



JQUERY

web application library

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com



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



<https://twitter.com/tutorialspoint>

About the Tutorial

jQuery is a fast and concise JavaScript library created by John Resig in 2006. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for Rapid Web Development.

Audience

This tutorial is designed for software programmers who wants to learn the basics of jQuery and its programming concepts in simple and easy ways. This tutorial will give you enough understanding on components of jQuery with suitable examples.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of HTML, CSS, JavaScript, Document Object Model (DOM) and any text editor. As we are going to develop web based application using jQuery, it will be good if you have understanding on how internet and web based applications work

Copyright & Disclaimer

© 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
Copyright & Disclaimer.....	i
Table of Contents	ii
1. OVERVIEW.....	1
What is jQuery?	1
How to use jQuery?.....	1
Local Installation	2
CDN Based Version.....	2
How to Call a jQuery Library Functions?	3
How to Use Custom Scripts?.....	4
Using Multiple Libraries	5
What is Next ?.....	6
2. BASICS	7
String	7
Numbers	7
Boolean.....	7
Objects.....	8
Arrays.....	8
Functions	8
Arguments	9
Context	9
Scope	10
Callback.....	10

Closures	11
Proxy Pattern	12
Built-in Functions	12
The Document Object Model	13
3. SELECTORS	15
The \$() Factory Function.....	15
How to Use Selectors?.....	17
jQuery - Element Name Selector	17
jQuery - Element ID Selector	19
jQuery - Element Class Selector.....	20
jQuery - Universal Selector.....	22
jQuery - Multiple Elements Selector.....	24
Selectors Examples.....	26
4. JQUERY ATTRIBUTES.....	31
Get Attribute Value	31
Set Attribute Value.....	32
Applying Styles.....	33
Attribute Methods	34
attr(properties) Method.....	35
attr(key, func) Method	37
removeAttr(name) Method	39
hasClass(class) Method	40
removeClass(class) Method	42
toggleClass(class) Method	43
html() Method.....	44
html(val) Method	46
text() Method.....	47

text(val) Method	48
val() Method	50
val(val) Method	51
5. DOM TRAVERSING	55
Find Elements by Index	55
Filtering Out Elements.....	57
Locating Descendent Elements.....	58
JQuery DOM Filter Methods.....	59
eq(index) Method.....	60
filter(selector) Method	61
filter(fn) Method	63
is(selector) Method	64
map(callback) Method.....	66
not(selector) Method	68
slice(start, end) Method	70
JQuery DOM Traversing Methods	71
add(selector) Method.....	73
andSelf() Method	75
children([selector]) Method	76
closest(selector) Method.....	77
contents() Method	79
end() Method	80
find(selector) Method	81
next([selector]) Method	83
nextAll([selector]) Method	84
offsetParent() Method	85
parent([selector]) Method.....	86

parents([selector]) Method	88
prev([selector]) Method	89
prevAll([selector]) Method	91
siblings([selector]) Method	92
6. CSS SELECTOR METHODS	94
Apply CSS Properties	94
Apply Multiple CSS Properties	94
Setting Element Width & Height	96
jQuery CSS Methods	97
css(name) Method.....	98
css(name, value) Method	100
css(properties) Method	101
height(val) Method.....	103
height() Method	105
innerHeight() Method	106
innerWidth() Method	108
offset() Method.....	110
offsetParent() Method	112
outerHeight([margin]) Method.....	114
outerWidth([margin]) Method	116
position() Method	118
scrollLeft(val) Method	120
scrollLeft() Method.....	121
scrollTop(val) Method	123
scrollTop() Method.....	125
width(val) Method	128
width() Method	129

7. DOM MANIPULATION.....	132
Content Manipulation	132
DOM Element Replacement	133
Removing DOM Elements	135
Inserting DOM Elements	136
DOM Manipulation Methods	138
after(content) Method	140
append(content) Method	141
appendTo(selector) Method	143
before(content) Method	144
clone(bool) Method.....	146
clone() Method.....	148
empty() Method	149
html(val) Method	151
html() Method.....	152
insertAfter(selector) Method	155
insertBefore(selector) Method	157
prepend(content) Method.....	159
prependTo(selector) Method	160
remove(expr) Method	162
replaceAll(selector) Method	164
replaceWith(content) Method.....	165
text(val) Method	167
text() Method.....	169
wrap(elem) Method	172
wrap(html) Method.....	174
wrapAll(elem) Method	175

