# jQuery Introduction (Source:w3schools.com)

---

The purpose of jQuery is to make it much easier to use JavaScript on your website.

---

## What is jQuery?

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that requires many lines of JavaScript code to accomplish, and wraps it into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

**Tip:** In addition, jQuery has plugins for almost any task out there.

---

## Why jQuery?

There are a lots of other JavaScript frameworks out there, but jQuery seems to be the most popular, and also the most extendable.

Many of the biggest companies on the Web use jQuery, like:

- Google
- Microsoft
- IBM
- Netflix

**Will jQuery work in all browsers?**

The jQuery team knows all about cross-browser issues, and they have written this knowledge into the jQuery library. jQuery will run exactly the same in all major browsers, including Internet Explorer 6!

# jQuery Install

---

## Adding jQuery to Your Web Pages

To use jQuery, you need to download the jQuery library (explained below), and include it on the pages you wish to use it.

The jQuery library is a single JavaScript file, and you reference to it using the HTML <script> tag:

```
<head>
<script src="jquery.js"></script>
</head>
```

Notice that the <script> tag should be inside the page's <head> section.

**Do you wonder why we do not have type="text/javascript" inside the <script> tag?**

This is not required in HTML5. JavaScript is the default scripting language in HTML5 and in all modern browsers!

---

## Downloading jQuery

There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

Both versions can be downloaded from jQuery.com.

**Tip:** Place the downloaded file in the same directory as the pages where you wish to use it.

---

## Alternatives to Downloading

If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).

Both Google and Microsoft host jQuery.

To use jQuery from Google, use one of the following:

## Google CDN:

```
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js">
</script>
</head>
```

Try it yourself »

**Get the latest available version with Google CDN:**

If you look at the Google URL above - the version of jQuery is specified in the URL (1.8.0). If you would like to use the latest version of jQuery, you can either remove a number from the end of the version string (for example 1.8), then Google will return the latest version available in the 1.8 series (1.8.0, 1.8.1, etc.), or you can take it up to the whole number (1), and Google will return the latest version available in the 1 series (from 1.1.0 to 1.9.9).

**One big advantage of using the hosted jQuery from Google or Microsoft:**

Many users already have downloaded jQuery from Google or Microsoft when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time. Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.

# jQuery Syntax

---

With jQuery you select (query) HTML elements and perform "actions" on them.

## jQuery Syntax

The jQuery syntax is tailor made for **selecting** HTML elements and perform some **action** on the element(s).

Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples:

$(this).hide() - hides the current element.

$("p").hide() - hides all <p> elements.

$(".test").hide() - hides all elements with class="test".

$("#test").hide() - hides the element with id="test".

**Are you familiar with CSS selectors?**

jQuery uses CSS syntax to select elements. You will learn more about the selector syntax in the next chapter of this tutorial.

---

## The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

```
$(document).ready(function(){

  // jQuery methods go here...

});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

It is good practice to wait for the document to be fully loaded and ready, before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

Here are some examples of actions that can fail if methods are run before the document is fully loaded:

- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet

**Tip:** The jQuery team has also created an even shorter method for the document ready event:

```
$(function(){

  // jQuery methods go here...

});
```

Use the syntax you prefer. We think that the document ready event is easier to understand when reading the code.

# jQuery Selectors

---

jQuery selectors are one of the most important parts of the jQuery library.

---

## jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s).

With jQuery selectors you can find elements based on their id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.

All type of selectors in jQuery, start with the dollar sign and parentheses: $().

---

# The element Selector

The jQuery element selector selects elements based on their tag names.

You can select all <p> elements on a page like this:

$("p")

**Example**

When a user clicks on a button, all <p> elements will be hidden:

# Example

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

# The #id Selector

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.

An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

To find an element with a specific id, write a hash character, followed by the id of the element:

$("#test")

**Example**

When a user clicks on a button, the element with id="test" will be hidden:

# Example

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

## The .class Selector

The jQuery class selector finds elements with a specific class.

To find elements with a specific class, write a period character, followed by the name of the class:

$(".test")

**Example**

When a user clicks on a button, the elements with class="test" will be hidden:

## Example

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
```

