Lecture 4

Manipulating Content with jQuery

First I want to talk about some of the methods that you'll be seeing with jQuery. JQuery has numerous methods that will help you manipulate the DOM, but I want to highlight some of the ones you'll run into the most. Many of the examples I give are derived from the examples on jquery.com. I highly suggest you bookmark the site as a reference. It has myriad examples and tutorials for every aspect of jQuery.

.attr() method

- .attr() = gets attribute value from first matched element

```
<p>Sidney is a <em title="tuxedo">black and white</em> cat.</p>

The title of the emphasis is:<div></div>

<script>
var title = $("em").attr("title");
//gets the title attribute of #em
 $("div").text(title);
//places the text from the title attribute into the div
</script>
```

- What would be returned is:

```
The title of the emphasis is:
tuxedo
```

This is a very useful one when you want to change an attribute. You can get the attribute, then change what you need to change.

In this example, we are pulling out the “em” element that has the attribute of title attached.

.html() method

- .html() = gets HTML content from matched element

```
<p>Sidney</p>
<button>Who is the best cat in the world</button>

<script>
$(document).ready(function(){
  $("button").click(function(){
    alert($("p").html());
// Similar to innerHTML - document.getElementById().innerHTML
  });
});
</script>
```

- What would be returned when button is clicked is:

```
<p>Sidney</p>
```

If the selector expression matches more than one element, only the first match will have its HTML content returned.

Each HTML element has an innerHTML property that defines both the HTML code and the text that occurs between that element's opening and closing tag.

.text() method

- .text() = gets text content from matched element

```
<p>Sidney</p>
<button>Who is the best cat in the world</button>

<script>
$(document).ready(function(){
  $("button").click(function(){
    alert($("p").text());
    });
});
</script>
```

- What would be returned when button is clicked is:

Sidney

So it stands to reason that there's a text method as well. One major difference between the html and text methods is that the text methods can be used on html and xml documents. The html method cannot be used on xml documents. Another major difference is that the html method returns everything, including the tags -- so it returns the P tags along with text. The text method only returns the text between the opening and closing P tags.

.toggleClass() method

- .toggleClass() = adds or removes classes from matched element

```
<style>
 p { margin-top: 2px; font-size:1.3em; }
  .cat { background:yellow; color:red; }
  .bestcat { background:white; color:black; }
</style>

<p class="cat">Sidney</p>

<script>
   $("p").click(function () {
     $(this).toggleClass("bestcat");
   });
</script>
```

This method allows you to toggle between class. For example, if you have an element with "bestcat" class attached, when the toggleClass method is used, it will switch it by toggling it on and off. What happens is that if the element already has the bestcat class, it will remove it. And if it doesn't have that class, it will add it. This is an alternative to addClass and removeClass, which we saw in Module 2. What addClass does is it only adds the class. removeClass will only remove the class. toggleClass will go back and forth.

.delay() method

- .delay() = delays execution of functions

```
<style>
div { width: 100px; height: 100px; float: left; }
.asumaroon { background-color: #903; }
.asugold { background-color: #fc3; }
</style>

<p><button>Click for ASU Maroon and Gold</button></p>
<div class=”asumaroon"></div>
<div class=”asugold"></div>

<script>
   $("button").click(function() {
     $("div.asumaroon”).slideUp(400).delay(600).fadeIn(400);
     $("div.asugold”).slideUp(400).fadeIn(400);
   });
</script>
```

This one is a biggie if you’re going to use any effects.

Sequential fade-ins, or the fading in of elements one after the other, establish a sense of professionalism for a website. Additionally, the fade-in helps guide the user’s eyes to the content you find most important.

What this code does is animate the hiding and showing of two divs, delaying the first before showing it.

.val() method

- val() = gets value of first matched element; great for form use

```
$('select.cat option:selected').val();
// gets value from dropdown select

$('select.cat').val();
// gets value from dropdown select but with less code

$('input:checkbox:checked').val();
// gets value from checked checkbox

$('input:radio[name=bestcat]:checked').val();
// gets value from set of radio buttons
```

You use select when dealing with dropdown lists, and you use input when dealing with checkboxes or radio buttons. This one is really useful if the user clicks on a checkbox or radio button and you want to find out which one they clicked on and grab that information in a variable or array.

Iteration

- Iterative statements = looping statements
- Iterative statements in JavaScript:
  - while
  - do while
  - for
  - for…in
- Iterative method (statement) in jQuery:
  - .each()

Looping statements determine how many times (from zero or more) a block of code is executed based on a specified condition.

.each() vs. jQuery.each()

- These are not the same!
- .each() method = iterates over jQuery object ONLY
- jQuery.each() function = iterates over ANY collection (object or array)

The each method iterates over a jQuery object, executing a function for each matched element.

.each() method

```
<ul>
   <li>Tuxedo</li>
   <li>Tabby</li>
   <li>Manx</li>
</ul>
 <p>Types of cats include:</p>
<script>
  $('li').each(function(index) {
     alert(index + ': ' + $(this).text());
  //Loops and retrieves the text value of each li
  });
</script>
```

- What returns in an alert box is:

  0: Tuxedo
  1: Tabby
  2: Manx

From jQuery.com - "The .each() method is designed to make DOM looping constructs concise and less error-prone. When called it iterates over the DOM elements that are part of the jQuery object. Each time the callback runs, it is passed the current loop iteration, beginning from 0. More importantly, the callback is fired in the context of the current DOM element, so the keyword this refers to the element."

What this code does is it sets a list of types of cats, and you can see the UL. Then there's a script that is iterating through each of the LI's in that list using .each. And what it will do is trigger an alert box that will write the index number of the LI (remember, JavaScript starts at 0 when dealing with index) and the text from the LI. Note that there is a single quote – colon – single quote, which if you see what returns, is putting that semicolon between the returned index number and the returned text.

I want to talk a little about the word "this". You can see that in

the part of the code $(this).text(). This is a carryover from JavaScript, and it's a keyword that is used in methods to refer to the object to which a method is being performed. So in this case, the each is iterating through each of the LI's, so the "this" keyword is doing the same: First it stands for the text at the index 0, which is Tuxedo, then index 1, which is Tabby, etc. The "this" keyword allows you not to have to type more code to deal with each LI individually. Anything that you can minify your code is a good thing. And, hey, if it saves you time, that's a good thing, too.

Assignment and reading for Module 4

- Reading:
  - Learning jQuery 4th Edition
  - Additional material
- Assignment 5: Student Resources