button class="button" id="remove">Remove Class</button>
</body>
</html>

```

Let's save the above code in an HTML file **removeclassexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Hello!

Add Class Remove Class

Click on the *Add Class* and *Remove Class* buttons to see the effect of classes on the box.

24. JQUERYUI – SHOW

This chapter will discuss the **show()** method, which is one of the methods used to manage jQueryUI visual effects. *show()* method displays an item using the indicated effect.

show() method toggles the visibility of the wrapped elements using the specified effect.

Syntax

The **show()** method has the following syntax:

```
.show( effect [, options ] [, duration ] [, complete ] )
```

Parameter	Description
effect	This is a String indicating which effect to use for the transition. This is a String and represents the effect to use when adjusting the element visibility. The effects are listed in the table below.
options	This is of type Object and indicates effect-specific settings and <u>easing</u> . Additionally, each effect has its own set of options that can be specified common across multiple effects described in the table <i>jQueryUI Effects</i> .
duration	This is of type Number or String and determines how long the animation will run. Its default value is <i>400</i> .
complete	This is a callback method called for each element when the effect is complete for this element.

jQueryUI Effects

The following table describes the various effects that can be used with the *effects()* method:

Effect	Description
blind	Shows or hides the element in the manner of a window blind: by moving the bottom edge down or up, or the right edge to the right or left, depending upon the specified <i>direction</i> and <i>mode</i> .
bounce	Causes the element to appear to bounce in the vertical or horizontal direction, optionally showing or hiding the element.

clip	Shows or hides the element by moving opposite borders of the element together until they meet in the middle, or vice versa.
drop	Shows or hides the element by making it appear to drop onto, or drop off of, the page.
explode	Shows or hides the element by splitting it into multiple pieces that move in radial directions as if imploding into, or exploding from, the page.
fade	Shows or hides the element by adjusting its opacity. This is the same as the core fade effects, but without options.
fold	Shows or hides the element by adjusting opposite borders in or out, and then doing the same for the other set of borders.
highlight	Calls attention to the element by momentarily changing its background color while showing or hiding the element.
puff	Expands or contracts the element in place while adjusting its opacity.
pulsate	Adjusts the opacity of the element on and off before ensuring that the element is shown or hidden as specified.
scale	Expands or contracts the element by a specified percentage.
shake	Shakes the element back and forth, either vertically or horizontally.
size	Resizes the element to a specified width and height. Similar to scale except for how the target size is specified.
slide	Moves the element such that it appears to slide onto or off of the page.
transfer	Animates a transient outline element that makes the element appear to transfer to another element. The appearance of the outline element must be defined via CSS rules for the ui-effects-transfer class, or the class specified as an option.

Examples

The following example demonstrates the use of *show()* method.

Show with Shake Effect

The following example demonstrates the use of *show()* method with *shake* effect.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI show Example</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .toggler { width: 500px; height: 200px; }
      #button { padding: .5em 1em; text-decoration: none; }
      #effect { width: 240px; height: 135px; padding: 0.4em; position:
relative; }
      #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
    </style>
    <script>
      $(function() {
        // run the currently selected effect
        function runEffect() {
          // run the effect
          $( "#effect" ).show( "shake", {times: 10,distance: 100}, 1000,
callback);
        };
        //callback function to bring a hidden box back
        function callback() {
          setTimeout(function() {
            $( "#effect:visible" ).removeAttr( "style" ).fadeOut();
          }, 1000 );
        };
        $( "#button" ).click(function() {
```

```

        runEffect();
        return false;
    });
    $( "#effect" ).hide();
});
</script>
</head>
<body>
    <div class="toggler">
        <div id="effect" class="ui-widget-content ui-corner-all">
            <h3 class="ui-widget-header ui-corner-all">Show</h3>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
                eiusmod tempor incididunt ut labore.
            </p>
        </div>
    </div>
    <a href="#" id="button" class="ui-state-default ui-corner-all">Run
    Effect</a>
</body>
</html>

```

Let's save the above code in an HTML file **showexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Show

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore.

