IS360

jQuery Lab 1

Ekedahl

# Abstract

**In this lab, you will:**

* Create a first jQuery script that will change CSS styles for a table.
* Create JQuery scripts that works with visual effects such as fade, show and hide, and sliding
* Create jQuery scripts that traverse the DOM.
* Create a script that performs a simple animation.

**A couple of points to make about jQuery**

* It only works well with “well-formed” HTML documents. Everything breaks if the document is not well-formed. It is also more useful with documents that have significant structure. The more structured the document, the more we get out of jQuery.
* Visual Studio does not provide much in the way of debugging. However, there is a jQuery debugger that can be added to Chrome. <https://chrome.google.com/webstore/category/apps?hl=en>. The Intellisense tools do help us (somewhat) with jQuery.
* Take a look at jQuery.com for the definitive reference. Again, w3Schools.com has a complete tutorial.

## SUMMARY OF jQuery ACTIONS (Event Methods)

|  |  |
| --- | --- |
| **Action** | **Description** |
| click() | The click() method is executed when the user clicks on the HTML element. While commonly used with buttons, it can also be used with other element types. |
| css() | Gets are sets a CSS style for the selected elements. |
| mouseout() | Fires when the mouse moves out of an element’s visible region. |
| mouseover() | Fires when the mouse passes over an element’s visible region. |
| focus() | Fires when an element gets focus. |
| blur() | Fires when an element loses focus. |

## SUMMARY OF jQuery ACTIONS (Traversal and Effect Methods)

|  |  |
| --- | --- |
| **Action** | **Description** |
| children() | Gets the children of the matched elements. Optionally, you can supply an argument containing another selector so as to further filter the child elements. |
| first() | Find the first element in a list of selected elements. |
| last() | Find the last element in a list of selected elements. |
| find() | Returns the descendent elements of the selected element. The find method requires a parameter. This parameter contains another selector. |
| fadeIn() | Makes the selected elements visible. The optional *speed* parameter controls the speed of the fade. (*slow, fast, milliseconds*) |
| fadeOut() | Makes the selected elements invisible. The optional *speed* parameter controls the speed of the fade. (*slow, fast, milliseconds*) |
| fadeToggle() | Toggles the element. (*slow, fast, milliseconds*) |
| hide() | Makes the selected elements invisible. An optional argument controls the speed at which an animation is performed. |
| show() | Makes the selected elements visible. An optional argument controls the speed at which an animation is performed. |

**In jQuery, the element being acted upon is often passed to the function parameter of a jQuery function as this. It can be made into a jQuery element object by using $(this) in the function parameter. The $(this) element object will be used extensively in this lab to select the current element.**

# Attaching the jQuery Hosted Libraries

The jQuery libraries can be referenced in a number of ways. They can be downloaded and stored in a specific Web application (site), or they can be used via a hosted library. In this lab, you will use the second approach. The benefit of using the hosted libraries is that it is easier to keep up with the latest version(s). Visit <https://developers.google.com/speed/libraries/> for more information about the Google hosted libraries. Note that jQuery is also hosted by Microsoft and other providers.

**HANDS-ON EXERCISE – Referencing the Hosted jQuery Libraries**

In this exercise, you will reference the Google Hosted Library for jQuery. To reference the hosted library, you reference external JavaScript with the src attribute in a <script> tag.

1. For this part of the lab, you will begin with an HTML 5 document that I created so that you do not have to waste time creating HTML 5 with which you are already familiar. Open the file named **IS360jQueryLab1.html** appearing on the Lecture Notes page of the Web site. This link appears with the link for this lab.
2. Add the following code (shown in bold) to the page header, so as to add the necessary reference to the jQuery library. *Note that you should reference the jQuery libraries before calling any jQuery functions.*

<head>

<title></title>

**<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">**

</script>

</head>  
  
Again, the preceding code references the jQuery library via a URL. Here, the minimized version is used. Note that any external JavaScript library can be imported in this way.

