LAB 10

JAVASCRIPT 3: JQUERY

What You Will Learn

- Extending JavaScript using jQuery
- Manipulating the DOM and handling events using jQuery
- Making asynchronous requests in JavaScript using jQuery

Approximate Time

The exercises in this lab should take approximately 45 minutes to complete.

JQUERY SELECTORS

PREPARING DIRECTORIES

1 If you haven't done so already, create a folder in your personal drive for all the labs for this course.

2 From the main `labs` folder (either downloaded from the textbook's web site using code provided with book or in a common location provided by your instructor), copy the folder titled `lab10` to your course folder created in step one.

jQuery is a powerful JavaScript framework that started as a concise set of selector mechanisms and continues to add new features such as animation and parsing capability. You will learn about JavaScript's prototype mechanism first and then use that mechanism with jQuery.

Exercise 10.1 — Set up jQuery

1 Examine `lab10-walkthrough01.html` in your browser and then in your editor of choice.

2 Add code to load jQuery from the Content Delivery Network in the `<head>` section of your page as follows (note the latest version number may be higher):

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
```

3 Because the CDN may be unavailable (for instance, you may not have internet access), you may want to have a fall back version of the script that can be loaded locally. After including the script from the CDN add the following:

```
<script type="text/javascript">
window.jQuery ||
   document.write('<script src="js/jquery-3.3.1.min.js"><\/script>');
</script>
```

Exercise 10.2 — Basic Selectors

1 Continue with your file from the previous exercise.

2 Edit js/`lab10-walkthrough02.js` by adding the following and then testing `lab10-walkthrough01.html` in browser.

```
var msg = $('#msgArea');
msg.val("here is some new text");
```

The first line makes use of the jQuery function. It selects the <textarea> element and sets its value.

3 Add the following code and test.

```
$("#pageTitle").html( "new Title");
```

The html() method can retrieve or set the HTML content of an element

4 Add the following code and test.

```
$('#msgArea').val("My class is " + $('#msgArea').attr("class") );
```

The attr() method can retrieve or set an HTML attribute of an element.

5 Add the following code and test.

```
var buttons = $('button');
```

```
buttons.css('background-color', 'red');
```

This selects all the <button> elements and changes their background color to red. Notice that you do not need to loop through the collection of jQuery nodes: one of the powerful features of jQuery is that the jQuery function always returns a set of jQuery nodes and thus all functions operate the same whether there is one or many nodes.

6 Add the following and test.

```
var temp = $('body');

temp.css("background-color", "ivory");
```

Even though the first line returns a set containing only a single element, the css() method still works because the jQuery function always returns a set.

7 Add the following and test.

```
$('.center-icons').addClass('selected');
```

It is common to reduce the number of global identifiers, so most jQuery developers will chain the method calls to the selection. In this example, we are using the addClass() method to add a class to the selected elements.

Exercise 10.3 — Advanced Selectors

1 Examine `lab10-walkthrough03.html` in your browser and then in your editor of choice.

2 Edit js/`lab10-walkthrough03.js` by adding the following and then testing `lab10-walkthrough03.html` in browser.

```
$(':password').css('background-color', 'yellow');
```

This selects all password fields and sets their background color to yellow.

3 Comment out this code.

4 Add the following and test.

```
$('form :nth-child(4n)').css('background-color', 'yellow');
```

This selects every four child element within the form element and changes its background color.

5 Add the following and test.

```
$('label:contains("word")').css("color", "black");
```

This selects all <label> elements that contains the text “word” and sets the selected elements text color to black.

Exercise 10.4 — jQuery Listeners

1 Examine `lab10-walkthrough04.html` in your browser and then in your editor of choice.

2 Edit js/`lab10-walkthrough04.js` by adding the following and then testing `lab10-walkthrough04.html` in browser.

```
$(".panel").click(function() {
    $("#message").html("You clicked in the panel");
});
```

Here we are defining a click event handler for the .

3 Edit the previous step with the following and test.

```
$(".panel").on("click", function() {
    $("#message").html("You clicked in the panel");
});
```

The on() method allows you to specify any event. The previous step actually made use of a shortcut method, but not every event has a corresponding shortcut method. Thus the on() method is more reliable.

4 Edit the previous step with the following and test.

```
$(function () {
    $(".panel").on("click", function() {
        $("#message").html("You clicked in the panel");
    });
});
```

In JavaScript you can't perform DOM manipulations until it is loaded. In the jQuery code so far, we have put the code at the bottom of the page to get around this limitation. This code is a shortcut for the jQuery document ready event, which fires once the DOM is ready for manipulation. With this code, we can place our jQuery code anywhere (and not just at the bottom of the page).

5 Replace the previous step with the following and test.