Run Effect

Click on the *Add Class* and *Remove Class* buttons to see the effect of classes on the box.

Show with Blind Effect

The following example demonstrates the use of *show()* method with *blind* effect.

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">

```

```

<title>jQuery UI show Example</title>
<link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<!-- CSS -->
<style>
    .toggler { width: 500px; height: 200px; }
    #button { padding: .5em 1em; text-decoration: none; }
    #effect { width: 240px; height: 135px; padding: 0.4em; position:
relative; }
    #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
</style>
<script>
    $(function() {
        // run the currently selected effect
        function runEffect() {
            // run the effect
            $( "#effect" ).show( "blind", {times: 10,distance: 100}, 1000,
callback);
        };
        //callback function to bring a hidden box back
        function callback() {
            setTimeout(function() {
                $( "#effect:visible" ).removeAttr( "style" ).fadeOut();
            }, 1000 );
        };
        // set effect from select menu value
        $( "#button" ).click(function() {
            runEffect();
            return false;
        });
        $( "#effect" ).hide();
    });
</script>
</head>
<body>

```

```

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">Show</h3>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
      eiusmod tempor incididunt ut labore.
    </p>
  </div>
</div>
<a href="#" id="button" class="ui-state-default ui-corner-all">Run
Effect</a>
</body>
</html>

```

Let's save the above code in an HTML file **showexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Show

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore.

Run Effect

25. JQUERYUI – SWITCHCLASS

This chapter will discuss the **switchClass()** method, which is a useful new class for manipulation. *switchClass()* method move from one CSS one CSS class to another, animating the transition from one state to the other.

Syntax

Added In Version 1.0 of jQueryUI

The **switchClass()** method has its basic syntax as follows:

```
.switchClass( removeClassName, addClassName [, duration ] [, easing ] [, complete ] )
```

Parameter	Description
removeClassName	This is a String and represents the CSS class name, or space-delimited list of class names, to be removed.
addClassName	This is of type String and represents one or more class names (space separated) to be added to the class attribute of each matched element.
duration	This is of type Number or String and optionally provides one of <i>slow</i> , <i>normal</i> , <i>fast</i> , or the duration of the effect in milliseconds. If omitted, the <i>animate()</i> method determines the default. Its default value is <i>400</i> .
easing	The name of the easing function to be passed to the <i>animate()</i> method.
complete	This is a callback method called for each element when the effect is complete for this element.

Added In Version 1.9 of jQueryUI

With version 1.9, this method now supports a *children* option, which will also animate descendant elements.

```
.switchClass( removeClassName, addClassName [, options ] )
```

Parameter	Description
removeClassName	This is a String and represents the CSS class name, or space-delimited list of class names, to be removed.

addClassName	This is of type String and represents one or more class names (space separated) to be added to the class attribute of each matched element.
options	<p>This represents all animation settings. All properties are optional. Possible values are:</p> <p>duration: A string or number determining how long the animation will run.. Its default value is <i>400</i>.</p> <p>easing: A string indicating which easing function to use for the transition. Its default value is <i>swing</i>. Possible values are here.</p> <p>complete: This is a callback method called for each element when the effect is complete for this element.</p> <p>children: This is a Boolean and represents whether the animation should additionally be applied to all descendants of the matched elements.</p> <p>queue: This is of type String/Boolean indicating whether to place the animation in the effects queue..</p>

Examples

The following example demonstrates the use of *switchClass()* method.

```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>jQuery UI Switch Class Example</title>
        <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
        <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <!-- CSS -->
        <style>
            .LargeClass
            {
                font-family: Arial;
                font-size: large;

                font-weight: bold;

