## 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 featurerich 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 put downloaded **jquery-2.1.3.min.js** file in a directory of your website, e.g. /jquery.

## 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.

<!doctype html>

<html>

<head>

<title>The jQuery Example</title>

<script src="https://www.tutorialspoint.com/jquery/jquery-3.6.0.js"></script>

<script>

$(document).ready(function() {

$("div").click(function() {alert("Hello, world!");});

});

</script>

</head>

<body>

<div>Click on this to see a dialogue box.</div>

</body>

</html>

## How to Use Custom Scripts?

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

/\* Filename: custom.js \*/

$(document).ready(function() {

$("div").click(function() {

alert("Hello, world!");

});

});

Let's keep this file in /jquery directory and then we can include custom.js file in our HTML file as follows.

<!doctype html>

<html>

<head>

<title>The jQuery Example</title>

<script src="https://www.tutorialspoint.com/jquery/jquery-3.6.0.js"></script>

<script src="https://www.tutorialspoint.com/jquery/custom.js"></script>

</head>

<body>

<div>Click on this to see a dialogue box.</div>

</body>

</html>

## Using Multiple Libraries

You can use multiple libraries all together without conflicting each others. For example, you can use jQuery and MooTooljavascript libraries together. You can check [jQuery noConflict](https://www.tutorialspoint.com/jquery/jquery-noconflict.htm) Method for more detail.

A jQuery Selector is a function which makes use of expressions to find out matching elements from a DOM based on the given criteria.

Selectors are used to select one or more HTML elements using jQuery.

Once an element is selected then we can perform various operations on that selected element.

The $() factory function

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

|  |  |
| --- | --- |
| **Sr.No.** | **Selector & Description** |
| 1 | **Tag Name**  Represents a tag name available in the DOM. For example **$('p')**selects all paragraphs <p> in the document. |
| 2 | **Tag ID**  Represents a tag available with the given ID in the DOM. For example **$('#some-id')** selects the single element in the document that has an ID of some-id. |
| 3 | **Tag Class**  Represents a tag available with the given class in the DOM. For example **$('.some-class')** selects all elements in the document that have a class of some-class. |

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

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 jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<script type="text/javascript" language="javascript">

$(document).ready(function(){

$(".myclass").css("background-color","yellow");

});

</script>

</head>

<body>

<div>

<pclass="myclass">This is a paragraph.</p>

<pid="myid">This is second paragraph.</p>

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

</div>

</body>

</html>

This will produce following result −



## 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.

|  |  |
| --- | --- |
| **Sr.No.** | **Selector & Description** |
| 1 | [**Name**](https://www.tutorialspoint.com/jquery/selector-element-name.htm)  Selects all elements which match with the given element **Name**. |
| 2 | [**#ID**](https://www.tutorialspoint.com/jquery/selector-element-id.htm)  Selects a single element which matches with the given **ID**. |
| 3 | [**.Class**](https://www.tutorialspoint.com/jquery/selector-element-class.htm)  Selects all elements which match with the given **Class**. |
| 4 | [**Universal (\*)**](https://www.tutorialspoint.com/jquery/selector-universal.htm)  Selects all elements available in a DOM. |
| 5 | [**Multiple Elements E, F, G**](https://www.tutorialspoint.com/jquery/selector-multiple-elements.htm)  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**.   Multipleelement.html  <html>  <head>  <title>The Selecter Example</title>  <scripttype="text/javascript"  src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">  </script>  <scripttype="text/javascript"language="javascript">  $(document).ready(function(){  $(".big, #div3").css("background-color","yellow");  });  </script>  </head>    <body>  <divclass="big"id="div1">  <p>This is first division of the DOM.</p>  </div>  <divclass="medium"id="div2">  <p>This is second division of the DOM.</p>  </div>  <divclass="small"id="div3">  <p>This is third division of the DOM</p>  </div>  </body>  </html> |

Here, you have different type of other useful selectors −

|  |  |
| --- | --- |
| **Sr.No.** | **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 *specialClass*. |
| 5 | **$("li:not(.myclass)")**  Selects all elements matched by <li> that do not have class = "myclass". |
| 6 | **$("a#specialID.specialClass")**  This selector matches links with an id of *specialID* and a class of *specialClass*. |
| 7 | **$("p a.specialClass")**  This selector matches links with a class of *specialClass* declared within <p> elements. |
| 8 | **$("ulli:first")**  This selector gets only the first <li> element of the <ul>. |
| 9 | **$("#container p")**  Selects all elements matched by <p> that are descendants of an element that has an id of *container*. |
| 10 | **$("li > ul")**  Selects all elements matched by <ul> that are children of an element matched by <li> |
| 11 | **$("strong + em")**  Selects all elements matched by <em> that immediately follow a sibling element matched by <strong>. |
| 12 | **$("p ~ ul")**  Selects all elements matched by <ul> that follow a sibling element matched by <p>. |
| 13 | **$("code, em, strong")**  Selects all elements matched by <code> or <em> or <strong>. |
| 14 | **$("p strong, .myclass")**  Selects all elements matched by <strong> that are descendants of an element matched by <p> as well as all elements that have a class of *myclass*. |
| 15 | **$(":empty")**  Selects all elements that have no children. |
| 16 | **$("p:empty")**  Selects all elements matched by <p> that have no children. |
| 17 | **$("div[p]")**  Selects all elements matched by <div> that contain an element matched by <p>. |
| 18 | **$("p[.myclass]")**  Selects all elements matched by <p> that contain an element with a class of *myclass*. |
| 19 | **$("a[@rel]")**  Selects all elements matched by <a> that have a rel attribute. |
| 20 | **$("input[@name = myname]")**  Selects all elements matched by <input> that have a name value exactly equal to *myname.* |
| 21 | **$("input[@name^=myname]")**  Selects all elements matched by <input> that have a name value beginning with *myname*. |
| 22 | **$("a[@rel$=self]")**  Selects all elements matched by <a> that have **rel** attribute value ending with *self*. |
| 23 | **$("a[@href\*=domain.com]")**  Selects all elements matched by <a> that have an href value containing domain.com. |
| 24 | **$("li:even")**  Selects all elements matched by <li> that have an even index value. |
| 25 | **$("tr:odd")**  Selects all elements matched by <tr> that have an odd index value. |
| 26 | **$("li:first")**  Selects the first <li> element. |
| 27 | **$("li:last")**  Selects the last <li> element. |
| 28 | **$("li:visible")**  Selects all elements matched by <li> that are visible. |
| 29 | **$("li:hidden")**  Selects all elements matched by <li> that are hidden. |
| 30 | **$(":radio")**  Selects all radio buttons in the form. |
| 31 | **$(":checked")**  Selects all checked box in the form. |
| 32 | **$(":input")**  Selects only form elements (input, select, textarea, button). |
| 33 | **$(":text")**  Selects only text elements (input[type = text]). |
| 34 | **$("li:eq(2)")**  Selects the third <li> element. |
| 35 | **$("li:eq(4)")**  Selects the fifth <li> element. |
| 36 | **$("li:lt(2)")**  Selects all elements matched by <li> element before the third one; in other words, the first two <li> elements. |
| 37 | **$("p:lt(3)")**  selects all elements matched by <p> elements before the fourth one; in other words the first three <p> elements. |
| 38 | **$("li:gt(1)")**  Selects all elements matched by <li> after the second one. |
| 39 | **$("p:gt(2)")**  Selects all elements matched by <p> after the third one. |
| 40 | **$("div/p")**  Selects all elements matched by <p> that are children of an element matched by <div>. |
| 41 | **$("div//code")**  Selects all elements matched by <code>that are descendants of an element matched by <div>. |
| 42 | **$("//p//a")**  Selects all elements matched by <a> that are descendants of an element matched by <p> |
| 43 | **$("li:first-child")**  Selects all elements matched by <li> that are the first child of their parent. |
| 44 | **$("li:last-child")**  Selects all elements matched by <li> that are the last child of their parent. |
| 45 | **$(":parent")**  Selects all elements that are the parent of another element, including text. |
| 46 | **$("li:contains(second)")**  Selects all elements matched by <li> that contain the text second. |

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# jQuery - Attributes

Some of the most basic components we can manipulate when it comes to DOM elements are the properties and attributes assigned to those elements.