---

## More Examples of jQuery Selectors

| Syntax | Description |
|---|---|
| $("*") | Selects all elements |
| $(this) | Selects the current HTML element |
| $("p.intro") | Selects all <p> elements with class="intro" |
| $("p:first") | Selects the first <p> element |
| $("ul li:first") | Selects the first <li> element of the first <ul> |
| $("ul li:first-child") | Selects the first <li> element of every <ul> |
| $("[href]") | Selects all elements with an href attribute |
| $("a[target='_blank']") | Selects all <a> elements with a target attribute value equal to "_blank" |
| $("a[target!='_blank']") | Selects all <a> elements with a target attribute value NOT equal to "_blank" |
| $(":button") | Selects all <button> elements and <input> elements of type="button" |
| $("tr:even") | Selects all even <tr> elements |
| $("tr:odd") | Selects all odd <tr> elements |

# jQuery Event Methods

---

jQuery is tailor-made to handle HTML/DOM events.

---

## jQuery Event Methods

Event handlers are methods that are called when "something happens" in HTML.

The term "**triggered (or "fired") by an event**" is often used.

It is common to put jQuery code into event handler methods in the <head> section.

In the example below, a function is called when the click event for the button is triggered:

## Example

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>

</html>
```

## Functions In a Separate File

If your website contains a lot of pages, and you want your jQuery functions to be easy to maintain, you can put your jQuery functions in a separate .js file.

When we demonstrate jQuery in this tutorial, the functions are added directly into the <head> section. However, sometimes it is preferable to place them in a separate file, like this (use the src attribute to refer to the .js file):

## Example

```
<head>
<script src="jquery.js"></script>
<script src="my_jquery_functions.js"></script>
</head>
```

---

## Some jQuery Event Methods

Here are some examples of event methods in jQuery:

| Event Method | Description |
|---|---|
| $(document).ready(function) | Specifies a function to execute when the DOM is fully loaded |
| $(*selector*).click(function) | Binds/Triggers the click event |
| $(*selector*).dblclick(function) | Binds/Triggers the double click event |
| $(*selector*).focus(function) | Binds/Triggers the focus event |
| $(*selector*).mouseover(function) | Binds/Triggers the mouseover event |

# jQuery Effects - Hide and Show

---

Hide, Show, Toggle, Slide, Fade, and Animate. WOW!

Click to show/hide pane

# jQuery hide() and show()

With jQuery, you can hide and show HTML elements with the hide() and show() methods:

## Example

```
$("#hide").click(function(){
  $("p").hide();
});

$("#show").click(function(){
  $("p").show();
});
```

**Syntax:**

$(*selector*).hide(*speed,callback*);

$(*selector*).show(*speed,callback*);

The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after hide (or show) completes.

The following example demonstrates the speed parameter with hide():

## Example

```
$("button").click(function(){
  $("p").hide(1000);
});
```

---

# jQuery toggle()

With jQuery, you can toggle between the hide() and show() methods with the toggle() method.

Shown elements are hidden and hidden elements are shown:

## Example

```
$("button").click(function(){
  $("p").toggle();
});
```

**Syntax:**

$(*selector*).toggle(*speed,callback*);

The optional speed parameter can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after the toggle() method completes.

# jQuery Effects - Fading

---

With jQuery you can fade elements in and out of visibility.

Click to fade in/out panel

---

## jQuery Fading Methods

With jQuery you can fade an element in and out of visibility.

jQuery has the following fade methods:

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

---

## jQuery fadeIn() Method

The jQuery fadeIn() method is used to fade in a hidden element.

**Syntax:**

$(*selector*).fadeIn(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after the fading completes.

The following example demonstrates the fadeIn() method with different parameters:

## Example