                color: Red;
```

```

    }
    .NormalClass
    {
        font-family: Arial;
        font-size: small;
        font-weight: bold;
        color: Blue;
    }
</style>
<script>
    $(function() {
        $('#btnSwitch').click(function(){
            $(".NormalClass").switchClass("NormalClass","LargeClass",'fast');
            $(".LargeClass").switchClass("LargeClass","NormalClass",'fast');
            return false;
        });
    });
</script>
</head>
<body>
    <div class="NormalClass">
        Tutorials Point Rocks!!!
    </div>
    <div class="NormalClass">
        Welcome to Tutorials Point!!!
    </div>
    <br />
    <input type="button" id="btnSwitch" value="Switch Class" />
</body>
</html>

```

Let's save the above code in an HTML file **switchclassexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

[Tutorials Point Rocks!!!](#)

[Welcome to Tutorials Point!!!](#)

Switch Class

Click on the *Switch Class* button to see the effect of classes on the box.

26. JQUERYUI – TOGGLE

This chapter will discuss the **toggle()** method of jQueryUI visual effects. *toggle()* method toggles the show () or hide () methods depending on whether the element is hidden or not.

Syntax

The **toggle()** method has the following syntax:

```
.toggle( effect [, options ] [, duration ] [, complete ] )
```

Parameter	Description
effect	This is a String indicating which effect to use for the transition. This is a String and represents the effect to use when adjusting the element visibility. The effects are listed in the table below.
options	This is of type Object and indicates effect-specific settings and <u>easing</u> . Additionally, each effect has its own set of options that can be specified common across multiple effects described in the table <i>jQueryUI Effects</i> .
duration	This is of type Number or String and determines how long the animation will run. Its default value is <i>400</i> .
complete	This is a callback method called for each element when the effect is complete for this element.

jQueryUI Effects

The following table describes the various effects that can be used with the effects() method:

Effect	Description
blind	Shows or hides the element in the manner of a window blind: by moving the bottom edge down or up, or the right edge to the right or left, depending upon the specified <i>direction</i> and <i>mode</i> .
bounce	Causes the element to appear to bounce in the vertical or horizontal direction, optionally showing or hiding the element.

clip	Shows or hides the element by moving opposite borders of the element together until they meet in the middle, or vice versa.
drop	Shows or hides the element by making it appear to drop onto, or drop off of, the page.
explode	Shows or hides the element by splitting it into multiple pieces that move in radial directions as if imploding into, or exploding from, the page.
fade	Shows or hides the element by adjusting its opacity. This is the same as the core fade effects, but without options.
fold	Shows or hides the element by adjusting opposite borders in or out, and then doing the same for the other set of borders.
highlight	Calls attention to the element by momentarily changing its background color while showing or hiding the element.
puff	Expands or contracts the element in place while adjusting its opacity.
pulsate	Adjusts the opacity of the element on and off before ensuring that the element is shown or hidden as specified.
scale	Expands or contracts the element by a specified percentage.
shake	Shakes the element back and forth, either vertically or horizontally.
size	Resizes the element to a specified width and height. Similar to scale except for how the target size is specified.
slide	Moves the element such that it appears to slide onto or off of the page.
transfer	Animates a transient outline element that makes the element appear to transfer to another element. The appearance of the outline element must be defined via CSS rules for the ui-effects-transfer class, or the class specified as an option.

Example

The following example demonstrates the use of *toggle()* method with different effect listed in the above table.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Toggle Example</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .toggler { width: 500px; height: 200px; }
      #button { padding: .5em 1em; text-decoration: none; }
      #effect { width: 240px; height: 135px; padding: 0.4em; position:
relative; }
      #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
    </style>
    <script>
      $(function() {
        function runEffect() {
          $( "#effect" ).toggle('explode', 300);
        };
        $( "#button" ).click(function() {
          runEffect();
          return false;
        });
      });
    </script>
  </head>
  <body>
    <div class="toggler">
      <div id="effect" class="ui-widget-content ui-corner-all">
        <h3 class="ui-widget-header ui-corner-all">Toggle</h3>
```

```
<p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
    eiusmod tempor incididunt ut labore.
</p>
</div>
</div>
<a href="#" id="button" class="ui-state-default ui-corner-all">Toggle</a>
</body>
</html>
```

Let us save the above code in an HTML file **toggleexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Toggle

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.

Click on the Toggle button to check how the classes are shown and hidden.

