What is jQuery

jQuery is a fast, small and feature-rich JavaScript library included in a single .js file.

jQuery makes a web developer's life easy. It provides many built-in functions using which you can accomplish various tasks easily and quickly.

jQuery Important Features

* DOM Selection: jQuery provides Selectors to retrieve DOM element based on different criteria like tag name, id, css class name, attribute name, value, nth child in hierarchy etc.
* DOM Manipulation: You can manipulate DOM elements using various built-in jQuery functions. For example, adding or removing elements, modifying html content, css class etc.
* Special Effects: You can apply special effects to DOM elements like show or hide elements, fade-in or fade-out of visibility, sliding effect, animation etc.
* Events: jQuery library includes functions which are equivalent to DOM events like click, dblclick, mouseenter, mouseleave, blur, keyup, keydown etc. These functions automatically handle cross-browser issues.
* Ajax: jQuery also includes easy to use AJAX functions to load data from servers without reloading whole page.
* Cross-browser support: jQuery library automatically handles cross-browser issues, so the user does not have to worry about it. jQuery supports IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+.

Advantages of jQuery

* Easy to learn: jQuery is easy to learn because it supports same JavaScript style coding.
* Write less do more: jQuery provides a rich set of features that increase developers' productivity by writing less and readable code.
* Excellent API Documentation: jQuery provides excellent online API documentation.
* Cross-browser support: jQuery provides excellent cross-browser support without writing extra code.
* Unobtrusive: jQuery is unobtrusive which allows separation of concerns by separating html and jQuery code.

jQuery is a lightweight JavaScript library that simplifies programming with JavaScript.    
  
   
  
**According to jQuery.com**  
jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.  
  
**Why should we use jQuery**   
OR  
**Advantages of using jQuery over raw JavaScript**  
  
The use of JQuery has several benefits over using the raw javascript.  
1. jQuery is cross-browser  
2. jQuery is lot more easy to use than raw javascript  
3. jQuery is extensible  
4. jQuery simplifies and has rich AJAX support  
5. jQuery has large development community and many plugins. Example autocomplete textbox plugin.  
6. Excellent documentation  
  
**How to use jQuery in a web application**  
Download the jQuery file from jQuery.com and reference it in your application just like any other JavaScript file.  
  
**What is the difference between jQuery 1.x and 2.x**  
If you want to support IE6/7/8, then use jQuery 1.x where as if you don't have the need to support IE6/7/8 then use jQuery 2.x. jQuery 2.x is smaller in size than jQuery 1.x.  
  
**Example :**Adding a click event handler for a button control using raw JavaScript. addEventListener() method is not supported in IE < 9. 

<script type="text/javascript">

    window.onload = function ()

    {

        // For all modern browsers

        if (document.addEventListener)

        {

            document.getElementById('button1')

                    .addEventListener('click', clickHandler, false);

        }

        else

        // For Internet Explorer < 9

        {

            document.getElementById('button1')

                    .attachEvent('onclick', clickHandler);

        }

        function clickHandler()

        {

            alert('jQuery Tutorial');

        }

    };

</script>

<input type="button" value="Click Me" id="button1" />

**Example :** Adding a click event handler for a button control using jQuery. With jQuery we have less code to achieve the same thing.We don't have to worry about cross-browser issues, as all this is taken care by jQuery.  
  
**Please Note :**If you want this example to work in IE 6/7/8, then use jQuery 1.x. If there is no need to support IE 6/7/8, then use jQuery 2.x.

<script type="text/javascript">

    $('document').ready(function () {

        $('#button1').click(function () {

            alert('jQuery Tutorial');

        });

    });

</script>

<input type="button" value="Click Me" id="button1" />

**Points to remember :**  
1. ready() function ensures that the DOM is fully loaded.  
2. $ is a shortcut for jQuery.  
3. All three of the following syntaxes are equivalent:  
$( document ).ready( handler )  
$().ready( handler ) (this is not recommended)  
$( handler )

**$(document).ready** is a jQuery event. It fires as soon as the DOM is loaded and ready to be manipulated by script. This is the earliest point in the page load process where the script can safely access elements in the page's html DOM. This event is fired before all the images, css etc.. are fully loaded.  
  
The following example works, because the jquery code that adds the event handler to the button is inside the ready() function, which ensures that the DOM is fully loaded before this piece of code is executed, so the JavaScript can find the button element in the DOM and adds the click event handler.

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#button1').click(function () {

                alert('jQuery Tuorial');

            });

        });

    </script>

</head>

<body>

    <input id="button1" type="button" value="Click Me" />

</body>

</html>

In the following example, we have removed the ready() method. When you click the button now, you don't get the alert. This is because the jQuery code is present before the button element, so by the time the jQuery code is executed the button element is not loaded into DOM. 

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $('#button1').click(function () {

            alert('jQuery Tuorial');

        });

    </script>

</head>

<body>

    <input id="button1" type="button" value="Click Me" />

</body>  
</html>

**To make this example work, you have 2 options**  
1. Place your jQuery code in $(document).ready function OR  
2. Place your script at the bottom of the page just before the closing </body> element  
  
**$(window).load event fires when the DOM and all the content on the page (images, css etc) is fully loaded.** Since the window load event waits for images, css etc to be fully loaded, this event fires after ready event.   
  
The following example proves the above point. When you run the page with the following script, notice that the alert in ready function is displayed before the alert in load function.

<script type="text/javascript">

    $(window).load(function () {

        alert('Window loaded');

    });

    $(document).ready(function () {

        alert('DOM Loaded and ready');

    });

</script>

In most cases, the script can be run as soon as the DOM hierarchy has been fully constructed. So ready() is usually the best place to write your JavaScript code.  
  
However, in your application there could be scenarios where you should be using $(window).load over $(document).ready. For example, let's say we want to display the actual image dimensions (Height and Width). To get the actual image dimensions, we will have to wait until the image is fully loded, so the jQuery code to get the height and width should be in $(window).load event.  
  
