jQuery Introduction

« Previous

Next Chapter »

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

What You Should Already Know

Before you start studying jQuery, you should have a basic knowledge of:

- HTML
- CSS
- JavaScript

If you want to study these subjects first, find the tutorials on our Home page.

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 require many lines of JavaScript code to accomplish, and wraps them 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 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, such as:

- Google
- Microsoft
- IBM
- Netflix

Will jQuery work in all browsers?

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

jQuery Get Started

« Previous

Next Chapter »

Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google

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.

The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section):

```
<head>
<script src="jquery-1.12.4.min.js"></script>
</head>
```

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

> **Do you wonder why we do not have type="text/javascript" inside the <scrip**
>
> This is not required in HTML5. JavaScript is the default scripting language in HTML5 a

---

jQuery CDN

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 or Microsoft, use one of the following:

Google CDN:

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
```

Try it Yourself »

Microsoft CDN:

```
<head>
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.12.4.min.js"></script>
</head>
```

Try it Yourself »

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

Many users already have downloaded jQuery from Google or Microsoft when visiting a
be loaded from cache when they visit your site, which leads to faster loading time. Als
that once a user requests a file from it, it will be served from the server closest to the
loading time.

jQuery Syntax

« Previous

Next Chapter »

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 performing 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 syn
> 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

« Previous

Next Chapter »

---

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).

jQuery selectors are used to "find" (or select) HTML elements based on their name, 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 selectors in jQuery start with the dollar sign and parentheses: $().

The element Selector

The jQuery element selector selects elements based on the element name.

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();
    });
});
```

Try it Yourself »

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 HTML 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();
    });
});
```

Try it Yourself »

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();
    });
});
```

Try it Yourself »

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 element of the first
$("ul li:first-child")	Selects the first element of every
$("[href]")	Selects all elements with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value eq
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT e
$(":button")	Selects all <button> elements and <input> elements of ty
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

Use our jQuery Selector Tester to demonstrate the different selectors.

For a complete reference of all the jQuery selectors, please go to our jQuery Selectors Reference.

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="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js">
</script>
<script src="my_jquery_functions.js"></script>
</head>
```

ing the code.

jQuery Event Methods

« Previous

Next Chapter »

jQuery is tailor-made to respond to events in an HTML page.

What are Events?

All the different visitor's actions that a web page can respond to are called events.

An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

The term **"fires/fired"** is often used with events. Example: "The keypress event is fired, the moment you press a key".

Here are some common DOM events:

Mouse Events	Keyboard Events	Form Events	Docume
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

---

jQuery Syntax For Event Methods

In jQuery, most DOM events have an equivalent jQuery method.

To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click();
```

The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function(){
  // action goes here!!
});
```

Commonly Used jQuery Event Methods

$(document).ready()

The $(document).ready() method allows us to execute a function when the document is fully loaded. This event is already explained in the jQuery Syntax chapter.

click()

The click() method attaches an event handler function to an HTML element.

The function is executed when the user clicks on the HTML element.

The following example says: When a click event fires on a <p> element; hide the current <p> element:

Example

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

Try it Yourself »

dblclick()

The dblclick() method attaches an event handler function to an HTML element.

The function is executed when the user double-clicks on the HTML element:

Example

```
$("p").dblclick(function(){
    $(this).hide();
});
```

Try it Yourself »

mouseenter()

The mouseenter() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer enters the HTML element:

Example

```
$("#p1").mouseenter(function(){
    alert("You entered p1!");
});
```

Try it Yourself »

mouseleave()

The mouseleave() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer leaves the HTML element:

Example

```
$("#p1").mouseleave(function(){
    alert("Bye! You now leave p1!");
});
```

Try it Yourself »

mousedown()

The mousedown() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element:

Example

```
$("#p1").mousedown(function(){
    alert("Mouse down over p1!");
});
```

Try it Yourself »

mouseup()

The mouseup() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is released, while the mouse is over the HTML element:

Example

```
$("#p1").mouseup(function(){
    alert("Mouse up over p1!");
});
```

Try it Yourself »

hover()

The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.

The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element:

Example

```
$("#p1").hover(function(){
    alert("You entered p1!");
},
function(){
    alert("Bye! You now leave p1!");
});
```

Try it Yourself »

focus()

The focus() method attaches an event handler function to an HTML form field.

The function is executed when the form field gets focus:

Example