27. JQUERYUI – TOGGLECLASS

This chapter will discuss the **toggleClass()** method, which is a useful new class for manipulation. *toggleClass()* method adds or removes one or more classes from each element in the set of matched elements.

Syntax

Added In Version 1.0 of jQueryUI

The **toggleClass()** method has its basic syntax as follows:

```
.toggleClass( className [, switch ] [, duration ] [, easing ] [, complete ] )
```

Parameter	Description
className	This is a String and represents the CSS class name, or space-delimited list of class names, to be added, removed, or toggled.
switch	This is of type Boolean and if specified, forces the <i>toggleClass()</i> method to add the class if <i>true</i> , or to remove the class if <i>false</i> .
duration	This is of type Number or String and optionally provides one of <i>slow</i> , <i>normal</i> , <i>fast</i> , or the duration of the effect in milliseconds. If omitted, the <i>animate()</i> method determines the default. Its default value is <i>400</i> .
easing	The name of the easing function to be passed to the <i>animate()</i> method.
complete	This is a callback method called for each element when the effect is complete for this element.

Added in Version 1.9 of jQueryUI

With version 1.9, this method now supports a *children* option, which will also animate descendant elements.

```
.toggleClass( className [, switch ] [, options ] )
```

Parameter	Description
className	This is a String and represents the CSS class name, or space-delimited list of class names, to be added, removed, or toggled.
switch	This is of type Boolean and if specified, forces the <i>toggleClass()</i> method to add the class if <i>true</i> , or to remove the class if <i>false</i> .

options	<p>This represents all animation settings. All properties are optional. Possible values are:</p> <p>duration: A string or number determining how long the animation will run.. Its default value is <i>400</i>.</p> <p>easing: A string indicating which easing function to use for the transition. Its default value is <i>swing</i>. Possible values are here.</p> <p>complete: This is a callback method called for each element when the effect is complete for this element.</p> <p>children: This is a Boolean and represents whether the animation should additionally be applied to all descendants of the matched elements.</p> <p>queue: This is of type String/Boolean indicating whether to place the animation in the effects queue.</p>
---------	---

Examples

The following example demonstrates the use of *toggleClass()* method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI Switch Class Example</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .class1 {
        border-width : 10px;
        border-color : grey;
        background-color : #cedc98;
        color : black;
      }
    </style>
    <script>
      function toggle ()
      {
```

```

        $("#para").toggleClass ("class1", 1000);
    }
</script>
</head>
<body>
    <button onclick=toggle()> Toggle </button>
    <p id="para" style=border-style:solid> Welcome to Tutorials Point </p>
</body>
</html>

```

Let us save the above code in an HTML file **toggleclassexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Toggle

Welcome to Tutorials Point

Click on the *Toggle* button to see how the CSS classes are changed for the text.

Unit IV – JQueryUI Utilities

28. JQUERYUI – POSITION

In this chapter we shall see one of the utility methods of jqueryUi, the *position()* method. The *position()* method allows you to position an element with respect to another element or mouse event.

jQuery UI extends the *.position()* method from jQuery core in a way that lets you describe how you want to position an element the same way you would naturally describe it to another person. Instead of working with numbers and math, you work with meaningful words (such as left and right) and relationships.

Syntax

Following is the syntax of the *position()* method:

```
.position( options )
```

Where *options* is of type Object and provides the information that specifies how the elements of the wrapped set are to be positioned. Following table lists the different *options* that can be used with this method:

Option	Description
<u>my</u>	<p>This option specifies the location of the wrapped elements (the ones being re-positioned) to align with the target element or location. By default its value is center.</p> <p>Two of these values are used to specify location: top, left, bottom, right, and center, separated by a space character, where the first value is the horizontal value, and the second the vertical. Whether the specified single value is considered horizontal or vertical depends upon which value you use (for example, top is taken as vertical, while right is horizontal).</p> <p>Example</p> <p>top, or bottom right.</p>
<u>at</u>	<p>This option is of type String and specifies the location of the target element against which to align the re-positioned elements. Takes the same values as the <i>my</i> option. By default its value is center.</p> <p>Example</p> <p>"right", or "left center"</p>
<u>of</u>	<p>This is of type Selector or Element or jQuery or Event. It identifies the target element against which the wrapped elements are to be re-positioned, or an</p>

	<p>Event instance containing mouse coordinates to use as the target location. By default its value is null.</p> <p>Example</p> <p>#top-menu</p>
<u>collision</u>	<p>This option is of type String and specifies the rules to be applied when the positioned element extends beyond the window in any direction. By default its value is flip.</p> <p>Accepts two (horizontal followed by vertical) of the following:</p> <p>flip: Flips the element to the opposing side and runs collision detection again for fit. If neither side fits, center is used as a fallback.</p> <p>fit: Keeps the element in the desired direction, but adjusts the position such that it will fit.</p> <p>flipfit: First applies the flip logic, placing the element on whichever side allows more of the element to be visible. Then the fit logic is applied to ensure as much of the element is visible as possible.</p> <p>none: Disables collision detection.</p> <p>If a single value is specified, it applies to both directions.</p> <p>Example</p> <p>"flip", "fit", "fit flip", "fit none"</p>
<u>using</u>	<p>This option is a function that replaces the internal function that changes the element position. Called for each wrapped element with a single argument that consists of an object hash with the <i>left</i> and <i>top</i> properties set to the computed target position, and the element set as the function context. By default its value is null.</p> <p>Example</p> <pre>{horizontal: "center", vertical: "left", important: "horizontal" }</pre>
<u>within</u>	<p>This option is a Selector or Element or jQuery element, and allows you to specify which element to use as the bounding box for collision detection. This can come</p>

in handy if you need to contain the positioned element within a specific section of your page. By default its value is window.
--

Example

The following example demonstrates the use of *position* method.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>jQuery UI position method Example</title>
    <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-
ui.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <!-- CSS -->
    <style>
      .positionDiv {
        position: absolute;
        width: 75px;
        height: 75px;
        background: #b9cd6d;
      }
      #targetElement{
        width: 300px;
        height: 500px;
        padding-top:50px;
      }
    </style>
    <script>
      $(function() {
        // Position the dialog offscreen to the left, but centered
        vertically
        $( "#position1" ).position({
          my: "center",
          at: "center",
          of: "#targetElement"
```



```

    });
    $( "#position2" ).position({
        my: "left top",
        at: "left top",
        of: "#targetElement"
    });
    $( "#position3" ).position({
        my: "right-10 top+10",
        at: "right top",
        of: "#targetElement"
    });
    $( document ).mousemove(function( event ) {
        $( "#position4" ).position({
            my: "left+3 bottom-3",
            of: event,
            collision: "fit"
        });
    });
});
</script>
</head>
<body>
    <div id="targetElement">
        <div class="positionDiv" id="position1">Box 1</div>
        <div class="positionDiv" id="position2">Box 2</div>
        <div class="positionDiv" id="position3">Box 3</div>
        <div class="positionDiv" id="position4">Box 4</div>
    </div>
</body>
</html>

```

Let us save the above code in an HTML file **positionmethodexample.htm** and open it in a standard browser which supports javascript, you must also see the following output. Now, you can play with the result:

Box 1
Box 2
Box 3

Box 4

In this example, we see that:

- *Box 1* is aligned to center (horizontally and vertically) *of* the div element.
- *Box 2* is aligned to left top (horizontally and vertically) *of* the div element.
- *Box 3* is displayed in the top right corner of the window, but leave some padding so that the message stands out more. This is done using the horizontal and vertical values of *my* or *at*.
- For *Box 4*, the *of* value is set as an event object. This is an event associated with a pointer and moves with the mouse event.

29. JQUERYUI – WIDGET FACTORY

Earlier, the only way to write custom controls in jQuery was to extend the *\$.fn* namespace. This works well for simple widgets. Suppose you build more stateful widgets, it quickly becomes cumbersome. To aid in the process of building widgets, Widget Factory was introduced in the jQuery UI, which removes most of the boilerplate that is typically associated with managing a widget.

