<http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery>

**Tutorials:Getting Started with jQuery**

Goes through the basics of jQuery, all the way up to building plugins.

**Original:** <http://jquery.bassistance.de/jquery-getting-started.html>  
**Author:** [Jörn Zaefferer](http://bassistance.de/)

Similar Tutorials: [jQuery Core](http://docs.jquery.com/Category:JQuery_Core), [Selectors](http://docs.jquery.com/Category:Selectors), [Attributes](http://docs.jquery.com/Category:Attributes), [Traversing](http://docs.jquery.com/Category:Traversing), [Manipulation](http://docs.jquery.com/Category:Manipulation), [Events](http://docs.jquery.com/Category:Events), [Effects](http://docs.jquery.com/Category:Effects), [Ajax](http://docs.jquery.com/Category:Ajax), [Plugins](http://docs.jquery.com/Category:Plugins)

This guide is an introduction to the jQuery library. Basic knowledge of JavaScript and the document object model (DOM) is required. It starts from ground up and tries to explain details where necessary. It covers a simple hello world example, selector and event basics, AJAX, FX and usage and authoring of plugins.

This guide contains no "click me" examples. The intention of providing only "copy me" code is to invite you to try it for yourself. Copy an example, see what it does, and modify it.

|  |
| --- |
| **CONTENTS**   * [1 Setup](http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery#Setup) * [2 Hello jQuery](http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery#Hello_jQuery) * [3 Find me: Using selectors and events](http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery#Find_me:_Using_selectors_and_events) * [4 Rate me: Using Ajax](http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery#Rate_me:_Using_Ajax) * [5 Animate me: Using Effects](http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery#Animate_me:_Using_Effects) * [6 Sort me: Using the tablesorter plugin](http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery#Sort_me:_Using_the_tablesorter_plugin) * [7 Plug me: Writing your own plugins](http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery#Plug_me:_Writing_your_own_plugins) * [8 Next steps](http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery#Next_steps) |

**SETUP**

To start, we need a copy of the jQuery library, which we can get from the [main download page](http://docs.jquery.com/Downloading_jQuery). The[jQuery Starterkit](http://docs.jquery.com/images/c/c7/Jquery-starterkit.zip) provides some markup and CSS to work with. After downloading and extracting its content we put jquery.js into the same directory and open starterkit.html and custom.js with your favorite editor and starterkit.html with a browser.

Now we have everything to start with the notorious "Hello world" example.

**Interesting links for this section:**

* [jQuery Starterkit](http://docs.jquery.com/images/c/c7/Jquery-starterkit.zip)
* [Downloading jQuery](http://docs.jquery.com/Downloading_jQuery)

**HELLO JQUERY**

We start with an empty html page:

<html>

<head>

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">

// we will add our javascript code here

</script>

</head>

<body>

<!-- we will add our HTML content here -->

</body>

</html>

This page just loads the jquery.js library (make sure the URL points to where you stored your copy of jquery! This example assumes that you store it in the same directory as this example file). Two comments indicate where we will expand this template with code.

As almost everything we do when using jQuery reads or manipulates the document object model (DOM), we need to make sure that we start adding events etc. as soon as the DOM is ready.

To do this, we register a ready event for the document.

$(document).ready(function() {

// do stuff when DOM is ready

});

Putting an alert into that function does not make much sense, as an alert does not require the DOM to be loaded. So lets try something a little more sophisticated: Show an alert when clicking a link.

Add the following to the <body>:

<a href="">Link</a>

Now update the $(document).ready handler:

$(document).ready(function() {

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

alert("Hello world!");

});

});

This should show the alert as soon as you click on the link. You are ready now to copy and paste this script into your custom.js file. Then, open starterkit.html in the browser and click any link. You should see a pop-up window with "Hello world!" message regardless of what link was clicked.

Let's have a look at what we are doing: $("a") is a jQuery selector, in this case, it selects all **a**elements. $ itself is an alias for the jQuery "class", therefore $() constructs a new jQuery object. The click() function we call next is a method of the jQuery object. It binds a click event to all selected elements (in this case, a single anchor element) and executes the provided function when the event occurs.