Most of these attributes are available through JavaScript as DOM node properties.

Some of the more common properties are −

* className
* tagName
* id
* href
* title
* rel
* src

Consider the following HTML markup for an image element −

<img id = "imageid" src = "image.gif" alt = "Image" class = "myclass" title = "This is an image"/>

In this element's markup, the tag name is img, and the markup for id, src, alt, class, and title represents the element's attributes, each of which consists of a name and a value.

jQuery gives us the means to easily manipulate an element's attributes and gives us access to the element so that we can also change its properties.

## Get Attribute Value

The **attr()** method can be used to either fetch the value of an attribute from the first element in the matched set or set attribute values onto all matched elements.

### Example

Following is a simple example which fetches title attribute of <em> tag and set <div id = "divid"> value with the same value

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

Src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/

jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

var title = $("em").attr("title");

$("#divid").text(title);

});

</script>

</head>

<body>

<div>

<em title="Bold and Brave">This is first paragraph.

</em>

<p id="myid">This is second paragraph.</p>

<div id="divid"></div>

</div>

</body>

</html>

This will produce following result −

## 

## Set Attribute Value

The **attr(name, value)** method can be used to set the named attribute onto all elements in the wrapped set using the passed value.

### Example

Following is a simple example which set **src** attribute of an image tag to a correct location −

<html>

<head>

<title>The jQuery Example</title>

<basehref="https://www.tutorialspoint.com"/>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("#myimg").attr("src","/jquery/images/jquery.jpg");

});

</script>

</head>

<body>

<div>

<imgid="myimg"src="/images/jquery.jpg"alt="Sample image"/>

</div>

</body>

</html>

This will produce following result −

## 

## Applying Styles

The **addClass( classes )** method can be used to apply defined style sheets onto all the matched elements.

You can specify multiple classes separated by space.

### Example

Following is a simple example which sets **class** attribute of a para <p> tag −

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("em").addClass("selected");

$("#myid").addClass("highlight");

});

</script>

<style>

.selected {color:red;}

.highlight {background:yellow;}

</style>

</head>

<body>

<emtitle="Bold and Brave">This is first paragraph.</em>

<pid="myid">This is second paragraph.</p>

</body>

</html>

This will produce following result −

## 

## Attribute Methods

Following table lists down few useful methods which you can use to manipulate attributes and properties −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | [**attr( properties )**](https://www.tutorialspoint.com/jquery/attr-properties.htm)  Set a key/value object as properties to all matched elements. |
| 2 | [**attr( key, fn )**](https://www.tutorialspoint.com/jquery/attr-key-function.htm)  Set a single property to a computed value, on all matched elements. |
| 3 | [**removeAttr( name )**](https://www.tutorialspoint.com/jquery/attr-remove-attribute.htm)  Remove an attribute from each of the matched elements. |
| 4 | [**hasClass( class )**](https://www.tutorialspoint.com/jquery/attr-has-class.htm)  Returns true if the specified class is present on at least one of the set of matched elements. |
| 5 | [**removeClass( class )**](https://www.tutorialspoint.com/jquery/attr-remove-class.htm)  Removes all or the specified class(es) from the set of matched elements. |
| 6 | [**toggleClass( class )**](https://www.tutorialspoint.com/jquery/attr-toggle-class.htm)  Adds the specified class if it is not present, removes the specified class if it is present. |
| 7 | [**html( )**](https://www.tutorialspoint.com/jquery/attr-html.htm)  Get the html contents (innerHTML) of the first matched element. |
| 8 | [**html( val )**](https://www.tutorialspoint.com/jquery/attr-html-val.htm)  Set the html contents of every matched element. |
| 9 | [**text( )**](https://www.tutorialspoint.com/jquery/attr-text.htm)  Get the combined text contents of all matched elements. |
| 10 | [**text( val )**](https://www.tutorialspoint.com/jquery/attr-text-val.htm)  Set the text contents of all matched elements. |
| 11 | [**val( )**](https://www.tutorialspoint.com/jquery/attr-val.htm)  Get the input value of the first matched element. |
| 12 | [**val( val )**](https://www.tutorialspoint.com/jquery/attr-val-val.htm)  Set the value attribute of every matched element if it is called on <input> but if it is called on <select> with the passed <option> value then passed option would be selected, if it is called on check box or radio box then all the matching check box and radiobox would be checked. |

# jQuery - DOM Traversing

jQuery is a very powerful tool which provides a variety of DOM traversal methods to help us select elements in a document randomly as well as in sequential method.

Most of the DOM Traversal Methods do not modify the jQuery object and they are used to filter out elements from a document based on given conditions.

## Find Elements by Index

Consider a simple document with the following HTML content −

<html>

<head>

<title>The JQuery Example</title>

</head>

<body>

<div>

<ul>

<li>list item 1</li>

<li>list item 2</li>

<li>list item 3</li>

<li>list item 4</li>

<li>list item 5</li>

<li>list item 6</li>

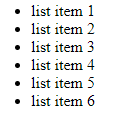
</ul>

</div>

</body>

</html>

This will produce following result –



* Above every list has its own index, and can be located directly by using **eq(index)** method as below example.
* Every child element starts its index from zero, thus, *list item 2* would be accessed by using **$("li").eq(1)** and so on.

### Example

Following is a simple example which adds the color to second list item.

<html>

<head>

<title>The JQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/

jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("li").eq(2).addClass("selected");

});

</script>

<style>

.selected {color:red;}

</style>

</head>

<body>

<div>

<ul>

<li>list item 1</li>

<li>list item 2</li>

<li>list item 3</li>

<li>list item 4</li>

<li>list item 5</li>

<li>list item 6</li>

</ul>

</div>

</body>

</html>

This will produce following result −

## 

## Filtering out Elements

The **filter( selector )** method can be used to filter out all elements from the set of matched elements that do not match the specified selector(s).

The *selector* can be written using any selector syntax.

### Example

Following is a simple example which applies color to the lists associated with middle class −

<html>

<head>

<title>The JQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("li").filter(".middle").addClass("selected");

});

</script>

<style>

.selected {color:red;}

</style>

</head>

<body>

<div>

<ul>

<liclass="top">list item 1</li>

<liclass="top">list item 2</li>

<liclass="middle">list item 3</li>

<liclass="middle">list item 4</li>

<liclass="bottom">list item 5</li>

<liclass="bottom">list item 6</li>

</ul>

</div>

</body>

</html>

This will produce following result −

## 

## Locating Descendant Elements

The **find( selector )** method can be used to locate all the descendant elements of a particular type of elements. The *selector* can be written using any selector syntax.