The jQueryUI Widget Factory is simply a function (*\$.widget*) that takes a string name and an object as arguments and creates a jQuery plugin and a "Class" to encapsulate its functionality.

Syntax

Following is the syntax of jQueryUI Widget Factory method:

```
jQuery.widget( name [, base ], prototype )
```

name: It is a string containing a *namespace* and the *widget name* (separated by a dot) of the widget to create.

base: The base widget to inherit from. This must be a constructor that can be instantiated with the ``new`` keyword. Defaults to *jQuery.Widget*.

prototype: The object to use as a prototype for the widget to inherit from. For instance, jQuery UI has a "mouse" plugin on which the rest of the interaction plugins are based. In order to achieve this, *draggable*, *droppable*, etc. all inherit from the mouse plugin like so: `jQuery.widget("ui.draggable", $.ui.mouse, {...});` If you do not supply this argument, the widget will inherit directly from the "base widget," *jQuery.Widget* (note the difference between lowercase "w" *jQuery.widget* and uppercase "W" *jQuery.Widget*).

Base Widget

Base widget is the widget used by the widget factory.

Options

The following table lists the different *options* that can be used with the base widget:

Option	Description
<u>disabled</u>	This option disables the widget if set to <i>true</i> . By default its value is false.
<u>hide</u>	This option determines how to animate the hiding of the element. By default its value is null.

	<p>This can be of type:</p> <p>Boolean: If set to false no animation will be used. Element will fade out with the default duration and the default easing if set to true.</p> <p>Number: The element will fade out with the specified duration and the default easing.</p> <p>String: The element will be hidden using the specified effect.</p> <p>Object: If the value is an object, then effect, delay, duration, and easing properties may be provided.</p> <p>Example</p> <pre>\$(".selector").widget({ hide: { effect: "explode", duration: 1000 } });</pre>
<u>show</u>	<p>This option determines how to animate the showing of the element. By default its value is null.</p> <p>This can be of type:</p> <p>Boolean: If set to false no animation will be used to show the element. Element will fade in with the default duration and the default easing if set to true.</p> <p>Number: The element will fade in with the specified duration and the default easing.</p> <p>String: The element will be shown using the specified effect.</p> <p>Object: If the value is an object, then effect, delay, duration, and easing properties may be provided.</p> <p>Example</p>

	<code>\$(".selector").widget({ show: { effect: "explode", duration: 1000 } });</code>
--	---

Methods

Following table lists the different *methods* that can be used with the base widget:

Action	Description
<u>_create()</u>	This method is the widget's constructor. There are no parameters, but <i>this.element</i> and <i>this.options</i> are already set.
<u>_delay(fn [, delay])</u>	This method invokes the provided function after a specified delay. Returns the timeout ID for use with <i>clearTimeout()</i> .
<u>_destroy()</u>	The public destroy() method cleans up all common data, events, etc. and then delegates out to this <i>_destroy()</i> method for custom, widget-specific, cleanup.
<u>_focusable(element)</u>	This method sets up element to apply the <i>ui-state-focus</i> class on focus. The event handlers are automatically cleaned up on destroy.
<u>_getCreateEventData()</u>	All widgets trigger the <i>create</i> event. By default, no data is provided in the event, but this method can return an object which will be passed as the create event's data.
<u>_getCreateOptions()</u>	This method allows the widget to define a custom method for defining options during instantiation. The user-provided options override the options returned by this method, which override the default options.
<u>_hide(element, option [, callback])</u>	This method hides an element immediately, using built-in animation methods, or using custom effects. See the <i>hide</i> option for possible option values.