```
$("button").click(function(){
  $("#div1").fadeIn();
  $("#div2").fadeIn("slow");
  $("#div3").fadeIn(3000);
});
```

---

## jQuery fadeOut() Method

The jQuery fadeOut() method is used to fade out a visible element.

**Syntax:**

$(*selector*).fadeOut(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after the fading completes.

The following example demonstrates the fadeOut() method with different parameters:

## Example

```
$("button").click(function(){
  $("#div1").fadeOut();
```

```
  $("#div2").fadeOut("slow");
  $("#div3").fadeOut(3000);
});
```

---

# jQuery fadeToggle() Method

The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.

If the elements are faded out, fadeToggle() will fade them in.

If the elements are faded in, fadeToggle() will fade them out.

**Syntax:**

$(*selector*).fadeToggle(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after the fading completes.

The following example demonstrates the fadeToggle() method with different parameters:

# Example

```
$("button").click(function(){
  $("#div1").fadeToggle();
  $("#div2").fadeToggle("slow");
  $("#div3").fadeToggle(3000);
});
```

# jQuery fadeTo() Method

The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1).

**Syntax:**

$(*selector*).fadeTo(*speed,opacity,callback*);

The required speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The required opacity parameter in the fadeTo() method specifies fading to a given opacity (value between 0 and 1).

The optional callback parameter is the name of a function to be executed after the function completes.

The following example demonstrates the fadeTo() method with different parameters:

## Example

```
$("button").click(function(){
  $("#div1").fadeTo("slow",0.15);
  $("#div2").fadeTo("slow",0.4);
  $("#div3").fadeTo("slow",0.7);
});
```

# jQuery Effects - Sliding

---

The jQuery slide methods slides elements up and down.

Click to slide down/up the panel

---

## jQuery Sliding Methods

With jQuery you can create a sliding effect on elements.

jQuery has the following slide methods:

- slideDown()
- slideUp()
- slideToggle()

---

## jQuery slideDown() Method

The jQuery slideDown() method is used to slide down an element.

**Syntax:**

$(*selector*).slideDown(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after the sliding completes.

The following example demonstrates the slideDown() method:

## Example

```
$("#flip").click(function(){
  $("#panel").slideDown();
});
```

---

# jQuery slideUp() Method

The jQuery slideUp() method is used to slide up an element.

**Syntax:**

$(*selector*).slideUp(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after the sliding completes.

The following example demonstrates the slideUp() method:

## Example

```
$("#flip").click(function(){
  $("#panel").slideUp();
});
```

## jQuery slideToggle() Method

The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.

If the elements are slide down, slideToggle() will slide them up.

If the elements are slide up, slideToggle() will slide them down.

$(*selector*).slideToggle(*speed,callback*);

The optional speed parameter can take the following values: "slow", "fast", milliseconds.

The optional callback parameter is the name of a function to be executed after the sliding completes.

The following example demonstrates the slideToggle() method:

## Example

```
$("#flip").click(function(){
  $("#panel").slideToggle();
});
```

## jQuery Effects Reference

For a complete overview of all jQuery effects, please go to our jQuery Effect Reference.

# jQuery Effects - Animation

The jQuery animate() method lets you create custom animations.

## jQuery Animations - The animate() Method

The jQuery animate() method is used to create custom animations.

**Syntax:**

$(*selector*).animate({*params*},*speed,callback*);

The required params parameter defines the CSS properties to be animated.

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after the animation completes.

The following example demonstrates a simple use of the animate() method; it moves a  element to the left, until it has reached a left property of 250px:

## Example

```
$("button").click(function(){
  $("div").animate({left:'250px'});
});
```

By default, all HTML elements have a static position, and cannot be moved.
To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!

---

## jQuery animate() - Manipulate Multiple Properties

Notice that multiple properties can be animated at the same time:

## Example