### Example

Following is an example which selects all the <span> elements available inside different <p> elements −

<html>

<head>

<title>The JQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/

jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("p").find("span").addClass("selected");

});

</script>

<style>

.selected {color:red;}

</style>

</head>

<body>

<p>This is 1st paragraph and <span>THIS IS RED</span></p>

<p>This is 2nd paragraph and <span>THIS IS ALSO RED</span></p>

</body>

</html>

This will produce following result −

## 

## JQuery DOM Filter Methods

Following table lists down useful methods which you can use to filter out various elements from a list of DOM elements −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [**eq( index )**](https://www.tutorialspoint.com/jquery/traversal-eq.htm)  Reduce the set of matched elements to a single element. |
| 2 | [**filter( selector )**](https://www.tutorialspoint.com/jquery/traversal-filter.htm)  Removes all elements from the set of matched elements that do not match the specified selector(s). |
| 3 | [**filter( fn )**](https://www.tutorialspoint.com/jquery/traversal-filter-fn.htm)  Removes all elements from the set of matched elements that do not match the specified function. |
| 4 | [**is( selector )**](https://www.tutorialspoint.com/jquery/traversal-is.htm)  Checks the current selection against an expression and returns true, if at least one element of the selection fits the given selector. |
| 5 | [**map( callback )**](https://www.tutorialspoint.com/jquery/traversal-map.htm)  Translate aset of elements in the jQuery object into another set of values in a jQuery array (which may, or may not contain elements). |
| 6 | [**not( selector )**](https://www.tutorialspoint.com/jquery/traversal-not.htm)  Removes elements matching the specified selector from the set of matched elements. |
| 7 | [**slice( start, [end] )**](https://www.tutorialspoint.com/jquery/traversal-slice.htm)  Selects a subset of the matched elements. |

## JQuery DOM Traversing Methods

Following table lists down other useful methods which you can use to locate various elements in a DOM −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | [**add( selector )**](https://www.tutorialspoint.com/jquery/traversal-add.htm)  Adds more elements, matched by the given selector, to the set of matched elements. |
| 2 | [**andSelf( )**](https://www.tutorialspoint.com/jquery/traversal-andself.htm)  Add the previous selection to the current selection. |
| 3 | [**children( [selector])**](https://www.tutorialspoint.com/jquery/traversal-children.htm)  Get a set of elements containing all of the unique immediate children of each of the matched set of elements. |
| 4 | [**closest( selector )**](https://www.tutorialspoint.com/jquery/traversal-closest.htm)  Get a set of elements containing the closest parent element that matches the specified selector, the starting element included. |
| 5 | [**contents( )**](https://www.tutorialspoint.com/jquery/traversal-contents.htm)  Find all the child nodes inside the matched elements (including text nodes), or the content document, if the element is an iframe. |
| 6 | [**end( )**](https://www.tutorialspoint.com/jquery/traversal-end.htm)  Revert the most recent 'destructive' operation, changing the set of matched elements to its previous state. |
| 7 | [**find( selector )**](https://www.tutorialspoint.com/jquery/traversal-find.htm)  Searches for descendant elements that match the specified selectors. |
| 8 | [**next( [selector] )**](https://www.tutorialspoint.com/jquery/traversal-next.htm)  Get a set of elements containing the unique next siblings of each of the given set of elements. |
| 9 | [**nextAll( [selector] )**](https://www.tutorialspoint.com/jquery/traversal-nextall.htm)  Find all sibling elements after the current element. |
| 10 | [**offsetParent( )**](https://www.tutorialspoint.com/jquery/traversal-offsetparent.htm)  Returns a jQuery collection with the positioned parent of the first matched element. |
| 11 | [**parent( [selector] )**](https://www.tutorialspoint.com/jquery/traversal-parent.htm)  Get the direct parent of an element. If called on a set of elements, parent returns a set of their unique direct parent elements. |
| 12 | [**parents( [selector] )**](https://www.tutorialspoint.com/jquery/traversal-parents.htm)  Get a set of elements containing the unique ancestors of the matched set of elements (except for the root element). |
| 13 | [**prev( [selector] )**](https://www.tutorialspoint.com/jquery/traversal-prev.htm)  Get a set of elements containing the unique previous siblings of each of the matched set of elements. |
| 14 | [**prevAll( [selector] )**](https://www.tutorialspoint.com/jquery/traversal-prevall.htm)  Find all sibling elements in front of the current element. |
| 15 | [**siblings( [selector] )**](https://www.tutorialspoint.com/jquery/traversal-siblings.htm)  Get a set of elements containing all of the unique siblings of each of the matched set of elements. |

# jQuery - CSS Selectors Methods

## Apply CSS Properties

This is very simple to apply any CSS property using JQuery method **css( PropertyName, PropertyValue )**.

Here is the syntax for the method −

**selector**.css( PropertyName, PropertyValue);

Here you can pass *PropertyName* as a javascript string and based on its value, *PropertyValue* could be string or integer.

### Example

Following is an example which adds font color to the second list item.

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("li").eq(2).css("color","red");

});

</script>

</head>

<body>

<div>

<ul>

<li>list item 1</li>

<li>list item 2</li>

<li>list item 3</li>

<li>list item 4</li>

<li>list item 5</li>

<li>list item 6</li>

</ul>

</div>

</body>

</html>

This will produce following result −

## 

## Apply Multiple CSS Properties

You can apply multiple CSS properties using a single JQuery method **CSS( {key1:val1, key2:val2....)**. You can apply as many properties as you like in a single call.

Here is the syntax for the method −

**selector**.css( {key1:val1, key2:val2....keyN:valN})

Here you can pass key as property and val as its value as described above.

### Example

Following is an example which adds font color as well as background color to the second list item.

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("li").eq(2).css({"color":"red","background-color":"green"});

});

</script>

</head>

<body>

<div>

<ul>

<li>list item 1</li>

<li>list item 2</li>

<li>list item 3</li>

<li>list item 4</li>

<li>list item 5</li>

<li>list item 6</li>

</ul>

</div>

</body>

</html>

This will produce following result −

## 

## Setting Element Width & Height

The **width( val )** and **height( val )** method can be used to set the width and height respectively of any element.