```
$(function () {
    // chaining handlers
    $(".panel")
        .on("mousemove",function (e) {
           $("#message").html("x=" + e.pageX + " y=" + e.pageY);
        })
        .on("mouseleave",function (e) {
```

```
        $("#message").html("goodbye!");
    })
    .on("click",function () {
        $("#message").html("stopped move reporting");
        $(".panel").off("mousemove");
    });
});
```

This example binds several events. Notice that the click event unbinds, or turns off, the mouse move event.

Exercise 10.5 — Inserting DOM Elements

1 Examine `lab10-walkthrough05.html` in your browser and then in your editor of choice.

2 Edit js/`lab10-walkthrough05.js` by adding the following and then testing `lab10-walkthrough05.html` in browser.

```
$(function () {
    // create a new DOM element
    var img = $('<img src="images/art/thumbs/13030.jpg" >');
    // and now add the new element after the panel
    var panel = $('.panel');
    panel.after(img);
});
```

3 Edit the previous step with the following and test.

```
$(function () {
    // create a new DOM element
    var img = $('<img src="images/art/thumbs/13030.jpg" >');
    // and now add the new element after the panel
    var panel = $('.panel');
    panel.before(img);
});
```

4 Modify the last line as follows and test.

```
panel.append(img);
```

5 Modify the last line as follows and test.

```
panel.prepend(img);
```

6 Modify the last line as follows and test.

```
img.appendTo(panel);
```

This provides an alternative to what you did in step 4.

7 Modify the last line as follows and test.

```
img.prependTo(panel);
```

This provides an alternative to what you did in step 5.

8 Modify the last line as follows and test.

```
img.insertBefore(panel);
```

This provides an alternative to what you did in step 3.

Exercise 10.6 —Simple jQuery Animation

1 Examine `lab10-walkthrough06.html` in your browser and then in your editor of choice.

2 Edit `js/lab10-walkthrough06.js` by adding the following and then testing `lab10-walkthrough06.html` in browser.

```
$(function() {
    $("#btnFadeIn").click(function() {
        $("figure").fadeIn(1000);
});
```

This fades the image in across 1000 ms. Try repeatedly clicking the fade in button. You will notice that once an image has faded in, repeated clicks do nothing.

3 Add the following event handler and test.

```
$(function() {
    $("#btnFadeIn").click(function() {
        $("figure").fadeIn(1000);
    });
    $("#btnFadeOut").click(function() {
        $("figure").fadeOut(1000);
    });
});
```

This fades the image out across 1000 ms.

4 Add the following event handler after the fade out and test.

```
$("#btnFade").click(function() {
    $("figure").fadeToggle(500);
});
```

The toggle() method will fade the content in or out depending on its current state.

5 Add the following event handlers for the slide buttons and test.

```
$("#btnSlideDown").click(function() {
    $("figure").slideDown("slow");
});
$("#btnSlideUp").click(function() {
    $("figure").slideUp("slow");
});
$("#btnSlide").click(function() {
    $("figure").slideToggle(2000);
});
```

The finished result should look similar to that shown in Figure 10.1

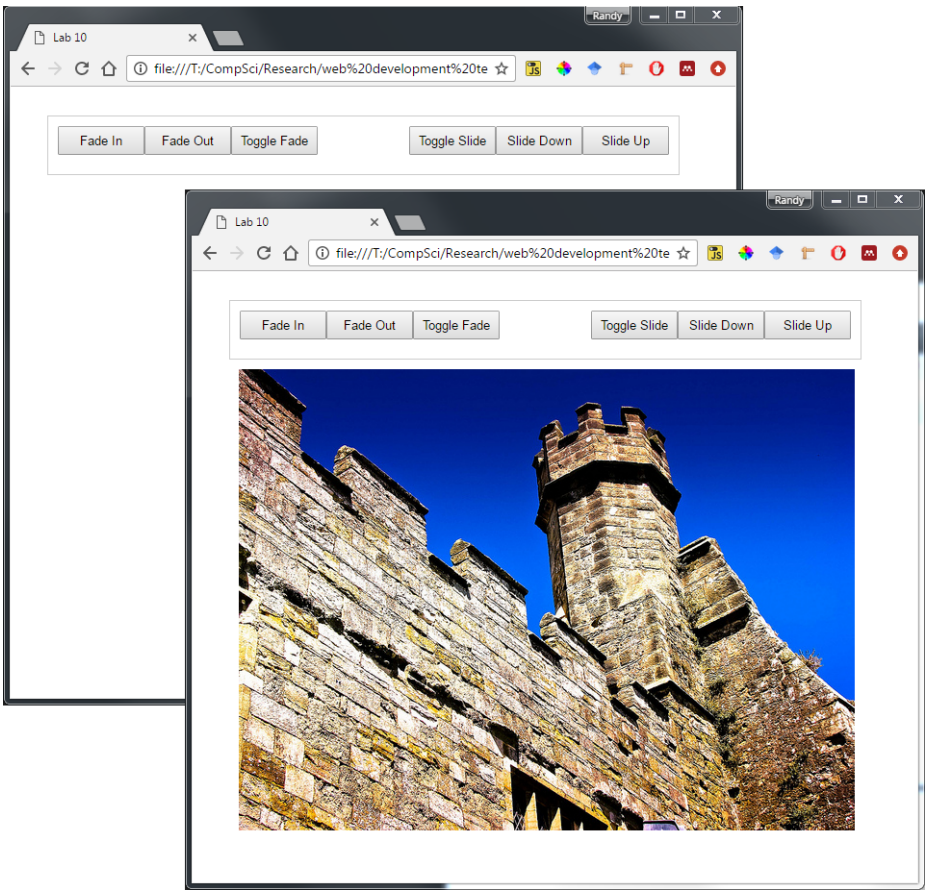

Figure 10.1 – Finished Exercise 10.6

Exercise 10.7 —More jQuery Animation

1 Examine `lab10-walkthrough07.html` in your browser and then in your editor of choice. In this example, you will dynamically generate a pop-up that displays a larger version of the image under the mouse.

2 Edit js/`lab10-walkthrough07.js` by adding the following and then testing `lab10-walkthrough07.html` in browser.

