JQUERY UI

user interface framework

tutorialspoint

SIMPLYEASYLEARNING

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

Table of Contents

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.

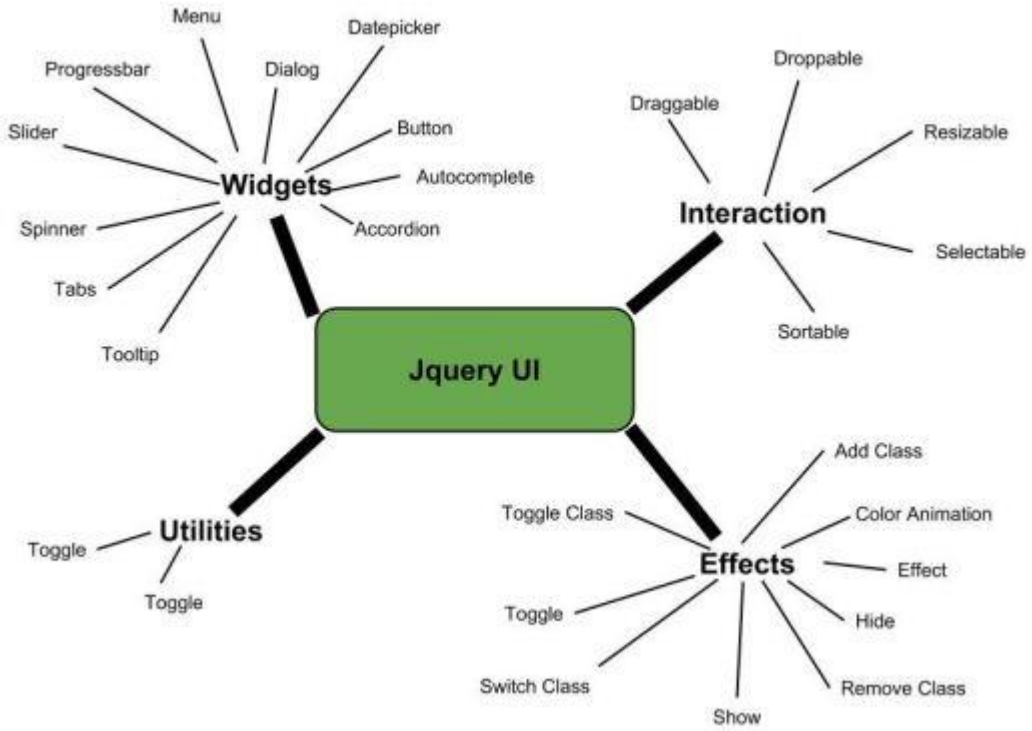
The set of plug-ins in JqueryUI includes interface interactions, effects, animations, widgets, and themes built on top of jQuery JavaScript Library.

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 accordian, 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.

