jQuery:
A JavaScript Utility Library

Plus: Anonymous Functions & Callbacks

Using external JavaScript libraries

- A library is simply a collection of functions and code that someone has written to make a set of tasks easier in a specific programming language
- By including the library in your code, you'll have access to its functions and methods

Exercise - understanding libraries

- Create a new file, `my_library.js`
- Inside of this file, write a function that does something and returns feedback in an alert or console message
- Include `my_library.js` in an HTML file
- Call the function in an arbitrary `<script>` block after your library has been loaded with a `<script>` tag:

```
<script>
  myFunction();
</script>
```

Common library problems

- Load the page with the JavaScript console open (Command + Option + J) to see if any errors are raised when the page and library are loaded
- Sometimes libraries conflict with each other due to similar naming conventions of functions and methods
- You may have to remove a library or edit it to get it to work with another library
- To check to see if a library is loaded, try calling one of its functions or methods. For instance, to make sure jQuery is loaded, just try calling `jQuery` in the JavaScript console

jQuery

- jQuery is a self described "write less, do more" library
- It can be marginally slower than using native JavaScript but exposes much JavaScript functionality in a very developer-friendly and cross-browser-friendly way
- The 2.x version does not support IE6, 7 or 8 if this is important to know for your project

Including the jQuery library

- Inside of the <head> section of your HTML file, include a copy of the jQuery library:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
```

- In the above case, our copy of the library is from the Google Content Delivery Network, meaning it should be loaded pretty quickly no matter where our user is located
- You could also just download the library and include it in your project directory, referencing it relatively

The Document Object Model (DOM)

- The DOM is a standard for interacting with and representing objects in HTML, XHTML, and XML
- It has been around since the "browser wars" of the 90s
- Web browsers implement a standard version of it to normalize how HTML elements are interacted with
- jQuery makes it incredibly easy to select elements in the DOM, which include practically any HTML tag or element

Selecting elements with jQuery

- jQuery can be used to select elements using a similar syntax to CSS
- Try this first line in the JavaScript console of a page that has jQuery included:

```
$("body")
```

- It should return an `Array` with selected elements:

```
> [<body>...</body>]
```

Selecting elements with jQuery

The format for selecting elements is:

```
$("<element name, class, id, or XPath>")
```

Here are some examples:

```
$("p")
$(".main-nav")
$("ul li")
```

Selecting elements: a closer look

- By calling `$`, we're just using a shorthand way of accessing the jQuery library's main `function`, that all others are descended from
- Try typing just a `$` in the JavaScript console and pressing enter to see the function itself
- The `("<element identifier")` after `$` is simply an argument being passed to the library's main function

JavaScript Events: monitoring the DOM

- An incredibly key feature of JavaScript is being able to monitor for certain events being fired by specific DOM elements
- For instance, we could monitor for a `click` event on a `<button>` element to fire a specific snippet of code

The `$(document).ready()` snippet

- This snippet ensures that your code doesn't run until your document fires a `ready` event
- The ready event fires when all of the page's DOM elements are loaded, even if multimedia elements aren't fully loaded
- By encapsulating all of our `jQuery` code to only run when the document is ready, the elements you're selecting will definitely be on the page when you access them

Anonymous functions

- are another way to write functions that makes them a "first-class citizen" data type meaning they can be:
  - given as an argument to a function (yes, a function taking a function)
  - returned by a function or block of code
  - assigned to a variable

Writing anonymous functions

```
function(){
  alert("I am anonymous!");
}
```

- Anonymous functions can be stored in variables:

```
var my_anon_func = function(){ alert("Har!"); }
```

Callbacks

- Anonymous functions are often used in **callback**s
- A **callback** is an anonymous function supplied to another function to be run when that function is done running

Callback Example

```
// Define an anonymous function called someOtherFunction
var someOtherFunction = function(){ alert('hello world'); }

// Declare a regular or anonymous function that takes another
// function as an argument, does something,
// then runs that other function
function someRunner(anyFunction){ console.log(2+2); anyFunction(); }

// Call the regular function, supplying the initial
// anonymous function as an argument
someRunner(someOtherFunction);
```

Using `$(document).ready()`

To use $(document).ready() and ensure the page is ready for your jQuery, you'll need to pass it an anonymous function (a callback) to execute when the `document` is `ready`

```
$(document).ready(
  //Code that executes when the document is ready
  function(){
    alert('The document is ready, sah!');
  }
);
```

The Fun Stuff

Try these out in your JavaScript console (Command + Option + J)

If you're using jQuery in an actual file/script, don't forget the `$(document).ready()` wrapper!

Showing and hiding

To show or hide an element on the page with jQuery, select it and call the `show()` or `hide()` function on your selection

```
$("body").hide();
```

```
$("body").show();
```

Fading in, fading out

To show or hide an element on the page using a fading animation, select it and call the `fadeIn()` or `fadeOut()` function on your selection

```
$("#section-one").fadeOut();

$("#section-one").fadeIn();
```

Fade over time

You can give the `fadeOut()` and `fadeIn()` functions a time in milliseconds as an argument over which they should fade in and out

```
$("#section-one").fadeOut(1000);
//Fade out over 1 second

$("#section-one").fadeIn(4000);
//Fade in over 4 seconds
```

Hiding with sliding

Another way of hiding and showing items on the page is by using the `slideUp()` or `slideDown()` methods

```
$("#section-one").slideUp(1000);
//Fade out over 1 second

$("#section-one").slideDown(4000);
//Fade in over 4 seconds
```

Animating CSS

Use jQuery's `animate()` method to animate most CSS properties that are numeric

```
$(".my-element").animate( {
   opacity: 0.25,
   width: 70%
} , 2000 );
```

The click event

- Remember that JavaScript events can be fired on any HTML element
- To monitor an HTML element for the click event and then fire a callback function with some code:

```
$("ul li a").click(
  function(){
    alert('imclicked!');
  }
);
```

Using a callback for a sequence

Here's an example of one animation triggering another when it's done, using an anonymous function as a callback:

```
var animateMenu = function(){
  $(".menu").animate( {opacity: .5} )
}
$(".menu").show(animateMenu)
```

Simplified syntax

The last example could also be written like this:

```
$(".menu").show( function(){
  $(".menu").animate( {opacity: .5} )
})
```

Exercise - using jQuery

- Create a basic HTML page with no styling that has at least 10 separate elements on it
- Use jQuery to create an animated "show"! Try changing an element's CSS, hiding, showing, fading, and sliding, and making the page run amok
- Once the "show" works, rig it up to only fire when the user clicks anywhere on the page
- If you finish this exercise, implement jQuery into some of the websites you built last week and try playing around with some other functions: http://api.jquery.com/