jQuery Tutorial for Beginners: Nothing But the Goods

Not too long ago I wrote an article for Six Revisions called “Getting Started with jQuery” that covered some important things (concept-wise) that beginning jQuery developers should know. This article is the complete opposite; there’s no concepts, no principles, and very little lecturing -- just some straight example code with brief descriptions demonstrating what you can do with jQuery.

This fast-paced tutorial should be able to get beginners up and running with jQuery very quickly, while also providing a good overview of what jQuery is capable of (although jQuery’s capabilities go far beyond this beginning tutorial).

Keep in mind that this tutorial is just a bunch of straightforward, superficial code examples and very brief explanations for beginners who want to avoid all the jargon and complexities. But I still highly recommend that all beginners get past this stuff by means of a good book, some more in-depth tutorials online, or by using the jQuery documentation.

Link to jQuery's Source Code Remotely

This is an optional technique that many developers are using today. Instead of hosting the jQuery source code on your own server, you can link to it remotely. This ensures the fastest possible download time of the source code, since many users visiting your site might have the file already cached in their browser. Here is how your <script> tag should look:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.0/jquery.min.js?ver=1.4.0"></scri
pt>
```

Running Code Unobtrusively When the DOM is Ready

The first thing you need to be up and running with jQuery is what’s called the “document ready” handler. Pretty much everything you code in jQuery will be contained inside one of these. This accomplishes two things: First, it ensures that the code does not run until the DOM is ready. This confirms that any elements being accessed are actually in existence, so the script won’t return any errors. Second, this ensures that your code is unobtrusive. That is, it’s separated from content (XHTML) and presentation (CSS).

Here is what it looks like:

```
$(document).ready(function() {
        // all jQuery code goes here
});
```

Although you will normally include your jQuery code inside the above handler, for brevity the rest of the code examples in this tutorial will not include the “ready” handler.

Selecting Elements in jQuery

The jQuery library allows you to select elements in your XHTML by wrapping them in $("") (you could also use single quotes), which is the jQuery wrapper. Here are some examples of “wrapped sets” in jQuery:

```
$("div"); // selects all HTML div elements
$("#myElement"); // selects one HTML element with ID "myElement"
$(".myClass"); // selects HTML elements with class "myClass"
$("p#myElement"); // selects HTML paragraph element with ID "myElement"
$("ul li a.navigation");
// selects anchors with class "navigation" that are nested in list items
```

jQuery supports the use of all CSS selectors, even those in CSS3. Here are some examples of alternate selectors:

```
$("p > a"); // selects anchors that are direct children of paragraphs
$("input[type=text]"); // selects inputs that have specified type
$("a:first"); // selects the first anchor on the page
$("p:odd"); // selects all odd numbered paragraphs
$("li:first-child"); // selects each list item that's the first child in its list
```

jQuery also allows the use of its own custom selectors. Here are some examples:

```
$(":animated"); // selects elements currently being animated
$(":button"); // selects any button elements (inputs or buttons)
$(":radio"); // selects radio buttons
$(":checkbox"); // selects checkboxes
$(":checked"); // selects checkboxes or radio buttons that are selected
$(":header"); // selects header elements (h1, h2, h3, etc.)
```

Manipulating and Accessing CSS Class Names

jQuery allows you to easily add, remove, and toggle CSS classes, which comes in handy for a variety of practical uses. Here are the different syntaxes for accomplishing this:

```
$("div").addClass("content"); // adds class "content" to all <div> elements
$("div").removeClass("content"); // removes class "content" from all <div> elements
$("div").toggleClass("content");
// toggles the class "content" on all <div> elements (adds it if it doesn't exist, //
and removes it if it does)
```

You can also check to see if a selected element has a particular CSS class, and then run some code if it does. You would check this using an if statement. Here is an example:

```
if ($("#myElement").hasClass("content")) {
        // do something here
}
```

You could also check a set of elements (instead of just one), and the result would return "true" if any one of the elements contained the class.

Manipulating CSS Styles with jQuery

CSS styles can be added to elements easily using jQuery, and it's done in a cross-browser fashion. Here are some examples to demonstrate this:

```
$("p").css("width", "400px"); // adds a width to all paragraphs
$("#myElement").css("color", "blue") // makes text color blue on element #myElement
$("ul").css("border", "solid 1px #ccc") // adds a border to all lists
```

Adding, Removing, and Appending Elements and Content

There are a number of ways to manipulate groups of elements with jQuery, including manipulating the content of those elements (whether text, inline elements, etc).

Get the HTML of any element (similar to innerHTML in JavaScript):

```
var myElementHTML = $("#myElement").html();
// variable contains all HTML (including text) inside #myElement
```

If you don't want to access the HTML, but only want the text of an element:

```
var myElementHTML = $("#myElement").text();
// variable contains all text (excluding HTML) inside #myElement
```

Using similar syntax to the above two examples, you can change the HTML or text content of a specified element:

```
$("#myElement").html("<p>This is the new content.</p>");
// content inside #myElement will be replaced with that specified
$("#myElement").text("This is the new content.");
// text content will be replaced with that specified
```

To append content to an element:

```
$("#myElement").append("<p>This is the new content.</p>");
// keeps content intact, and adds the new content to the end
$("p").append("<p>This is the new content.</p>");
// add the same content to all paragraphs
```

jQuery also offers use of the commands appendTo(), prepend(), prependTo(), before(), insertBefore(), after(), and insertAfter(), which work similarly to append() but with their own unique characteristics that go beyond the scope of this simple tutorial.

Dealing with Events in jQuery

Specific event handlers can be established using the following code:

```
$("a").click(function() {
        // do something here
        // when any anchor is clicked
});
```

The code inside function() will only run when an anchor is clicked. Some other common events you might use in jQuery include:

```
blur, focus, hover, keydown, load, mousemove, resize, scroll, submit, select.
```

Showing and Hiding Elements with jQuery

The syntax for showing, hiding an element (or toggling show/hide) is:

```
$("#myElement").hide("slow", function() {
        // do something once the element is hidden
}

$("#myElement").show("fast", function() {
        // do something once the element is shown
}

$("#myElement").toggle(1000, function() {
        // do something once the element is shown/hidden
}
```

Remember that the "toggle" command will change whatever state the element currently has, and the parameters are both optional. The first parameter indicates the speed of the showing/hiding. If no speed is set, it will occur instantly, with no animation. A number for "speed" represents the speed in milliseconds. The second parameter is an optional function that will run when the command is finished executing.

You can also specifically choose to fade an element in or out, which is always done by animation:

```
$("#myElement").fadeOut("slow", function() {
        // do something when fade out finished
}

$("#myElement").fadeIn("fast", function() {
        // do something when fade in finished
}
```

To fade an element only partially, either in or out:

```
$("#myElement").fadeTo(2000, 0.4, function() {
        // do something when fade is finished
}
```

The second parameter (0.4) represents "opacity", and is similar to the way opacity is set in CSS. Whatever the opacity is to start with, it will animate (fadeTo) until it reaches the setting specified, at the speed set (2000 milliseconds). The optional function (called a "callback function") will run when the opacity change is complete. This is the way virtually all callback functions in jQuery work.

jQuery Animations and Effects

You can slide elements, animate elements, and even stop animations in mid-sequence. To slide elements up or down:

```
$("#myElement").slideDown("fast", function() {
        // do something when slide down is finished
}

$("#myElement").slideUp("slow", function() {
        // do something when slide up is finished
}

$("#myElement").slideToggle(1000, function() {
        // do something when slide up/down is finished
}
```

To animate an element, you do so by telling jQuery the CSS styles that the item should change to. jQuery will set the new styles, but instead of setting them instantly (as CSS or raw JavaScript would do), it does so gradually, animating the effect at the chosen speed:

```
$("#myElement").animate(
        {
                opacity: .3,
                width: "500px",
                height: "700px"
        }, 2000, function() {
                // optional callback after animation completes
        }
);
```

Animation with jQuery is very powerful, and it does have its quirks (for example, to animate colors, you need a special plugin). It's worth taking the time to learn to use the animate command in-depth, but it is quite easy to use even for beginners.

This is Just the Beginning

There is so much more you can do with jQuery beyond these basics I've introduced here. I highly recommend that all developers buy a good book on jQuery, and also take the time to learn some important JavaScript concepts (like anonymous functions, closures, scope, etc.) in order to be able to use jQuery in a more powerful way.

And please note that I've introduced a lot of commands and syntaxes in a very superficial manner, without explaining many of the problems you could have with some of these, so if you do run into problems, you can check out the jQuery documentation.