### Example

Following is a simple example which sets the width of first division element where as rest of the elements have width set by style sheet

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("div:first").width(100);

$("div:first").css("background-color","blue");

});

</script>

<style>

div {

width:70px; height:50px;float:left;

margin:5px;background:red;cursor:pointer;

}

</style>

</head>

<body>

<div></div>

<div>d</div>

<div>d</div>

<div>d</div>

<div>d</div>

</body>

</html>

This will produce following result −

## 

## JQuery CSS Methods

Following table lists down all the methods which you can use to play with CSS properties −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [**css( name )**](https://www.tutorialspoint.com/jquery/css-name.htm)  Return a style property on the first matched element. |
| 2 | [**css( name, value )**](https://www.tutorialspoint.com/jquery/css-name-value.htm)  Set a single style property to a value on all matched elements. |
| 3 | [**css( properties )**](https://www.tutorialspoint.com/jquery/css-properties.htm)  Set a key/value object as style properties to all matched elements. |
| 4 | [**height( val )**](https://www.tutorialspoint.com/jquery/css-height-val.htm)  Set the CSS height of every matched element. |
| 5 | [**height( )**](https://www.tutorialspoint.com/jquery/css-height.htm)  Get the current computed, pixel, height of the first matched element. |
| 6 | [**innerHeight( )**](https://www.tutorialspoint.com/jquery/css-innerheight.htm)  Gets the inner height (excludes the border and includes the padding) for the first matched element. |
| 7 | [**innerWidth( )**](https://www.tutorialspoint.com/jquery/css-innerwidth.htm)  Gets the inner width (excludes the border and includes the padding) for the first matched element. |
| 8 | [**offset( )**](https://www.tutorialspoint.com/jquery/css-offset.htm)  Get the current offset of the first matched element, in pixels, relative to the document. |
| 9 | [**offsetParent( )**](https://www.tutorialspoint.com/jquery/css-offsetparent.htm)  Returns a jQuery collection with the positioned parent of the first matched element. |
| 10 | [**outerHeight( [margin] )**](https://www.tutorialspoint.com/jquery/css-outerheight.htm)  Gets the outer height (includes the border and padding by default) for the first matched element. |
| 11 | [**outerWidth( [margin] )**](https://www.tutorialspoint.com/jquery/css-outerwidth.htm)  Get the outer width (includes the border and padding by default) for the first matched element. |
| 12 | [**position( )**](https://www.tutorialspoint.com/jquery/css-position.htm)  Gets the top and left position of an element relative to its offset parent. |
| 13 | [**scrollLeft( val )**](https://www.tutorialspoint.com/jquery/css-scrollleft-val.htm)  When a value is passed in, the scroll left offset is set to that value on all matched elements. |
| 14 | [**scrollLeft( )**](https://www.tutorialspoint.com/jquery/css-scrollleft.htm)  Gets the scroll left offset of the first matched element. |
| 15 | [**scrollTop( val )**](https://www.tutorialspoint.com/jquery/css-scrolltop-val.htm)  When a value is passed in, the scroll top offset is set to that value on all matched elements. |
| 16 | [**scrollTop( )**](https://www.tutorialspoint.com/jquery/css-scrolltop.htm)  Gets the scroll top offset of the first matched element. |
| 17 | [**width( val )**](https://www.tutorialspoint.com/jquery/css-width-val.htm)  Set the CSS width of every matched element. |
| 18 | [**width( )**](https://www.tutorialspoint.com/jquery/css-width.htm)  Get the current computed, pixel, width of the first matched element. |

# jQuery - DOM Manipulation

JQuery provides methods to manipulate DOM in efficient way. You do not need to write big code to modify the value of any element's attribute or to extract HTML code from a paragraph or division.

JQuery provides methods such as .attr(), .html(), and .val() which act as getters, retrieving information from DOM elements for later use.

## Content Manipulation

The **html( )** method gets the html contents (innerHTML) of the first matched element.

Here is the syntax for the method −

*selector*.html( )

### Example

Following is an example which makes use of .html() and .text(val) methods.

Here .html() retrieves HTML content from the object and then .text( val ) method sets value of the object using passed parameter −

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("div").click(function(){

var content = $(this).html();

$("#result").text( content );

});

});

</script>

<style>

#division{ margin:10px;padding:12px; border:2px solid #666; width:60px;}

</style>

</head>

<body>

<p>Click on the square below:</p>

<spanid="result"></span>

<divid="division"style="background-color:blue;">

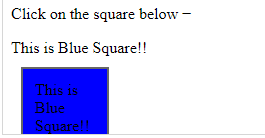
This is Blue Square!!

</div>

</body>

</html>

This will produce following result –



## DOM Element Replacement

You can replace a complete DOM element with the specified HTML or DOM elements.

The **replaceWith( content )** method serves this purpose very well.

Here is the syntax for the method −

*selector*.replaceWith( content )

Here content is what you want to have instead of original element. This could be HTML or simple text.

### Example

Following is an example which would replace division element with "<h1>JQuery is Great </h1>" −

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("div").click(function(){

$(this).replaceWith("<h1>JQuery is Great</h1>");

});

});

</script>

<style>

#division{ margin:10px;padding:12px; border:2px solid #666; width:60px;}

</style>

</head>

<body>

<p>Click on the square below:</p>

<spanid="result"></span>

<divid="division"style="background-color:blue;">

This is Blue Square!!

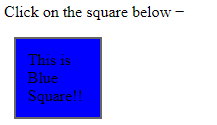
</div>

</body>

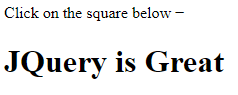
</html>

This will produce following result –

Before Clicking



After clicking



## Removing DOM Elements

There may be a situation when you would like to remove one or more DOM elements from the document.

JQuery provides two methods to handle the situation.

The **empty( )** method remove all child nodes from the set of matched elements where as the method

**remove( expr )** method removes all matched elements from the DOM.

Here is the syntax for the method −

*selector*.remove( [ expr ])

or

*selector*.empty( )

You can pass optional parameter *expr* to filter the set of elements to be removed.

### Example

Following is an example where elements are being removed as soon as they are clicked −

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("div").click(function(){

$(this).remove();

});

});

</script>

<style>

.div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}

</style>

</head>

<body>

<p>Click on any square below:</p>

<spanid="result"></span>

<divclass="div"style="background-color:blue;"></div>

<divclass="div"style="background-color:green;"></div>

<divclass="div"style="background-color:red;"></div>

</body>

</html>

This will produce following result −

## Before Click event

## 

## After clicking on blue

## 

## Inserting DOM Elements

There may be a situation when you would like to insert new one or more DOM elements in your existing document.

JQuery provides various methods to insert elements at various locations.

The **after( content )** method insert content after each of the matched elements where as the method

**before( content )** method inserts content before each of the matched elements.

Here is the syntax for the method −

*selector*.after( content )

or

*selector*.before( content )

Here content is what you want to insert. This could be HTML or simple text.