**Example :** If you use $(document).ready() instead of $(window).load() the height and width will be displayed as 0.

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(window).load(function () {

            $('#div1').html("Height = " + $('#Image1').height()

                + "<br/>" + "Width = " + $('#Image1').width())

        });

    </script>

</head>

<body>

    <div id="div1"></div>

    <img src="Chrysanthemum.jpg" id="Image1" />

</body>

</html>

**$(document).ready** is a jQuery event. It fires as soon as the DOM is loaded and ready to be manipulated by script. This is the earliest point in the page load process where the script can safely access elements in the page's html DOM. This event is fired before all the images, css etc.. are fully loaded.  
  
The following example works, because the jquery code that adds the event handler to the button is inside the ready() function, which ensures that the DOM is fully loaded before this piece of code is executed, so the JavaScript can find the button element in the DOM and adds the click event handler.

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#button1').click(function () {

                alert('jQuery Tuorial');

            });

        });

    </script>

</head>

<body>

    <input id="button1" type="button" value="Click Me" />

</body>

</html>

In the following example, we have removed the ready() method. When you click the button now, you don't get the alert. This is because the jQuery code is present before the button element, so by the time the jQuery code is executed the button element is not loaded into DOM.

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $('#button1').click(function () {

            alert('jQuery Tuorial');

        });

    </script>

</head>

<body>

    <input id="button1" type="button" value="Click Me" />

</body>  
</html>

**To make this example work, you have 2 options**  
1. Place your jQuery code in $(document).ready function OR  
2. Place your script at the bottom of the page just before the closing </body> element  
  
**$(window).load event fires when the DOM and all the content on the page (images, css etc) is fully loaded.** Since the window load event waits for images, css etc to be fully loaded, this event fires after ready event.   
  
The following example proves the above point. When you run the page with the following script, notice that the alert in ready function is displayed before the alert in load function.

<script type="text/javascript">

    $(window).load(function () {

        alert('Window loaded');

    });

    $(document).ready(function () {

        alert('DOM Loaded and ready');

    });

</script>

In most cases, the script can be run as soon as the DOM hierarchy has been fully constructed. So ready() is usually the best place to write your JavaScript code.  
  
However, in your application there could be scenarios where you should be using $(window).load over $(document).ready. For example, let's say we want to display the actual image dimensions (Height and Width). To get the actual image dimensions, we will have to wait until the image is fully loded, so the jQuery code to get the height and width should be in $(window).load event.  
  
**Example :** If you use $(document).ready() instead of $(window).load() the height and width will be displayed as 0.

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(window).load(function () {

            $('#div1').html("Height = " + $('#Image1').height()

                + "<br/>" + "Width = " + $('#Image1').width())

        });

    </script>

</head>

<body>

    <div id="div1"></div>

    <img src="Chrysanthemum.jpg" id="Image1" />

</body>

</html>

**What is a CDN**  
**CDN** stands for **Content Delivery Network**. A CDN is a system of distributed servers that hosts resources such as images, CSS, JavaScript files etc.  
  
Companies like Microsoft, Google, Yahoo etc have a free public CDN from which we can load jQuery instead of hosting it on our own web server.  
  
**Microsoft jQuery CDN**  
<http://www.asp.net/ajax/cdn#jQuery_Releases_on_the_CDN_0>  
  
**Google jQuery CDN**  
<https://developers.google.com/speed/libraries/devguide#jquery>   
  
**Advantages of using a CDN**  
1. **Distributed CDN servers** : The jQuery file can be downloaded from the CDN server that is closest to the user   
  
2. **Browser Caching** : jQuery is used on many popular websites. If a user has already visited a webpage that uses jQuery from a CDN, and then if he arrives at your page, the jQuery file has already been cached by the browser so there is no need to download it again.  
  
3. **Parallel Downloads** : There is a browser limit on how many files can be concurrently downloaded from a given domain. This number varies from browser to browser. For example, if the browser allows only 2 concurrent downloads from a given domain, the 3rd download is blocked until one of the previous files has been fully downloaded. Since the jQuery file is on a CDN, it is being downloaded from a different domain. So this means the browser allows another 2 parallel downloads from the CDN server.  
  
4. **Reduced server load** : The HTTP request for jQuery file is handled by the CDN server, so the load on your web server is reduced. This also means there is a saving on your website bandwidth consumption which in turn will reduce your hosting cost.  
  
**Disadvantages of using a CDN**  
Your clients may block the CDN. So you may have to request your clients to whitelist the CDN.  
  
**What if the required jQuery file cannot be downloaded from CDN**  
Let assume that, the CDN is down or because of some network issue we are not able to download jQuery from CDN. In this case we will have to fallback to use jQuery file that we hosted on our own server.   
  
Here is the code that falls back to use jQuery on your web server, if it can't be downloaded from CDN. If jQuery is successfully downloaded, jQuery property is added to the window object. If this property is not found then jQuery is not downloaded. So in this case we are writing a script tag to fallback to the local jQuery file.

<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.3.min.js"></script>

<script>

    window.jQuery || document.write("<script src='Scripts/jquery-2.1.3.js'><\/script>");

</script>

**What are jQuery selectors**  
One of the most important concept in jQuery is selectors. jQuery selectors allow you to select and manipulate HTML elements.   
  
**Different selectors in jQuery**  
jQuery selectors allow you to select html elements in the DOM by  
1. Element ID  
2. Element Tag Name  
3. Element Class Name  
4. Element attribute  
5. Element Attribute values and many more   
  
**Id selector in jquery**  
To find an HTML element by ID, use the jQuery #id selector   
  
**Example :** The following example finds button with ID **button1** and attaches the click event handler.

<html>

<head>

    <title></title>

    <script src="jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#button1').click(function () {

                alert('jQuery Tuorial');

            });

        });

    </script>

</head>

<body>

    <input id="button1" type="button" value="Click Me" />

</body>

</html>

**Changes the background colour of the button to yellow**

$(document).ready(function () {

    $('#button1').css('background-color', 'yellow');

});

**Important points to remember about jQuery #id selector**   
  
1. **jQuery #id selector** uses the JavaScript **document.getElementById()** function   
  
2. **jQuery #id selector** is the most efficient among all jQuery selectors. If you know the id of an element that you want to find, then always use the #id selector.   
  
