JQUREY

OVERVIEW

WhatisjQuery?

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto: **Write less, do more**. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery:

- **DOM manipulation:** The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called**Sizzle**.
- **Event handling:** The jQuery offers an elegant way to capture a wide variety of events,such as a user clicking on a link, without the need to clutter the HTML code itself withevent handlers.
- **AJAX Support:** The jQuery helps you a lot to develop a responsive and feature-rich site using AJAX technology.
- **Animations:** The jQuery comes with plenty of built-in animation effects which you canuse in your websites.
- **Lightweight:** The jQuery is very lightweight library - about 19KB in size (Minified andgzipped).
- **Cross Browser Support:** The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- **Latest Technology:** The jQuery supports CSS3 selectors and basic XPath syntax.

How to use jQuery?

There are two ways to use jQuery.

- **Local Installation –** You can download jQuery library on your local machine and include it in your HTML code.
- **CDN Based Version –** You can include jQuery library into your HTML code directly from Content Delivery Network (CDN).

Local Installation

- Go to the https://jquery.com/download/ to download the latest version available.
- Now, insert downloaded **jquery-2.1.3.min.js** file in a directory of your website, e.g. /jquery.

Example:

Now, you can include *jquery* library in your HTML file as follows:

```
<html>
   <head>
      <title>The jQuery Example</title>

      <script type="text/javascript"  src="/jquery/jquery-2.1.3.min.js"></script>
      <script type="text/javascript">
         $(document).ready(function(){ doc
            ument.write("Hello, World!");
         });
      </script>

   </head>
   <body>
      <h1>Hello</h1>
   </body>
</html>
```

This will produce the following result-

Hello, World!

CDNBasedVersion

You can include jQuery library into your HTML code directly from Content Delivery Network(CDN). Google and Microsoft provides content deliver for the latest version.

We are using Google CDN version of the library throughout this tutorial.

Example:

Now let us rewrite above example using jQuery library from Google CDN.

```
<html>
   <head>
      <title>The jQuery Example</title>

      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>
      <script type="text/javascript">
         $(document).ready(function(){ doc

            ument.write("Hello, World!");

         });
      </script>

   </head>
   <body>
      <h1>Hello</h1>
   </body>
</html>
```

This will produce the following result:

Hello, World!

HowtoCallajQueryLibraryFunctions?

As almost everything, we do when using jQuery reads or manipulates the document object model (DOM), we need to make sure that we start adding

events etc. as soon as the DOM isready.
If you want an event to work on your page, you should call it inside the
$ (document).ready()function. Everything inside it will load as soon as the DOM isloaded and before the page contents are loaded.
To do this, we register a ready event for the document as follows:

```
$(document).ready(function() {
    // do stuff when DOM is ready
 });
```

To call upon any jQuery library function, use HTML script tags as shown below:

```
<html>
<head>
<title>The jQuery Example</title>

    <script type="text/javascript"
    src="/jquery/jquery-
    1.3.2.min.js"></script>
    <script type="text/javascript" language="javascript">
    // <![CDATA[
    $(document).ready(function() {
        $("div").click(function()
          {alert("Hello world!");
        });
    });
    // ]]>
    </script>

</head>
<body>

<div id="newdiv">
Click on this to see a dialogue box.
</div>

</body>
```

```
</html>
```

This will produce the following result:

```
Click on this to see a dialogue box.
```

HowtoUseCustomScripts?

It is better to write our custom code in the custom JavaScript file : **custom.js**, as follows:

```
/* Filename: custom.js */
$(document).ready(function() {
   $("div").click(function()
        {alert("Hello world!");
   });
});
```

Now we can include **custom.js** file in our HTML file as follows:

```
<html>
<head>
<title>The jQuery Example</title>

    <script type="text/javascript"
    src="/jquery/jquery-
    1.3.2.min.js"></script>
    <script type="text/javascript"
    src="/jquery/custom.js"></script>
</head>
<body>

<div id="newdiv">
Click on this to see a dialogue box.
</div>

</body>
```

```
</html>
```

This will produce the following result:

Click on this to see a dialogue box.

UsingMultipleLibraries

You can use multiple libraries all together without conflicting each others. For example, you can use jQuery and MooTool javascript libraries together. You can check jQuery noConflict Method for more detail.

jQuery noConflict() Method

Many JavaScript libraries use $ as a function or variable name, just as jQuery does. In jQuery'scase, $ is just an alias for jQuery, so all the functionality is available without using $.