wrapAll(html) Method.....	177
wrapInner(elem) Method	179
wrapInner(html) Method	181
8. EVENTS HANDLING.....	184
Binding Event Handlers	184
Removing Event Handlers	186
Event Types	186
The Event Object	188
The Event Attributes	188
The Event Methods	191
preventDefault() Method	192
isDefaultPrevented() Method.....	193
stopPropagation() Method.....	195
isPropagationStopped() Method	196
stopImmediatePropagation() Method.....	198
isImmediatePropagationStopped() Method.....	200
Event Manipulation Methods.....	202
bind(type, [data], fn) Method.....	203
off(events [, selector] [, handler(eventObject)]) Method	205
hover(over, out) Method.....	207
on(events [, selector] [, data], handler) Method	208
one(type, [data], fn) Method.....	210
ready(fn) Method	212
trigger(event, [data]) Method	213
triggerHandler(event, [data]) Method.....	215
unbind([type], [fn]) Method	216
Event Helper Methods	218

Trigger Methods.....	218
Binding Methods.....	218
9. AJAX.....	222
Loading Simple Data.....	222
Getting JSON Data.....	223
Passing Data to the Server	225
jQuery AJAX Methods	226
jQuery AJAX Events.....	249
10. EFFECTS.....	264
Showing and Hiding Elements	264
Toggling the Elements	266
jQuery Effect Methods.....	267
UI Library Based Effects.....	300

1. OVERVIEW

What is jQuery?

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 with event 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 can use in your websites.
- **Lightweight:** The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
- **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(){
        document.write("Hello, World!");
      });
    </script>

  </head>

  <body>

    <h1>Hello</h1>

  </body>
</html>
```

This will produce the following result –

```
Hello, World!
```

CDN Based Version

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(){
        document.write("Hello, World!");
      });
    </script>
  </head>

  <body>

    <h1>Hello</h1>

  </body>
</html>
```

This will produce the following result:

```
Hello, World!
```

How to Call a jQuery Library Functions?

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 is ready.

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 is loaded 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">  
    // <br/>    $(document).ready(function() {<br/>      $("div").click(function() {<br/>        alert("Hello world!");<br/>      });<br/>    });<br/>    // ]]&gt;<br/>  &lt;/script&gt;<br/><br/>&lt;/head&gt;<br/>&lt;body&gt;<br/>&lt;div id="newdiv"&gt;<br/>Click on this to see a dialogue box.</pre></div><div data-bbox="855 853 889 869" data-label="Page-Footer">12</div><div data-bbox="115 867 420 936" data-label="Page-Footer"><img alt="Tutorialspoint logo" data-bbox="115 867 205 936"/><p><b>tutorialspoint</b><br/>SIMPLYEASYLEARNING</p></div>
```

```

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

```

This will produce the following result:

```

Click on this to see a dialogue box.

```

How to Use Custom Scripts?

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.

Using Multiple Libraries

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's case, \$ 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 alias the 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.
```

What is Next ?

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

2. 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 concepts but 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 are immutable, 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 literal as 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 // ==> Zara  
emp.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 defined using *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 method as 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 example shows 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 **this** context 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 all arguments 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 variable will have only two scopes.

- **Global Variables:** A global variable has global scope which means it is defined everywhere in your JavaScript code.

- **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the 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 state is 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 callback is 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 handler can return false as follows:

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

Closures

Closures are created whenever a variable that is defined outside the current scope is accessed from 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 visible to the outside world. It should be noted that **data hiding** is the very basis of object-oriented programming.

Proxy Pattern

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 subject and allow it to be accessed remotely. We can save 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 to manipulate 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 first occurrence 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 that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reverse()	Reverses the order of the elements of an array -- the first becomes 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 specified location 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](#).

The Document Object 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 other elements 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 also their 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 are not aware of DOM, then I would suggest you to go through our simple tutorial on [DOM Tutorial](#).

3. 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 elements from a DOM based on the given criteria.

The **\$()** Factory Function

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 <code>\$('p')</code> selects all paragraphs <code><p></code> in the document.
2	Tag ID Represents a tag available with the given ID in the DOM. For example <code>\$('#some-id')</code> 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 <code>\$('.some-class')</code> 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 using any 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 the elements 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. They get 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 - Element Name Selector

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 found elements.

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 Selector 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 like periods 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 found element.

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 background as below:

```
<html>
  <head>
    <title>The Selector 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>
```

30

```
</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 the elements.

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 found elements.

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 color to its background

```
<html>
  <head>
    <title>The Selector 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 Selector 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>
  </body>
</html>
```

```

    </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, F or G.

You can specify any number of selectors to combine into a single result. Here order of the DOM 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 found elements.