```
$("button").click(function(){
  $("div").animate({
    left:'250px',
    opacity:'0.5',
    height:'150px',
    width:'150px'
  });
});
```

**Is it possible to manipulate ALL CSS properties with the animate() method?**

Yes, almost! However, there is one important thing to remember: all property names must be camel-cased when used with the animate() method: You will need to write paddingLeft instead of padding-left, marginRight instead of margin-right, and so on.

Also, color animation is not included in the core jQuery library.
If you want to animate color, you need to download the Color Animations plugin from jQuery.com.

---

## jQuery animate() - Using Relative Values

It is also possible to define relative values (the value is then relative to the element's current value). This is done by putting += or -= in front of the value:

## Example

```
$("button").click(function(){
  $("div").animate({
    left:'250px',
    height:'+=150px',
    width:'+=150px'
  });
});
```

---

## jQuery animate() - Using Pre-defined Values

You can even specify a property's animation value as "show", "hide", or "toggle":

## Example

```
$("button").click(function(){
  $("div").animate({
    height:'toggle'
  });
});
```

---

## jQuery animate() - Uses Queue Functionality

By default, jQuery comes with queue functionality for animations.

This means that if you write multiple animate() calls after each other, jQuery creates an "internal" queue with these method calls. Then it runs the animate calls ONE by ONE.

So, if you want to perform different animations after each other, we take advantage of the queue functionality:

## Example 1

```
$("button").click(function(){
  var div=$("div");
  div.animate({height:'300px',opacity:'0.4'},"slow");
  div.animate({width:'300px',opacity:'0.8'},"slow");
  div.animate({height:'100px',opacity:'0.4'},"slow");
  div.animate({width:'100px',opacity:'0.8'},"slow");
});
```

The example below first moves the  element to the right, and then increases the font size of the text:

## Example 2

```
$("button").click(function(){
  var div=$("div");
  div.animate({left:'100px'},"slow");
  div.animate({fontSize:'3em'},"slow");
});
```

---

# jQuery Stop Animations

---

The jQuery stop() method is used to stop animations or effects before it is finished.

Click to slide down/up the panel

---

---

## jQuery stop() Method

The jQuery stop() method is used to stop an animation or effect before it is finished.

The stop() method works for all jQuery effect functions, including sliding, fading and custom animations.

**Syntax:**

$(*selector*).stop(*stopAll,goToEnd*);

The optional stopAll parameter specifies whether also the animation queue should be cleared or not. Default is false, which means that only the active animation will be stopped, allowing any queued animations to be performed afterwards.

The optional goToEnd parameter specifies whether or not to complete the current animation immediately. Default is false.

So, by default, the stop() method kills the current animation being performed on the selected element.

The following example demonstrates the stop() method, with no parameters:

## Example

```
$("#stop").click(function(){
  $("#panel").stop();
});
```

# jQuery Callback Functions

---

A callback function is executed after the current effect is 100% finished.

---

## jQuery Callback Functions

JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.

To prevent this, you can create a callback function.

A callback function is executed after the current effect is finished.

Typical syntax: **$(*selector*).hide(*speed,callback*);**

**Examples**

The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

## Example with Callback

```
$("button").click(function(){
  $("p").hide("slow",function(){
    alert("The paragraph is now hidden");
  });
});
```

The example below has no callback parameter, and the alert box will be displayed before the hide effect is completed:

## Example without Callback

```
$("button").click(function(){
  $("p").hide(1000);
  alert("The paragraph is now hidden");
});
```

# jQuery - Chaining

---

With jQuery, you can chain together actions/methods.

Chaining allows us to run multiple jQuery methods (on the same element) within a single statement.

---

## jQuery Method Chaining

Until now we have been writing jQuery statements one at a time (one after the other).

However, there is a technique called chaining, that allows us to run multiple jQuery commands, one after the other, on the same element(s).

**Tip:** This way, browsers do not have to find the same element(s) more than once.

To chain an action, you simply append the action to the previous action.

The following example chains together the css(), slideUp(), and slideDown() methods. The "p1" element first changes to red, then it slides up, and then it slides down:

## Example