3. HTML element IDs must be unique on the page. **jQuery #id selector** returns only the first element, if you have 2 or more elements with the same ID.

<html>

<head>

    <title></title>

    <script src="jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#button1').css('background-Color', 'yellow');

        });

    </script>

</head>

<body>

    <input id="button1" type="button" value="Click Me" />

    <input id="button1" type="button" value="Click Button" />

</body>

</html>

4. JavaScript's **document.getElementById()** function throws an error if the element with the given id is not found, where as **jQuery #id selector** will not throw an error. To check if an element is returned by the #id selector use length property.

<html>

<head>

    <title></title>

    <script src="jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            if ($('#button1').length > 0) {

                alert('Element found')

            }

            else {

                alert('Element not found')

            }

        });

    </script>

</head>

<body>

    <input id="button1" type="button" value="Click Me" />

</body>

</html>

5. JavaScript's **document.getElementById()** and **jQuery(#id) selector** are not the same. **document.getElementById()** returns a **raw DOM object** where as **jQuery('#id') selector** returns a jQuery object that wraps the DOM object and provides jQuery methods. This is the reason you are able to call jQuery methods like css(), click() on the object returned by jQuery. To get the underlying DOM object from a jQuery object write $('#id')[0]  
  
6. **document.getElementById()** is faster than **jQuery('#id') selector**. Use document.getElementById() over jQuery('#id') selector unless you need the extra functionality provided by the jQuery object.

To select the elements by tag name use **jQuery Element Selector**  
  
**Syntax :** $(element)

$('td') // Selects all td elements

$('div a') // Select all anchor elements that are descendants of div element

$('div, span, a') // Selects all div, span and anchor elements

Alerts the total count of td elements on the page

<html>

<head>

    <title></title>

    <script src="jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            alert($('td').length);

        });

    </script>

</head>

<body>

    <table style="border:1px solid black; border-collapse:collapse">

        <tr>

            <td>C#</td>

            <td>ASP.NET</td>

            <td>SQL Server</td>

        </tr>

        <tr>

            <td>ADO.NET</td>

            <td>jQuery</td>

            <td>JavaScript</td>

        </tr>

        <tr>

            <td>AJAX</td>

            <td>CSS</td>

            <td>HTML</td>

        </tr>

        <tr>

            <td>VB</td>

            <td>Dot NET</td>

            <td>Visual C++</td>

        </tr>

        <tr>

            <td>Oracle</td>

            <td>Java</td>

            <td>J2EE</td>

        </tr>

    </table>

    <br /><br />

    <div>

        DIV 1

        <br />

        <a href="http://pragimtech.com">PragimTech</a>

    </div>

    <br /><br />

    <a href="http://google.com">Google</a>

    <br /><br />

    <div>DIV 2</div>

    <br /><br />

    <span>SPAN 1</span>

    <br /><br />

    <div>DIV 3</div>

</body>

</html>

Selects all the tr elements on the page and changes their background colour to red

<script type="text/javascript">

    $(document).ready(function () {

        $('tr').css('background-Color', 'red');

    });

</script>

Alerts the HTML content of the table

<script type="text/javascript">

    $(document).ready(function () {

        alert($('table').html());

    });

</script>

Alerts the HTML content of each table row

<script type="text/javascript">

    $(document).ready(function () {

        $('table tr').each(function () {

            alert($(this).html());

        });

    });

</script>

Select and changes the background colour of all the div, span and anchor elements

<script type="text/javascript">

    $(document).ready(function () {

        $('div, span, a').css('background-Color', 'yellow');

    });

</script>

Select all anchor elements that are descendants of div element

<html>

<head>

    <title></title>

    <script src="jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('div a').css('background-Color', 'yellow');

        });

    </script>

</head>

<body>

    <div>

        <a href="http://pragimtech.com">PragimTech</a>

    </div>

    <br />

    <a href="http://microsoft.com">Microsoft</a>

</body>

</html>

Changes the background color of even rows to gray and odd rows to yellow on both the tables.

<html>

<head>

    <title></title>

    <script src="jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('tr:even').css('background-Color', 'gray');

            $('tr:odd').css('background-Color', 'yellow');

        });

    </script>

</head>

<body>

    <table id="table1" border="1">

        <tr>

            <td>C#</td>

            <td>ASP.NET</td>

            <td>SQL Server</td>

        </tr>

        <tr>

            <td>ADO.NET</td>

            <td>jQuery</td>

            <td>JavaScript</td>

        </tr>

        <tr>

            <td>AJAX</td>

            <td>CSS</td>

            <td>HTML</td>

        </tr>

        <tr>

            <td>VB</td>

            <td>Dot NET</td>

            <td>Visual C++</td>

        </tr>

        <tr>

            <td>Oracle</td>

            <td>Java</td>

            <td>J2EE</td>

        </tr>

    </table>

    <br />

    <table id="table2" border="1">

        <tr>

            <td>Mark</td>

            <td>Mary</td>

            <td>Mike</td>

        </tr>

        <tr>

            <td>John</td>

            <td>Jade</td>

            <td>Joy</td>

        </tr>

        <tr>

            <td>Rob</td>

            <td>Roy</td>

            <td>Rachel</td>

        </tr>

        <tr>

            <td>Sam</td>

            <td>Sara</td>

            <td>Smith</td>

        </tr>

        <tr>

            <td>Todd</td>

            <td>Tom</td>

            <td>Theo</td>

        </tr>

    </table>

</body>

</html>

To change the background color of even rows to gray and odd rows to yellow just for one of the table, use #id selector along with element selector.

<script type="text/javascript">

    $(document).ready(function () {

        $('#table1 tr:even').css('background-Color', 'gray');

        $('#table1 tr:odd').css('background-Color', 'yellow');

    });

</script>

**Class Elements:**

jQuery class selectors uses JavaScript's native **getElementsByClassName**() function if the browser supports it.