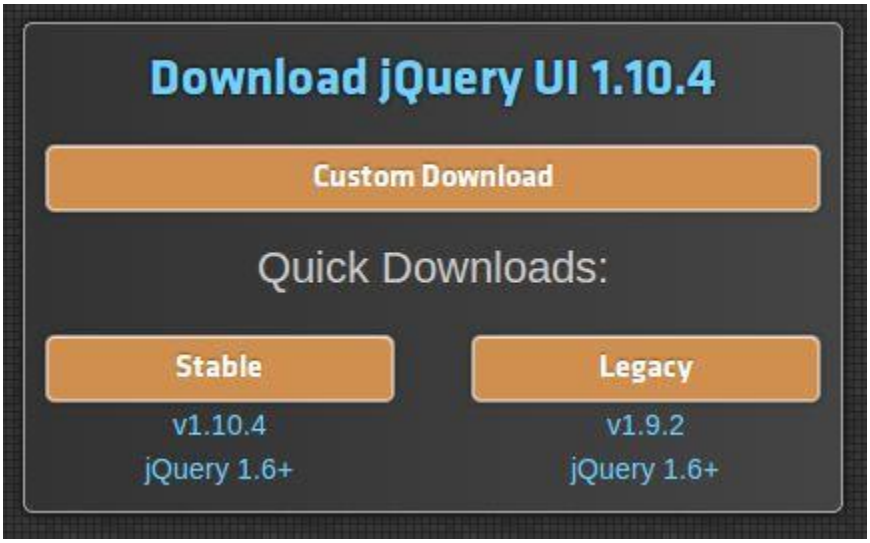
2. JQUERYUI – ENVIRONMENT SETUP

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

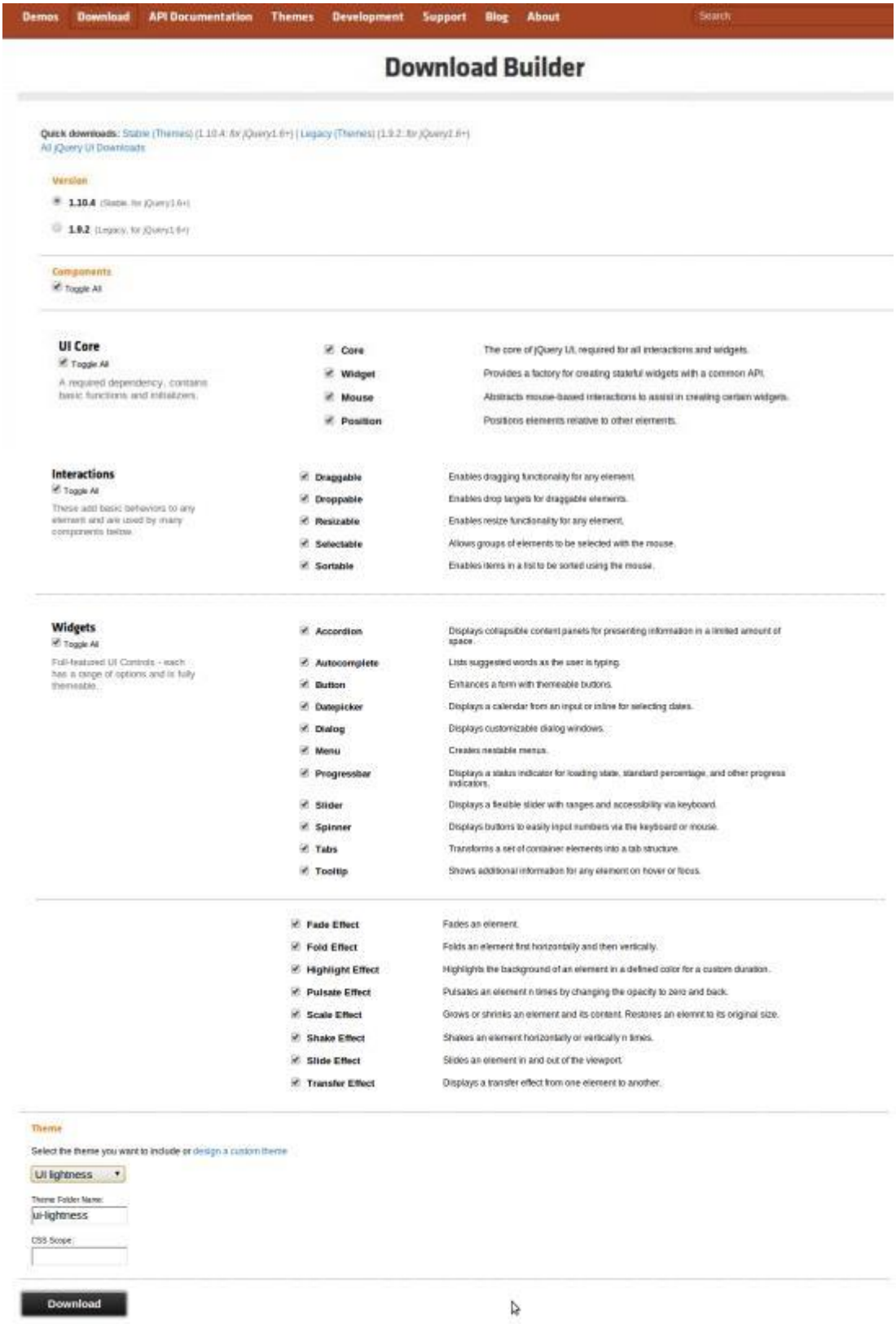
When you open the link http://jqueryui.com/, you will see there are three options to download JqueryUI library:



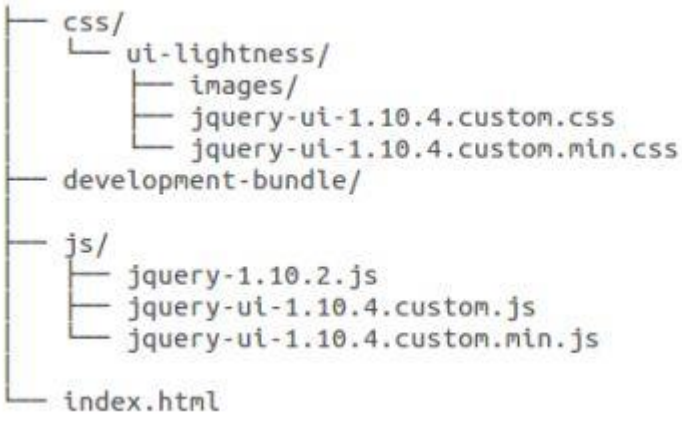
- **Custom Download** : Click on this button to download a customized version of library.
- **Stable** : Click on this button to get the stable and latest version of JqueryUI library.
- **Legacy** : Click on this button to get the previous major release of the JqueryUI library.