```
$("input").focus(function(){
    $(this).css("background-color", "#cccccc");
});
```

Try it Yourself »

blur()

The blur() method attaches an event handler function to an HTML form field.

The function is executed when the form field loses focus:

Example

```
$("input").blur(function(){
    $(this).css("background-color", "#ffffff");
});
```

Try it Yourself »

The on() Method

The on() method attaches one or more event handlers for the selected elements.

Attach a click event to a <p> element:

Example

```
$("p").on("click", function(){
    $(this).hide();
});
```

Try it Yourself »

Attach multiple event handlers to a <p> element:

Example

```
$("p").on({
    mouseenter: function(){
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function(){
        $(this).css("background-color", "lightblue");
    },
    click: function(){
        $(this).css("background-color", "yellow");
    }
});
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 »

jQuery Event Methods

For a full jQuery event reference, please go to our jQuery Events Reference.

jQuery Effects - Hide and Show

« Previous

Next Chapter »

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

Click to show/hide panel

Examples

jQuery hide()
Demonstrates a simple jQuery hide() method.

jQuery hide()
Another hide() demonstration. How to hide parts of text.

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();
});
```

Try it Yourself »

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 a function to be executed after the hide() or show() method completes (you will learn more about callback functions in a later chapter).

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

Example

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

Try it Yourself »

---

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();
});
```

Try it Yourself »

Syntax:

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

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

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

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

---

jQuery Effects Reference

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

jQuery Effects - Fading

« Previous

Next Chapter »

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

Click to fade in/out panel

Examples

jQuery fadeIn()
Demonstrates the jQuery fadeIn() method.

jQuery fadeOut()
Demonstrates the jQuery fadeOut() method.

jQuery fadeToggle()
Demonstrates the jQuery fadeToggle() method.

jQuery fadeTo()
Demonstrates the jQuery fadeTo() method.

---

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 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);
});
```

Try it Yourself »

---

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 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);
});
```

Try it Yourself »

---

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 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);
});
```

Try it Yourself »

---

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 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);
});
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 »

---

jQuery Effects Reference

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

jQuery Effects - Sliding

« Previous

Next Chapter »

The jQuery slide methods slide elements up and down.

Click to slide down/up the panel

Examples

jQuery slideDown()
Demonstrates the jQuery slideDown() method.

jQuery slideUp()
Demonstrates the jQuery slideUp() method.

jQuery slideToggle()
Demonstrates the jQuery SlideToggle() method.

---

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 a function to be executed after the sliding completes.

The following example demonstrates the slideDown() method:

Example

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

Try it Yourself »

---

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 a function to be executed after the sliding completes.

The following example demonstrates the slideUp() method:

Example

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

Try it Yourself »

---

jQuery slideToggle() Method

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

If the elements have been slid down, slideToggle() will slide them up.

If the elements have been slid 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 a function to be executed after the sliding completes.

The following example demonstrates the slideToggle() method:

Example

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

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

---

jQuery Effects Reference

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

jQuery Effects - Animation

« Previous

Next Chapter »

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

Start Animation

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 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 right, until it has reached a left property of 250px:

Example

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

Try it Yourself »

> 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 ele
> 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'
    });
});
```

Try it Yourself »

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 used with the animate() method: You will need to write paddingLeft instead of paddin 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 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'
    });
});
```

Try it Yourself »

---

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'
    });
});
```

Try it Yourself »

---

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");
});
```

Try it Yourself »

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");
});
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

---

jQuery Effects Reference

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

jQuery Stop Animations

« Previous

Next Chapter »

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

Start sliding Stop sliding

Click to slide down/up the panel

Examples

jQuery stop() sliding
Demonstrates the jQuery stop() method.

jQuery stop() animation (with parameters)
Demonstrates the jQuery stop() method.

---

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();
});
```

Try it Yourself »

---

jQuery Effects Reference

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

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");
    });
});
```

Try it Yourself »

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");
});
```

Try it Yourself »

jQuery - Chaining

« Previous

Next Chapter »

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);
```

Try it Yourself »

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);
```

Try it Yourself »

jQuery throws away extra whitespace and executes the lines above as one long line of code.

jQuery - Get Content and Attributes

« Previous

Next Chapter »

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 make 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 interfac*
> *scripts to dynamically access and update the content, structure, and style of a docum*

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

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

- 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());
});
```

Try it Yourself »

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());
});
```