$('.small') // Selects all elements with class small

$('.small,.big') // Selects all elements with class small or big

$('div.small,.big') // Selects div elements with class small and any element with class big

Selects all elements **with class "small"** and sets 5px solid red border

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('.small').css('border', '5px solid red');

        });

    </script>

</head>

<body>

    <span class="small">

        Span 1

    </span>

    <br /><br />

    <div class="small">

        Div 1

    </div>

    <br />

    <span class="big">

        Span 2

    </span>

    <p class="big">This is a paragraph</p>

</body>

</html>

Selects all elements **with class "small" or "big"** and sets 5px solid red border

<script type="text/javascript">

    $(document).ready(function () {

        $('.small, .big').css('border', '5px solid red');

    });

</script>

Selects all elements **with class "small" and all span elements with class "big"** and sets 5px solid red border

<script type="text/javascript">

    $(document).ready(function () {

        $('.small, span.big').css('border', '5px solid red');

    });

</script>

Selects all elements with **class small that are nested in a an element with id=div2** and sets 5px solid red border

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#div2 .small').css('border', '5px solid red');

        });

    </script>

</head>

<body>

    <div id="div1" class="small">

        DIV 1

    </div>

    <br />

    <div id="div2">

        Div 2

        <br />

        <div class="small">

            DIV 3

        </div>

        <br />

        <span class="small">

            SPAN

        </span>

    </div>

</body>

</html>

Selects all elements **with class small** and sets 5px solid red border. Notice div1 has 2 classes - small and big.

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('.small').css('border', '5px solid red');

        });

    </script>

</head>

<body>

    <div class="small big">

        DIV 1

    </div>

    <br />

    <div class="small">

        DIV 2

    </div>

</body>

</html>

Selects all elements **that has both the classes - small and big**. There should be no space between the class names.

<script type="text/javascript">

    $(document).ready(function () {

        $('.small.big').css('border', '5px solid red');

    });

</script>

If you have a **space between the two class names then we are trying to find descendants**, i.e. find elements with class big that are descendants of an element with class small.

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('.small .big').css('border', '5px solid red');

        });

    </script>

</head>

<body>

    <div class="small big">

        DIV 1

    </div>

    <br />

    <div class="small">

        DIV 2

        <div class="big">

            DIV 3

        </div>

    </div>

</body>

</html>

**Another way to selects all elements that has both the classes - small and big is by using filter method**. But this approach is slower because it will first create a list of objects with class "small" and then removes elements that does not have class "big"

<script type="text/javascript">

    $(document).ready(function () {

        $('.small').filter('.big').css('border', '5px solid red');

    });

</script>

**Attribute Selector:**

**Syntax :**  
$('[attribute]')  
$('[attribute="value"]')

$('[title]') // Selects all elements that have title attribute

$('div[title]') // Selects all div elements that have title attribute

$('[title="divTitle"]') // Selects all elements that have title attribute value - divTitle

$('div[title="divTitle"]') // Selects all div elements that have title attribute value - divTitle

**Selects all elements with title attribute** and sets 5px solid red border

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('[title]').css('border', '5px solid red');

        });

    </script>

</head>

<body>

    <div title="div1Title">

        DIV 1

    </div>

    <br />

    <div title="div2Title">

        DIV 2

    </div>

    <p title="pTitle">

        This is a paragraph

    </p>

    <span title="div1Title">

        SAPN 1

    </span>

    <br /><br />

    <span>

        SPAN 2

    </span>

</body>  
</html>

**Selects all div elements with title attribute** and sets 5px solid red border

<script type="text/javascript">

    $(document).ready(function () {

        $('div[title]').css('border', '5px solid red');

    });

</script>

**Selects all elements with title attribute value - div1Title**, and sets 5px solid red border

<script type="text/javascript">

    $(document).ready(function () {

        $('[title="div1Title"]').css('border', '5px solid red');

    });

</script>

**Selects all div elements with title attribute value - div1Title**, and sets 5px solid red border

<script type="text/javascript">

    $(document).ready(function () {

        $('div[title="div1Title"]').css('border', '5px solid red');

    });

</script>

**Selects all div elements with both title and style attributes**, and sets 5px solid black border

<html>

<head>

    <title></title>

    <script src="Scripts/jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('div[title][style]').css('border', '5px solid black');

        });

    </script>

</head>

<body>

    <div title="divTitle" style="background-color:red">

        Red DIV

    </div>

    <br />

    <div title="divTitle" style="background-color:green">

        Green DIV

    </div>

    <br />

    <div title="divTitle">

        Normal Div

    </div>

    <br />

    <div>

        Normal Div without any attributes

    </div>

</body>

</html>

Selects all div elements with **title attribute value - divTitle**, and **style attribute value - background-color:red**, and sets 5px solid black border

<script type="text/javascript">

    $(document).ready(function () {

        $('div[title="divTitle"][style="background-color:red"]')

            .css('border', '5px solid black');

    });

</script>

**Selects all div elements with either title or style attributes**, and sets 5px solid black border

<script type="text/javascript">

    $(document).ready(function () {

        $('div[title],[style]').css('border', '5px solid black');

    });

</script>

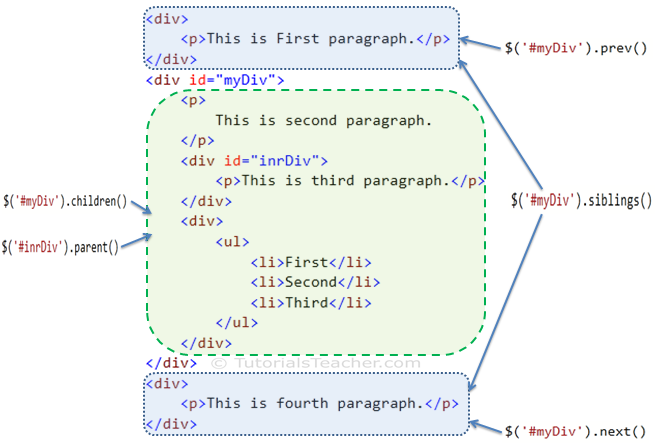
# Traversing DOM Elements using jQuery

The jQuery library includes various methods to traverse DOM elements in a DOM hierarchy.

