Introduction to jQuery– Part One

Website Development 2

Lecture Outline

- What is jQuery?
- Selectors
- Filters
- Adding content to a page
- Modifying CSS properties
- Setting attributes of tags

What is jQuery?

- jQuery is an open source JavaScript library that simplifies the interactions between an HTML document, or more precisely the Document Object Model (aka the DOM), and JavaScript
- jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library.
- jQuery is not a language but it is a well written JavaScript code. As quoted on an official jQuery website, "it is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development."

What is jQuery?

- The jQuery library contains the following features:
  - HTML/DOM manipulation
  - CSS manipulation
  - HTML event methods
  - Effects and animations
  - AJAX
  - Utilities

How to use jQuery

- Include jQuery in your webpage by using link to jQuery JavaScript file hosted online:

```
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

Lecture Outline

- What is jQuery?
- Selectors
- Filters
- Adding content to a page
- Modifying CSS properties
- Setting attributes of tags

Selectors

- The jQuery library harnesses the power of Cascading Style Sheets (CSS) selectors to let us quickly and easily access elements or groups of elements in the Document Object Model (DOM).
- A jQuery Selector is a function which makes use of expressions to find out matching elements from a DOM based on the given criteria.
- Simply you can say, selectors are used to select one or more HTML elements using jQuery. Once an element is selected then we can perform various operations on that selected element.

Selectors

- jQuery selectors start with the dollar sign and parentheses - $().
- $ is shorthand for jQuery.
- $() makes use of following three building blocks while selecting elements in a given document:
  - Elements/tags
  - IDs
  - Classes

Tag Selectors

- You can select any page element by its tag name. For example: to select every <a> tag on the page, you'd write: **$('a');**
- In the following example, the paragraphs are highlighted (background colour has changed) because the elements with the <p> tag are selected and modified.

```
$(document).ready(function(){
  $('p').css('background-color','#ac9986');
});
```

Id Selectors

- You can select any page element that has an ID applied to it using jQuery and a CSS ID selector.
- For example, say you have the following HTML in a Web page:

  Special message
- To select the element using jQuery:

  $('#message');
- You have to use the CSS-syntax for defining an ID selector ('#message').

Id Selectors

- The following example selects an element with an id of *p2* and changes the background colour to yellow.

```
$(document).ready(function(){
  $("#p2").css("background-color", "yellow");
});
```

Class Selectors

- You can select all page elements with the same class name. For example: to select every element on the page with the class name submenu, you'd write:

  $('.submenu');
- In the following example, any element which has a class **friends** are highlighted (background colour has changed) because the elements with the class friends are selected and modified.

```
$(document).ready(function(){
  $('.friends').css('background-color','#ac9986');
});
```

Attribute Selectors

- Attribute Selectors let you select elements based on whether the element has a particular attribute, and even check to make sure the attribute matches a specific value.

- **$('img[alt]')** select all <img> tags with the alt attribute set.
- **$('input[type="text"]')** select all text boxes in a form.
- **$('a[href^="http://"]')** select all links that point outside your site (i.e. links that start with http://).
- **$('a[href^="mailto:"]')** select all mailto: links.
- **$('a[href$=".pdf"]')** select all links that point to PDF files (i.e. links that end with .pdf).
- **$('a[href*="wit.ie"]')** select all links that point to wit.ie (i.e. links that include wit.ie).

Document.ready

- You might have noticed that all jQuery functions, in our examples, are inside a document.ready() function.
- As almost everything we do when using jQuery reads or manipulates the document object model (DOM), we need to make sure that we start adding events etc. as soon as the DOM is ready.
- If you want an event to work on your page, you should call it inside the $(document).ready()function.
- Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.

Document.ready

```
$(document).ready(function(){
  $('a[href^="http://"]').css({
   'font-weight':'bold',
   'font-style':'italic',
   'color':'black'
  });
});
```

Lecture Outline

- What is jQuery?
- Selectors
- Filters
- Adding content to a page
- Modifying CSS properties
- Setting attributes of tags

Filters

- jQuery also provides a way to filter your selections based on certain characteristics.
- For example, the **:even** filter lets you select every even element in a collection of elements.
- In addition, you can find elements that contain particular tags, specific text, elements that are hidden from view, and even elements that do not match a particular selector.
- For Example, to find every row of a table, the jQuery selector would be as follows: **$('tr');** and **$('tr':even);** finds every even row.

Filters

- **:even** and **:odd** select every other element in a group.
- **:first** and **:last** matches the first selected element and last selected element respectively.
- **:not()** finds elements that do not match a particular selector type.
  - For example, say you want to select every <a> tag except ones with a class of navButton.
  - You could do that as follows: **$('a:not(.navButton)');**
- **:has()** finds elements that contain another selector.
  - For example, say you want to find all <li> tags, but only if they have an <a> tag inside them.
  - You could do that as follows: **$('li:has(a)');**

Filters

- **:contains()** finds elements that contain specific text.
  - For example, to find every link that says "Click Me!" you can create a jQuery object like this: **$('a:contains("Click Me!")');**
- **:hidden** locates elements that are hidden, which includes elements that either have the CSS display property set to none, elements that are hidden using jQuery's hide() function, or hidden form fields.
  - For example, if you have hidden several tags, you can find them and make them visible as follows: **$('div:hidden').show();**
- **:visible** is the opposite of hidden. It locates elements that are visible on the page.

Automatic loops

- If jQuery returns a set of elements, you would think you need a loop to process each element in turn, but because looping through a collection of elements is so common, jQuery functions have that feature built right in.
- In other words, when you apply a jQuery function to a selection of elements, you do not need to create a loop yourself, since the function does it automatically.
- For example, to select all images inside a tag with an ID of slideshow and then hide those images, you write this in jQuery:

```
$('#slideshow img').hide();
```

Automatic loops

- In the following example we hide all paragraphs (<p>) using: **$('p').fadeOut(1500)** when a button with an ID of *btn1* is selected; and we display/show all paragraphs: **$('p').fadeIn(1500)** when a button with an ID of *btn2* is selected.

```
$(document).ready(function(){
  $('button#btn1').on('click', function(){
     $('p').fadeOut(1500);
  });

  $('button#btn2').on('click', function(){
     $('p').fadeIn(2500);
  });

});
```

Chaining functions

- Sometimes you will want to perform several operations on a selection of elements.
- For example, say you want to set the width and height of a tag (with an ID of popup) using JavaScript.
- jQuery lets you do this as follows:

```
$('#popup').width(300).height(300);
```

Chaining functions

- jQuery uses a unique principle called chaining, which lets you add functions one after the other.
- Each function is connected to the next by a period(.), and operates on the same jQuery collection of elements as the previous function.
- For example, say you not only want to set the width and height of the tag but also want to add text inside the and make it fade into view (assuming it is not visible at present).
- You can do that very succinctly as follows:

```
$('#popup').width(300).height(100).text("This is an example of Chaining").fadeIn(1000);
```

Lecture Outline

- What is jQuery?
- Selectors
- Filters
- Working with content on a page
- Modifying CSS properties
- Setting attributes of tags

Working with content on a page

- jQuery provides many functions for manipulating elements and content on a page, from simply replacing HTML, to precisely positioning new HTML in relation to a selected element, to completely removing tags and content from the page.
  - **.html()**
  - **.text()**
  - **.append()**
  - **.prepend()**
  - **.before()**
  - **.after()**

Retrieving content

- To illustrate the **.html()** method we are going to work with the following HTML:

```
<div id = "errors">
<h2>Errors</h2>
</div>
```

- To retrieve the HTML currently inside the selection, just add .html() after the jQuery selection, as in the following example:

```
alert($('#errors').html());
```

- This example produces an alert with the text <h2>Errors</h2>.

Adding content

- If you supply a string as an argument to **.html()**, you can set the html contents of every matched element.

```
$('#errors').html('<h3>More Errors</h3>');
```

- **text()** returns the text contents of all matched elements.
- If you supply a string as an argument to **.text()**, you can set the text contents of all matched elements.
- **Note:** Any HTML tags that are passed to text() are encoded and displayed and not treated as tags.

Adding content

- **.append()** adds HTML as the last child element of the selected element. The append() function is a great way to add an item to the end of a bulleted (<ul>) list or a numbered (<ol>) list.
- As an example, we will add a fruit list item to the end of an already existing list:

```
$('#fruits').append('<li>Orange</li>');
```

- **prepend()** is just like append(), but adds HTML after the opening tag of the selected element.
- As an example, we will add a fruit list item to the start of an already existing list:

```
$('#fruits').prepend('<li>Orange</li>');
```

Lecture Outline

- What is jQuery?
- Selectors
- Filters
- Working with content on a page
- Modifying CSS properties
- Setting attributes of tags

Modifying CSS properties

- Adding, removing, and changing elements is not the only thing jQuery is good at. You will often want to, for example, add a class to a tag or change a CSS property of an element.
- We can create some really advanced visual effects by simply using jQuery to add, remove, or change a class applied to an element.
- jQuery provides several functions for manipulating a tag's class attribute:
  - addClass()
  - removeClass()
  - toggleClass()

Modifying CSS properties

- **addClass()** adds a specified class to an element.
- You can add the addClass() after a jQuery selection and pass the function a string, which represents the class name you wish to add.
- For example, to add the class externalLink to all links pointing outside your site, you can use this code:

```
$('a[href^="http://"]').addClass('externalLink');
```

- This would take HTML like this:

```
<a href = "http://www.oreilly.com/">
```

- And change it to the following:

```
<a href = "http://www.oreilly.com/" class = "externalLink">
```

Modifying CSS properties

- **removeClass()** is the opposite to addClass().
- It removes a single class, multiple classes, or all classes from each element in the set of matched elements.
- For example, if you wanted to remove a class named highlight from a with an ID of alertbox, you would do this:

```
$('#alertbox').removeClass('highlight');
```

- **toggleClass()** allows you to toggle a particular class - meaning it will add the class if it does not already exist, or remove the class if it does.
- Toggling is a popular way to show an element in either an on or off state.

Lecture Outline

- What is jQuery?
- Selectors
- Filters
- Working with content on a page
- Modifying CSS properties
- Working with tag attributes

Working with tag attributes

- jQuery's css() function lets you directly change CSS properties of an element, so instead of simply applying a class style to an element, you can immediately add a border or background colour, or set a width or positioning property.
- You can use the css() function in three ways:
  - Get the current value of a CSS property
  - Set the CSS property for an element
  - Change multiple CSS properties at once

Working with tag attributes

Get the current value of a CSS property

- For example, say you want to find the background colour of a tag with an ID of main:

```
var bgColor = $('#main').css('background-color');
```

- After this code runs, the variable bgColor will contain a string with the element's background colour value.

Working with tag attributes

Set the CSS property for an element

- To use the function in this way, you must supply two arguments to the function: the CSS property name and a value.
- For example, to change the font size for the tag to 200%, you could do this:

```
$('body').css('font-size', '200%');
```

- To add a black, one pixel border around all paragraphs with a class of highlight:

```
$('.highlight').css('border', '1px solid black');
```

Working with tag attributes

Change multiple CSS properties at once

- If you want to change more than one CSS property on an element, you do not need to resort to multiple uses of the .css() function.
- Instead, we can pass what is called an object literal to the .css() function. Think of an object literal as a list of property name/value pairs.

```
$('.highlight').css(
 {
  'background-color' : '#FF0000',
  'border': '2px solid #FE0037'
 }
);
```

Working with tag attributes

- jQuery includes general purpose functions for handling HTML attributes such as the **attr()** and **removeAttr()** functions.
- The **attr()** method can be used to either fetch the value of an attribute from the first element in the matched set or set attribute values onto all matched elements. For example, to determine the current graphic file a particular <img> points to, you pass the string 'src' to the function:

```
var imagefile = $('#banner img').attr('src');
```

Working with tag attributes

- If you pass a second argument to the attr() function, you can set the tag's attribute.
- For example, to swap in a different image, you can change an <img> tag's src property like this:

```
$('#banner img').attr('src', 'images/newImage.jpg');
```

Working with tag attributes

- If you want to completely remove an attribute from a tag, use the **removeAttr()** function.
- For example, this code removes the bgColor property from the tag:

```
$('body').removeAttr('bgColor');
```

Anonymous functions

- As stated previously, jQuery functions have automatic loops built in.
- For example, to hide all external referenced links in a web page the following would suffice: **$('a[href^="http://"]').hide();**
- However, what if you wanted to retrieve the URL for each selected link and do something with it (such as display it somewhere else on the page) - how do we achieve this?
- The problem is that there is a few tasks to complete, but the solution is rather simple. jQuery does not have a built-in function that performs all the tasks you require, but we can use the **each()** function.

Anonymous functions

- To use the each() function, you pass a special kind of argument to it - an anonymous function. The anonymous function is simply a function containing the steps that you wish to perform on each selected element. It is called anonymous because you don't give it a name.
- Here is how you incorporate an anonymous function as part of the each() function:

```
$('selector').each(
  function() {
   //code goes here
  }
);
```

Anonymous functions

- The each() function works like a loop - meaning the instructions inside the anonymous function will run once for each element that you have retrieved.
- For example, say you have 10 images on a page and add the following JavaScript code:

```
$('img').each(function() {
    alert('Image found');
});
```

- Ten alert dialog boxes will appear.

Anonymous functions

- When using the each() function, you will want to access or set attributes of each element - for example, to find the URL for each external link.
- To access the current element through the loop, you use a special keyword called **this**.
- The this keyword refers to whatever element is calling the anonymous function. So the first time through the loop this refers to the first element in the jQuery selection, while the second time through the loop, this refers to the second element.
- We write this as **$(this)**.

Anonymous functions

- In this example, the URL for each external link is extracted and added to an unordered list with an ID of *bibList*.

```
$('a[href^="http://"]').each(function() {
    var extLink = $(this).attr('href');
    $('#bibList').append('<li>' + extLink + '</li>');
});
```