```
$(function () {

    $('figure img').on('mouseover', function (e) {

        // construct preview filename based on existing img
        var alt = $(this).attr('alt');
        var src = $(this).attr('src');
```

```
        var newsrc = src.replace("small","medium");

        // make dynamic element with larger preview image and caption
        var preview = $('<div id="preview"></div>');
        var image = $('<img src="' + newsrc + '">');
        var caption = $('<p>' + alt + '</p>');
        preview.append(image);
        preview.append(caption);
        $('body').append(preview);

        // make small image gray
        $(this).addClass("gray");

        // and then fade preview in
        $("#preview").fadeIn(1000);
    });
});
```

This will dynamically add a that contains a larger version of the image the mouse is over at the end of the document.

3 Add the following two functions.

```
function removePreview() {
    // remove the dynamic element and the gray class
    $(this).removeClass("gray");
    $("#preview").remove();
}

function movePreview(e) {
    // position preview based on mouse coordinates
    $("#preview")
        .css("top", (e.pageY - 10) + "px")
        .css("left", (e.pageX + 30) + "px");
}
```

4 Add the following event triggers and test.

```
$("figure img").on("mouseleave",  removePreview);
$("figure img").on("mousemove", movePreview);
```

The result should look similar to that shown in Figure 10.2.

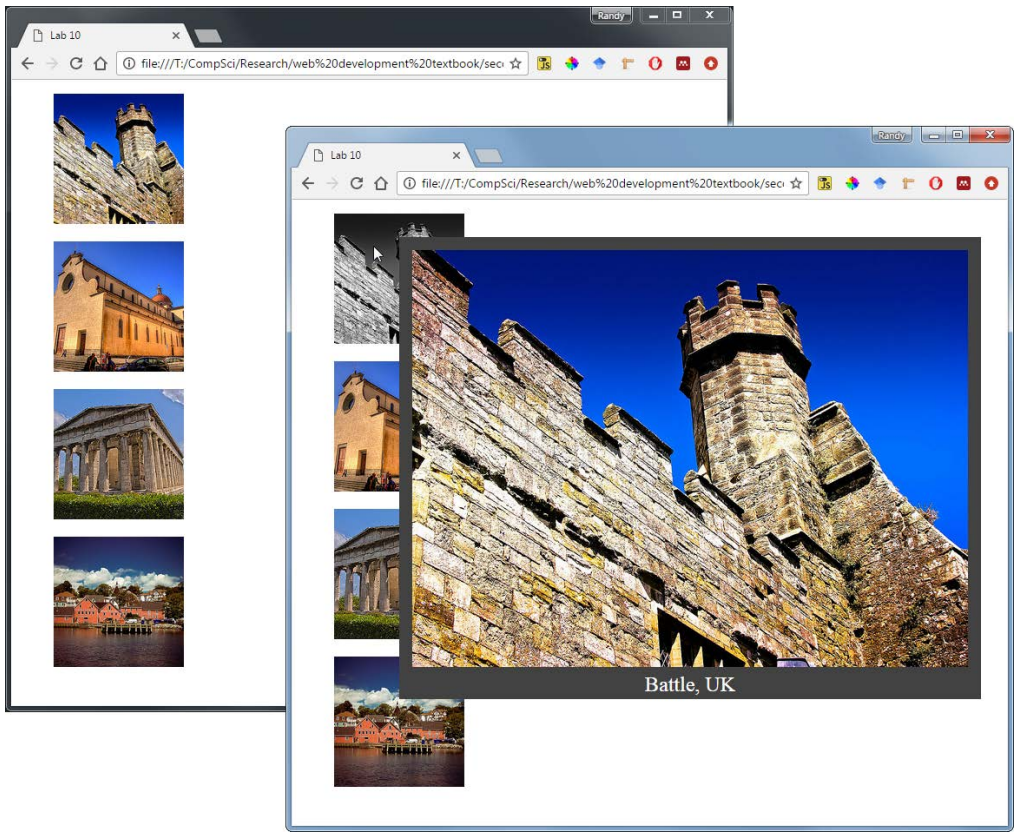

Figure 10.2 – Finished Exercise 10.7

Exercise 10.8 — Complex Animations

1 Examine `lab10-walkthrough08.html` in your browser and then in your editor of choice.

2 Edit js/`lab10-walkthrough08.js` by adding the following and then testing `lab10-walkthrough08.html` in browser.