### Example

Following is an example where <div> elements are being inserted just before the clicked element −

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("div").click(function(){

$(this).before('<div class="div"></div>');

});

});

</script>

<style>

.div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}

</style>

</head>

<body>

<p>Click on any square below:</p>

<spanid="result"></span>

<divclass="div"style="background-color:blue;"></div>

<divclass="div"style="background-color:green;"></div>

<divclass="div"style="background-color:red;"></div>

</body>

</html>

This will produce following result –

Before Click

## 

## After clicking on Green box

## 

## DOM Manipulation Methods

Following table lists down all the methods which you can use to manipulate DOM elements −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [**after( content )**](https://www.tutorialspoint.com/jquery/dom-after.htm)  Insert content after each of the matched elements. |
| 2 | [**append( content )**](https://www.tutorialspoint.com/jquery/dom-append.htm)  Append content to the inside of every matched element. |
| 3 | [**appendTo( selector )**](https://www.tutorialspoint.com/jquery/dom-appendto.htm)  Append all of the matched elements to another, specified, set of elements. |
| 4 | [**before( content )**](https://www.tutorialspoint.com/jquery/dom-before.htm)  Insert content before each of the matched elements. |
| 5 | [**clone( bool )**](https://www.tutorialspoint.com/jquery/dom-clone-bool.htm)  Clone matched DOM Elements, and all their event handlers, and select the clones. |
| 6 | [**clone( )**](https://www.tutorialspoint.com/jquery/dom-clone.htm)  Clone matched DOM Elements and select the clones. |
| 7 | [**empty( )**](https://www.tutorialspoint.com/jquery/dom-empty.htm)  Remove all child nodes from the set of matched elements. |
| 8 | [**html( val )**](https://www.tutorialspoint.com/jquery/dom-html-val.htm)  Set the html contents of every matched element. |
| 9 | [**html( )**](https://www.tutorialspoint.com/jquery/dom-html.htm)  Get the html contents (innerHTML) of the first matched element. |
| 10 | [**insertAfter( selector )**](https://www.tutorialspoint.com/jquery/dom-insertafter.htm)  Insert all of the matched elements after another, specified, set of elements. |
| 11 | [**insertBefore( selector )**](https://www.tutorialspoint.com/jquery/dom-insertbefore.htm)  Insert all of the matched elements before another, specified, set of elements. |
| 12 | [**prepend( content )**](https://www.tutorialspoint.com/jquery/dom-prepend.htm)  Prepend content to the inside of every matched element. |
| 13 | [**prependTo( selector )**](https://www.tutorialspoint.com/jquery/dom-prependto.htm)  Prepend all of the matched elements to another, specified, set of elements. |
| 14 | [**remove( expr )**](https://www.tutorialspoint.com/jquery/dom-remove-expr.htm)  Removes all matched elements from the DOM. |
| 15 | [**replaceAll( selector )**](https://www.tutorialspoint.com/jquery/dom-replaceall.htm)  Replaces the elements matched by the specified selector with the matched elements. |
| 16 | [**replaceWith( content )**](https://www.tutorialspoint.com/jquery/dom-replacewith.htm)  Replaces all matched elements with the specified HTML or DOM elements. |
| 17 | [**text( val )**](https://www.tutorialspoint.com/jquery/dom-text-val.htm)  Set the text contents of all matched elements. |
| 18 | [**text( )**](https://www.tutorialspoint.com/jquery/dom-text.htm)  Get the combined text contents of all matched elements. |
| 19 | [**wrap( elem )**](https://www.tutorialspoint.com/jquery/dom-wrap-elem.htm)  Wrap each matched element with the specified element. |
| 20 | [**wrap( html )**](https://www.tutorialspoint.com/jquery/dom-wrap-html.htm)  Wrap each matched element with the specified HTML content. |
| 21 | [**wrapAll( elem )**](https://www.tutorialspoint.com/jquery/dom-wrapall-elem.htm)  Wrap all the elements in the matched set into a single wrapper element. |
| 22 | [**wrapAll( html )**](https://www.tutorialspoint.com/jquery/dom-wrapall-html.htm)  Wrap all the elements in the matched set into a single wrapper element. |
| 23 | [**wrapInner( elem )**](https://www.tutorialspoint.com/jquery/dom-wrapinner-elem.htm)  Wrap the inner child contents of each matched element (including text nodes) with a DOM element. |
| 24 | [**wrapInner( html )**](https://www.tutorialspoint.com/jquery/dom-wrapinner-html.htm)  Wrap the inner child contents of each matched element (including text nodes) with an HTML structure. |

# jQuery - Events Handling

We have the ability to create dynamic web pages by using events.

Events are actions that can be detected by your Web Application.

Following are the examples events −

* A mouse click
* A web page loading
* Taking mouse over an element
* Submitting an HTML form
* A keystroke on your keyboard, etc.

When these events are triggered, you can then use a custom function to do pretty much whatever you want with the event. These custom functions call Event Handlers.

## Binding Event Handlers

Using the jQuery Event Model, we can establish event handlers on DOM elements with the **bind()** method as follows −

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

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

alert('Hi there!');

});

});

</script>

<style>

.div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}

</style>

</head>

<body>

<p>Click on any square below to see the result:</p>

<divclass="div"style="background-color:blue;">ONE</div>

<divclass="div"style="background-color:green;">TWO</div>

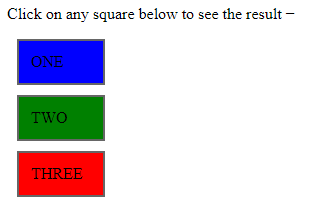
<divclass="div"style="background-color:red;">THREE</div>

</body>

</html>

This code will cause the division element to respond to the click event; when a user clicks inside this division thereafter, the alert will be shown.

This will produce following result −



The full syntax of the bind() command is as follows −

*selector*.bind( eventType[, eventData], handler)

Following is the description of the parameters −

* **eventType** − A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.
* **eventData** − This is optional parameter is a map of data that will be passed to the event handler.
* **handler** − A function to execute each time the event is triggered.

## Removing Event Handlers

Typically, once an event handler is established, it remains in effect for the remainder of the life of the page. There may be a need when you would like to remove event handler.

jQuery provides the **unbind()** command to remove an exiting event handler. The syntax of unbind() is as follows −

*selector*.unbind(eventType, handler)

or

*selector*.unbind(eventType)

Following is the description of the parameters −

* **eventType** − A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.
* **handler** − If provided, identifies the specific listener that's to be removed.

## Event Types

The following are cross platform and recommended event types which you can bind using JQuery −

|  |  |
| --- | --- |
| **Sr.No.** | **Event Type & Description** |
| 1 | **blur**  Occurs when the element loses focus. |
| 2 | **change**  Occurs when the element changes. |
| 3 | **click**  Occurs when a mouse click. |
| 4 | **dblclick**  Occurs when a mouse double-click. |
| 5 | **error**  Occurs when there is an error in loading or unloading etc. |
| 6 | **focus**  Occurs when the element gets focus. |
| 7 | **keydown**  Occurs when key is pressed. |
| 8 | **keypress**  Occurs when key is pressed and released. |
| 9 | **keyup**  Occurs when key is released. |
| 10 | **load**  Occurs when document is loaded. |
| 11 | **mousedown**  Occurs when mouse button is pressed. |
| 12 | **mouseenter**  Occurs when mouse enters in an element region. |
| 13 | **mouseleave**  Occurs when mouse leaves an element region. |
| 14 | **mousemove**  Occurs when mouse pointer moves. |
| 15 | **mouseout**  Occurs when mouse pointer moves out of an element. |
| 16 | **mouseover**  Occurs when mouse pointer moves over an element. |
| 17 | **mouseup**  Occurs when mouse button is released. |
| 18 | **resize**  Occurs when window is resized. |
| 19 | **scroll**  Occurs when window is scrolled. |
| 20 | **select**  Occurs when a text is selected. |
| 21 | **Submit**  Occurs when form is submitted. |
| 22 | **Unload**  Occurs when documents is unloaded. |

## The Event Object

The callback function takes a single parameter; when the handler is called the JavaScript event object will be passed through it.

The event object is often unnecessary and the parameter is omitted, as sufficient context is usually available when the handler is bound to know exactly what needs to be done when the handler is triggered, however there are certain attributes which you would need to be accessed.