```
$("#p1").css("color","red").slideUp(2000).slideDown(2000);
```

We could also have added more method calls if needed.

**Tip**: When chaining, the line of code could become quite long. However, jQuery is not very strict on the syntax; you can format it like you want, including line breaks and indentations.

This also works just fine:

## Example

```
$("#p1").css("color","red")
  .slideUp(2000)
  .slideDown(2000);
```

# jQuery - Get Content and Attributes

jQuery contains powerful methods for changing and manipulating HTML elements and attributes.

---

## jQuery DOM Manipulation

One very important part of jQuery, is the possibility to manipulate the DOM.

jQuery comes with a bunch of DOM related methods, that makes it easy to access and manipulate elements and attributes.

**DOM = Document Object Model**

The DOM defines a standard for accessing HTML and XML documents:

*"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

---

# Get Content - text(), html(), and val()

Three simple, but useful, jQuery methods for DOM manipulation is:

- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields

The following example demonstrates how to get content with the jQuery text() and html() methods:

# Example

```
$("#btn1").click(function(){
  alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
  alert("HTML: " + $("#test").html());
});
```

The following example demonstrates how to get the value of an input field with the jQuery val() method:

# Example

```
$("#btn1").click(function(){
  alert("Value: " + $("#test").val());
});
```

# Get Attributes - attr()

The jQuery attr() method is used to get attribute values.

The following example demonstrates how to get the value of the href attribute in a link:

## Example

```
$("button").click(function(){
  alert($("#w3s").attr("href"));
});
```

# jQuery - Set Content and Attributes

## Set Content - text(), html(), and val()

We will use the same three methods from the previous page to **set content**:

- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields

The following example demonstrates how to set content with the jQuery text(), html(), and val() methods:

## Example

```
$("#btn1").click(function(){
  $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
  $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
  $("#test3").val("Dolly Duck");
});
```

---

## A Callback Function for text(), html(), and val()

All of the three jQuery methods above: text(), html(), and val(), also come with a callback function. The callback function has two parameters: the index of the current element in the list of elements selected and the original (old) value. You then return the string you wish to use as the new value from the function.

The following example demonstrates text() and html() with a callback function:

## Example

```
$("#btn1").click(function(){
  $("#test1").text(function(i,origText){
    return "Old text: " + origText + " New text: Hello world!
    (index: " + i + ")";
  });
});

$("#btn2").click(function(){
  $("#test2").html(function(i,origText){
    return "Old html: " + origText + " New html: Hello <b>world!</b>
    (index: " + i + ")";
  });
});
```

---

## Set Attributes - attr()

The jQuery attr() method is also used to set/change attribute values.

The following example demonstrates how to change (set) the value of the href attribute in a link:

## Example

```
$("button").click(function(){
  $("#w3s").attr("href","http://www.w3schools.com/jquery");
});
```

The attr() method also allows you to set multiple attributes at the same time.

The following example demonstrates how to set both the href and title attributes at the same time:

## Example

```
$("button").click(function(){
  $("#w3s").attr({
    "href" : "http://www.w3schools.com/jquery",
    "title" : "W3Schools jQuery Tutorial"
  });
});
```

---

## A Callback Function for attr()

The jQuery method attr(), also come with a callback function. The callback function has two parameters: the index of the current element in the list of elements selected and the original (old) attribute value. You then return the string you wish to use as the new attribute value from the function.

The following example demonstrates attr() with a callback function:

## Example

```
$("button").click(function(){
  $("#w3s").attr("href", function(i,origValue){
    return origValue + "/jquery";
  });
});
```

# jQuery - Add Elements

---

With jQuery, it is easy to add new elements/content.

---

## Add New HTML content

We will look at four jQuery methods that is used to add new content:

- append() - Inserts content at the end of the selected elements

- prepend() - Inserts content at the beginning of the selected elements
- after() - Inserts content after the selected elements
- before() - Inserts content before the selected elements

---

## jQuery append() Method

The jQuery append() method inserts content AT THE END of the selected HTML elements.

## Example