Run **$.noConflict()** method to give control of the $ variable back to whichever library first implemented it. This helps us to make sure that jQuery doesn't conflict with the $ object of other libraries.

Here is a simple way of avoiding any conflict:

```
// Import other Library
// Import jQuery Library
$.noConflict();
// Code that uses other library's $ can follow here.
```

This technique is especially effective in conjunction with the .ready() method's ability to aliasthe jQuery object, as within the .ready() we can use $ if we wish without fear of conflicts later:

```
// Import other library
// Import jQuery
$.noConflict();
jQuery(document).ready(function($) {
    // Code that uses jQuery's $ can follow here.
});
// Code that uses other library's $ can follow here.
```

Whatis Next?

Do not worry too much if you did not understand the above examples. You are going to graspthem very soon in subsequent chapters. In the next chapter, we would try to cover few basicconcepts which are coming from conventional JavaScript.

BASICS

jQuery is a framework built using JavaScript capabilities. So, you can use all the functions and other capabilities available in JavaScript. This chapter would explain most basic conceptsbut frequently used in jQuery.

String

A string in JavaScript is an immutable object that contains none, one or many characters. Following are the valid examples of a JavaScript String:

"This is JavaScript String"
'This is JavaScript String'
'This is "really" a JavaScript String'
"This is 'really' a JavaScript String"

Numbers

Numbers in JavaScript are double-precision 64-bit format IEEE 754 values. They areimmutable, just as strings. Following are the valid examples of a JavaScript Numbers:

5350
120.27
0.26

Boolean

A boolean in JavaScript can be either **true** or **false**. If a number is zero, it defaults to false.If there is an empty string, it defaults to false.

Following are the valid examples of a JavaScript Boolean:

True	//	true
False	//	false
0	//	false
1	//	true
...	//	false
hello"	//	true

Objects

JavaScript supports Object concept very well. You can create an object using the object literalas follows:

```
  var emp =
      { name:
      "Zara",age:
      10
  };
```

You can write and read properties of an object using the dot notation as follows:

```
// Getting object properties
emp.name // ==> Zaraemp.age // ==> 10

// Setting object properties
emp.name = "Daisy"  // <== Daisy

emp.age   =   20      // <== 20
```

Arrays

You can define arrays using the array literal as follows:

```
  var x = [];
  var y = [1, 2, 3, 4, 5];
```

An array has a **length** property that is useful for iteration:

```
  var x = [1, 2, 3, 4, 5];
  for (var i = 0; i < x.length; i++) {
      // Do something with x[i]
   }
```

Functions

A function in JavaScript can be either named or anonymous. A named function can be definedusing *function* keyword as follows:

```
  function named(){
    // do some stuff here
  }
```

An anonymous function can be defined in similar way as a normal function but it would not have any name. An anonymous function can be assigned to a variable or passed to a methodas shown below.

```
  var handler = function (){
    // do some stuff here
  }
```

JQuery makes a use of anonymous functions very frequently as follows:

```
  $(document).ready(function(){
    // do some stuff here
  });
```

Arguments

JavaScript variable *arguments* is a kind of array which has *length* property. Following exampleshows it very well:

```
function func(x){
 console.log(typeof x, arguments.length);
}
func();                    //==> "undefined", 0
func(1);                   //==> "number", 1
func("1", "2", "3");       //==> "string", 3
```

The arguments object also has a *callee* property, which refers to the function you're inside.For example:

```
function func() {
    return arguments.callee;
}
func();          // ==> func
```

Context

JavaScript famous keyword **this** always refers to the current context. Within a function
thiscontext can change, depending on how the function is called:

```
  $(document).ready(function() {
     // this refers to window.document
  });

  $("div").click(function() {
     // this refers to a div DOM element
  });
```

You can specify the context for a function call using the function-built-in methods **call()** and**apply()** methods. The difference between them is how they pass arguments. Call passes allarguments through as arguments to the function, while apply accepts an array as the arguments.

```
  function scope() {
     console.log(this, arguments.length);
  }
```

```
scope() // window, 0
scope.call("foobar", [1,2]);       //==> "foobar", 1
scope.apply("foobar", [1,2]);   //==> "foobar", 2
```

Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variablewill have only two scopes.