Try it Yourself »

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"));
});
```

Try it Yourself »

The next chapter explains how to set (change) content and attribute values.

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

---

jQuery HTML Reference

For a complete overview of all jQuery HTML methods, please go to our jQuery HTML/CSS Reference.

jQuery - Set Content and Attributes

« Previous

Next Chapter »

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");
});
```

Try it Yourself »

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 + ")";
```

```
    });
});
```

Try it Yourself »

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");
});
```

Try it Yourself »

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"
    });
});
```

Try it Yourself »

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";
    });
});
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 »

jQuery HTML Reference

For a complete overview of all jQuery HTML methods, please go to our jQuery HTML/CSS Reference.

jQuery - Add Elements

« Previous

Next Chapter »

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

Add New HTML Content

We will look at four jQuery methods that are 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.");
```

Try it Yourself »

---

jQuery prepend() Method

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

Example

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

Try it Yourself »

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.";
    $("body").append(txt1, txt2, txt3);      // Append the new elements
}
```

Try it Yourself »

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");
```

Try it Yourself »

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("b");     // Create with DOM
    txt3.innerHTML = "jQuery!";
    $("img").after(txt1, txt2, txt3);           // Insert new
elements after <img>
}
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

---

jQuery HTML Reference

For a complete overview of all jQuery HTML methods, please go to our jQuery HTML/CSS Reference.

jQuery - Remove Elements

« Previous

Next Chapter »

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();
```

Try it Yourself »

---

jQuery empty() Method

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

Example

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

Try it Yourself »

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="test":

Example

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

Try it Yourself »

This example removes all <p> elements with class="test" and class="demo":

Example

```
$("p").remove(".test, .demo");
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

---

jQuery HTML Reference

For a complete overview of all jQuery HTML methods, please go to our jQuery HTML/CSS Reference.

jQuery - Get and Set CSS Classes

« Previous

Next Chapter »

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

Toggle class

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");
});
```

Try it Yourself »

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

Example

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

Try it Yourself »

---

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");
});
```

Try it Yourself »

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");
});
```

Try it Yourself »

jQuery css() Method

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

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

jQuery CSS Reference

For a complete overview of all jQuery CSS methods, please go to our jQuery HTML/CSS Reference.

jQuery - css() Method

« Previous

Next Chapter »

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");
```

Try it Yourself »

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");
```

Try it Yourself »

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%"})
;
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

---

jQuery CSS Reference

For a complete overview of all jQuery CSS methods, please go to our jQuery HTML/CSS Reference.

jQuery - Dimensions

« Previous

Next Chapter »

With jQuery, it is easy to work with the dimensions of elements and browser window.

---

jQuery Dimension Methods

jQuery has several important methods for working with dimensions:

- width()
- height()
- innerWidth()
- innerHeight()
- outerWidth()
- outerHeight()

---

jQuery Dimensions

---

jQuery width() and height() Methods

The width() method sets or returns the width of an element (excludes padding, border and margin).

The height() method sets or returns the height of an element (excludes padding, border and margin).

The following example returns the width and height of a specified element:

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Width: " + $("#div1").width() + "</br>";
    txt += "Height: " + $("#div1").height();
    $("#div1").html(txt);
});
```

Try it Yourself »

---

jQuery innerWidth() and innerHeight() Methods

The innerWidth() method returns the width of an element (includes padding).

The innerHeight() method returns the height of an element (includes padding).

The following example returns the inner-width/height of a specified element:

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Inner width: " + $("#div1").innerWidth() + "</br>";
    txt += "Inner height: " + $("#div1").innerHeight();
    $("#div1").html(txt);
});
```

Try it Yourself »

---

jQuery outerWidth() and outerHeight() Methods

The outerWidth() method returns the width of an element (includes padding and border).

The outerHeight() method returns the height of an element (includes padding and border).

The following example returns the outer-width/height of a specified element:

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Outer width: " + $("#div1").outerWidth() + "</br>";
    txt += "Outer height: " + $("#div1").outerHeight();
    $("#div1").html(txt);
});
```

Try it Yourself »

The outerWidth(true) method returns the width of an element (includes padding, border, and margin).

The outerHeight(true) method returns the height of an element (includes padding, border, and margin).