```
$(function() {
    $("button").click(function(){
        $("#box").animate({left: '495px'});
    });
```

```
});
```

This will animate the box element's left CSS position property.

3 Modify the click handler as follows and test.

```
$("button").click(function(){
    $("#box").animate({
        left: '495px',
        opacity: '1',
        height: '50px'
    });
});
```

In this example, we are animating three CSS properties.

4 Modify the click handler as follows and test.

```
$("button").click(function(){
    $("#box")
        .animate({
            left: '495px',
            opacity: '1',
            height: '50px'
        })
        .animate({top: '400px', opacity: '0.2'})
        .animate({height: '200px'}, 2000);
});
```

In this example, three animations are defined: each runs after the previous one finishes.

Exercise 10.9 — GET REQUESTS

1 Examine `lab10-walkthrough09.html` in your browser and then in your editor of choice. In this exercise, we are going to consume an external web service.

2 In a browser, make the following request:

`http://www.randyconnolly.com/funwebdev/services/travel/cities.php`

This service returns a list of cities in the JSON format.

3 Edit js/`lab10-walkthrough09.js` by adding the following and then testing `lab10-`

walkthrough09.html in browser.

```
$(function() {

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

        var url =
"http://www.randyconnolly.com/funwebdev/services/travel/cities.php";

        $.get(url, function (data, status) {

                        var list = "";

                        // loop through JSON data and add each city to list

                        for (var i=0; i < data.length; i++) {

                            list += data[i].name + "<br>";

                        }

                        $("#results").html(list);

                    });

    });

});
```

This will make an asynchronous GET request from the web service.

4 While our code will work, it doesn't handle errors. The next exercise will do so in a more robust manner, but until then, we can make use of the status variable. Add the following conditional and test.

```
if (status == "success") {

    var list = "";

    for (var i=0; i < data.length; i++) {

        list += data[i].name + "<br>";

    }

    $("#results").html(list);

}
```

5 Another approach to looping through the returned data is to make use of the jQuery each() method. Modify your click event handler as follows and test.

```
$("#consume").click(function() {

    var url =
"http://www.randyconnolly.com/funwebdev/services/travel/cities.php";

    $("#results").html("<ul></ul>");

    $.get(url, function (data, status) {

            $.each(data, function(index,value) {
```

```
            var li = $('<li/>').html(value.name);
            li.appendTo("div#results ul");
        });
    }
});
});
```

Exercise 10.10 — jQXHR Handling

1 Examine `lab10-walkthrough10.html` in your browser and then in your editor of choice. In this exercise, we are going to consume an external web service but use the jqXHR object for more robust error handling.

2 Edit js/`lab10-walkthrough10.js` by adding the following and then testing `lab10-walkthrough10.html` in browser.

```
$(function() {
    $('.animLoading').hide();
    $("#consume").click(function() {
        $('.animLoading').show();
        var url =
  "http://www.randyconnolly.com/funwebdev/services/travel/cities.php";

        $.get(url)
            // done will happen first
            .done(function(data) {
                $.each(data, function(index,value) {
                    var li = $('<li/>').html(value.name);
                    li.appendTo("div#results ul");
                });
            })
            // if error occurs will execute, but before always()
            .fail(function(xhr,status,error) {
                alert("failed loading data - status=" + status +
                        " error=" + error);
            })
```

```
            // always should be last
            .always(function(data) {
                $('.animLoading').fadeOut("slow");
            });
    });
});
```

SUBMIT YOUR COMPLETED LAB FILES

After completing all the exercises above, please rename the directory into lab10_yourname, and compress your completed files inside into ***lab10_yourname.zip***. Please submit the zip file via ***dropbox*** on ***elearning*** under ***lab10 Code Submission*** **before Sunday 11:00PM.** Each lab is worth 1.5% of your total grade.