- **Global Variables:** A global variable has global scope which means it is definedeverywhere in your JavaScript code.
- **Local Variables:** A local variable will be visible only within a function where it isdefined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable withthe same name:

```
var myVar = "global";          // ==> Declare a global variable
function ( ) {
var myVar = "local";           // ==> Declare a local variable
document.write(myVar);       // ==> local }
```

Callback

A callback is a plain JavaScript function passed to some method as an argument or option. Some callbacks are just events, called to give the user a chance to react when a certain stateis triggered. jQuery's event system uses such callbacks everywhere for example:

```
$("body").click(function(event)
    { console.log("clicked: " +
    event.target);
 });
```

Most callbacks provide arguments and a context. In the event-handler example, the callbackis called with one argument, an Event. Some callbacks are required to return something, others make that return value optional. To prevent a form submission, a submit event handlercan return false as follows:

```
$("#myform").submit(function
    () {return false;
  });
```

Closures

Closures are created whenever a variable that is defined outside the current scope is accessedfrom within some inner scope. Following example shows how the variable **counter** is visible within the create, increment, and print functions, but not outside of them:

```
function create() {


    var counter = 0;
    return {
       increment: function()
          {counter++;
       },
   print: function()
          { console.log(counter);  }
    }
}
var c = create();
c.increment();
c.print();          // ==>1
```

This pattern allows you to create objects with methods that operate on data that isn't visibleto the outside world. It should be noted that **data hiding** is the very basis of object-orientedprogramming.

ProxyPattern

A proxy is an object that can be used to control access to another object. It implements the same interface as this other object and passes on any method invocations to it. This other object is often called the real subject. A proxy can be instantiated in place of this real subjectand allow it to be accessed remotely. We can saves jQuery's setArray method in a closure and overwrites it as follows:

```
(function() {
   // log all calls to setArray
   var proxied = jQuery.fn.setArray;


   jQuery.fn.setArray = function()
      {console.log(this, arguments);
      return proxied.apply(this, arguments);
   };
} ) ( );
```

The above wraps its code in a function to hide the proxied variable. The proxy then logs all calls to the method and delegates the call to the original method. Using apply(this, arguments) guarantees that the caller won't be able to notice the difference between the original and the proxied method.

Built-in Functions

JavaScript comes along with a useful set of built-in functions. These methods can be used tomanipulate Strings, Numbers and Dates.
Following are the important JavaScript functions:

Method	Description
charAt()	Returns the character at the specified index.
concat()	Combines the text of two strings and returns a new string.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the index within the calling String object of the firstoccurrence of the specified value, or -1 if not found.
length()	Returns the length of the string.

pop()	Removes the last element from an array and returns thatelement.
push()	Adds one or more elements to the end of an array and returnsthe new length of the array.
reverse()	Reverses the order of the elements of an array -- the firstbecomes the last, and the last becomes the first.
sort()	Sorts the elements of an array.
substr()	Returns the characters in a string beginning at the specifiedlocation through the specified number of characters.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns the string representation of the number's value.
toUpperCase()	Returns the calling string value converted to uppercase.

A complete list of **JavaScript** built-in function is available here – Built-in Functions.

TheDocumentObject Model

The Document Object Model is a tree structure of various elements of HTML as follows:

```
<html>
<head>
   <title>the title</title>
</head>
<body>
   <div>
      <p>This is a paragraph.</p>
      <p>This is second paragraph.</p>
      <p>This is third paragraph.</p>
      </div>
</body>
</html>
```

This will produce the following result:

This is a paragraph.

This is second paragraph.

This is third paragraph

Following are the important points about the above tree structure:

- The <html> is the ancestor of all the other elements; in other words, all the otherelements are descendants of <html>.
- The <head> and <body> elements are not only descendants, but children of <html>,as well.
- Likewise, in addition to being the ancestor of <head> and <body>, <html> is alsotheir parent.
- The <p> elements are children (and descendants) of <div>, descendants of <body>and <html>, and siblings of each other <p> elements.

While learning jQuery concepts, it will be helpful to have understanding on DOM, if you arenot aware of DOM, then I would suggest you to go through our simple tutorial on DOM Tutorial.

