The Learning House

427 S 4th Street #300

Louisville KY 40202

Introduction to jQuery

.NET Cohort

Coding Bootcamp

SOFTWARE GUILD

Lesson Goals

1. Learn how to include jQuery in your project
2. Learn basic selector syntax
3. Learn some of the common, built-in functions
4. Introduction to plug-ins

Why jQuery?

jQuery makes it easy to handle differences between browsers and perform common tasks against the DOM *(document object model*, a fancy way of saying HTML).

Tasks like selecting DOM elements, adding effects like CSS classes, animations, and even executing Ajax requests to servers are trivial using jQuery.

It is also very popular, so there's a ton of examples, documentation, and plug-ins (e.g. jQuery Ui) for it.

Including jQuery in Your Page

We have a couple ways of doing this:

1. Link the script in the <head>

   <script type="text/javascript" src="jquery-2.0.3.js"></script>

2. Reference the script in our BundleConfig and include it in our Views or Shared Layout page
3. Use a CDN (Content Delivery Network) like Google's

   <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/2.0.3/jquery.min.js"></script>

Self-Host or CDN

Self-host if you want full control over the file and version you use or if your IT policy doesn't allow external references.

CDN if you want it to be cached by someone else (speed), to save bandwidth by using an external reference, or to stay on the current version by referencing a version-less path.

jQuery Command Format

Every jQuery statement starts with the jQuery alias, a selector, then an action and its parameters.

Alias → Selector → Action → Parameters

The alias calls the jQuery library, the selector chooses what DOM elements to manipulate, the action is the function to be performed, and the parameters customize the function.

jQuery Statement Examples

Alias	Selector	Action	Parameters
jQuery	‘p’	.css	(‘color’, ‘red’);
$	‘p’	.css	(‘color’, ‘red’);

Both of these statements select all of the paragraph tags in the DOM and add a CSS property of color set to the value red.

jQuery(‘p’).css(‘color’, ‘red’);
$(‘p’).css(‘color’, ‘red’);

Note that we can use jQuery or the dollar sign as the alias for the jQuery library. The dollar sign is more common.

First Things First:
Make Sure the Page is Ready!

To interact with HTML elements on the page, we first have to make sure they are loaded. Because web pages load asynchronously, we want to wait until everything is done before running our scripts. jQuery handles this with the *Document Ready Event*

```
/// <reference path="jquery-2.0.3.min.js" />

$(document).ready(function () {
    alert('The page is all loaded up and ready to go!');
});
```

* The reference path blurb is a Visual Studio trick to enable IntelliSense.

$(document).ready()

You're going to see this in just about every page. Most of jQuery's functionality needs all the elements to be loaded to work properly. You should only need to do this once, though.

It's so common that you can shortcut it, like so:

```
$(function() {
    alert('The page is all loaded up and ready to go!');
});
```

Selecting DOM Elements

This is the absolute core of jQuery. Nothing will work well without a target, so jQuery gives us various ways to select items to manipulate.

Selectors always go in parenthesis after the jQuery alias, like so:

```
$(function() {
    jQuery('SELECTORS GO HERE');
    $('SELECTORS GO HERE');
});
```

Selecting by Tag

The first type of selector selects every element on the page of a specific tag. For example, if we want all table rows:

$('tr')

Any HTML tag can be selected in this way:

$('p')

$('div')

$('input')

Getting More Specific

It’s rare that we would want all of the divs; usually, we target specific elements. If we put an id on an element, we can select just that element by using the same syntax as CSS:

$(‘#myElementId’)

We can also select all elements with a CSS class:

$(‘.myCSSClass’)

Drilling Down

Say we had a table with an id of customers and we wanted to select only the rows in the body. We can chain selectors together with spaces to "drill down" into elements, like so:

$('#customers tbody tr')

Protip: Because jQuery selectors return an array of elements, you can alert the .length of the selector for troubleshooting.

e.g. alert($('#customers tbody tr').length + ' elements found!');

Selecting Multiple Elements

Space separation allows us to drill down, but comma separation allows us to select multiple elements across the DOM.

$(‘p,div,h1’)

This would select all paragraph, div, and h1 tags.

Filtering

jQuery has some built-in filters that we can append to our selectors to get even more specific. Say, in our previous example, we only wanted the even rows:

$('#customers tbody tr:even')

Some of the most common filters are :even, :odd, :first, :last, and :eq(#).

E.g. :eq(3) will select the 3rd element.

Manipulating CSS Properties

jQuery allows us to read and write CSS properties using the .css function:

```
$(document).ready(function () {
    var fontSize = $('#customers tbody tr:first').css('font-size');
    alert(fontSize);
});
```

```
$(document).ready(function () {
    $('#customers tbody tr:even').css('background-color', '#dddddd');
});
```

Manipulating CSS Classes

Working directly with CSS properties is generally not the best way to go about things. It is better practice to reference CSS classes and add/remove the whole class at a time.

```
$(document).ready(function () {
    $('#customers tbody tr:even').addClass('text-info');
    $('#customers tbody tr:even').removeClass('text-info');
});
```

Hiding and Showing Elements… and Events!

jQuery has 3 methods for hiding and showing elements:

- hide(): hides the element
- show(): shows the element
- toggle(): if hidden, shows the element; if visible, hides the element

We can also register events to selected items. In our example, we register with the click() of a button.