The following table lists jQuery methods for traversing DOM elements.

| **jQuery Methods** | **Description** |
| --- | --- |
| children() | Get all the child elements of the specified element(s) |
| each() | Iterate over specified elements and execute specified call back function for each element. |
| find() | Get all the specified child elements of each specified element(s). |
| first() | Get the first occurrence of the specified element. |
| next() | Get the immediately following sibling of the specified element. |
| parent() | Get the parent of the specified element(s). |
| prev() | Get the immediately preceding sibling of the specified element. |
| siblings() | Get the siblings of each specified element(s) |

The following figure shows how the jQuery traversing methods get DOM elements.

[](http://www.tutorialsteacher.com/Content/images/jquery/jq-traversing-methods.png)Traversing DOM Elements

Let's look at some of the important jQuery traversing methods.

## jQuery each() Method

The jQuery each() method iterates over each specified element (specified using selector) and executes callback function for each element.

Syntax:

$('selector expression').each(callback function);

To begin, specify a selector to get the reference of elements and call each() method with callback function, which will be executed for each element.

Example: jQuery each() method

$('p').each(function (index) {

alert('index' + index + ', text: ' + $(this).text());

});

<div>

<p>This is First paragraph.</p>

</div>

<div id="myDiv">

<p>

This is second paragraph.

</p>

<div id="inrDiv">

<p>This is third paragraph.</p>

</div>

<div>

<ul>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ul>

</div>

</div>

<div>

<p>This is fourth paragraph.</p>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-61)

Result:

Index:0, text: This is first paragraph.

Index:1, text: This is second paragraph.

Index:2, text: This is third paragraph.

Index:3, text: This is fourth paragraph.

## jQuery children() Method

The jQuery children() method get the child element of each element specified using selector expression.

Syntax:

$('selector expression').children();

http://www.tutorialsteacher.com/Content/images/tips.pngElements returned from children() method can be iterated using each() method.

First, specify a selector to get the reference of an element(s) and call children() method to get all the child elements.

Example: jQuery children() method

$('#myDiv').children().each(function (index) {

alert('Index: ' + index + ', html: ' + $(this).html());

});

<div>

<p>This is First paragraph.</p>

</div>

<div id="myDiv">

<p>

This is second paragraph.

</p>

<div id="inrDiv">

<p>This is third paragraph.</p>

</div>

<div>

<ul>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ul>

</div>

</div>

<div>

<p>This is fourth paragraph.</p>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-62)

Result:

Index:0, html: <p>

This is second paragraph.

</p>

Index:1, html: <div id="inrDiv">

<p>This is third paragraph.</p>

</div>

Index:2, html: <div>

<ul>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ul>

</div>

## jQuery find() Method

The jQuery find() method returns all the matching child elements of specified element(s).

Syntax:

$('selector expression').find('selector expression to find child elements');

Specify a selector to get the reference of an element(s) whose child elements you want to find and then call find() method with selector expression to get all the matching child elements. You can iterate child elements using each method.

Example: jQuery find() method

$('#myDiv').find('p').each(function(index){

alert('index' + index + ', text: ' + $(this).text());

});

<div>

<p>This is First paragraph.</p>

</div>

<div id="myDiv">

<p>

This is second paragraph.

</p>

<div id="inrDiv">

<p>This is third paragraph.</p>

</div>

<div>

<ul>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ul>

</div>

</div>

<div>

<p>This is fourth paragraph.</p>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-63)

Result:

Index:0, text: This is second paragraph.

Index:1, text: This is third paragraph.

## jQuery next() Method

The jQuery next() method gets the immediately following sibling of the specified element.

Syntax:

$('selector expression').next();

Specify a selector to get the reference of an element of which you want to get next element and then call next() method.

Example: jQuery next() method

alert('Next element to #myDiv: ' + $('#myDiv').next().html());

alert('Next element to #inrDiv: ' + $('#inrDiv').next().html());

<div>

<p>This is First paragraph.</p>

</div>

<div id="myDiv">

<p>

This is second paragraph.

</p>

<div id="inrDiv">

<p>This is third paragraph.</p>

</div>

<div>

<ul>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ul>

</div>

</div>

<div>

<p>This is fourth paragraph.</p>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-64)

Result:

Next element to #myDiv: <div>

<p>This is fourth paragraph.</p>

</div>

Next element to #inrDiv: <ul>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ul>

## jQuery parent() Method

The jQuery parent() method gets the immediate parent element of the specified element.

Syntax:

$('selector expression').parent();

Specify a selector to get the reference of an element of which you want to get the parent element and then call parent() method.

Example: jQuery parent() method

alert('Parent element of #inrDiv: ' + $('#inrDiv').parent().html());

<div>

<p>This is First paragraph.</p>

</div>

<div id="myDiv">

<p>

This is second paragraph.

</p>

<div id="inrDiv">

<p>This is third paragraph.</p>

</div>

<div>

<ul>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ul>

</div>

</div>

<div>

<p>This is fourth paragraph.</p>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-65)

Result:

Parent element of #inrDiv: <p>

This is second paragraph.

</p>

<div id="inrDiv">

<p>This is third paragraph.</p>

</div>

<div>

<ul>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ul>

</div>

## jQuery siblings() Method

The jQuery siblings()method gets all siblings of the specified DOM element(s).

Syntax:

$('selector expression').siblings();

Specify a selector to get the reference of an element of which you want to get the siblings and call siblings() method.

Tips: you can iterate sibling elements using each() method.

Example: jQuery siblings() method

$('#myDiv').siblings().css({"color": "green", "border": "2px solid green"});

<div>

<p>This is First paragraph.</p>

</div>

<div id="myDiv">

<p>

This is myDiv.

</div>

<div>

<p>This is second paragraph.</p>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-66)

Visit [Traversing methods reference](http://www.tutorialsteacher.com/jquery/jquery-dom-traversing-methods-reference) to know all the traversing DOM methods.

http://www.tutorialsteacher.com/Content/images/bulb-glow.pngPoints to Remember :

1. The jQuery traversing methods allow you to iterate DOM elements in a DOM hierarchy.
2. Use the selector to get the reference of an element(s) and then call jQuery traversing methods to edit it.
3. Important DOM manipulation methods: each(), children(), find(), first(), parent(), next(), previous(), siblings() etc.