```
$("p").append("Some appended text.");
```

---

## jQuery prepend() Method

The jQuery prepend() method inserts content AT THE BEGINNING of the selected HTML elements.

## Example

```
$("p").prepend("Some prepended text.");
```

---

## Add Several New Elements With append() and prepend()

In both examples above, we have only inserted some text/HTML at the beginning/end of the selected HTML elements.

However, both the append() and prepend() methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML (like we have done in the examples above), with jQuery, or with JavaScript code and DOM elements.

In the following example, we create several new elements. The elements are created with text/HTML, jQuery, and JavaScript/DOM. Then we append the new elements to the text with the append() method (this would have worked for prepend() too) :

## Example

```
function appendText()
{
var txt1="<p>Text.</p>";              // Create element with HTML
var txt2=$("<p></p>").text("Text.");   // Create with jQuery
var txt3=document.createElement("p");  // Create with DOM
txt3.innerHTML="Text.";
$("p").append(txt1,txt2,txt3);        // Append the new elements
}
```

---

## jQuery after() and before() Methods

The jQuery after() method inserts content AFTER the selected HTML elements.

The jQuery before() method inserts content BEFORE the selected HTML elements.

## Example

```
$("img").after("Some text after");

$("img").before("Some text before");
```

---

## Add Several New Elements With after() and before()

Also, both the after() and before() methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML (like we have done in the example above), with jQuery, or with JavaScript code and DOM elements.

In the following example, we create several new elements. The elements are created with text/HTML, jQuery, and JavaScript/DOM. Then we insert the new elements to the text with the after() method (this would have worked for before() too) :

## Example

```
function afterText()
{
var txt1="<b>I </b>";                  // Create element with HTML
var txt2=$("<i></i>").text("love ");     // Create with jQuery
var txt3=document.createElement("big");  // Create with DOM
txt3.innerHTML="jQuery!";
```

```
$("img").after(txt1,txt2,txt3);         // Insert new elements after img
}
```

# jQuery - Remove Elements

With jQuery, it is easy to remove existing HTML elements.

---

## Remove Elements/Content

To remove elements and content, there are mainly two jQuery methods:

- remove() - Removes the selected element (and its child elements)
- empty() - Removes the child elements from the selected element

---

## jQuery remove() Method

The jQuery remove() method removes the selected element(s) and its child elements.

## Example

```
$("#div1").remove();
```

---

## jQuery empty() Method

The jQuery empty() method removes the child elements of the selected element(s).

## Example

```
$("#div1").empty();
```

---

## Filter the Elements to be Removed

The jQuery remove() method also accepts one parameter, which allows you to filter the elements to be removed.

The parameter can be any of the jQuery selector syntaxes.

The following example removes all <p> elements with class="italic":

## Example

```
$("p").remove(".italic");
```

# jQuery - Get and Set CSS Classes

---

With jQuery, it is easy to manipulate the CSS of elements.

---

## jQuery Manipulating CSS

jQuery has several methods for CSS manipulation. We will look at the following methods:

- addClass() - Adds one or more classes to the selected elements
- removeClass() - Removes one or more classes from the selected elements
- toggleClass() - Toggles between adding/removing classes from the selected elements
- css() - Sets or returns the style attribute

---

## Example Stylesheet

The following stylesheet will be used for all the examples on this page:

```
.important
{
font-weight:bold;
font-size:xx-large;
}

.blue
{
color:blue;
}
```

## jQuery addClass() Method

The following example shows how to add class attributes to different elements. Of course you can select multiple elements, when adding classes:

## Example