This is similar to the following code:

<a href="" onclick="alert('Hello world')">Link</a>

The difference is quite obvious: We don't need to write an onclick for every single element. We have a clean separation of structure (HTML) and behavior (JS), just as we separate structure and presentation by using CSS.

With this in mind, we explore selectors and events a little further.

**Interesting links for this section:**

* [jQuery Base](http://docs.jquery.com/Core)
* [jQuery Expressions](http://docs.jquery.com/Selectors)
* [jQuery Basic Events](http://docs.jquery.com/Events)

**FIND ME: USING SELECTORS AND EVENTS**

jQuery provides two approaches to select elements. The first uses a combination of CSS and XPath selectors passed as a string to the jQuery constructor (eg. $("div > ul a")). The second uses several methods of the jQuery object. Both approaches can be combined.

To try some of these selectors, we select and modify the first ordered list in our starterkit.

To get started, we want to select the list itself. The list has an ID "orderedlist". In classic JavaScript, you could select it by using document.getElementById("orderedlist"). With jQuery, we do it like this:

$(document).ready(function() {

$("#orderedlist").addClass("red");

});

The starterkit provides a stylesheet with a class "red" that simply adds a red background. Therefore, when you reload the page in your browser, you should see that the first ordered list has a red background. The second list is not modified.

Now lets add some more classes to the child elements of this list.

$(document).ready(function() {

$("#orderedlist > li").addClass("blue");

});

This selects all child lis of the element with the id orderedlist and adds the class "blue".

Now for something a little more sophisticated: We want to add and remove the class when the user hovers the li element, but only on the last element in the list.

$(document).ready(function() {

$("#orderedlist li:last").hover(function() {

$(this).addClass("green");

},function(){

$(this).removeClass("green");

});

});

There are many other selectors similar to [CSS](http://docs.jquery.com/DOM/Traversing/Selectors#CSS_Selectors) and [XPath](http://docs.jquery.com/DOM/Traversing/Selectors#XPath_Selectors) syntax. More examples and a list of all available expressions can be found [here](http://docs.jquery.com/DOM/Traversing/Selectors).

For every onxxx event available, like onclick, onchange, onsubmit, there is a jQuery equivalent.[Some other events](http://docs.jquery.com/Events), like ready and hover, are provided as convenient methods for certain tasks.

You can find a complete list of all events in the [jQuery Events Documentation](http://docs.jquery.com/Events).

With those selectors and events you can already do a lot of things, but there is more.

$(document).ready(function() {

$("#orderedlist").find("li").each(function(i) {

$(this).append( " BAM! " + i );

});

});

find() allows you to further search the descendants of the already selected elements, therefore$("#orderedlist").find("li") is mostly the same as $("#orderedlist li").

each() iterates over every element and allows further processing. Most methods, likeaddClass(), use each() themselves.

In this example, append() is used to append some text to it and set it as text to the end of each element.

Another task you often face is to call methods on DOM elements that are not covered by jQuery. Think of a form you would like to reset after you submitted it successfully via AJAX.

$(document).ready(function() {

// use this to reset a single form

$("#reset").click(function() {

$("form")[0].reset();

});

});

This code selects the first form element and calls reset() on it. In case you had more than one form, you could also do this:

$(document).ready(function() {

// use this to reset several forms at once

$("#reset").click(function() {

$("form").each(function() {

this.reset();

});

});

});

This would select all forms within your document, iterate over them and call reset() for each. Note that in an .each() function, this refers to the actual element. Also note that, since thereset() function belongs to the form element and not to the jQuery object, we cannot simply call$("form").reset() to reset all the forms on the page.

An additional challenge is to select only certain elements from a group of similar or identical ones. jQuery provides filter() and not() for this. While filter() reduces the selection to the elements that fit the filter expression, not() does the contrary and removes all elements that fit the expression. Think of an unordered list where you want to select all li elements that have no ul children.

$(document).ready(function() {

$("li").not(":has(ul)").css("border", "1px solid black");

});

This selects all li elements that have a ul element as a child and removes all elements from the selection. Therefore all li elements get a border, except the one that has a child ul.