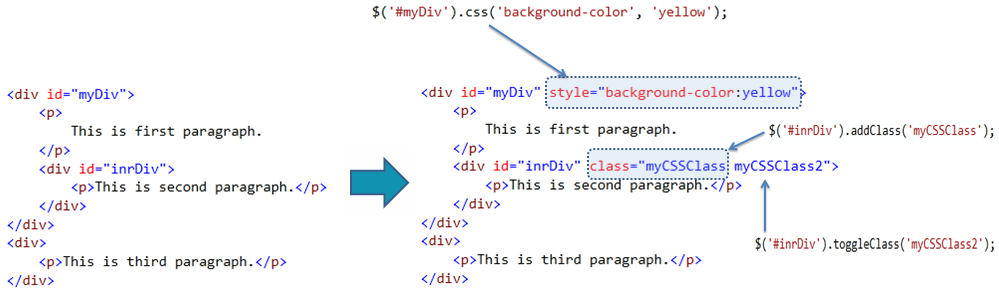
# **CSS Manipulation using jQuery**

The jQuery library includes various methods to manipulate style properties and CSS class of DOM element(s).

The following table lists jQuery methods for styling and css manipulation.

| jQuery Methods | Description |
| --- | --- |
| css() | Get or set style properties to the specified element(s). |
| addClass() | Add one or more class to the specified element(s). |
| hasClass() | Determine whether any of the specified elements are assigned the given CSS class. |
| removeClass() | Remove a single class, multiple classes, or all classes from the specified element(s). |
| toggleClass() | Toggles between adding/removing classes to the specified elements |

The following figure shows how jQuery methods changes style and css class of the DOM elements.

[](http://www.tutorialsteacher.com/Content/images/jquery/jq-css-manipulation.png)jQuery Methods for Style & CSS Manipulation

Let's have an overview of important jQuery methods for style and css manipulation.

## **jQuery css() Method**

The jQuery css() method gets or sets style properties to the specified element(s).

Syntax:

$('selector expression').css('style property name','value');

$('selector expression').css({

'style property name':'value',

});

Specify a selector to get the reference of an elements to which you want to set the style property and then call css() method with style property name and value parameter. You can also set multiple style properties by passing JSON object with 'style property name':'value'.

Example: jQuery css() Method

$('#myDiv').css('background-color','yellow');

$('p').css({'background-color': 'red','width':'400px'});

$('#myDiv').css('background-color'); // returns rgb(255,255,0) for yellow color

<div id="myDiv">

<p>This is first paragraph.</p>

</div>

<div>

<p>This is second paragraph.</p>

</div>

<div >

<p>This is third paragraph.</p>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-71)

In the above example, we set background-color of #myDiv and font styles to all <p> elements. The same way, we can get value of any style properties using css() method by specifying property name as first parameter.

## **jQuery addClass() method**

The jQuery addClass() method adds single or multiple css class to the specified element(s).

Syntax:

$('selector expression').addClass('css class name');

First specify a selector to get the reference of an elements to which you want to set the css property and then call addClass() method with one or multiple class names as a string parameter. Multiple class names must be separated by space.

Example: jQuery addClass() Method

$('#myDiv').addClass('yellowDiv');

$('p').addClass('impPrg');

<div id="myDiv">

<p>

This is first paragraph.

</p>

</div>

<div>

<p>This is second paragraph.</p>

</div>

<div >

<p>This is third paragraph.</p>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-72)

In the above example, we set css class of individual <div> element (#myDiv) as well as multiple <p> elements using addClass() method.

## **jQuery toggleClass() Method**

The jQuery toggleClass() method toggles between adding/removing classes to the specified elements.

Syntax:

$('selector expression').toggleClass('css class name');

Specify a selector to get the reference of an elements to which you want to toggle css classes and then call toggleClass() method with css class name as a string parameter.

Example: jQuery toggleClass() Method

$('#myDiv').toggleClass('redDiv');

<div id="myDiv" class="yellowDiv">

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-73)

In the above example, css class yellowDiv will be first added into div element and then removed. Thus, css class will be added or removed consecutively.

Visit [Manipulation methods reference](http://www.tutorialsteacher.com/jquery/jquery-dom-methods-reference) to know all the CSS manipulation methods in jQuery.

http://www.tutorialsteacher.com/Content/images/bulb-glow.pngPoints to Remember :

1. The jQuery CSS methods allow you to manipulate CSS class or style properties of DOM elements.
2. Use the selector to get the reference of an element(s) and then call jQuery css methods to edit it.
3. Important DOM manipulation methods: css(), addClass(), hasClass(), removeClass(), toggleClass() etc.

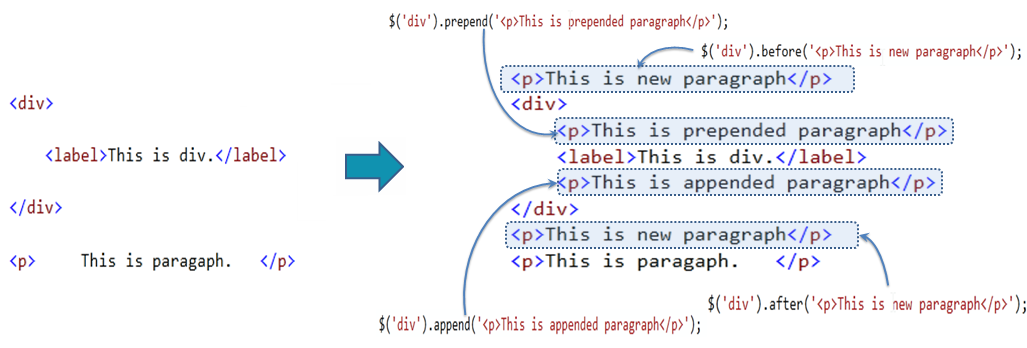
# DOM Manipulation Methods in jQuery

jQuery provides various methods to add, edit or delete DOM element(s) in the HTML page.