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Outer width (+margin): " + $("#div1").outerWidth(true)
+ "</br>";
    txt += "Outer height (+margin): " +
$("#div1").outerHeight(true);
    $("#div1").html(txt);
});
```

Try it Yourself »

---

jQuery More width() and height()

The following example returns the width and height of the document (the HTML document) and window (the browser viewport):

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Document width/height: " + $(document).width();
    txt += "x" + $(document).height() + "\n";
    txt += "Window width/height: " + $(window).width();
    txt += "x" + $(window).height();
    alert(txt);
});
```

Try it Yourself »

The following example sets the width and height of a specified element:

Example

```
$("button").click(function(){
    $("#div1").width(500).height(500);
});
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 »

---

jQuery CSS Reference

For a complete overview of all jQuery CSS methods, please go to our jQuery HTML/CSS Reference.

jQuery Traversing

« Previous

Next Chapter »

What is Traversing?

jQuery traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until you reach the elements you desire.

The image below illustrates a family tree. With jQuery traversing, you can easily move up (ancestors), down (descendants) and sideways (siblings) in the family tree, starting from the selected (current) element. This movement is called traversing - or moving through - the DOM.

Illustration explained:

- The element is the **parent** of <ul>, and an **ancestor** of everything inside of it
- The <ul> element is the **parent** of both <li> elements, and a **child** of
- The left <li> element is the **parent** of , **child** of <ul> and a **descendant** of
- The element is a **child** of the left <li> and a **descendant** of <ul> and
- The two <li> elements are **siblings** (they share the same parent)
- The right <li> element is the **parent** of <b>, **child** of <ul> and a **descendant** of
- The <b> element is a **child** of the right <li> and a **descendant** of <ul> and

> An ancestor is a parent, grandparent, great-grandparent, and so on.
> A descendant is a child, grandchild, great-grandchild, and so on.
> Siblings share the same parent.

Traversing the DOM

jQuery provides a variety of methods that allows us to traverse the DOM.

The largest category of traversal methods are tree-traversal.

The next chapters will show us how to travel up, down and sideways in the DOM tree.

---

jQuery Traversing Reference

For a complete overview of all jQuery Traversing methods, please go to our jQuery Traversing Reference.

jQuery Traversing - Ancestors

« Previous

Next Chapter »

An ancestor is a parent, grandparent, great-grandparent, and so on.

With jQuery you can traverse up the DOM tree to find ancestors of an element.

Traversing Up the DOM Tree

Three useful jQuery methods for traversing up the DOM tree are:

- parent()
- parents()
- parentsUntil()

jQuery parent() Method

The parent() method returns the direct parent element of the selected element.

This method only traverse a single level up the DOM tree.

The following example returns the direct parent element of each elements:

Example

```
$(document).ready(function(){
    $("span").parent();
});
```

Try it Yourself »

---

jQuery parents() Method

The parents() method returns all ancestor elements of the selected element, all the way up to the document's root element (<html>).

The following example returns all ancestors of all elements:

Example

```
$(document).ready(function(){
    $("span").parents();
});
```

Try it Yourself »

You can also use an optional parameter to filter the search for ancestors.

The following example returns all ancestors of all elements that are <ul> elements:

Example

```
$(document).ready(function(){
    $("span").parents("ul");
});
```

Try it Yourself »

---

jQuery parentsUntil() Method

The parentsUntil() method returns all ancestor elements between two given arguments.

The following example returns all ancestor elements between a and a element:

Example

```
$(document).ready(function(){
    $("span").parentsUntil("div");
});
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

---

jQuery Traversing Reference

For a complete overview of all jQuery Traversing methods, please go to our jQuery Traversing Reference.

jQuery Traversing - Descendants

« Previous

Next Chapter »

A descendant is a child, grandchild, great-grandchild, and so on.

With jQuery you can traverse down the DOM tree to find descendants of an element.

Traversing Down the DOM Tree

Two useful jQuery methods for traversing down the DOM tree are:

- children()
- find()

---

jQuery children() Method

The children() method returns all direct children of the selected element.

This method only traverse a single level down the DOM tree.

The following example returns all elements that are direct children of each elements:

Example

```
$(document).ready(function(){
    $("div").children();
});
```

Try it Yourself »

You can also use an optional parameter to filter the search for children.

The following example returns all <p> elements with the class name "first", that are direct children of :

Example

```
$(document).ready(function(){
    $("div").children("p.first");
});
```

Try it Yourself »

---

jQuery find() Method

The find() method returns descendant elements of the selected element, all the way down to the last descendant.