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):



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	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. **Syntax** $(".selector").draggable({ addClasses: false });

appendTo	Specifies the element in which the draggable helper should be appended to while dragging. By default its value is "parent". **Syntax** $(".selector").draggable({ appendTo: "body"});
axis	This option constrains dragging to either the horizontal (x) or vertical (y) axis. Possible values: "x", "y". **Syntax** $(".selector").draggable({ axis: "x" });
cancel	You can use this option to prevent dragging from starting on specified elements. By default its value is "input,textarea, button,select,option". **Syntax** $(".selector").draggable({ cancel: ".title" });
connectToSortable	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". **Syntax** $(".selector").draggable({ connectToSortable: "#my-sortable" });
containment	Constrains dragging to within the bounds of the specified element or region. By default its value is "false".

	Syntax $(".selector").draggable({ containment: "parent" });
cursor	Specifies the cursor CSS property when the element moves. It represents the shape of the mouse pointer. By default its value is "auto". By default its value is "auto". Other possible values are: "crosshair" (across) "default" (an arrow) "pointer" (hand) "move" (two arrows cross) "e-resize" (expand to the right) "ne-resize" (expand up right) "nw-resize" (expand up left) "n-resize" (expand up) "se-resize" (expand down right)

	"sw-resize" (expand down left) "s-resize" (expand down) "auto" (default) "w-resize" (expand left) "text" (pointer to write text) "wait" (hourglass) "help" (help pointer) **Syntax** $(".selector").draggable({ cursor: "crosshair" });
cursorAt	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". **Syntax** $(".selector").draggable($(".selector").draggable({ cursorAt: { left: 5 } }););
delay	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". **Syntax**

	$(".selector").draggable({ delay: 300 });
disabled	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". **Syntax** $(".selector").draggable({ disabled: true });
distance	Number of pixels that the mouse must be moved before the displacement is taken into account. By default its value is "1". **Syntax** $(".selector").draggable({ distance: 10 });
grid	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". **Syntax** $(".selector").draggable({ grid: [50, 20] });
handle	If specified, restricts dragging from starting unless the mousedown occurs on the specified element(s). By default its value is "false". **Syntax**

	$(".selector").draggable({ handle: "h2" });
helper	Allows for a helper element to be used for dragging display. By default its value is "original". **Syntax** $(".selector").draggable({ helper: "clone" });
iframeFix	Prevent iframes from capturing the mousemove events during a drag. By default its value is "false". **Syntax** $(".selector").draggable({ iframeFix: true });
opacity	Opacity of the element moved when moving. By default its value is "false". **Syntax** $(".selector").draggable({ opacity: 0.35 });
refreshPositions	If set to *true*, all droppable positions are calculated on every mousemove. By default its value is "false". **Syntax** $(".selector").draggable({ refreshPositions: true });

revert	Indicates whether the element is moved back to its original position at the end of the move. By default its value is "false". **Syntax** $(".selector").draggable({ revert: true });
revertDuration	Duration of displacement (in milliseconds) after which the element returns to its original position (see options.revert). By default its value is "500". Duration of displacement (in milliseconds) after which the element returns to its original position (see options.revert). By default its value is "500". **Syntax** $(".selector").draggable({ revertDuration: 200 });
scope	Used to group sets of draggable and droppable items, in addition to droppable's accept option. By default its value is "default". **Syntax** $(".selector").draggable({ scope: "tasks" });
scroll	When set to *true* (the default), the display will scroll if the item is moved outside the viewable area of the window. By default its value is "true". **Syntax** $(".selector").draggable({ scroll: false });

scrollSensitivity	Indicates how many pixels the mouse must exit the window to cause scrolling of the display. By default its value is "20". **Syntax** $(".selector").draggable({ scrollSensitivity: 100 });
scrollSpeed	Indicates the scrolling speed of the display once scrolling begins. By default its value is "20". **Syntax** $(".selector").draggable({ scrollSpeed: 100 });
snap	Adjusts the display of the item being moved on other elements (which are flown). By default its value is "false". **Syntax** $(".selector").draggable({ snap: true });
snapMode	Specifies how the adjustment should be made between the moved element and those indicated in *options.snap*. By default its value is "both". **Syntax** $(".selector").draggable({ snapMode: "inner" });
snapTolerance	Maximum number of pixels in the difference in position necessary to establish the adjustment. By default its value is "20". **Syntax**