SELECTORS

The jQuery library harnesses the power of Cascading Style Sheets (CSS) selectors to let us quickly and easily access elements or groups of elements in the Document Object Model (DOM).
A jQuery Selector is a function which makes use of expressions to find out matching elementsfrom a DOM based on the given criteria.

The$()FactoryFunction

All type of selectors available in jQuery, always start with the dollar sign and parentheses: **$()**. The factory function **$()** makes use of the following three building blocks while selecting elements in a given document:

S.N.	Selector & Description
1	**Tag Name** Represents a tag name available in the DOM. For example $('p') selects all paragraphs <p> in the document.
2	**Tag ID** Represents a tag available with the given ID in the DOM. For example $('#some-id') selects the single element in the document that has an ID of some-id.
3	**Tag Class** Represents a tag available with the given class in the DOM. For example $('.some-class') selects all elements in the document that have a class of some-class.

All the above items can be used either on their own or in combination with other selectors. All the jQuery selectors are based on the same principle except some tweaking.

NOTE: The factory function **$()** is a synonym of **jQuery()** function. So in case you are usingany other JavaScript library where **$** sign is conflicting with some thing else then you can replace **$** sign by **jQuery** name and you can use function **jQuery()** instead of **$()**.

Example

Following is a simple example which makes use of Tag Selector. This would select all theelements with a tag name **p**.

```
<html>
<head>
<title>the title</title>
    <script type="text/javascript"
    src="/jquery/jquery-1.3.2.min.js"></script>
```

```
<script type="text/javascript" language="javascript">
$(document).ready(function
    () {var pars = $("p");
    for( i=0; i<pars.length; i++ ){
        alert("Found paragraph: " + pars[i].innerHTML);
    }
});
</script>
</head>
<body>
    <div>
        <p class="myclass">This is a paragraph.</p>
        <p id="myid">This is second paragraph.</p>
        <p>This is third paragraph.</p>
    </div>
</body>
</html>
```

This will produce the the following result:

```
This is a paragraph.

This is second paragraph.

This is third paragraph.
```

How to Use Selectors?

The selectors are very useful and would be required at every step while using jQuery. Theyget the exact element that you want from your HTML document. Following table lists down few basic selectors and explains them with examples.

S.N.	Selector & Description
1	**Name** Selects all elements which match with the given element **Name**.
2	**#ID** Selects a single element which matches with the given **ID**.

3	**.Class** Selects all elements which matches with the given **Class**.
4	**Universal (*)** Selects all elements available in a DOM.
5	**Multiple Elements E, F, G** Selects the combined results of all the specified selectors **E**, **F** or **G**.

jQuery-ElementNameSelector

Description
The element selector selects all the elements that have a tag name of T.

Syntax
Here is the simple syntax to use this selector −

```
$('tagname')
```

Parameters
Here is the description of all the parameters used by this selector −

- **tagname –** Any standard HTML tag name like div, p, em, img, li etc.

Returns
Like any other jQuery selector, this selector also returns an array filled with the foundelements.

Example:

- **$('p')** − Selects all elements with a tag name of **p** in the document.
- **$('div')** − Selects all elements with a tag name of **div** in the document.

Following example would select all the divisions and will apply yellow color to their background −

```
<html>
    <head>
        <title>The Selecter Example</title>

        <script type="text/javascript"
```

```
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
        </script>

        <script type="text/javascript" language="javascript">
            $(document).ready(function() {
                /* This would select all the divisions */
                $("div").css("background-color", "yellow");
            });
        </script>
    </head>
    <body>

        <div class="big" id="div1">
            <p>This is first division of the DOM.</p>
        </div>

        <div class="medium" id="div2">
            <p>This is second division of the DOM.</p>
        </div>

        <div class="small" id="div3">
            <p>This is third division of the DOM</p>
         </div>

     </body>
</html>
```

This will produce the following result:

```
This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM
```

jQuery-Element ID Selector

Description

The element ID selector selects a single element with the given id attribute.

Syntax

Here is the simple syntax to use this selector −

```
$('#elementid')
```

Parameters

Here is the description of all the parameters used by this selector −

- **Elementid:** This would be an element ID. If the id contains any special characters likeperiods or colons you have to escape those characters with backslashes.

Returns

Like any other jQuery selector, this selector also returns an array filled with the foundelement.

Example

- **$('#myid')** − Selects a single element with the given id myid.
- **$('div#yourid')** − Selects a single division with the given id yourid.