The [expression] syntax is taken from XPath and can be used to filter by attributes. Maybe you want to select all anchors that have a name attribute:

$(document).ready(function() {

$("a[name]").css("background", "#eee" );

});

This adds a background color to all anchor elements with a name attribute.

More often than selecting anchors by name, you might need to select anchors by their "href" attribute. This can be a problem as browsers behave quite inconsistently when returning what they think the "href" value is (Note: This problem was fixed recently in jQuery, available in any versions after 1.1.1). To match only a part of the value, we can use the contains select "\*=" instead of an equals ("="):

$(document).ready(function() {

$("a[href\*='/content/gallery']").click(function() {

// do something with all links that point somewhere to /content/gallery

});

});

Until now, all selectors were used to select children or filter the current selection. There are situations where you need to select the previous or next elements, known as siblings. Think of a FAQ page, where all answers are hidden first, and shown, when the question is clicked. The jQuery code for this:

$(document).ready(function() {

$('#faq').find('dd').hide().end().find('dt').click(function() {

$(this).next().slideToggle();

});

});

Here we use some chaining to reduce the code size and gain better performance, as '#faq' is only selected once. By using end(), the first find() is undone, so we can start search with the nextfind() at our #faq element, instead of the dd children.

Within the click handler, the function passed to the click() method, we use $(this).next() to find the next sibling starting from the current dt. This allows us to quickly select the answer following the clicked question.

In addition to siblings, you can also select parent elements (also known as ancestors for those more familiar with XPath). Maybe you want to highlight the paragraph that is the parent of the link the user hovers. Try this:

$(document).ready(function(){

$("a").hover(function(){

$(this).parents("p").addClass("highlight");

},function(){

$(this).parents("p").removeClass("highlight");

});

});

For all hovered anchor elements, the parent paragraph is searched and a class "highlight" added and removed.

Lets go one step back before continuing: jQuery is a lot about making code shorter and therefore easier to read and maintain. The following is a shortcut for the $(document).ready(callback) notation:

$(function() {

// code to execute when the DOM is ready

});

Applied to the Hello world! example, we end with this:

$(function() {

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

alert("Hello world!");

});

});

Now, with the basics at hand, we want to explore some other aspects, starting with AJAX.

**Interesting links for this chapter:**

* [jQuery API documentation](http://docs.jquery.com/API)
* [Visual jQuery](http://visualjquery.com/) - A categorized browsable API documentation.
* [jQuery Selectors](http://docs.jquery.com/DOM/Traversing/Selectors)
* [jQuery Events](http://docs.jquery.com/Events)
* [jQuery DOM Traversing](http://docs.jquery.com/DOM/Traversing)

**RATE ME: USING AJAX**

In this part we write a small Ajax application, that allows the user to rate something, just like it is done on youtube.com.

We need some server code for this. My example uses a php file that reads the "rating" parameter and returns the number of ratings and the average rating. Have a look at [rate.php](http://jquery.bassistance.de/rate.phps) for the server-side code.

We don't want this example to work without Ajax, although it may be possible, we therefore generate the necessary markup with jQuery and append it to a container div with an ID of "rating".

$(document).ready(function() {

// generate markup

$("#rating").append("Please rate: ");

for ( var i = 1; i <= 5; i++ ) {

$("#rating").append("<a href='#'>" + i + "</a> ");

}

// add markup to container and apply click handlers to anchors

$("#rating a").click(function(e) {

// stop normal link click

e.preventDefault();

// send request

$.post("rate.php", {rating: $(this).html()}, function(xml) {

// format and output result

$("#rating").html(

"Thanks for rating, current average: " +

$("average", xml).text() +

", number of votes: " +

$("count", xml).text()

);

});

});

});

This snippet generates five anchor elements and appends them to the container element with the id "rating". Afterwards, for every anchor inside the container, a click handler is added. When the anchor is clicked, a post request is send to rate.php with the content of the anchor as a parameter. The result returned as a XML is then added to the container, replacing the anchors.

If you don't have a web server with PHP installed at hand, you can look at an [online example](http://jquery.bassistance.de/example-rateme.html). For a very nice example of a rating system that even works without JavaScript, visit [softonic.de](http://www.softonic.de/ie/51494) and click on "Kurz bewerten!"