	$(".selector").draggable({ snapTolerance: 30 });
stack	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". **Syntax** $(".selector").draggable({ stack: ".products" });
zIndex	Z-index for the helper while being dragged. By default its value is "false". **Syntax** $(".selector").draggable({ zIndex: 100 });

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 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
destroy()	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");
disable()	Disable drag functionality. Elements cannot be moved until the next call to the draggable("enable") method. **Syntax** $(".selector").draggable("disable");
enable()	Reactivates drag management. The elements can be moved again. **Syntax** $(".selector").draggable("enable");

option(optionName)	Gets the value currently associated with the specified*optionName*. Where *optionName* is name of the option to get and is of type *String*. **Syntax** var isDisabled = $(".selector").draggable("option", "disabled");
option()	Gets an object containing key/value pairs representing the current draggable options hash. **Syntax** var options = $(".selector").draggable("option");
option(optionName, value)	Sets the *value* of the draggable option associated with the specified *optionName*. Where *optionName* is the name of the option to set and *value* is the value to set for the option. **Syntax** $(".selector").draggable("option", "disabled", true);
option(options)	Sets one or more options for the draggable. Where *options* is a map of option-value pairs to set. **Syntax** $(".selector").draggable("option", { disabled: true });
widget()	Returns a jQuery object containing the draggable element. **Syntax** var widget = $(".selector).draggable("widget");

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

create(event, ui)	Triggered when the draggable is created. Where *event* is of type *Event*, and *ui* is of type *Object*. **Syntax** $(".selector").draggable(create: function(event, ui) {});
drag(event, ui)	Triggered while the mouse is moved during the dragging. Where *event* is of type *Event*, and *ui* is of type *Object* like helper, position, offset. **Syntax** $(".selector").draggable(drag: function(event, ui) {});
start(event, ui)	Triggered when dragging starts. Where *event* is of type *Event*, and *ui* is of type *Object* like helper, position, offset. **Syntax** $(".selector").draggable(start: function(event, ui) {});
stop(event, ui)	Triggered when dragging stops. Where *event* is of type *Event*, and *ui* is of type *Object* like helper, position, offset. **Syntax** $(".selector").draggable(stop: function(event, ui) {});

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

accept	This option is used when you need to control which draggable elements are to be accepted for dropping. By default its value is * meaning that every item is accepted by droppable. This can be of type: Selector: This type indicates which draggable elements are accepted. 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. **Syntax** $(".selector").droppable({ accept: ".special" });
activeClass	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 *options.accept*) is being dragged. By default its value is false. **Syntax** $(".selector").droppable({ activeClass: "ui-state-highlight" });
addClasses	This option when set to *false* will prevent the *ui-droppable* class from being added to the droppable elements. By default its value is true. This may be desired as a performance optimization when calling .droppable() init on hundreds of elements. **Syntax** $(".selector").droppable({ addClasses: false });
disabled	This option when set to *true* disables the droppable. By default its value is false.

	Syntax $(".selector").droppable({ disabled: true });
greedy	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 *true*, any parent droppables will not receive the element. **Syntax** $(".selector").droppable({ greedy: true });
hoverClass	This option is *String* representing one or more CSS classes to be added to the element of droppable when an accepted element (an element indicated in *options.accept*) moves into it. By default its value is false. **Syntax** $(".selector").droppable({ hoverClass: "drop-hover" });
scope	This option is used to restrict the droppable action of draggable elements only to items that have the same *options.scope* (defined in draggable (options)). By default its value is "default". **Syntax** $(".selector").droppable({ scope: "tasks" });

tolerance	This option is a *String* that specifies which mode to use for testing whether a draggable is hovering over a droppable. By default its value is "intersect". Possible values are: fit: Draggable covers the droppable element in full. intersect: Draggable overlaps the droppable element at least 50% in both directions. pointer: Mouse pointer overlaps the droppable element. touch: Draggable overlaps the droppable any amount of touching. **Syntax** $(".selector").droppable({ tolerance: "fit" });

End of ebook preview
If you liked what you saw…
Buy it from our store @ **https://store.tutorialspoint.com**