```
$("button").click(function(){
  $("h1,h2,p").addClass("blue");
  $("div").addClass("important");
});
```

You can also specify multiple classes within the addClass() method:

## Example

```
$("button").click(function(){
  $("#div1").addClass("important blue");
});
```

## jQuery removeClass() Method

The following example shows how to remove a specific class attribute from different elements:

## Example

```
$("button").click(function(){
  $("h1,h2,p").removeClass("blue");
});
```

## jQuery toggleClass() Method

The following example will show how to use the jQuery toggleClass() method. This method toggles between adding/removing classes from the selected elements:

## Example

```
$("button").click(function(){
  $("h1,h2,p").toggleClass("blue");
});
```

---

## jQuery css() Method

The jQuery css() method will be explained in the next chapter.

---

# jQuery - css() Method

---

## jQuery css() Method

The css() method sets or returns one or more style properties for the selected elements.

---

## Return a CSS Property

To return the value of a specified CSS property, use the following syntax:

css("*propertyname*");

The following example will return the background-color value of the FIRST matched element:

## Example

```
$("p").css("background-color");
```

---

## Set a CSS Property

To set a specified CSS property, use the following syntax:

css("*propertyname*","*value*");

The following example will set the background-color value for ALL matched elements:

## Example

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

---

## Set Multiple CSS Properties

To set multiple CSS properties, use the following syntax:

css({"*propertyname*":"*value*","*propertyname*":"*value*",...});

The following example will set a background-color and a font-size for ALL matched elements:

## Example

```
$("p").css({"background-color":"yellow","font-size":"200%"});
```

# jQuery - AJAX Introduction

---

AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.

---

## What is AJAX?

AJAX = Asynchronous JavaScript and XML.

In short; AJAX is about loading data in the background and display it on the webpage, without reloading the whole page.

Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.

You can learn more about AJAX in our AJAX tutorial.

---

## What About jQuery and AJAX?

jQuery provides several methods for AJAX functionality.

With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post - And you can load the external data directly into the selected HTML elements of your web page!

**Without jQuery, AJAX coding can be a bit tricky!**

Writing regular AJAX code can be a bit tricky, because different browsers have different syntax for AJAX implementation. This means that you will have to write extra code to test for different browsers. However, the jQuery team has taken care of this for us, so that we can write AJAX functionality with only one single line of code.

---

## jQuery AJAX Methods

In the next chapters we will look at the most important jQuery AJAX methods.

# jQuery - AJAX load() Method

---

## jQuery load() Method

The jQuery load() method is a simple, but powerful AJAX method.

The load() method loads data from a server and puts the returned data into the selected element.

**Syntax:**

$(*selector*).load(*URL,data,callback*);

The required URL parameter specifies the URL you wish to load.

The optional data parameter specifies a set of querystring key/value pairs to send along with the request.

The optional callback parameter is the name of a function to be executed after the load() method is completed.

**Here is the content of our example file: "demo_test.txt":**

<h2>jQuery and AJAX is FUN!!!</h2>
This is some text in a paragraph.

The following example loads the content of the file "demo_test.txt" into a specific  element:

# Example

$("#div1").load("demo_test.txt");

It is also possible to add a jQuery selector to the URL parameter.

The following example loads the content of the element with id="p1", inside the file "demo_test.txt", into a specific  element:

# Example

$("#div1").load("demo_test.txt #p1");

The optional callback parameter specifies a callback function to run when the load() method is completed. The callback function can have different parameters:

- responseTxt - contains the resulting content if the call succeed
- statusTXT - contains the status of the call
- xhr - contains the XMLHttpRequest object

The following example displays an alert box after the load method() completes. If the load() method has succeed, it displays "External content loaded successfully!", and if it fails it displays an error message:

# Example