More documentation of the Ajax methods of jQuery can be found in the [Ajax Documentation](http://docs.jquery.com/Ajax) or on[Visual jQuery](http://visualjquery.com/) filed under Ajax.

A very common problem encountered when loading content by Ajax is this: When adding event handlers to your document that should also apply to the loaded content, you have to apply these handlers after the content is loaded. To prevent code duplication, you can delegate to a function. Example:

function addClickHandlers() {

$("a.remote", this).click(function() {

$("#target").load(this.href, addClickHandlers);

});

}

$(document).ready(addClickHandlers);

Now addClickHandlers is called once when the DOM is ready and then everytime when a user clicked a link with the class remote and the content has finished loading.

Note the $("a.remote", this) query, this is passed as a context: For the document ready event, this refers to the document, and it therefore searches the entire document for anchors with class remote. When addClickHandlers is used as a callback for load(), this refers to a different element: In the example, the element with id target. This prevents that the click event is applied again and again to the same links, causing a crash eventually.

Another common problem with callbacks are parameters. You have specified your callback but need to pass an extra parameter. The easiest way to achieve this is to wrap the callback inside another function:

// get some data

var foobar = ...;

// specify handler, it needs data as a paramter

function handler(data) {

//...

}

// add click handler and pass foobar!

$('a').click(function() {

handler(foobar);

});

// if you need the context of the original handler, use apply:

$('a').click(function() {

handler.apply(this, [foobar]);

});

With Ajax this simple we can cover quite a lot of "Web 2.0". Now that we've looked at some basic Ajax, let's add some simple effects and animations to the page.

**Interesting links for this chapter:**

* [jQuery Ajax Documentation](http://docs.jquery.com/Ajax)
* [jQuery API](http://docs.jquery.com/API) - Contains description and examples for append and all other jQuery methods
* [ThickBox](http://jquery.com/demo/thickbox/) - A jQuery plugin that uses jQuery to enhance the famous lightbox

**ANIMATE ME: USING EFFECTS**

Simple animations with jQuery can be achieved with show() and hide().

$(document).ready(function(){

$("a").toggle(function(){

$(".stuff").hide('slow');

},function(){

$(".stuff").show('fast');

});

});

You can create any combination of animations with animate(), eg. a slide with a fade:

$(document).ready(function(){

$("a").toggle(function(){

$(".stuff").animate({ height: 'hide', opacity: 'hide' }, 'slow');

},function(){

$(".stuff").animate({ height: 'show', opacity: 'show' }, 'slow');

});

});

Much fancier effects can be achieved with the [interface plugin collection](http://interface.eyecon.ro/). The site provides demos and documentation. While Interface is at the top of jQuery's plugin list, there are lots of others. The next part shows how to use the tablesorter plugin.

**Interesting links for this chapter:**

* [jQuery Effects Documentation](http://docs.jquery.com/Effects)
* [Interface plugin](http://interface.eyecon.ro/)

**SORT ME: USING THE TABLESORTER PLUGIN**

The tablesorter plugin allows sorting of tables on the client side. You include jQuery, and the plugin, and tell the plugin which tables you want to sort.

To try this example you need to [download the tablesorter plugin](http://tablesorter.com/jquery.tablesorter.zip) and add this line to starterkit.html (below the jquery include):

<script src="jquery.tablesorter.js"></script>

After including the plugin, you can call it like this:

$(document).ready(function(){

$("#large").tablesorter();

});

Try clicking the headers of the table and see how it is sorted ascending on first click and descending on second.

The table could use some row highlighting, we can add those by passing some options:

$(document).ready(function() {

$("#large").tablesorter({

// striping looking

widgets: ['zebra']

});

});

There are more examples and documentation about the available options at the [tablesorter homepage](http://tablesorter.com/docs/).

Most plugins can be used like this: Include the plugin file and call the plugin method on some elements, passing some optional settings to customize the plugin.

A up-to-date list of available plugins can be found [on the jQuery Plugin site](http://docs.jquery.com/Plugins).