Following example would select second division and will apply yellow color to its backgroundas below:

```
<html>
    <head>
        <title>The Selecter Example</title>
        <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
        </script>

        <script type="text/javascript" language="javascript">
            $(document).ready(function() {
                /* This would select second division only*/
                $("#div2").css("background-color", "yellow");
            });
        </script>
```

```
   </head>
   <body>

      <div class="big" id="div1">
         <p>This is first division of the DOM.</p>
      </div>

      <div class="medium" id="div2">
         <p>This is second division of the DOM.</p>
      </div>

      <div class="small" id="div3">
         <p>This is third division of the DOM</p>
      </div>

   </body>
</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

jQuery-Element Class Selector

Description

The element class selector selects all the elements which match with the given class of theelements.

Syntax

Here is the simple syntax to use this selector:

```
$('.classid')
```

Parameters

Here is the description of all the parameters used by this selector −

- **classid –** This is class ID available in the document.

Returns

Like any other jQuery selector, this selector also returns an array filled with the foundelements.

Example

- **$('.big')** − Selects all the elements with the given class ID **big**.
- **$('p.small')** − Selects all the paragraphs with the given class ID **small**.
- **$('.big.small')** − Selects all the elements with a class of **big** and **small**.

Following example would select all divisions with class .big and will apply yellow colorto its background

```
<html>
    <head>
        <title>The Selecter Example</title>

        <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
        </script>

        <script type="text/javascript" language="javascript">
            $(document).ready(function() {
                /* This would select second division only*/
                $(".big").css("background-color", "yellow");
            });
        </script>

    </head>
    <body>

        <div class="big" id="div1">
            <p>This is first division of the DOM.</p>
        </div>

        <div class="medium" id="div2">

            <p>This is second division of the DOM.</p>
        </div>

        <div class="small" id="div3">
            <p>This is third division of the DOM</p>
        </div>

    </body>
</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

jQuery- Universal Selector

Description
The universal selector selects all the elements available in the document.

Syntax
Here is the simple syntax to use this selector –

```
 $('*')
```

Parameters
Here is the description of all the parameters used by this selector –
***** – A symbolic star.

Returns
Like any other jQuery selector, this selector also returns an array filled with the found elements.

Example

- **$('*')** selects all the elements available in the document.

Following example would select all the elements and will apply yellow color to their background. Try to understand that this selector will select every element including head, body etc.

```
<html>
    <head>
        <title>The Selecter Example</title>
        <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
        </script>

        <script type="text/javascript" language="javascript">
            $(document).ready(function() {
                /* This would select all the elements */
                $("*").css("background-color", "yellow");
            });
        </script>

    </head>
    <body>

        <div class="big" id="div1">
            <p>This is first division of the DOM.</p>
        </div>

        <div class="medium" id="div2">
            <p>This is second division of the DOM.</p>
            </div>

        <div class="small" id="div3">
            <p>This is third division of the DOM</p>
        </div>

    </body>
</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

jQuery–Multiple Elements Selector

Description

This Multiple *Elements* selector selects the combined results of all the specified selectors E, For G.

You can specify any number of selectors to combine into a single result. Here order of theDOM elements in the jQuery object aren't necessarily identical.

Syntax

Here is the simple syntax to use this selector –

```
$('E, F, G,........')
```

Parameters

Here is the description of all the parameters used by this selector –

- **E –** Any valid selector
- **F –** Any valid selector
- **G –** Any valid selector

Returns

Like any other jQuery selector, this selector also returns an array filled with the foundelements.

Example

- **$('div, p') –** selects all the elements matched by **div** or **p**.
- **$('p strong, .myclass') –** selects all elements matched by **strong** that are descendants of an element matched by **p** as well as all elements that have a classof **myclass**.
- **$('p strong, #myid') –** selects a single elements matched by **strong** that is descendant of an element matched by **p** as well as element whose id is **myid**.

Following example would select elements with class ID **big** and element with ID **div3** and willapply yellow color to its background –

```
<html>
    <head>
        <title>The Selecter Example</title>

        <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
        </script>

        <script type="text/javascript" language="javascript">
            $(document).ready(function() {
                $(".big, #div3").css("background-color", "yellow");
            });
        </script>
    </head>
    <body>

        <div class="big" id="div1">
            <p>This is first division of the DOM.</p>
        </div>

