

What You Will Learn

- Extending JavaScript using jQuery
- Building and using external JavaScript (and jQuery) files
- Manipulating the DOM and handling events using jQuery
- Making asynchronous requests in JavaScript using jQuery
- Using Bootstrap to expedite process of building interfaces and styles
- Using external fonts such as font-awesome

The jQuery version used throughout the handout may not be the latest one.

Preparing Directories

1. Create a folder in your personal drive for this tutorial (call it tutorial3).
2. From Canvas → Modules → Module 3 → Tutorial 3, download the file **tutorial3.zip** and extract the files in the folder tutorial3) you created in the first step.
3. **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 to about JavaScript's prototype mechanism first and then use that mechanism with jQuery.

1. Become Familiar with Files

- 1 Open the **css** folder and examine the various CSS files that are already provided.
- 2 Open the **js** folder and examine the various JavaScript files that are already provided
- 3 Throughout the tutorial, we use CSS class names such as container, row, col-sm-4, etc. These are prebuilt CSS styles that are part of the Bootstrap framework. Please visit the following this [link](#) to become familiar with Bootstrap. Using Bootstrap, it becomes faster and easier to build responsive web pages. That is, as a developer, you can take advantage of using the existing CSS styles and you can customize your own.

2. Setup jQuery & Basic Selectors

- 1 Examine tut3-walkthrough02.html in your browser and then in your editor of choice.
- 2 Add code to load jQuery from the Content Delivery Network (CDN) in the <head> section of your page as follows (note the latest version number may be higher):

```
<script src="http://code.jquery.com/jquery-3.1.1.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-3.1.1.min.js"></script>');
</script>
```
- 4 Continue with your file from the step 1: setup jQuery.

- 5 Edit js/tut3-walkthrough02.js by adding the following and then testing tut3-walkthrough02.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.

- 6 Add the following code.

```
$('#first').html = "new Title";
```

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

- 7 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.

- 8 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.

- 9 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.

- 10 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.

3. Advanced Selectors

- 1 Examine tut3-walkthrough03.html in your browser and then in your editor of choice.

- 2 Edit js/tut3-walkthrough03.js by adding the following and then testing tut3-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.

4. jQuery Listeners

- 1 Examine tut3-walkthrough04.html in your browser and then in your editor of choice.

- 2 Edit js/tut3-walkthrough04.js by adding the following and then testing tut3-walkthrough04.html in browser.

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

Here we are defining a click event handler for the <div>.

- 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.

5. Inserting DOM Elements

- 1 Examine tut3-walkthrough05.html in your browser and then in your editor of choice.
- 2 Edit js/tut3-walkthrough05.js by adding the following and then testing tut3-walkthrough05.html in browser.

```
$(function () {  
    // create a new DOM element  
    var img = $('');  
    // 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 = $('');  
    // and now add the new element before 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.

6. Simple jQuery Animation

- 1 Examine `tut3-walkthrough06.html` in your browser and then in your editor of choice.
- 2 Edit `js/tut3-walkthrough06.js` by adding the following and then testing `tut3-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 1

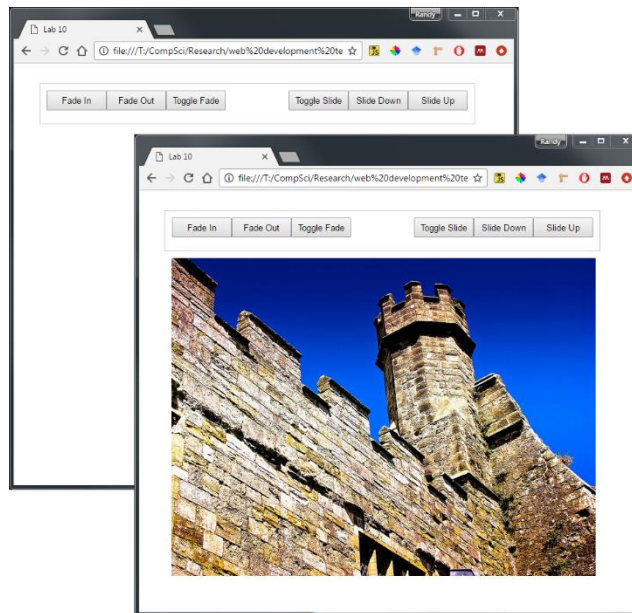


Figure 1 – Finished Step 6.

7. GET REQUESTS

- 1 Examine tut3-walkthrough07.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://faculty.washington.edu/ealmasri/cities.php`

This service returns a list of cities in the JSON format.
- 3 Edit js/tut3-walkthrough07.js by adding the following and then testing tut3-walkthrough09.html in browser.

```
$(function() {  
    $("#consume").click(function() {  
        var url = "http://faculty.washington.edu/ealmasri/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.

8. jqXHR Handling

- 1 Examine tut3-walkthrough08.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/tut3-walkthrough08.js by adding the following and then testing tut3-walkthrough08.html in browser.

```
$(function() {  
    $('.animLoading').hide();  
    $("#consume").click(function() {  
        $('.animLoading').show();  
        var url = "http://faculty.washington.edu/ealmasri/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");  
    });  
});  
});
```