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 class of **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 will apply yellow color to its background –

```
<html>
  <head>
    <title>The Selector 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 understanding on 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 <i>specialClass</i> .
5	<code>\$("li:not(.myclass)")</code> Selects all elements matched by <code></code> that do not have <code>class="myclass"</code> .
6	<code>\$("a#specialID.specialClass")</code> This selector matches links with an id of <i>specialID</i> and a class of <i>specialClass</i> .
7	<code>\$("p a.specialClass")</code> This selector matches links with a class of <i>specialClass</i> declared within <code><p></code> elements.
8	<code>\$("ul li:first")</code> This selector gets only the first <code></code> element of the <code></code> .
9	<code>\$("#container p")</code> Selects all elements matched by <code><p></code> that are descendants of an element that has an id of <i>container</i> .
10	<code>\$("li > ul")</code> Selects all elements matched by <code></code> that are children of an element matched by <code></code>
11	<code>\$("strong + em")</code> Selects all elements matched by <code></code> that immediately follow a sibling element matched by <code></code> .
12	<code>\$("p ~ ul")</code> Selects all elements matched by <code></code> that follow a sibling element matched by <code><p></code> .
13	<code>\$("code, em, strong")</code> Selects all elements matched by <code><code></code> or <code></code> or <code></code> .

14	<pre>\$("p strong, .myclass")</pre> <p>Selects all elements matched by <code></code> that are descendants of an element matched by <code><p></code> as well as all elements that have a class of <i>myclass</i>.</p>
15	<pre>\$(":empty")</pre> <p>Selects all elements that have no children.</p>
16	<pre>\$("p:empty")</pre> <p>Selects all elements matched by <code><p></code> that have no children.</p>
17	<pre>\$("div[p]")</pre> <p>Selects all elements matched by <code><div></code> that contain an element matched by <code><p></code>.</p>
18	<pre>\$("p[.myclass]")</pre> <p>Selects all elements matched by <code><p></code> that contain an element with a class of <i>myclass</i>.</p>
19	<pre>\$("a[@rel]")</pre> <p>Selects all elements matched by <code><a></code> that have a <code>rel</code> attribute.</p>
20	<pre>\$("input[@name=myname]")</pre> <p>Selects all elements matched by <code><input></code> that have a <code>name</code> value exactly equal to <i>myname</i>.</p>
21	<pre>\$("input[@name^=myname]")</pre> <p>Selects all elements matched by <code><input></code> that have a <code>name</code> value beginning with <i>myname</i>.</p>
22	<pre>\$("a[@rel\$=self]")</pre> <p>Selects all elements matched by <code><a></code> that have <code>rel</code> attribute value ending with <i>self</i>.</p>
23	<pre>\$("a[@href*=domain.com]")</pre>

	Selects all elements matched by <a> that have a href value containing domain.com.
24	<code>\$("li:even")</code> Selects all elements matched by that have an even index value.
25	<code>\$("tr:odd")</code> Selects all elements matched by <tr> that have an odd index value.
26	<code>\$("li:first")</code> Selects the first element.
27	<code>\$("li:last")</code> Selects the last element.
28	<code>\$("li:visible")</code> Selects all elements matched by that are visible.
29	<code>\$("li:hidden")</code> Selects all elements matched by that are hidden.
30	<code>\$(":radio")</code> Selects all radio buttons in the form.
31	<code>\$(":checked")</code> Selects all checked boxes in the form.
32	<code>\$(":input")</code> Selects only form elements (input, select, textarea, button).
33	<code>\$(":text")</code> Selects only text elements (input[type=text]).

34	<code>\$("li:eq(2)")</code> Selects the third element.
35	<code>\$("li:eq(4)")</code> Selects the fifth element.
36	<code>\$("li:lt(2)")</code> Selects all elements matched by element before the third one; in other words, the first two elements.
37	<code>\$("p:lt(3)")</code> Selects all elements matched by <p> elements before the fourth one; in other words the first three <p> elements.
38	<code>\$("li:gt(1)")</code> Selects all elements matched by after the second one.
39	<code>\$("p:gt(2)")</code> Selects all elements matched by <p> after the third one.
40	<code>\$("div/p")</code> Selects all elements matched by <p> that are children of an element matched by <div>.
41	<code>\$("div//code")</code> Selects all elements matched by <code> that are descendants of an element matched by <div>.
42	<code>\$("//p//a")</code> Selects all elements matched by <a> that are descendants of an element matched by <p>
43	<code>\$("li:first-child")</code>

	Selects all elements matched by that are the first child of their parent.
44	<code>\$("li:last-child")</code> Selects all elements matched by that are the last child of their parent.
45	<code>\$(":parent")</code> Selects all elements that are the parent of another element, including text.
46	<code>\$("li:contains(second)")</code> Selects all elements matched by that contain the text second.

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

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