# Creating a First jQuery Script

Recall that the basic jQuery syntax is made up of a selector and an action as follows:

$(*selector*).*action*()

* The $ indicates that we are accessing jQuery.
* The (*selector*) is just about the same as a CSS selector and serves the same purpose. It selects elements on which actions will be performed. As the selector is a string, it should be quoted.
* The *action*() is performed on the selected elements. Here, *action()* is a placeholder and should be replaced with the actual jQuery action.

All jQuery scripts should be placed inside of a document ready event. This event fires after a document has been fully loaded and prevents jQuery scripts from running until the document has been fully processed (loaded).

The older jQuery syntax required that you handle document ready as follows:

$(document).ready(function(){  
 // *jQuery code.*  
});

The newer abbreviated syntax is as follows:

$(function(){  
 // *jQuery code.*  
}

A great explanation of the jQuery syntax appears at the following URL [http://www.smashingmagazine.com/2014/05/mystery-jquery-object-syntax-basic-introduction](http://www.smashingmagazine.com/2014/05/mystery-jquery-object-syntax-basic-introduction/)

**HANDS-ON EXERCISE – Creating a First jQuery Script**

In this exercise, you will work with a simple table. One common task with tables is to format the alternating rows of a table so that it’s easy to follow the content of a row – especially a wide row. To do this, you can use the jQuery .css method (action). The purpose of the jQuery .css method is to get or set a CSS style.

* When called with one argument, the method will get a CSS style.
* When called with two arguments. The first argument contains the CSS style to set and the second argument contains the value of that CSS style.

1. Create a second <script> block in the <head> section of the document **immediately after** the <script> block that you just created.
2. Next, add the following jQuery code that will apply a style to all odd elements.

$(function ()  
 {

$("tr:odd").css("background-color", "lightgrey")

});

1. View the page in the browser. You should see the table formatted such that every other element has a light grey background. Like JavaScript, jQuery usually fails passively so if you make a mistake, nothing will likely happen.  
     
   Note the following about the above code:

* The procedure implicitly calls the $(document).ready function.
* $(function () tells the system that we are registering a jQuery function.
* The selector $(“tr:odd”) gets the odd <tr> elements from every row in all tables. The selector $(“tr:even”) will get the even rows. In other words, select selectors get every other row from a table.
* The action calls the css method to set the value of a css background-color property for the selected elements to lightgrey. Note that both arguments are quoted because they are strings.

1. Try it on your own by setting the even <tr> elements to a different background color, such as lightgreen.
2. Format the <th> tag so that the background color is white. Use jQuery. Do not just create a CSS style.

# Changing Formatting with mouseover and mouseout

Remember the DOM mouseover event fires when the mouse passes over an element’s visible region and the mouseout event fires when the mouse passes out of an element’s visible region. You can program these actions to change formatting when the mouse passes over and out of an element. In this exercise, you will select the <h1> tags. When the mouse passes over or out of one of these <h1> tags, you will change the foreground color.

1. On the same page that you have been editing and in the same <script> block in which you created the last jQuery code, create the following two functions:

$(function () {

$("h1").mouseover(function () {

$(this).css("color", "maroon");

});

});

$(function () {

$("h1").mouseout(function () {

$(this).css("color", "black");

});

});

Note the following about the code that you just wrote:

* The selector $(“h1”) selects all of the <h1> elements. So $(“h1”).mouseover() registers the mouseover event for all of the <h1> elements.
* Next, we register the jQuery function that will execute for the event.
* The next statement in the function uses $(this) to select the current <h1> element an set the CSS color attribute to maroon. css("color", "maroon"); Remember that $(this) represents the currently selected element, which I this case is one of the <h1> tags.
* The second function works the same way for the mouseout event resetting the color back to black.

1. View the page in the browser. Move the mouse over the various <h1> elements in the document. They foreground color should be maroon when the mouse passes over the element and black when the mouse passes off of the element.

# Using jQuery (toggle, hide, and show) to Control Visibility

In this exercise, you will work with the various visibility effects. Specifically, you will show and hide elements when the user clicks on them. There are a couple of ways to do this.

