

# Introduction to jQuery

Instructor: Frank Stepanski

## Overview

[jQuery](#) aims to change the way that web developers think about creating rich functionality in their pages. Rather than spending time juggling the complexities of advanced JavaScript, designers can leverage their existing knowledge of Cascading Style Sheets (CSS), Extensible Hypertext Markup Language (XHTML), and good old straightforward JavaScript to manipulate page elements directly, making more rapid development a reality.

## Audience for this Class

This class is considered a beginner class and can be taken by either a web designer or web developer with no previous experience with jQuery.

For a **web designer**, good knowledge of XHTML and CSS is required. It would be very helpful if you have existing knowledge or experience of JavaScript. This class uses programming syntax and terminology so pre-existing knowledge of JavaScript or a programming language is required.

For a **web developer**, knowledge of programming language syntax and terminology is already assumed so no other experience would be necessary.

## Tools Needed

There is no specific tool to develop in jQuery so you can use any free text or web editor like [Notepad++](#) or [HTML-Kit](#) or [Komodo Edit](#) or commercial products like [Adobe Dreamweaver](#) or [Microsoft Web Expression](#).

## Why jQuery?

If you've spent any time at all trying to add dynamic functionality to your pages (as well as figure out browser differences), you've found that you're constantly following a pattern of selecting an element or group of elements and operating upon those elements in some fashion. You could be hiding or revealing the elements, adding a CSS class to them, animating them, or modifying their attributes.

Using raw JavaScript can result in dozens of lines of code for each of these tasks. The creators of jQuery specifically created the library to make common tasks trivial.

So the basic concept of jQuery could be described as:

1. Creating something new
2. Selecting it
3. Then doing something with it.

## About jQuery

It's one of the most popular JavaScript libraries around and was created by [John Resig](#) during his college days at the Rochester Institute of Technology. Its popularity has been helped by a number of high-profile sites using jQuery such as the BBC, Digg, Intel, MSNBC, and Technorati.

The core features of jQuery are:

- Gives developers a common set of functions for all browsers
- Uses selectors which is an expression for identifying target elements on a page that allows us to easily identify and grab the elements we need
- Gives access to page elements without having to wait for all images to load in place of using the browser's onload event, which delays anything you do until the page is fully loaded
- Let's you create and delete HTML.
- Has a great selection of animation and visual effect
- Contains enhancements to basic JavaScript constructs such as iteration and array manipulation.

The real power in this jQuery statement comes from the *selector*, an expression for identifying target elements on a page that allows us to easily identify and grab the elements we need; in this case, even every `<tr>` element in all tables.

The jQuery library provides a general-purpose abstraction layer for common web scripting, and is therefore useful in almost every scripting situation. Its extensible nature means that we could never cover all possible uses and functions in a single book, as plugins are constantly being developed to add new abilities.

## Downloading jQuery

jQuery is available in two versions: a packed version for use in production sites and an uncompressed version that is more readable (if you want to review the source).

[http://docs.jquery.com/Downloading\\_jQuery](http://docs.jquery.com/Downloading_jQuery)

No installation is required. To use jQuery, you just need to place it in a public location.

Since JavaScript is an interpreted language, there is no compilation or build phase to worry about. Whenever we need a page to have jQuery available, we will simply refer to the file's location from the HTML document.

Just include the file in the same location as your HTML page and you're ready to use jQuery.

```
<script src=jquery-1.4.4.js" type="text/javascript"></script>
```

## Google CDN

An alternative method for including the jQuery library that's worth considering is via the Google **Content Delivery Network (CDN)**. A CDN is a network of computers that are specifically designed to serve content to users in a fast and scalable manner. These servers are often distributed geographically, with each request being served by the nearest server in the network.

So, instead of hosting the jQuery files on your own web server as we did above, you have the option of letting Google pick up part of your bandwidth bill. You benefit from the speed and reliability of Google’s vast infrastructure, with the added bonus of the option to always use the latest version of jQuery.

```
<script type="text/javascript" src="http://ajax.googleapis.com/
ajax/libs/jquery/1.4.4/jquery.min.js"></script>
```

### **Uncompressed or compressed?**

If you had a poke around on the jQuery download page, you might have also spied a couple of different download format options: compressed (also called **minified**), and uncompressed (also called “development”).

Typically, you’ll want to use the minified version for your production code, where the jQuery source code is compressed: spaces and line breaks have been removed and variable names are shortened. The result is exactly the same jQuery library, but contained in a JavaScript file that’s much smaller than the original. This is great for reducing bandwidth costs for you, and speeding up page requests for the end user.

### **The Document Object Model**

One of the most powerful aspects of jQuery is its ability to make selecting elements in the DOM easy.