The following example returns all elements that are descendants of :

Example

```
$(document).ready(function(){
    $("div").find("span");
});
```

Try it Yourself »

The following example returns all descendants of :

Example

```
$(document).ready(function(){
    $("div").find("*");
});
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

---

jQuery Traversing Reference

For a complete overview of all jQuery Traversing methods, please go to our jQuery Traversing Reference.

jQuery Traversing - Siblings

« Previous

Next Chapter »

Siblings share the same parent.

With jQuery you can traverse sideways in the DOM tree to find siblings of an element.

Traversing Sideways in The DOM Tree

There are many useful jQuery methods for traversing sideways in the DOM tree:

- siblings()
- next()
- nextAll()
- nextUntil()
- prev()
- prevAll()
- prevUntil()

jQuery siblings() Method

The siblings() method returns all sibling elements of the selected element.

The following example returns all sibling elements of <h2>:

Example

```
$(document).ready(function(){
    $("h2").siblings();
});
```

Try it Yourself »

You can also use an optional parameter to filter the search for siblings.

The following example returns all sibling elements of <h2> that are <p> elements:

Example

```
$(document).ready(function(){
    $("h2").siblings("p");
});
```

Try it Yourself »

jQuery next() Method

The next() method returns the next sibling element of the selected element.

The following example returns the next sibling of <h2>:

Example

```
$(document).ready(function(){
    $("h2").next();
});
```

Try it Yourself »

---

jQuery nextAll() Method

The nextAll() method returns all next sibling elements of the selected element.

The following example returns all next sibling elements of <h2>:

Example

```
$(document).ready(function(){
    $("h2").nextAll();
});
```

Try it Yourself »

---

jQuery nextUntil() Method

The nextUntil() method returns all next sibling elements between two given arguments.

The following example returns all sibling elements between a <h2> and a <h6> element:

Example

```
$(document).ready(function(){
    $("h2").nextUntil("h6");
});
```

Try it Yourself »

---

jQuery prev(), prevAll() & prevUntil() Methods

The prev(), prevAll() and prevUntil() methods work just like the methods above but with reverse functionality: they return previous sibling elements (traverse backwards along sibling elements in the DOM tree, instead of forward).

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 » Exercise 6 »

---

jQuery Traversing Reference

For a complete overview of all jQuery Traversing methods, please go to our jQuery Traversing Reference.

jQuery Traversing - Filtering

« Previous

Next Chapter »

Narrow Down The Search For Elements

The three most basic filtering methods are first(), last() and eq(), which allow you to select a specific element based on its position in a group of elements.

Other filtering methods, like filter() and not() allow you to select elements that match, or do not match, a certain criteria.

---

jQuery first() Method

The first() method returns the first element of the selected elements.

The following example selects the first <p> element inside the first element:

Example

```
$(document).ready(function(){
    $("div p").first();
});
```

Try it Yourself »

---

jQuery last() Method

The last() method returns the last element of the selected elements.

The following example selects the last <p> element inside the last element:

Example

```
$(document).ready(function(){
    $("div p").last();
});
```

Try it Yourself »

---

jQuery eq() method

The eq() method returns an element with a specific index number of the selected elements.

The index numbers start at 0, so the first element will have the index number 0 and not 1. The following example selects the second <p> element (index number 1):

Example

```
$(document).ready(function(){
    $("p").eq(1);
});
```

Try it Yourself »

---

jQuery filter() Method

The filter() method lets you specify a criteria. Elements that do not match the criteria are removed from the selection, and those that match will be returned.

The following example returns all <p> elements with class name "intro":

Example

```
$(document).ready(function(){
    $("p").filter(".intro");
});
```

Try it Yourself »

---

jQuery not() Method

The not() method returns all elements that do not match the criteria.

Tip: The not() method is the opposite of filter().

The following example returns all <p> elements that do not have class name "intro":

Example

```
$(document).ready(function(){
    $("p").not(".intro");
});
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 » Exercise 6 »

---

jQuery Traversing Reference

For a complete overview of all jQuery Traversing methods, please go to our jQuery Traversing Reference.

jQuery - AJAX Introduction

« Previous

Next Chapter »

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

jQuery AJAX Example

Let jQuery AJAX Change This Text

```
Get External Content
```

Try it Yourself »