In this example, you will apply the following strategy. The following tasks will be performed when the user clicks on an <article> element. Note that the document is structured such that each <article> element has a first child named <h1>. Following this <h1> element are several <p> elements. Both the <h1> and <p> elements are immediate children of the <article> element.

* Register the click() method for each article element.
* Using $(this), toggle all of children of the current <article> element. Note that this will hide all children so the <h1> tag will not be visible. Thus, there will be no way for the user to make the <article> visible again.
* The next statement will use the find() action to find the <h1> element that is the current child of the <article> element and make it visible by calling the show() method.
* These two tasks will be performed when the user moves the mouse over a tag (mouseover) and off of a tag (mouseout).

**HANDS-ON EXERCISE: Controlling Visibility with toggle() and show()**

1. On the same page with which you have been working create the following jQuery code.

$(function () {

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

$(this).children().toggle(500);

$(this).find('h1').show(500);

});

});

1. View the page in the browser. Click on each of the articles. If the detail content is visible, it will be hidden. If the detail content is hidden, it will be displayed.

Note the following about the code that you just wrote:

* The first fragment selects all <article> elements, and registers the click() method.
* The fragment $(this).children().toggle(500); first selects the current <article> element. All of the current children of the <article> are then selected. Finally, the visibility of each child is toggled. A 500 millisecond delay appears at the argument to toggle().
* The next statement again selects the current <article> element using $(this). The find() method is called to select the <h1> children of the <article>. The show() method displays the <h1> tag.

# Using jQuery (fadeIn and fadeOut) to Control Visibility

In addition to simply showing and hiding element, you can make those same elements fade in and fade out slowly using the fadeIn() and fadeOut() methods. Both methods accept an argument. You can use the literal string values slow or fast, or a numeric value. The numeric value controls the number of milliseconds over which to perform the fade.

In this exercise, you will show and hide an image as buttons are clicked. You will implement this using two functions that will use button ID selectors.

**HANDS-ON EXERCISE: Controlling Visibility with fadeIn() and fadeOut()**

1. At the end of the same <script> block that you have been using, create the following two jQuery functions.

$(function () {

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

$("#NVLogo").fadeIn(1500);

});

});

$(function () {

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

$("#NVLogo").fadeOut(1500);

});

});

Note the following about the above code.

* The two functions work almost identically.
* The first selects the button with the ID btnFadeIn (ID selector), and registers the click() activity. The next statement selects the image named NVLogo, and calls the fadeIn() method. The value 1500 tells jQuery that the fade should take 1500 milliseconds.
* The second procedure works the same way. However, fadeout() is called instead of fadeIn().

# Using jQuery (slideDown and slideUp) to Control Visibility

slideUp() and slideDown() work the same way as fadeIn() and fadeout(). However, the visual effect works like an accordion.

**HANDS-ON EXERCISE: Controlling Visibility with slideDown() and slideUp()**

1. At the end of the same <script> block that you have been using, create the following jQuery function.

$(function () {

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

$("#NVLogo").slideDown(1500);

});

});

1. On our own, create the function to slideUp().
2. Again, view the page in a Browser to verify that your code is working.

# Using jQuery for Animation

The jQuery animate() method allows you to create custom animations in which you can move an object from one place to another on a form, or change other styles. Roughly speaking, animation involves the following:

* As always, a selector is used to select the element(s) that will be animated.
* The animate() method accepts up to three arguments.
  + The required first argument, named *params*, defines the CSS properties that will be animated. The object is animated from the current property value to the new property value.
  + The second argument defines the speed at which the animation is performed. Again, the speed can be “slow”, “fast”, or *milliseconds.*
  + The final argument contains the optional callback function that will execute when the animation is complete.

**HANDS-ON EXERCISE: Creating movement with animate()**

In this exercise, you will use the animate method to move the contents of a <div> tag across the screen.

1. At the end of the same <script> block that you have been using, create the following jQuery function:

$(function () {

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

$("#divAnimate").animate({ left: '7500px'},"slow");

$("#divAnimate").animate({ left: '0px'}, "slow");

});

});

Note the following about this function:

* The function registers the click method for the button named btnAnimate.
* The contents of the <div> tag are moved across the page and back.