```
$("button").click(function(){
  $("#div1").load("demo_test.txt",function(responseTxt,statusTxt,xhr){
    if(statusTxt=="success")
      alert("External content loaded successfully!");
    if(statusTxt=="error")
      alert("Error: "+xhr.status+": "+xhr.statusText);
  });
});
```

# jQuery - AJAX get() and post() Methods

---

The jQuery get() and post() methods is used to request data from the server with an HTTP GET or POST request.

---

## HTTP Request: GET vs. POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

GET is basically used for just getting (retrieving) some data from the server. **Note:** The GET method may return cached data.

POST can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request.

To learn more about GET and POST, and the differences between the two methods, please read our HTTP Methods GET vs POST chapter.

---

## jQuery $.get() Method

The $.get() method requests data from the server with an HTTP GET request.

**Syntax:**

$.get(*URL,callback*);

The required URL parameter specifies the URL you wish to request.

The optional callback parameter is the name of a function to be executed if the request succeeds.

The following example uses the $.get() method to retrieve data from a file on the server:

## Example

```
$("button").click(function(){
  $.get("demo_test.asp",function(data,status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

The first parameter of $.get() is the URL we wish to request ("demo_test.asp").

The second parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

**Tip:** Here is how the ASP file looks like ("demo_test.asp"):

```
<%
response.write("This is some text from an external ASP file.")
%>
```

# jQuery $.post() Method

The $.post() method requests data from the server using an HTTP POST request.

**Syntax:**

$.post(*URL,data,callback*);

The required URL parameter specifies the URL you wish to request.

The optional data parameter specifies some data to send along with the request.

The optional callback parameter is the name of a function to be executed if the request succeeds.

The following example uses the $.post() method to send some data along with the request:

## Example

```
$("button").click(function(){
  $.post("demo_test_post.asp",
  {
    name:"Donald Duck",
    city:"Duckburg"
  },
  function(data,status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

The first parameter of $.post() is the URL we wish to request ("demo_test_post.asp").

Then we pass in some data to send along with the request (name and city).

The ASP script in "demo_test_post.asp" reads the parameters, process them, and return a result.

The third parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

**Tip:** Here is how the ASP file looks like ("demo_test_post.asp"):

```
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Dear " & fname & ". ")
Response.Write("Hope you live well in " & city & ".")
%>
```

# jQuery - The noConflict() Method

---

What if you wish to use other frameworks on your pages, while still using jQuery?

---

# jQuery and Other JavaScript Frameworks

As you already know; jQuery uses the $ sign as a shortcut for jQuery.

**What if other JavaScript frameworks also use the $ sign as a shortcut?**

Some other popular JavaScript frameworks are: MooTools, Backbone, Sammy, Cappuccino, Knockout, JavaScript MVC, Google Web Toolkit, Google Closure, Ember, Batman, and Ext JS.

Some of the other frameworks also use the $ character as a shortcut (just like jQuery), and then you suddenly have two different frameworks using the same shortcut, which might result in that your scripts stop working.

The jQuery team have already thought about this, and implemented the noConflict() method.

---

# The jQuery noConflict() Method

The noConflict() method releases the hold on the $ shortcut identifier, so that other scripts can use it.

You can of course still use jQuery, simply by writing the full name instead of the shortcut:

# Example

```
$.noConflict();
jQuery(document).ready(function(){
  jQuery("button").click(function(){
    jQuery("p").text("jQuery is still working!");
  });
});
```

You can also create your own shortcut very easily. The noConflict() method returns a reference to jQuery, that you can save in a  variable, for later use. Here is an example:

# Example

```
var jq = $.noConflict();
jq(document).ready(function(){
  jq("button").click(function(){
```

```
  jq("p").text("jQuery is still working!");
 });
});
```

If you have a block of jQuery code which uses the $ shortcut and you do not want to change it all, you can pass the $ sign in as a parameter to the ready method. This allows you to access jQuery using $, inside this function - outside of it, you will have to use "jQuery":

## Example

```
$.noConflict();
jQuery(document).ready(function($){
 $("button").click(function(){
  $("p").text("jQuery is still working!");
 });
});
```

---