## The Event Attributes

The following event properties/attributes are available and safe to access in a platform independent manner –

|  |  |
| --- | --- |
| **Sr.No.** | **Property & Description** |
| 1 | **altKey**  Set to true if the Alt key was pressed when the event was triggered, false if not. The Alt key is labeled Option on most Mac keyboards. |
| 2 | **ctrlKey**  Set to true if the Ctrl key was pressed when the event was triggered, false if not. |
| 3 | **data**  The value, if any, passed as the second parameter to the bind() command when the handler was established. |
| 4 | **keyCode**  For keyup and keydown events, this returns the key that was pressed. |
| 5 | **metaKey**  Set to true if the Meta key was pressed when the event was triggered, false if not. The Meta key is the Ctrl key on PCs and the Command key on Macs. |
| 6 | **pageX**  For mouse events, specifies the horizontal coordinate of the event relative from the page origin. |
| 7 | **pageY**  For mouse events, specifies the vertical coordinate of the event relative from the page origin. |
| 8 | **relatedTarget**  For some mouse events, identifies the element that the cursor left or entered when the event was triggered. |
| 9 | **screenX**  For mouse events, specifies the horizontal coordinate of the event relative from the screen origin. |
| 10 | **screenY**  For mouse events, specifies the vertical coordinate of the event relative from the screen origin. |
| 11 | **shiftKey**  Set to true if the Shift key was pressed when the event was triggered, false if not. |
| 12 | **target**  Identifies the element for which the event was triggered. |
| 13 | **timeStamp**  The timestamp (in milliseconds) when the event was created. |
| 14 | **type**  For all events, specifies the type of event that was triggered (for example, click). |
| 15 | **which**  For keyboard events, specifies the numeric code for the key that caused the event, and for mouse events, specifies which button was pressed (1 for left, 2 for middle, 3 for right). |

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

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

alert('Event type is '+event.type);

alert('pageX : '+event.pageX);

alert('pageY : '+event.pageY);

alert('Target : '+event.target.innerHTML);

});

});

</script>

<style>

.div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}

</style>

</head>

<body>

<p>Click on any square below to see the result:</p>

<divclass="div"style="background-color:blue;">ONE</div>

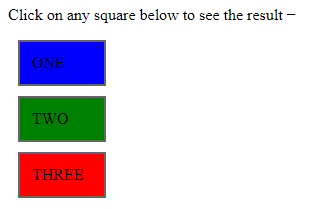
<divclass="div"style="background-color:green;">TWO</div>

<divclass="div"style="background-color:red;">THREE</div>

</body>

</html>

This will produce following result –



## The Event Methods

There is a list of methods which can be called on an Event Object −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [**preventDefault()**](https://www.tutorialspoint.com/jquery/events-preventdefault.htm)  Prevents the browser from executing the default action. |
| 2 | [**isDefaultPrevented()**](https://www.tutorialspoint.com/jquery/events-isdefaultprevented.htm)  Returns whether event.preventDefault() was ever called on this event object. |
| 3 | [**stopPropagation()**](https://www.tutorialspoint.com/jquery/events-stoppropagation.htm)  Stops the bubbling of an event to parent elements, preventing any parent handlers from being notified of the event. |
| 4 | [**isPropagationStopped()**](https://www.tutorialspoint.com/jquery/events-ispropagationstopped.htm)  Returns whether event.stopPropagation() was ever called on this event object. |
| 5 | [**stopImmediatePropagation()**](https://www.tutorialspoint.com/jquery/events-stopimmediatepropagation.htm)  Stops the rest of the handlers from being executed. |
| 6 | [**isImmediatePropagationStopped()**](https://www.tutorialspoint.com/jquery/events-isimmediatepropagationstopped.htm)  Returns whether event.stopImmediatePropagation() was ever called on this event object. |

## Event Manipulation Methods

Following table lists down important event-related methods −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [**bind( type, [data], fn )**](https://www.tutorialspoint.com/jquery/events-bind.htm)  Binds a handler to one or more events (like click) for each matched element. Can also bind custom events. |
| 2 | [**off( events [, selector ] [, handler(eventObject) ] )**](https://www.tutorialspoint.com/jquery/events-off.htm)  This does the opposite of live, it removes a bound live event. |
| 3 | [**hover( over, out )**](https://www.tutorialspoint.com/jquery/events-hover.htm)  Simulates hovering for example moving the mouse on, and off, an object. |
| 4 | [**on( events [, selector ] [, data ], handler )**](https://www.tutorialspoint.com/jquery/events-on.htm)  Binds a handler to an event (like click) for all current − and future − matched element. Can also bind custom events. |
| 5 | [**one( type, [data], fn )**](https://www.tutorialspoint.com/jquery/events-one.htm)  Binds a handler to one or more events to be executed once for each matched element. |
| 6 | [**ready( fn )**](https://www.tutorialspoint.com/jquery/events-ready.htm)  Binds a function to be executed whenever the DOM is ready to be traversed and manipulated. |
| 7 | [**trigger( event, [data] )**](https://www.tutorialspoint.com/jquery/events-trigger.htm)  Trigger an event on every matched element. |
| 8 | [**triggerHandler( event, [data] )**](https://www.tutorialspoint.com/jquery/events-triggerhandler.htm)  Triggers all bound event handlers on an element. |
| 9 | [**unbind( [type], [fn] )**](https://www.tutorialspoint.com/jquery/events-unbind.htm)  This does the opposite of bind, it removes bound events from each of the matched elements. |

## Event Helper Methods

jQuery also provides a set of event helper functions which can be used either to trigger an event to bind any event types mentioned above.

## Trigger Methods

Following is an example which would triggers the blur event on all paragraphs −

$("p").blur();

## Binding Methods

Following is an example which would bind a **click** event on all the <div> −

$("div").click( function () {

// do something here

});

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **blur( )**  Triggers the blur event of each matched element. |
| 2 | **blur( fn )**  Bind a function to the blur event of each matched element. |
| 3 | **change( )**  Triggers the change event of each matched element. |
| 4 | **change( fn )**  Binds a function to the change event of each matched element. |
| 5 | **click( )**  Triggers the click event of each matched element. |
| 6 | **click( fn )**  Binds a function to the click event of each matched element. |
| 7 | **dblclick( )**  Triggers the dblclick event of each matched element. |
| 8 | **dblclick( fn )**  Binds a function to the dblclick event of each matched element. |
| 9 | **error( )**  Triggers the error event of each matched element. |
| 10 | **error( fn )**  Binds a function to the error event of each matched element. |
| 11 | **focus( )**  Triggers the focus event of each matched element. |
| 12 | **focus( fn )**  Binds a function to the focus event of each matched element. |
| 13 | **keydown( )**  Triggers the keydown event of each matched element. |
| 14 | **keydown( fn )**  Bind a function to the keydown event of each matched element. |
| 15 | **keypress( )**  Triggers the keypress event of each matched element. |
| 16 | **keypress( fn )**  Binds a function to the keypress event of each matched element. |
| 17 | **keyup( )**  Triggers the keyup event of each matched element. |
| 18 | **keyup( fn )**  Bind a function to the keyup event of each matched element. |
| 19 | **load( fn )**  Binds a function to the load event of each matched element. |
| 20 | **mousedown( fn )**  Binds a function to the mousedown event of each matched element. |
| 21 | **mouseenter( fn )**  Bind a function to the mouseenter event of each matched element. |
| 22 | **mouseleave( fn )**  Bind a function to the mouseleave event of each matched element. |
| 23 | **mousemove( fn )**  Bind a function to the mousemove event of each matched element. |
| 24 | **mouseout( fn )**  Bind a function to the mouseout event of each matched element. |
| 25 | **mouseover( fn )**  Bind a function to the mouseover event of each matched element. |
| 26 | **mouseup( fn )**  Bind a function to the mouseup event of each matched element. |
| 27 | **resize( fn )**  Bind a function to the resize event of each matched element. |
| 28 | **scroll( fn )**  Bind a function to the scroll event of each matched element. |
| 29 | **select( )**  Trigger the select event of each matched element. |
| 30 | **select( fn )**  Bind a function to the select event of each matched element. |
| 31 | **submit( )**  Trigger the submit event of each matched element. |
| 32 | **submit( fn )**  Bind a function to the submit event of each matched element. |
| 33 | **unload( fn )**  Binds a function to the unload event of each matched element. |