The following table lists some important methods to add/remove new DOM elements.

| **Method** | **Description** |
| --- | --- |
| append() | Inserts content to the end of element(s) which is specified by a selector. |
| before() | Inserts content (new or existing DOM elements) before an element(s) which is specified by a selector. |
| after() | Inserts content (new or existing DOM elements) after an element(s) which is specified by a selector. |
| prepend() | Insert content at the beginning of an element(s) specified by a selector. |
| remove() | Removes element(s) from DOM which is specified by selector. |
| replaceAll() | Replace target element(s) with specified element. |
| wrap() | Wrap an HTML structure around each element which is specified by selector. |

The following figure shows how the DOM manipulation methods add new elements.

[](http://www.tutorialsteacher.com/Content/images/jquery/manipulation-methods.png)DOM Manipulation Methods

Let's have a quick overview of important DOM manipulation methods.

## jQuery after() Method

The jQuery after() method inserts content (new or existing DOM elements) after target element(s) which is specified by a selector.

Syntax:

$('selector expression').after('content');

First of all, specify a selector to get the reference of target element(s) after which you want to add the content and then call after() method. Pass the content string as a parameter. Content string can be any valid HTML element.

Example: jQuery after() Method

$('#div1').after('<div style="background-color:yellow"> New div </div>');

<div id="div1">div 1

</div>

<div id="div2">div 2

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-31)

Result:

<div id="div1">div 1

</div>

<div style="background-color:yellow"> New div </div>

<div id="div2">div 2

</div>

## jQuery before() Method

The jQuery before() method inserts content (new or existing DOM elements) before target element(s) which is specified by a selector.

Syntax:

$('selector expression').before('content');

Specify a selector to get the reference of target element(s) before which you want to add the content and then call before() method. Pass the content string that can be any valid HTML element as parameter.

Example: jQuery before() Method

$('#div1').before('<div style="background-color:yellow"> New div </div>');

<div id="div1">div 1

</div>

<div id="div2">div 2

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-32)

Result:

<div style="background-color:yellow"> New div </div>

<div id="div1">div 1

</div>

<div id="div2">div 2

</div>

## jQuery append() Method

The jQuery append() method inserts content to the end of target element(s) which is specified by a selector.

Syntax:

$('selector expression').append('content');

First specify a selector expression to get the reference of an element(s) to which you want to append content, then call append() method and pass content string as a parameter.

Example: jQuery append() Method

$('p').append('World!');

<p>Hello </p>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-33)

Result:

<p>Hello World!</p>

## jQuery prepend() Method

The jQuery prepend() method inserts content at the beginning of an element(s) specified by a selector.

Syntax:

$('selector expression').prepend('content');

First specify a selector expression to get the reference of an element(s) to which you want to prepend the content, then call prepend() method and pass content string as a parameter.

Example: jQuery prepend() Method

$('div').prepend('<p>This is prepended paragraph</p>');

<div>

<label>This is div.</label>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-34)

Result:

<div>

<p>This is prepended paragraph</p>

<label>This is div.</label>

</div>

## jQuery remove() Method

The jQuery remove() method removes element(s) as specified by a selector.

Syntax:

$('selector expression').remove();

First specify a selector expression to get the reference of an element(s) which you want to remove from the document and then call remove() method.

Example: jQuery remove() Method

$('label').remove();

<div>This is div.

<label>This is label.</label>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-35)

Result:

<div>

This is div.

</div>

## jQuery replaceAll() Method

The jQuery replaceAll() method replaces all target elements with specified element(s).

Syntax:

$('content string').replaceAll('selector expression');

Here, syntax is different. First specify a content string as replacement element(s) and then call replaceAll() method with selector expression to specify a target element(s).

Example: jQuery replaceAll() Method

$('<span>This is span</span>').replaceAll('p');

<div>

<p>This is paragraph.</p>

</div>

<p>This is another paragraph.</p>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-36)

Result:

<div>

<span>This is span</span>

</div>

<span>This is span</span>

## jQuery wrap() Method

The jQuery wrap() method wrap each target element with specified content element.

Syntax:

$('selector expression').wrap('content string');

Specify a selector to get target elements and then call wrap method and pass content string to wrap the target element(s).

Example: jQuery wrap() Method

$('span').wrap('<p></p>');

<div>

<span>This is span.</span>

</div>

<span>This is span.</span>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-37)

Result:

<div>

<p> <span>This is span.</span></p>

</div>

<p><span>This is span.</span></p>

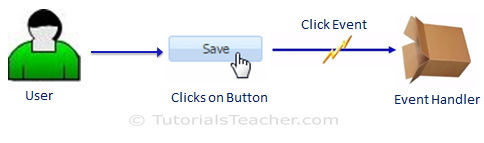
http://www.tutorialsteacher.com/Content/images/bulb-glow.pngPoints to Remember :

1. jQuery DOM manipulation methods allows you to add, edit or delete DOM element(s) in the web page.
2. Use the selector to get the reference of an element(s) and then call manipulate methods to edit it.
3. Important DOM manipulation methods: append(), prepend(), before(), after(), remove(), replaceAll(), wrap() etc.

# jQuery Events

You will learn about jQuery events in this section.

In most web applications, the user does some action to perform an operation. For example, user clicks on save button to save the edited data in a web page. Here, clicking on the button is a user's action, which triggers click event and click event handler (function) saves data.

[](http://www.tutorialsteacher.com/Content/images/jquery/jquery-event.png)Event

## jQuery Event Methods

The jQuery library provides methods to handle DOM events. Most jQuery methods correspond to native DOM events.

The following table lists all jQuery methods and corresponding DOM events.

| **Category** | **jQuery Method** | **DOM Event** |
| --- | --- | --- |
| Form events | blur | onblur |
|  | change | onchange |
|  | focus | onfocus |
|  | focusin | onfocusin |
|  | select | onselect |
|  | submit | onsubmit |
| Keyboard events | keydown | onkeydown |
|  | keypress | onkeypress |
|  | keyup | onkeyup |
|  | focusout |  |
| Mouse events | click | onclick |
|  | dblclick | ondblclick |
|  | focusout |  |
|  | hover |  |
|  | mousedown | onmousedown |
|  | mouseenter | onmouseenter |
|  | mouseleave | onmouseleave |
|  | mousemove | onmousemove |
|  | mouseout | onmouseout |
|  | mouseover | onmouseover |
|  | mouseup | onmouseup |
|  | Toggle |  |
| Browser events | Error | onerror() |
|  | Resize | onresize |
|  | Scroll | onscroll |
| Document loading | Load | onload |
|  | Ready |  |
|  | Unload | onunload |

## Event Handling

To handle DOM events using jQuery methods, first get the reference of DOM element(s) using jQuery selector and invoke appropriate jQuery event method.