When you are using jQuery more often, you may find it useful to package your own code as a plugin, either to reuse it for yourself or your company, or to share it with the community. The next chapter gives some hints on how to structure a plugin.

**Interesting links for this chapter:**

* [Plugins for jQuery](http://docs.jquery.com/Plugins)
* [Tablesorter Plugin](http://tablesorter.com/jquery.tablesorter.zip)

**PLUG ME: WRITING YOUR OWN PLUGINS**

Writing your own plugins for jQuery is quite easy. If you stick to the following rules, it is easy for others to integrate your plugin, too.

**Plugin Naming**

Find a name for your plugin, lets call our example "foobar". Create a file named jquery.[yourpluginname].js, eg. jquery.foobar.js

**Adding a Custom Method**

Create one or more plugin methods by extending the jQuery object, eg.:

jQuery.fn.foobar = function() {

// do something

};

Which will then be accessible by performing:

$(...).foobar();

**Default Settings:**

Create default settings that can be changed by the user, eg.:

jQuery.fn.foobar = function(options) {

var settings = jQuery.extend({

value: 5, name: "pete", bar: 655

}, options);

};

You can then call the plugin without options, using the defaults:

$("...").foobar();

Or with some options:

$("...").foobar({ value: 123, bar: 9 });

**Documentation**

If you release your plugin, you should provide some examples and documentation, too. There are[lots of plugins](http://docs.jquery.com/Plugins) available as a great reference.

Now you should have the basic idea of plugin writing. Lets use this knowledge and write one of our own.

**Checkbox Plugin**

Something lots of people, trying to manipulate forms with jQuery, ask for, is checking and unchecking of radio buttons or checkboxes. They end up with code like this:

$(":checkbox").each(function() {

this.checked = true;

this.checked = false; // or, to uncheck

this.checked = !this.checked; // or, to toggle

});

Whenever you have an each in your code, you might want to rewrite that as a plugin, pretty straightforward:

jQuery.fn.check = function() {

return this.each(function() {

this.checked = true;

});

};

This plugin can now be used:

$(":checkbox").check();

Now you could write plugins for both uncheck() and toggleCheck(), too. But instead we extend our plugin to accept some options.

jQuery.fn.check = function(mode) {

// if mode is undefined, use 'on' as default

var mode = mode || 'on';

return this.each(function() {

switch(mode) {

case 'on':

this.checked = true;

break;

case 'off':

this.checked = false;

break;

case 'toggle':

this.checked = !this.checked;

break;

}

});

};

By providing a default for the option, the user can omit the option or pass one of "on", "off", and "toggle", eg.:

$(":checkbox").check();

$(":checkbox").check('on');

$(":checkbox").check('off');

$(":checkbox").check('toggle');

**Optional Settings**

With more than one optional setting, this approach gets complicated, because the user must pass null values if he wants to omit the first parameter and only use the second.

The use of the tablesorter in the last chapter demonstrates the use of an object literal to solve this problem. The user can omit all parameters or pass an object with a key/value pair for every setting he wants to override.

For an exercise, you could try to rewrite the Voting code from the fourth section as a plugin. The plugin skeleton should look like this:

jQuery.fn.rateMe = function(options) {

// instead of selecting a static container with

// $("#rating"), we now use the jQuery context

var container = this;

var settings = jQuery.extend({

url: "rate.php"

// put more defaults here

}, options);

// ... rest of the code ...

// if possible, return "this" to not break the chain

return this;

});

And allowing you to run the plugin like so:

$(...).rateMe({ url: "test.php" });

**NEXT STEPS**

If you plan to develop more JavaScript, you should consider the Firefox extension called [FireBug](http://www.joehewitt.com/software/firebug/). It provides a console (nice to replace alerts), a debugger and other useful stuff for the daily JavaScript development.

If you have problems you can't solve, ideas you want to share or just the need to express your opinion on jQuery, feel free to post to the [jQuery mailing list](http://jquery.com/discuss/).

For anything related to this guide post a comment on my [blog](http://bassistance.de/index.php/2006/09/12/jquery-getting-started-guide/) or contact me directly.

**Whats left...**Thanks a lot to John Resig for this great library! Thanks to the jQuery community for providing John with enough coffee and everything else!