# jQuery - Ajax

AJAX is an acronym standing for Asynchronous JavaScript and XML and this technology helps us to load data from the server without a browser page refresh.

JQuery is a great tool which provides a rich set of AJAX methods to develop next generation web application.

## Loading Simple Data

This is very easy to load any static or dynamic data using JQuery AJAX. JQuery provides **load()** method to do the job −

### Syntax

Here is the simple syntax for **load()** method −

**[selector].**load( URL, [data], [callback] );

Here is the description of all the parameters −

* **URL** − The URL of the server-side resource to which the request is sent. It could be a CGI, ASP, JSP, or PHP script which generates data dynamically or out of a database.
* **data** − This optional parameter represents an object whose properties are serialized into properly encoded parameters to be passed to the request. If specified, the request is made using the **POST** method. If omitted, the **GET** method is used.
* **callback** − A callback function invoked after the response data has been loaded into the elements of the matched set. The first parameter passed to this function is the response text received from the server and second parameter is the status code.

### Example

Consider the following HTML file with a small JQuery coding −

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("#driver").click(function(event){

$('#stage').load('/jquery/result.html');

});

});

</script>

</head>

<body>

<p>Click on the button to load /jquery/result.html file −</p>

<divid="stage"style="background-color:cc0;">

STAGE

</div>

<inputtype="button"id="driver"value="Load Data"/>

</body>

</html>

Here **load()** initiates an Ajax request to the specified URL **/jquery/result.html** file. After loading this file, all the content would be populated inside <div> tagged with ID *stage*. Assuming, our /jquery/result.html file has just one HTML line −

<h1>THIS IS RESULT...</h1>

When you click the given button, then result.html file gets loaded.

## Getting JSON Data

There would be a situation when server would return JSON string against your request. JQuery utility function **getJSON()** parses the returned JSON string and makes the resulting string available to the callback function as first parameter to take further action.

### Syntax

Here is the simple syntax for **getJSON()** method −

**[selector].**getJSON( URL, [data], [callback] );

Here is the description of all the parameters −

* **URL** − The URL of the server-side resource contacted via the GET method.
* **data** − An object whose properties serve as the name/value pairs used to construct a query string to be appended to the URL, or a preformatted and encoded query string.
* **callback** − A function invoked when the request completes. The data value resulting from digesting the response body as a JSON string is passed as the first parameter to this callback, and the status as the second.

### Example

Consider the following HTML file with a small JQuery coding −

[Live Demo](http://tpcg.io/dGRqqF)

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("#driver").click(function(event){

$.getJSON('/jquery/result.json',function(jd){

$('#stage').html('<p> Name: '+ jd.name +'</p>');

$('#stage').append('<p>Age : '+jd.age+'</p>');

$('#stage').append('<p> Sex: '+jd.sex+'</p>');

});

});

});

</script>

</head>

<body>

<p>Click on the button to load result.json file −</p>

<divid="stage"style="background-color:#eee;">

STAGE

</div>

<inputtype="button"id="driver"value="Load Data"/>

</body>

</html>

Here JQuery utility method **getJSON()** initiates an Ajax request to the specified URL **result.json** file. After loading this file, all the content would be passed to the callback function which finally would be populated inside <div> tagged with ID *stage*. Assuming, our result.json file has following json formatted content −

{

"name": "Zara Ali",

"age" : "67",

"sex": "female"

}

When you click the given button, then result.json file gets loaded.

## Passing Data to the Server

Many times you collect input from the user and you pass that input to the server for further processing. JQuery AJAX made it easy enough to pass collected data to the server using **data** parameter of any available Ajax method.

### Example

This example demonstrate how can pass user input to a web server script which would send the same result back and we would print it −

[Live Demo](http://tpcg.io/aU6ni2)

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("#driver").click(function(event){

var name = $("#name").val();

$("#stage").load('/jquery/result.php',{"name":name});

});

});

</script>

</head>

<body>

<p>Enter your name and click on the button:</p>

<inputtype="input"id="name"size="40"/><br/>

<divid="stage"style="background-color:cc0;">

STAGE

</div>

<inputtype="button"id="driver"value="Show Result"/>

</body>

</html>

Here is the code written in **result.php** script −

<?php

if( $\_REQUEST["name"]){

$name = $\_REQUEST['name'];

echo "Welcome ". $name;

}

?>

Now you can enter any text in the given input box and then click "Show Result" button to see what you have entered in the input box.

## JQuery AJAX Methods

You have seen basic concept of AJAX using JQuery. Following table lists down all important JQuery AJAX methods which you can use based your programming need −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | [**jQuery.ajax( options )**](https://www.tutorialspoint.com/jquery/ajax-jquery-ajax.htm)  Load a remote page using an HTTP request. |
| 2 | [**jQuery.ajaxSetup( options )**](https://www.tutorialspoint.com/jquery/ajax-jquery-ajaxsetup.htm)  Setup global settings for AJAX requests. |
| 3 | [**jQuery.get( url, [data], [callback], [type] )**](https://www.tutorialspoint.com/jquery/ajax-jquery-get.htm)  Load a remote page using an HTTP GET request. |
| 4 | [**jQuery.getJSON( url, [data], [callback] )**](https://www.tutorialspoint.com/jquery/ajax-jquery-getjson.htm)  Load JSON data using an HTTP GET request. |
| 5 | [**jQuery.getScript( url, [callback] )**](https://www.tutorialspoint.com/jquery/ajax-jquery-getscript.htm)  Loads and executes a JavaScript file using an HTTP GET request. |
| 6 | [**jQuery.post( url, [data], [callback], [type] )**](https://www.tutorialspoint.com/jquery/ajax-jquery-post.htm)  Load a remote page using an HTTP POST request. |
| 7 | [**load( url, [data], [callback] )**](https://www.tutorialspoint.com/jquery/ajax-load.htm)  Load HTML from a remote file and inject it into the DOM. |
| 8 | [**serialize( )**](https://www.tutorialspoint.com/jquery/ajax-serialize.htm)  Serializes a set of input elements into a string of data. |
| 9 | [**serializeArray( )**](https://www.tutorialspoint.com/jquery/ajax-serializearray.htm)  Serializes all forms and form elements like the .serialize() method but returns a JSON data structure for you to work with. |

## JQuery AJAX Events

You can call various JQuery methods during the life cycle of AJAX call progress. Based on different events/stages following methods are available −

You can go through all the [AJAX Events](https://www.tutorialspoint.com/jquery/ajax-events.htm).