<u>_hoverable(element)</u>	This method Sets up element to apply the ui-state-hover class on hover. The event handlers are automatically cleaned up on destroy.
<u>_init()</u>	Any time the plugin is called with no arguments or with only an option hash, the widget is initialized; this includes when the widget is created.
<u>_off(element, eventName)</u>	This method unbinds event handlers from the specified element(s).
<u>_on([suppressDisabledCheck] [, element], handlers)</u>	Binds event handlers to the specified element(s). Delegation is supported via selectors inside the event names, e.g., "click .foo".
<u>_setOption(key, value)</u>	This method is called from the _setOptions() method for each individual option. Widget state should be updated based on changes.
<u>_setOptions(options)</u>	This method is called whenever the option() method is called, regardless of the form in which the option() method was called.
<u>_show(element, option [, callback])</u>	Shows an element immediately, using built-in animation methods, or using custom effects. See the show option for possible option values.
<u>_super([arg] [, ...])</u>	This method invokes the method of the same name from the parent widget, with any specified arguments. Essentially .call().
<u>_superApply(arguments)</u>	Invokes the method of the same name from the parent widget, with the array of arguments.
<u>_trigger(type [, event] [, data])</u>	This method triggers an event and its associated callback. The option with the name equal to type is invoked as the callback.
<u>_destroy()</u>	This method removes the widget functionality completely. This will

	return the element back to its pre-init state.
<u>disable()</u>	This method disables the widget.
<u>enable()</u>	This method enables the widget.
<u>option(optionName)</u>	This method gets the value currently associated with the specified <i>optionName</i> .
<u>option()</u>	This method gets an object containing key/value pairs representing the /current widget options hash.
<u>option(optionName, value)</u>	This method sets the value of the widget option associated with the specified optionName.
<u>option(options)</u>	This method sets one or more options for the widget.
<u>widget()</u>	This method returns a jQuery object containing the original element or other relevant generated element.

Events

Event Method	Description
<u>create(event, ui)</u>	<p>This event is triggered when a widget is created.</p> <p>Syntax</p> <pre>\$(".selector").widget({ create: function(event, ui) {} });</pre>

jQueryUI widget factory Lifecycle

The jQueryUI widget factory, provides an object-oriented way to manage the lifecycle of a widget. These lifecycle activities include:

Creating and destroying a widget: For example,

```
$( "#elem" ).progressbar();
```

Changing widget options: For example

```
$( "#elem" ).progressbar({ value: 20 });
```

Making "super" calls in subclassed widgets: For example

```
$( "#elem" ).progressbar( "value" );
or
$( "#elem" ).progressbar( "value", 40 );
```

Event notifications: For example

```
$( "#elem" ).bind( "progressbarchange", function() {
    alert( "The value has changed!" );
});
```

Example

Now let us create a custom widget in the following example. We will create a button widget. We will see how to create options, methods and events in a widget in the following examples :

Creating Custom Widget

Let us first create a simple custom widget.

```
<!doctype html>
<head>
    <meta charset="utf-8">
    <title>jQuery UI Widget - Default functionality</title>
    <link rel="stylesheet"
href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.css">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script>
        $(function() {
            $.widget("iP.myButton", {
                _create: function() {
                    this._button = $("<button>");
                    this._button.text("My first Widget Button");
                    this._button.width(this.options.width)
                    this._button.css("background-color", this.options.color);
```



```

        this._button.css("position", "absolute");
        this._button.css("left", "100px");
        $(this.element).append(this._button);
    },
    });
    $("#button1").myButton();
});
</script>
</head>
<body>
    <div id="button1"></div>
</body>
</html>

```

Let us save the above code in an HTML file **widgetfactoryexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Adding Options To Custom Widget

In the previous example, we used the `_create` function to create a custom control. But users generally want to customize the control by setting and modifying options. We can define an options object which stores the default values for all of the options you define. `_setOption` function is used for this purpose. It is called for each individual option that the user sets. Here we are setting *width* and *background-color* of the button.

```

<!doctype html>
<head>
    <meta charset="utf-8">
    <title>jQuery UI Widget - Default functionality</title>
    <link rel="stylesheet"
href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.css">
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <script>
        $(function() {
            $.widget("iP.myButton", {
                _create: function() {
                    this._button = $("<button>");
                    this._button.text("My first Widget Button");
                    this._button.width(this.options.width)