        <div class="medium" id="div2">
            <p>This is second division of the DOM.</p>
        </div>

        <div class="small" id="div3">
            <p>This is third division of the DOM</p>
        </div>

    </body>
</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

Selectors Examples

Similar to above syntax and examples, the following examples would give you understandingon using different type of other useful selectors:

S.N.	Selector & Description
1	$('*') This selector selects all elements in the document.
2	$("p > *") This selector selects all elements that are children of a paragraph element.
3	$("#specialID") This selector function gets the element with id="specialID".
4	$(".specialClass") This selector gets all the elements that have the class of *special Class*.
5	$("li:not(.myclass)") Selects all elements matched by <li> that do not have class="myclass".
6	$("a#specialID.specialClass") This selector matches links with an id of *specialID* and a class of *specialClass*.
7	$("p a.specialClass") This selector matches links with a class of *specialClass* declared within <p>elements.
8	$("ul li:first") This selector gets only the first <li> element of the <ul>.
9	$("#container p") Selects all elements matched by <p> that are descendants of an element thathas an id of *container*.
10	$("li > ul") Selects all elements matched by <ul> that are children of an element matchedby <li>
11	$("strong + em") Selects all elements matched by <em> that immediately follow a sibling elementmatched by <strong>.

12	$("p ~ ul") Selects all elements matched by <ul> that follow a sibling element matched by <p>.
13	$("code, em, strong") Selects all elements matched by <code> or <em> or <strong>.
14	$("p strong, .myclass") Selects all elements matched by <strong> that are descendants of an elementmatched by <p> as well as all elements that have a class of *myclass*.
15	$(":empty") Selects all elements that have no children.
16	$("p:empty") Selects all elements matched by <p> that have no children.
17	$("div[p]") Selects all elements matched by <div> that contain an element matched by <p>.
18	$("p[.myclass]") Selects all elements matched by <p> that contain an element with a classof*myclass*.
19	$("a[@rel]") Selects all elements matched by <a> that have a rel attribute.
20	$("input[@name=myname]") Selects all elements matched by <input> that have a name value exactly equalto*myname*.
21	$("input[@name^=myname]") Selects all elements matched by <input> that have a name value beginningwith*myname*.
22	$("a[@rel$=self]") Selects all elements matched by <a> that have rel attribute value ending with *self*.
23	$("a[@href*=domain.com]") Selects all elements matched by <a> that have a href value containingdomain.com.

24	$("li:even") Selects all elements matched by <li> that have an even index value.
25	$("tr:odd") Selects all elements matched by <tr> that have an odd index value.
26	$("li:first") Selects the first <li> element.
27	$("li:last") Selects the last <li> element.
28	$("li:visible") Selects all elements matched by <li> that are visible.
29	$("li:hidden") Selects all elements matched by <li> that are hidden.
30	$(":radio") Selects all radio buttons in the form.
31	$(":checked") Selects all checked boxes in the form.
32	$(":input") Selects only form elements (input, select, textarea, button).
33	$(":text") Selects only text elements (input[type=text]).
34	$("li:eq(2)") Selects the third <li> element.
35	$("li:eq(4)") Selects the fifth <li> element.
36	$("li:lt(2)") Selects all elements matched by <li> element before the third one; in otherwords, the first two <li> elements.
37	$("p:lt(3)") Selects all elements matched by <p> elements before the fourth one; in otherwords the first three <p> elements.
38	$("li:gt(1)") Selects all elements matched by <li> after the second one.
39	$("p:gt(2)") Selects all elements matched by <p> after the third one.

40	$("div/p") Selects all elements matched by <p> that are children of an element matched by <div>.
41	$("div//code") Selects all elements matched by <code>that are descendants of an elementmatched by <div>.
42	$("//p//a") Selects all elements matched by <a> that are descendants of an elementmatched by <p>
43	$("li:first-child") Selects all elements matched by <li> that are the first child of their parent.
44	$("li:last-child") Selects all elements matched by <li> that are the last child of their parent.
45	$(":parent") Selects all elements that are the parent of another element, including text.
46	$("li:contains(second)") Selects all elements matched by <li> that contain the text second.

You can use all the above selectors with any HTML/XML element in generic way. For exampleif selector **$("li:first")** works for <li> element then **$("p:first")** would also work for <p> element.