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | [**ajaxComplete( callback )**](https://www.tutorialspoint.com/jquery/ajaxcomplete.htm)  Attach a function to be executed whenever an AJAX request completes. |
| 2 | [**ajaxStart( callback )**](https://www.tutorialspoint.com/jquery/ajaxstart.htm)  Attach a function to be executed whenever an AJAX request begins and there is none already active. |
| 3 | [**ajaxError( callback )**](https://www.tutorialspoint.com/jquery/ajaxerror.htm)  Attach a function to be executed whenever an AJAX request fails. |
| 4 | [**ajaxSend( callback )**](https://www.tutorialspoint.com/jquery/ajaxsend.htm)  Attach a function to be executed before an AJAX request is sent. |
| 5 | [**ajaxStop( callback )**](https://www.tutorialspoint.com/jquery/ajaxstop.htm)  Attach a function to be executed whenever all AJAX requests have ended. |
| 6 | [**ajaxSuccess( callback )**](https://www.tutorialspoint.com/jquery/ajaxsuccess.htm)  Attach a function to be executed whenever an AJAX request completes successfully. |

# jQuery - Plugins

A plug-in is piece of code written in a standard JavaScript file. These files provide useful jQuery methods which can be used along with jQuery library methods.

There are plenty of jQuery plug-in available which you can download from repository link at <https://jquery.com/plugins>.

## How to use Plugins

To make a plug-in's methods available to us, we include plug-in file very similar to jQuery library file in the <head> of the document.

We must ensure that it appears after the main jQuery source file, and before our custom JavaScript code.

Following example shows how to include **jquery.plug-in.js** plugin −

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scriptsrc="jquery.plug-in.js"type="text/javascript"></script>

<scriptsrc="custom.js"type="text/javascript"></script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

.......your custom code.....

});

</script>

</head>

<body>

.............................

</body>

</html>

## How to develop a Plug-in

This is very simple to write your own plug-in. Following is the syntax to create a a method −

jQuery.fn.methodName = methodDefinition;

Here *methodNameM* is the name of new method and *methodDefinition* is actual method definition.

The guideline recommended by the jQuery team is as follows −

* Any methods or functions you attach must have a semicolon (;) at the end.
* Your method must return the jQuery object, unless explicity noted otherwise.
* You should use this.each to iterate over the current set of matched elements - it produces clean and compatible code that way.
* Prefix the filename with jquery, follow that with the name of the plugin and conclude with .js.
* Always attach the plugin to jQuery directly instead of $, so users can use a custom alias via noConflict() method.

For example, if we write a plugin that we want to name *debug*, our JavaScript filename for this plugin is −

jquery.debug.js

The use of the **jquery.** prefix eliminates any possible name collisions with files intended for use with other libraries.

## Example

Following is a small plug-in to have warning method for debugging purpose. Keep this code in *jquery.debug.js* file −

jQuery.fn.warning=function(){

returnthis.each(function(){

alert('Tag Name:"'+ $(this).prop("tagName")+'".');

});

};

Here is the example showing usage of warning() method. Assuming we put *jquery.debug.js* file in same directory of html page.

[Live Demo](http://tpcg.io/1wF90g)

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scriptsrc="jquery.debug.js"type="text/javascript">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("div").warning();

$("p").warning();

});

</script>

</head>

<body>

<p>This is paragraph</p>

<div>This is division</div>

</body>

</html>

This would alert you with following result −

This is paragraph

This is division

# jQuery - flickerplate

Flickerplate is a jQuery plugin for creating a slider which allows you cycle through images with animated arrows and dots navigation.

A Simple of flickerplate example as shown below −

<!DOCTYPE html>

<html>

<head>

<metacharset="utf-8">

<metaname="viewport"content="width = device-width,

initial-scale = 1.0, maximum-scale = 1.0, user-scalable = no">

<scriptsrc="/jquery/src/flickerplate/js/min/jquery-v1.10.2.min.js"

type="text/javascript">

</script>

<scriptsrc="/jquery/src/flickerplate/js/min/modernizr-custom-v2.7.1.min.js"

type="text/javascript">

</script>

<scriptsrc="/jquery/src/flickerplate/js/min/hammer-v2.0.3.min.js"

type="text/javascript">

</script>

<linkhref="/jquery/src/flickerplate/css/flickerplate.css"

type="text/css"rel="stylesheet">

<scriptsrc="/jquery/src/flickerplate/js/min/flickerplate.min.js"

type="text/javascript">

</script>

<script>

$(function(){

$('.flicker-example').flickerplate({

auto\_flick:true,

auto\_flick\_delay:8,

flick\_animation:'transform-slide'

});

});

</script>

<linkhref="/jquery/src/flickerplate/css/demo.css"

type="text/css"rel="stylesheet">

</head>

<body>

<divclass="flicker-example">

<ul>

<lidata-background="https://genblock.com/wp-content/uploads/2015

/05/download-circles-abstract-wallpaper-

abstract-photo-abstract-wallpaper.jpg">

<imgsrc="https://www.tutorialspoint.com/about/images/mohtashim.jpg"

style="margin-left:428px;">

<divclass="flick-title">Mohtashim M.</div>

<divclass="flick-sub-text">

Mohtashim is an MCA from AMU (Aligarah) and a Project

Management Professional. He has more than 17 years of

experience in Telecom and Datacom industries covering

complete SDLC. He is managing in-house innovations,

business planning, implementation, finance and the overall

business development of Tutorials Point.

</div>

</li>

<lidata-background="https://genblock.com/wp-content/uploads/2015

/05/download-circles-abstract-wallpaper-

abstract-photo-abstract-wallpaper.jpg">

<imgsrc="https://www.tutorialspoint.com/about/images/gopal\_verma.jpg"

style="margin-left:428px;">

<divclass="flick-title">Gopal K Verma</div>

<divclass="flick-sub-text">

Gopal is an MCA from GJU (Hisar) and a Cisco Certified Network

Professional. He has more than 11 years of experience in core

data networking and telecommunications. He develops contents

for Computer Science related subjects. He is also involved in

developing Apps for various Mobile devices.

</div>

</li>

<lidata-background="https://genblock.com/wp-content/uploads/2015

/05/download-circles-abstract-wallpaper-

abstract-photo-abstract-wallpaper.jpg">

<imgsrc="https://www.tutorialspoint.com/about/images/mukesh\_kumar.jpg"

style="margin-left:428px;">

<divclass="flick-title">Mukesh Kumar</div>

<divclass="flick-sub-text">

Mukesh Kumar, having 7+years experience in writing on various

topics ranging from IT products and services, legal, medical,

online advertisement & education to e-commerce businesses.

He also has experience of text & copy-editing, & online

research. He has done two masters – MA (Geography) from

University of Delhi and MA (Mass Communication &

Journalism) from Kurukshetra University.

</div>

</li>

</ul>

</div>

</body>

</html>