The [Document Object Model](#) is a family-tree structure of sorts. HTML, like other markup languages, uses this model to describe the relationships of things on a page. When we refer to these relationships, we use the same terminology that we use when referring to family relationships—parents, children, and so on.

A simple example can help us understand how the family tree metaphor applies to a document:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>the title</title>
</head>
  <body>
    <div>
      <p>This is a paragraph.</p>
      <p>This is another paragraph.</p>
      <p>This is yet another paragraph.</p>
    </div>
  </body>
</html>
```

Here, `<html>` is the **ancestor** of all the other elements; in other words, all the other elements are **descendants** of `<html>`. The `<head>` and `<body>` elements are not only descendants, but **children** of `<html>`, as well.

Likewise, in addition to being the ancestor of `<head>` and `<body>`, `<html>` is also their **parent**. The `<p>` elements are children (and descendants) of `<div>`, descendants of `<body>` and `<html>`, and **siblings** of each other.

An important point to note before we begin is that the resulting set of elements from selectors and methods is always wrapped in a jQuery object. These jQuery objects are very easy to work with when we want to actually do something with the things that we find on a page.

We can easily bind **events** to these objects and add slick **effects** to them, as well as **chain** multiple modifications or effects together. Nevertheless, jQuery objects are different from regular DOM elements or node lists, and as such do not necessarily provide the same methods and properties for some tasks.

## The `$()` factory function

The fundamental operation in jQuery is selecting a part of the document. This is done with the `$()` construct. Typically, it takes a string as a parameter, which can contain any CSS selector expression.

## Making Sure the Page is Ready: `$(document).ready(function)`

While JavaScript provides the *load* event for executing code when a page is rendered, this event does not get triggered until all assets (images, script, frames and other embedded objects) have been completely loaded.

```
$(document).ready(function() {  
    alert('ready event was triggered');  
});
```

With this code in place, an alert will be displayed when the page is loaded.

**Note:** If your script requires images to be loaded first before executing then you can use the *load()* method instead:

```
$(document).load(function() {  
    alert('all assets have been loaded');  
});
```

The shortcut version would be:

```
$(function() {  
    alert('all assets have been loaded');  
});
```

## CSS Selectors

You will often need to select page elements in online work. The selection capabilities of browsers are minimal and mostly involve the JavaScript [getElementById](#) function, which requires you to add an ID value to any element you want to select (making selection of multiple items difficult or at least time consuming).

CSS offers a much stronger set of tools for selecting page elements to set styles, and Query adopts many of the CSS selectors. Using the jQuery selectors lets you select page elements so you can work on them in JavaScript, not just with CSS styles.

SELECTOR	PURPOSE
<i>selector, selector, ...</i>	Finds all of the specified selectors
<i>.class1.class2</i>	Finds all elements with both <i>.class1</i> and <i>.class2</i> applied
<i>parent &gt; child</i>	Finds all <i>child</i> elements that are direct children of elements of type <i>parent</i>
<i>ancestor descendant</i>	Finds all <i>descendant</i> elements that are contained within elements of type <i>ancestor</i>
<i>prev + next</i>	Finds all <i>next</i> elements that are next to a <i>prev</i> element
<i>prev ~ siblings</i>	Finds all sibling elements that come after <i>prev</i> and match the <i>siblings</i> selector

Figure 1 – Basic and Hierarchal Selectors in jQuery

Documentation: <http://api.jquery.com/category/selectors/>

Let's go through some examples so you get a feel for how to use these basic selectors.

```
<body>
  <ul id="list1">
    <li class="a">item 1</li>
    <li class="a">item 2</li>
    <li class="b">item 3</li>
    <li class="b">item 4</li>
  </ul>
  <p class="a">This is paragraph 1</p>
  <p id="para1">This is paragraph 2</p>
  <p class="b">This is paragraph 3</p>
  <p>This is paragraph 4</p>
</body>
```

**Note:** Comment out (//) all but one statement at a time. You can disregard the part of the code that has `.css` added to it. We will discuss that in a later lesson.

## Basic Selectors:

```
<script type="text/javascript">
  $("document").ready(function() {
    //$("p").css("border", "3px solid red");
    //$(".a").css("border", "3px solid red");
    //$("#list1").css("border", "3px solid red");
    //$("p.b").css("border", "3px solid red");
  });
</script>
```

1. \$("p") – Selects all paragraphs.
2. \$(".a") – Selects everything that has a class of “a”.
3. \$("#list1") – Selects the element that has an ID of “list1”.
4. \$("p.b") – Selects all paragraphs that have a class of “b”.

## Filters

Filters work with selectors to provide even more fine-grained control over how elements are selected in the document. jQuery filters fall into six main categories: Basic, Content, Visibility, Attribute, Child and Form.

Note: Some documentation doesn't differentiate between selectors and filters. They just “mush” them all together. 😊

FILTER	PURPOSE
:first	Selects only the first instance of the selector's returned set
:last	Selects only the last instance of the selector's returned set
:even	Selects only even-numbered elements in the selector's returned set
:odd	Selects only odd-numbered elements in the selector's returned set
:eq(n)	Filters out elements that are not positioned at the given index
:gt(n)	Includes elements that are past the given index
:lt(n)	Includes elements that are before the given index
:header	Selects all header elements (H1, H2, H3, etc)
:animated	Selects all elements that are currently being animated in some way
:not(selector)	Includes elements that do not match the given selector

Figure 2 – Basic Filters in jQuery

**Note:** Comment out (//) all but one statement at a time. Disregard the part of the code that has .css added to it. We will discuss that in a later lesson.

### Basic Filters:

```
<script type="text/javascript">
  $("document").ready(function() {
    //$("p:first").css("border", "3px solid red");
    //$("p:last").css("border", "3px solid red");
    //$("p:even").css("border", "3px solid red");
    //$("p:odd").css("border", "3px solid red");
    //$(".a:first").css("border", "3px solid red");
    //$(".b:even").css("border", "3px solid red");
    //$("p:gt(1)").css("border", "3px solid red");
  });
</script>
```

5. \$("p:first") – Selects the **first** paragraph.
6. \$("p:last") – Selects the **last** paragraph.
7. \$("p:even") – Selects all **even** paragraphs (starts at 0, so the first is ‘even’).
8. \$(".a:first") – Selects the first element that has an “a” class applied to it.
9. \$(".b:even") – Selects all even number elements that have a class of “b”.
10. \$("p:odd") – Selects all **odd** paragraphs.
11. \$("p:gt(1)") – Select all paragraphs that are greater than index of 1 (3<sup>rd</sup> and 4<sup>th</sup>)

## Attribute Filters

Attribute filters provide the ability to further filter out the results a selector statement based upon attributes that are on the elements being selected.

FILTER	PURPOSE
[ <i>attribute</i> ]	Includes elements in the result set if they have the specified <i>attribute</i>
[ <i>attribute=value</i> ]	Includes elements in the result set if they have the specified <i>attribute</i> and it has the given <i>value</i>
[ <i>attribute!=value</i> ]	Includes elements in the result set only if they have the specified <i>attribute</i> and it doesn't have the given <i>value</i>
[ <i>attribute^=value</i> ]	Includes elements that have the specified <i>attribute</i> and it starts with the specified <i>value</i>
[ <i>attribute\$=value</i> ]	Includes elements that have the specified <i>attribute</i> and it ends with the specified <i>value</i>
[ <i>attribute*=value</i> ]	Includes elements that have the specified <i>attribute</i> and it contains the specified <i>value</i>
[ <i>attrFilter1</i> ][ <i>attrFilterN</i> ]	Includes elements that match all of the specified attribute filters

Figure 3 – Attribute Filters in jQuery

**Note:** Comment out (//) all but one statement at a time. Disregard the part of the code that has .css added to it. We will discuss that in a later lesson.

## Attribute Filters:

```
<script type="text/javascript">
  $( "document" ).ready(function() {
    //$( "p[class]").css("border", "3px solid red");
    //$( "p[id=para1]").css("border", "3px solid red");
    //$( "p[id^=para]").css("border", "3px solid red");
    //$( "p[id^=para][lang*=en-]").css("border", "3px solid red");
  });
</script>
```

12. \$("p[class]") – Selects all the paragraphs that have a class.
13. \$("p[id=para1]") – Selects all paragraphs that have an id equal to “para1”.
14. \$("p: [id^=para1]") – Selects all paragraphs with an id that starts with “para”.
15. \$("p: [id^=para1][lang\*=en-]") – Selects all paragraphs with an id that starts with “para” and has a lang attribute that contains the text of “en-”.

## Try for Practice

Now that you have a good idea of the basics of selecting HTML elements in jQuery using a few of selectors and filters you can practice some more with this online example from w3schools:

[http://www.w3schools.com/jquery/tryselect.asp?filename=tryselect\\_basic](http://www.w3schools.com/jquery/tryselect.asp?filename=tryselect_basic)

## **Additional Resources:**

1. [http://docs.jquery.com/Main\\_Page](http://docs.jquery.com/Main_Page)
2. <http://api.jquery.com/browser/>

## **Assignment for Lesson 1**

1. Review the lesson.
2. Download [jQuery](#) and review the examples.
3. Create a simple script using jQuery that selects all the rows in a table.
4. Post a link on the class forum for week one

Copyright 2011 © Frank Stepanski

Lessons, files and content of these classes cannot be reproduced and/or published without the express written consent of the author.