```

```

        this._button.css("background-color", this.options.color);
        this._button.css("position", "absolute");
        this._button.css("left", "100px");
        $(this.element).append(this._button);
    },
    _setOption: function(key, value) {
        switch (key) {
            case "width":
                this._button.width(value);
                break;
            case "color":
                this._button.css("background-color",value);
                break;
        }
    },
    });
    $("#button2").myButton();
    $("#button2").myButton("option", {width:100,color:"#cedc98"});
});
</script>
</head>
<body>
    <div id="button2"></div>
</body>
</html>

```

Let us save the above code in an HTML file **widgetfactoryexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Adding Methods to Custom Widget

In the following example we will add methods that the user can make use of and these are very easy to build into the framework. We will write a Move method, that shifts the button a specified horizontal distance. To make this work, we also need to set the position and left properties in the `_create` function:

```

this._button.css("position", "absolute");
this._button.css("left", "100px");

```

Following this, the user can now call your method in the usual jQuery UI way:

```

this._button.css("position", "absolute");
this._button.css("left", "100px");
$("#button3").myButton("move", 200);
<!doctype html>
<head>
  <meta charset="utf-8">
  <title>jQuery UI Widget - Default functionality</title>
  <link rel="stylesheet"
href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.css">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <script>
    $(function() {
      $.widget("iP.myButton", {
        _create: function() {
          this._button = $("<button>");
          this._button.text("My first Widget Button");
          this._button.width(this.options.width)
          this._button.css("background-color", this.options.color);
          this._button.css("position", "absolute");
          this._button.css("left", "100px");
          $(this.element).append(this._button);
        },

        move: function(dx) {
          var x = dx+parseInt(this._button.css("left"));
          this._button.css("left", x);
          if(x>400){ this._trigger("outbounds",{}, {position:x}); }
        }
      });
      $("#button3").myButton();
      $("#button3").myButton("move", 200);
    });
  </script>
</head>

```

```

<body>
    <div id="button3"></div>
</body>
</html>

```

Let us save the above code in an HTML file **widgetfactoryexample.htm** and open it in a standard browser which supports javascript, you must also see the following output:

Adding Events To Custom Widget

In this example we will demonstrate how to create an event. To create an event all you have to do is use the `_trigger` method. The first parameter is the name of the event, the second any standard event object you want to pass and the third any custom event object you want to pass.

Here we are firing an event when if the button moves beyond `x=400`. All you have to do is to add to the move function:

```

if(x<400){ this._trigger("outbounds",{}, {position:x}); }

```

In this case the event is called `outbounds` and an empty event object is passed with a custom event object that simply supplies the position as its only property.

The entire move function is:

```

move: function(dx) {
    var x = dx+parseInt(this._button.css("left"));
    this._button.css("left", x);
    if(x<400){ this._trigger("outbounds",{},
                                {position:x}); }
}

```

The user can set the event handling function by simply defining an option of the same name.

```

$("button4").myButton("option",
    {width: 100,
      color: "red",
      outbounds:function(e,ui){
          alert(ui.position);}
    });
<!doctype html>
<head>
    <meta charset="utf-8">

```

```

<title>jQuery UI Widget - Default functionality</title>
<link rel="stylesheet"
href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<script>
    $(function() {
        $.widget("iP.myButton", {
            _create: function() {
                this._button = $("<button>");
                this._button.text("My first Widget Button");
                this._button.width(this.options.width)
                this._button.css("background-color", this.options.color);
                this._button.css("position", "absolute");
                this._button.css("left", "100px");
                $(this.element).append(this._button);
            },
            move: function(dx) {
                var x = dx+parseInt(this._button.css("left"));
                this._button.css("left", x);
                if(x>400){ this._trigger("outbounds",{}, {position:x}); }
            }
        });
        $("#button4").myButton();
        $("#button4").on("mybuttonoutbounds", function(e, ui) {
            alert("out");
        });
        $("#button4").myButton("move", 500);
    });
</script>
</head>
<body>
    <div id="button4"></div>
</body>
</html>

```

Let us save the above code in an HTML file **widgetfactoryexample.htm** and open it in a standard browser which supports javascript, an alert box opens up.