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 differen for AJAX implementation. This means that you will have to write extra code to test for browsers. However, the jQuery team has taken care of this for us, so that we can wri 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

« Previous

Next Chapter »

---

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>
<p id="p1">This is some text in a paragraph.</p>
```

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

Example

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

Try it Yourself »

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");
```

Try it Yourself »

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 succeeds
- 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 succeeded, 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);
    });
});
```

Try it Yourself »

---

jQuery AJAX Reference

For a complete overview of all jQuery AJAX methods, please go to our jQuery AJAX Reference.

jQuery - AJAX get() and post() Methods

« Previous

Next Chapter »

The jQuery get() and post() methods are 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 ourHTTP 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);
    });
});
```

Try it Yourself »

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);
    });
});
```

Try it Yourself »

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, processes them, and returns 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 AJAX Reference

For a complete overview of all jQuery AJAX methods, please go to our jQuery AJAX Reference.

jQuery - The noConflict() Method

« Previous

Next Chapter »

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.

There are many other popular JavaScript frameworks like: Angular, Backbone, Ember, Knockout, and more.

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

If two different frameworks are using the same shortcut, one of them might 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!");
    });
});
```

Try it Yourself »

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!");
    });
});
```

Try it Yourself »

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!");
    });
});
```

Try it Yourself »

---

jQuery Misc Reference

For a complete overview of all jQuery Misc methods, please go to our jQuery Misc Reference.

jQuery Examples

芦 Previous

Next Chapter 禄

Do you want to develop your jQuery selector skills?

Please try our jQuery Selector Tester

---

jQuery Selectors

$("p").hide()
Demonstrates the jQuery hide() method, hiding all <p> elements.

$("#test").hide()
Demonstrates the jQuery hide() method, hiding the element with id="test".

$(".test").hide()
Demonstrates the jQuery hide() method, hiding all elements with class="test".

$(this).hide()
Demonstrates the jQuery hide() method, hiding the current HTML element.

Examples explained

---

jQuery Events

jQuery click()
Demonstrates the jQuery click() event.

jQuery dblclick()
Demonstrates the jQuery dblclick() event.

jQuery mouseenter()
Demonstrates the jQuery mouseenter() event.

jQuery mouseleave()
Demonstrates the jQuery mouseleave() event.

jQuery mousedown()
Demonstrates the jQuery mousedown() event.

jQuery mouseup()
Demonstrates the jQuery mouseup() event.

jQuery hover()
Demonstrates the jQuery hover() event.

jQuery focus() and blur()
Demonstrates the jQuery focus() and blur() events.

Examples explained

---

jQuery Hide/Show

jQuery hide()
Demonstrates the jQuery hide() method.

jQuery hide() and show()
Demonstrates jQuery hide() and show() methods.

jQuery toggle()
jQuery toggle() toggles between hide() and show().

jQuery hide()
Another hide demonstration. How to hide parts of text.

Examples explained

---

jQuery Fade

jQuery fadeIn()
Demonstrates the jQuery fadeIn() method.

jQuery fadeOut()
Demonstrates the jQuery fadeOut() method.

jQuery fadeToggle()
Demonstrates the jQuery fadeToggle() method.

jQuery fadeTo()
Demonstrates the jQuery fadeTo() method.

Examples explained

---

jQuery Slide

jQuery slideDown()
Demonstrates the jQuery slideDown() method.

jQuery slideUp()
Demonstrates the jQuery slideUp() method.

jQuery slideToggle()
Demonstrates the jQuery slideToggle() method.

Examples explained

---

jQuery Animate

jQuery animate()
Demonstrates a simple use of the jQuery animate() method.

jQuery animate() - manipulate multiple CSS properties
Demonstrates that you can manipulate multiple CSS properties with the jQuery animate() method.

jQuery animate() - using relative values
Demonstrates that you can use relative values in the jQuery animate() method.

jQuery animate() - using pre-defined values
Demonstrates that you can use the pre-defined values "hide", "show", "toggle" in the jQuery animate() method.

jQuery animate()
Demonstrates more using the jQuery animate() method (several animate() calls after each other).

jQuery animate()
Demonstrates more using the jQuery animate() method (several animate() calls after each other).

Examples explained

jQuery Stop Animations

jQuery stop() sliding
Demonstrates the jQuery stop() method.

jQuery stop() animation (with parameters)
Demonstrates the jQuery stop() method.