The following example shows how to handle button click event.

Example:Handle Button Click Event

$('#saveBtn').click(function () {

alert('Save button clicked');

});

<input type="button" value="Save" id="saveBtn" />

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-91)

In the above example, we first use id selector to get a reference of 'Save' button and then call click method. We have specified handler function as a callback function, which will be called whenever click event of Save button is triggered.

### Event Object

jQuery passes an [event object](http://api.jquery.com/category/events/event-object/) to every event handler function. The event object includes important properties and methods for cross-browser consistency e.g. target, pageX, pageY, relatedTarget etc.

Example: jQuery Event Object

$('#saveBtn').click(function (eventObj) {

alert('X =' + eventObj.pageX + ', Y =' + eventObj.pageY);

});

<input type="button" value="Save" id="saveBtn" />

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-92)

### *this* Keyword in Event Handler

**this** keyword in an event handler represents a DOM element which raised an event.

Example: this in Event Handler

$(':button').click(function (eventObj) {

alert(this.value + ' ' + this.type + ' clicked');

});

<input type="button" value="Save" id="saveBtn" />

<input type="button" value="Delete" id="delBtn" />

<input type="button" value="Clear" id="clearBtn" />

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-93)

### Hover Events

jQuery provides various methods for mouse hover events e.g. mouseenter, mouseleave, mousemove, mouseover, mouseout and mouseup.

Example: Hover Events

$('#myDiv').mouseenter(function (data) {

$(this).css('background-color','green');

});

$('#myDiv').mouseleave(function (data) {

$(this).css('background-color','red');

});

<div id="myDiv" style="width:100px;height:100px">

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-94)

You can use **hover()** method instead of handling mouseenter and mouseleave events separately.

Example: hover() Method

$('#myDiv').hover(function () {

$(this).css('background-color','green');

},

function () {

$(this).css('background-color','red');

});

<div id="myDiv" style="width:100px;height:100px">

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-95)

## Event Binding using on()

jQuery allows you to attach an event handler for one or more events to the selected elements using **on** method.

Internally all of the shorthand methods uses on() method. The on() method gives you more flexibility in event binding.

Syntax:

on(types, selector, data, fn )

* Types = One or more space-separated event types and optional namespaces
* Selector = selector string
* Data = data to be passed to the handler in event.data when an event is triggered.
* Fn = A function to execute when the event is triggered.

Example: Event Binding using on

$('#saveBtn').on('click',function () {

alert('Save Button clicked');

});

<input type="button" value="Save" id="saveBtn" />

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-96)

You can use selector to filter the descendants of the selected elements that trigger the event.

Example: Event Binding using on

$('#myDiv').on('click',':button', function () {

alert('Button clicked');

});

<div id="myDiv" >

<input type="button" value="Save" id="saveBtn" />

<input type="button" value="Add" id="addBtn" />

</div>

<input type="button" value="Delete" id="delBtn" />

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-97)

In the above example, we specify ':button' selector. So click event triggered by buttons in< div> tag whose id is myDiv, will only be handled.

## Binding Multiple Events

You can also specify multiple event types separated by space.

Example: Multiple Events Binding

$( 'myDiv' ).on('mouseenter mouseleave', function() {

$(this).text('The mouse entered or left from the div' );

});

<div id="myDiv" style="width:100px;height:100px">

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-98)

### Specify Named Function as Event Handler

You can create separate functions and specify that as a handler. This is useful if you want to use the same handler for different events or events on different elements.

Example:Binding Named Function to Event

var mouseHandler = function() {

alert( "The mouse entered" );

};

$('#myDiv').on('mouseenter', mouseHandler);

<div id="myDiv" style="width:100px;height:100px">

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-99)

jQuery on() method is replacement of live() and delegate() method.

## Event Bubbling

The following example demonstrates event bubbling in jQuery.

Example:Event Bubbling

$('div').click(function (event) {

alert( event.target.tagName + ' clicked');

});

<div>

<p>

<span>This is span.</span>

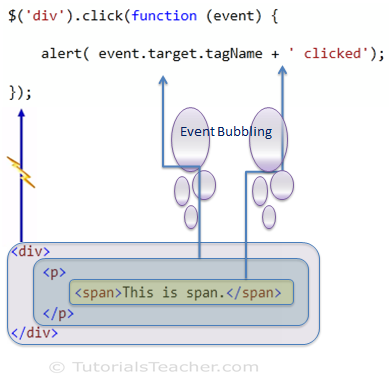
</p>

</div>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-901)

As you can see in the above example, we have handled click event of <div> element in jQuery. So if you click on div, it will display alert message 'DIV clicked'. However, if you click on span, still it will popup alert message SPAN clicked even though we have not handled click event of <span>. This is called event bubbling. Event bubbles up to the **document** level in DOM hierarchy till it finds it.

The following figure illustrates event bubbling.

[](http://www.tutorialsteacher.com/Content/images/jquery/jquery-event-bubbling.png)jQuery Event Bubbling

Visit [event methods reference](http://www.tutorialsteacher.com/jquery/jquery-event-reference) to know about all the event methods in jQuery.

http://www.tutorialsteacher.com/Content/images/bulb-glow.pngPoints to Remember :

1. The jQuery event methods allow you to attach event handler or fire native DOM events.
2. Use the selector to get the reference of an element(s) and then call jQuery event methods to fire it or attach an event handler.
3. Important DOM manipulation methods: click(), dblClick(), change(), submit(), keyup(), keydown(), mouseenter(), mouseleave(), hover() etc.