Week 4 – Introducing jQuery

Introduction

jQuery is a JavaScript library. It provides us with simple methods for achieving complex tasks. jQuery's syntax allows us to side-step the intricacies of vanilla JavaScript.

How jQuery Works

Including jQuery

jQuery is an external JavaScript library. In order to use jQuery in your HTML document you must *include* the library in the page. This is done using the `<script>` tag in the `<head>` section of the page.

You can include jQuery from numerous online sources but this means you have to be connected to the Internet in order to use the library. We advise downloading the library into the same folder as your HTML documents.

At the time of writing, the current version of jQuery is v.3.3.1, which can be downloaded from http://jquery.com/download/. There may be a new version out by the time you read this. There are several types available; you want the *compressed, production jQuery*.

Once you have downloaded jQuery, you can include it in your HTML document:

```
<script src="jquery-3.5.0.min.js"></script>
```

Note: This assumes that a) you have placed the jQuery file in the same folder as your HTML documents and b) you have not changed its filename.

What jQuery Does

Once jQuery is included in an HTML page it allows you to manipulate the CSS properties of selectors (HTML tags, classes and IDs) via various events such as when the page loads or when an item is clicked. jQuery also comes packaged with some simplistic functions such as fades and slides as well as the more complex .animate() function which allows the animation of any CSS property that accepts a numerical value, e.g. font-size, top, width, border-width.

jQuery also provides a host of other useful tools for navigating and manipulating HTML documents: more on this next week.

Using jQuery

jQuery code will be typed in the `<head>` section of the document. You will require an additional set of `<script>` tags to the ones used to include the jQuery library.

```
<script>
$(document).ready(function() {
   //our code goes here
});
</script>
```

Inside the `<script> </script>` we always call the "document ready function". This is a function that ensures the page is fully loaded before any code we write can be run.

jQuery and Selectors

jQuery works by targeting selectors and then performing some action on them. If you don't understand selectors then you need to go back to the previous week and ensure that you do before continuing.

jQuery has its own syntax and it's different to HTML and CSS. jQuery identifies selectors like this:

```
$('selector')
```

Some examples:

Targeting an HTML tag	`$('body')` `$('p')`
Targeting a class	`$('.myClass')`
Targeting an ID	`$('#myID')`

Note that if you're targetting a class or an ID you need to include the prefix full stop or hash.

Once you've identified your selector you can then 'chain' an *action*. In this example, we use jQuery's inbuilt fadeOut() effect to make all the paragraphs disappear:

```
$('p').fadeOut();
```

The method is the same regardless of which effect you are using or what selector you are targetting:

```
$(selector).action();
```

List of jQuery Effects

jQuery comes with a few inbuilt effects that are very easy to use:

Effect	Syntax
hide: instantly hides the targeted selector	`$('selector').hide();`
show: instantly shows the targeted selector	`$('selector').show();`
fadeOut: hides the targeted selector by fading it out	`$('selector').fadeOut();`
fadeIn: shows the targeted selector by fading it in	`$('selector').fadeIn();`
slideDown: shows the targeted selector by sliding it down	`$('selector').slideDown();`
slideUp: hides the targeted selector by sliding it up	`$('selector').slideUp();`

The effects can also accept a numerical value (measured in milliseconds) to set the duration of the effect. For example:

`$('p').fadeIn(500);` sets all paragraphs to fade in over 500ms (half a second).

Note: show(), fadeIn() and slideUp() will not work if the targeted selector is already visible on the page. Similarly, hide(), fadeOut() and slideDown() will not work if the targeted selector is already hidden.

You can hide elements when the page loads by either using jQuery's .hide() or setting the selector's CSS display property to none, e.g. p { display: none; }

Triggering Effects

There are two primary ways to trigger effects.

Page Load

With this method, effects are triggered when the page loads. For example:

```
<script>
$(document).ready(function() {
   $('p').hide();
});
</script>
```

Binding to Events

With this method we bind events such as `click()` or `mouseenter()` to selectors. When the event is triggered an action is performed. For example:

```
<script>
$(document).ready(function() {
   $('p').click(function() {
      $('h1').fadeOut();
   });
});
</script>
```

In this example if a paragraph is clicked all the headings on the page will fade out. Let's break this down a bit:

`$('p').click()` tells the browser to listen for clicks on `<p>` tags. On its own, it won't do very much. We need to use a *function* to get it to perform an action. Functions are a staple of programming. The easiest way to understand functions is to see them as actions. When a function is run, something happens. In this case, `<h1>` tags fade out.

There are two types of functions: named functions and anonymous functions. We may deal with named functions later when it comes to your major projects but for now we'll just be dealing with anonymous functions.

animate()

The inbuilt effects are useful, but sometimes you'll want to do something a bit more complex and this is where animate() comes into play. jQuery's animate() allows us to animate one or more CSS properties.

What you can animate

You can animate any CSS property that accepts a numerical value.

How to do it

Using animate is a bit more complex. Its basic construction is as follows:

```
$('selector').animate({
   property: value,
   property: value,
   property: value
```

```
}, {
   option: value,
   option: value
});
```

Note: when using multiple properties or options, a comma is used to separate them, but there is no trailing comma for the last entry.

Ok, let's look at a working example:

```
$('p').animate({
   opacity: 0.4,
   fontSize: 20
}, {
   duration: 3000
});
```

In this example <p> tags will become semi-transparent and increase to 20px in size over a three second duration.

Note: jQuery doesn't require you to specify pixels (px) as the unit of value, just the plain number is all it requires. Additionally, any CSS properties that contain a hyphen are 'camelCased' when used with jQuery.

You can, of course, move elements using animate(). You can animate the margins or paddings of an element but this will generally affect other elements on the page by pushing them around also. For example, setting or animating a top margin on one element will push down everything below it. You can, however, use `position: relative` to get around this. A relative position element can be animated all over the page without affecting the position of any other element. The only problem with this is that the browser will leave whitespace in the place originally occupied by the relative position element. You can also animate an absolute position element.

More on Events

Earlier we looked at how to bind a click event to a selector. There are a few other events of which you should be aware, and we'll also take this opportunity to introduce you to the `.css()` feature of jQuery.

Events	Syntax
Click	`$('selector').click();`
Mouseenter	`$('selector').mouseenter();`
Mouseleave	`$('selector').mouseleave();`

The real benefit of these events is that in plain old HTML only hyperlinks react to these events. jQuery, however, means that we can make any element respond to a click, mouseenter or mouseleave. For example:

```
$('h1').mouseenter(function() {
   //perform some action
});
```

Setting CSS with jQuery

jQuery also offers us a method of setting CSS properties for any targeted selector using the `.css()` feature. For example:

```
$('p').css({
   color: '#ff0000',
   fontSize: 24
});
```

Note: Any CSS property that uses a hyphen is camelCased in jQuery. Additionally, any value that is NOT a number must be enclosed in quotation marks (either single or double).

This can, of course, be triggered from any event. We'll go into this in more detail in next week's workshop.