Examples explained

jQuery HTML Get Content and Attributes

jQuery text() and html() - Get content
Get content with the jQuery text() and html() methods.

jQuery val() - Get content
Get the value of a form field with the jQuery val() method.

jQuery attr() - Get attribute value
Get the value of an attribute with the jQuery attr() method.

Examples explained

jQuery HTML Set Content and Attributes

jQuery text(), html(), and val() - Set content
Set content with the jQuery text(), html() and val() methods.

jQuery text() and html() - Set content with a callback function
Set content + using a callback function inside text() and html().

jQuery attr() - Set attribute value
Set attribute value with the jQuery attr() method.

jQuery attr() - Set multiple attribute values
Set multiple attribute values with the jQuery attr() method.

jQuery attr() - Set attribute value with a callback function
Set attribute value + using a callback function inside attr().

Examples explained

---

jQuery HTML Add Elements/Content

jQuery append()
Insert content at the end of the selected HTML elements.

jQuery prepend()
Insert content at the beginning of the selected HTML elements.

jQuery append() - Insert several new elements
Create new elements with text/HTML, jQuery, and JavaScript/DOM. Then append the new elements to the text.

jQuery after() and before()
Insert content after and before the selected HTML elements.

jQuery after() - Insert several new elements
Create new elements with text/HTML, jQuery, and JavaScript/DOM. Then insert the new elements after the selected element.

Examples explained

---

jQuery HTML Remove Elements/Content

jQuery remove()
Remove the selected element(s).

jQuery empty()
Remove all child elements of the selected element(s).

jQuery remove() - with a parameter
Filter the elements to be removed

Examples explained

---

jQuery Get and Set CSS Classes

jQuery addClass()
Add class attributes to different elements.

jQuery addClass() - Multiple classes
Specify multiple classes within the addClass() method.

jQuery removeClass()
Remove a specific class attribute from different elements.

jQuery toggleClass()
Toggle between adding/removing classes from the selected elements.

Examples explained

---

jQuery css() Method

jQuery css() - return CSS property
Return the value of a specified CSS property from the FIRST matched element.

jQuery css() - set CSS property
Set a specified CSS property for ALL matched elements.

jQuery css() - set CSS properties
Set multiple CSS properties for ALL matched elements.

Examples explained

jQuery Dimensions

jQuery - return width() and height()
Return the width and height of a specified element.

jQuery - return innerWidth() and innerHeight()
Return the inner-width/height of a specified element.

jQuery - return outerWidth() and outerHeight()
Return the outer-width/height of a specified element.

jQuery - return outerWidth(true) and outerHeight(true)
Return the outer-width/height (including margins) of a specified element.

jQuery - return width() and height() of document and window
Return the width and height of the document (the HTML document) and window (the browser viewport).

jQuery - set width() and height()
Sets the width and height of a specified element.

Examples explained

jQuery Traversing Ancestors

jQuery parent()
Demonstrates the jQuery parent() method.

jQuery parents()
Demonstrates the jQuery parents() method.

jQuery parentsUntil()
Demonstrates the jQuery parentsUntil() method.

Examples explained

jQuery Traversing Descendants

jQuery children()
Demonstrates the jQuery children() method.

jQuery find()
Demonstrates the jQuery find() method.

Examples explained

---

jQuery Traversing Siblings

jQuery siblings()
Demonstrates the jQuery siblings() method.

jQuery next()
Demonstrates the jQuery next() method.

jQuery nextAll()
Demonstrates the jQuery nextAll() method.

jQuery nextUntil()
Demonstrates the jQuery nextUntil() method.

Examples explained

---

jQuery Traversing Filtering

jQuery first()
Demonstrates the jQuery first() method.

jQuery last()
Demonstrates the jQuery last() method.

jQuery eq()
Demonstrates the jQuery eq() method.

jQuery filter()
Demonstrates the jQuery filter() method.

jQuery not()
Demonstrates the jQuery not() method.

Examples explained

---

jQuery AJAX load() Method

jQuery load()
Load the content of a file into a element.

jQuery load()
Load the content of a specific element inside a file, into a element.

jQuery load() - with callback
Use of the jQuery load() method with a callback function.

Examples explained

---

jQuery AJAX get() and post() Methods

jQuery get()
Use the $.get() method to retrieve data from a file on the server.

jQuery post()
Use the $.post() method to send some data along with the request.

Examples explained