There are more events in the documentation:

http://api.jquery.com/category/events

dblclick(), change(), blur(), focus(), hover(), keypress(), mouseenter(), mouseleave(), etc…

```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>

</head>
<body>
    <div>
        <p id="spoiler">This is something that will spoil the movie</p>
        <input type="button" id="showSpoiler" value="Show Spoiler" />
    </div>
    <script src="../Scripts/jquery-2.0.3.js"></script>
    <script type="text/javascript">
        $(document).ready(function () {
            $('#spoiler').hide();

            $('#showSpoiler').click(function () {
                $('#spoiler').show();
            });
        });
    </script>
</body>
</html>
```

Adding HTML Dynamically

jQuery can also add custom HTML dynamically to the DOM.

insertBefore() and insertAfter() places an HTML snippet into the DOM before or after a selected element.

prependTo() and appendTo() places HTML content inside a tag at the beginning or end.

remove() completely removes HTML.

```
<div>
    <p id="myText">This is some text</p>
</div>
<script src="../Scripts/jquery-2.0.3.js"></script>
<script type="text/javascript">
    $(document).ready(function () {
        $('<p>This goes before</p>').insertBefore('#myText');
        $('<p>This goes after</p>').insertAfter('#myText');

        $('<strong>First! </strong>').prependTo('#myText');
        $('<strong> Last!</strong>').appendTo('#myText');
    });
</script>
```

```
▼<div>
    <p>This goes before</p>
  ▼<p id="myText">
      <strong>First! </strong>
      "This is some text"
      <strong> Last!</strong>
    </p>
    <p>This goes after</p>
  </div>
```

Setting HTML and Text

text() and html() functions will read or write the text and HTML from an element.

```
<div>
    <p></p>
    <h2></h2>
</div>
<script src="../Scripts/jquery-2.0.3.js"></script>
<script type="text/javascript">
    $(document).ready(function () {
        $('p').html('<strong>Warning!</strong> Text has been replaced … ');
        $('h2').text('<strong>Warning!</strong> Title elements can be …');
    });
</script>
```

Warning! Text has been replaced …

Warning! Title elements can be …

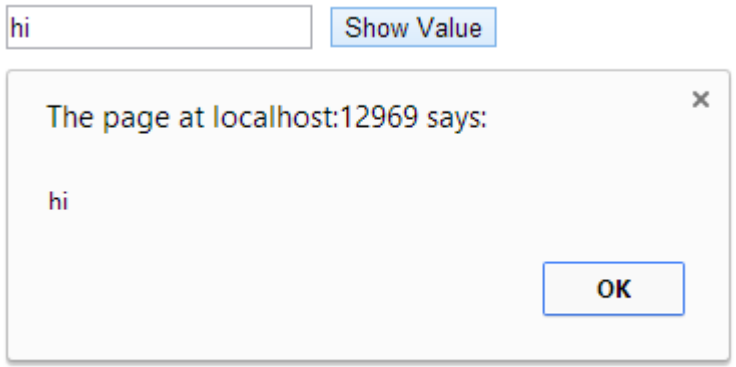
SOFTWARE GUILD

Reading Data

The val() method will read data from data elements like inputs.

Later on, we will introduce two-way binding frameworks like Knockout.js that make it easy for us to handle data.

```
<div>
    <input id="name" type="text"/>
    <button id="btnShowValue">Show Value</button>
</div>
<script src="../Scripts/jquery-2.0.3.js"></script>
<script type="text/javascript">
    $(document).ready(function () {
        $('#btnShowValue').click(function() {
            alert($('#name').val());
        });
    });
</script>
```



Animations

jQuery also has animations for changing properties. There are far too many to cover in slides…

Please go to CodeAcademy's jQuery track and do the effects exercises because, well… they're really well done.

Most of the time, in real applications, you will get animation scripts from a plug-in, not write them yourself.

Plug-ins

One of the main reasons why jQuery is so popular is that it has tons of plug-in scripts that were written by the community.

Go to plugins.jQuery.com and take a look! There are hundreds of plug-ins to do things like animations, forms and inputs, special controls (date pickers, grids, etc.).

Do not reinvent the wheel! (Unless you are practicing, of course.) In the UI world, if something hasn't already been done, it's probably dumb and you should think twice before doing it.

Conclusion

jQuery is a useful and popular framework that smooths out the differences between browsers.

As usual in the open source community, there are many frameworks, but the widespread use of jQuery and the abundance of plug-ins makes it an easy tool to put in your belt.

We covered the basics of how selectors work and many of the most common built-in functions. The rest is up to you to study on your own, or better yet, just find a framework/plug-in that does what you want!