jQuery

Recap & Intro

- Previously we talked DOM manipulation and event handling (tedious)
- Today we will talk about jQuery
- Introduce Ajax

jQuery

- jQuery is a LIBRARY not a language
- jQuery is the #1 most used library on the web
- Same pattern - select something, watch for an event, do something when it happens
- jQuery "Wraps" dom elements and makes them more convenient to work with.

Selecting Elements

Just like querySelectorAll()

```
var elements = $('#myDivWithAnId .css .query');
```

You can further refine using a filter function.

```
var elements = $('#myDivWithAnId .css .query').filter(function(){
  return this.innerHTML === 'myElement';
});
```

Attaching Event Handlers

html

```
<div id="firstDiv">
    <button id="myButton" name="button">Click ME</button>
</div>
<div id="someOtherElement">
    <h3>hello world</h3>
</div>
```

```
$('#myButton').on('click', function(event){
  //Invoked when #myButton gets a click event
  this.innerHTML = "button clicked";
})
```

Alternatively, you might see this syntax: Just like `$('#myElement').on('event', handlerFunction)` ... except `this` refers to the element.

```
$('#myButton').click(function(event){
  //Invoked when #myButton gets a click event
  this.innerHTML = "button clicked";
})
```

Manipulating DOM

Just like innerHTML, appendChild etc, but more powerful.

```
$('#myButton').on('click', function(event){
  //Update button html
  $(this).html("button clicked");

  //Create a new element
  var newEl = '<a>look at me!</a>';

  //Append new element
  $('#someOtherElement').append(newEl);

  //Add an attribute
  $('#someOtherElement').append(newEl).attr({'href' :
'http://techtalentsouth.com'})

  //Remove it
  newEl.remove();

})
```

An aside on `this`

```
//in jQuery:
  $(this).html("button clicked");
// is equal to:
  this.innerHTML = 'button clicked'
```

try `console.log($(this))` inside one of your functions and inspect the result

`$(this)` will return a DOM element.
It is important to remember that everytime jQuery sees `$(this)`, it will lookup the DOM element, which can impact performance over time. To address the issue of multiple lookups, you can store `$(this)` in a variable so the object is now stored as a reference in memory, like so:

```
var self = $(this);
```

Yes, this is similar to the closure hack.

bringing it all together

```
$('#myButton').click(function(event){
    var self = $(this);

    self.html('FOO')

    $('#someOtherElement').css('background-color', 'green');

    console.log('button clicked!');

    self.css('color', 'red');
 });
```

Document.ready

Most of your jQuery scripts should be loaded on the DOM's document.ready event. Pass a function into jQuery object to run it on DOM ready.

```
  $(document).ready(function() {
    console.log('this runs second');
  })

  console.log('this runs first');
```

This makes sure the document exists before your code tries to manipulate it.

TODO: jQuery basics exercise

- Use the following html

```
<html>
    <body>
        <header>
            <ul>
                <li class="first">Item 1</li>
                <li class="selected">Item 2</li>
                <li class="last">Item 3</li>
            </ul>
        </header>
        <div class="col">
            <section>
                <h2>Section 1</h2>
            </section>
            <section class="current">
                <h2 class="highlight">Section 2</h2>
            </section>
            <section>
                <h1>Section 3</h1>
            </section>
        </div>
    </body>
</html>
```

Using jQuery:

- Load jQuery (using a CDN)
- Get the top-level header element
- Get all the section elements
- Get the section element with class="current"
- Get the section that comes after the current section
- Get the h2 node from the section before the 'current' section
- Get the div that contains the section that has an h2 with a class of 'highlight'
- Get all the sections that contain an H2 and store them in an Array

jQuery basics Answers

```
$("h1");
$("section").find("*");
var current = $(".current");
current.next():
current.prev().find("h2")[0];
$(".highlight").parent().parent();
var allH2 = Array.from($("section").find("h2"));
```

jQuery Next Steps

Updating Styles and Classes

```
//Edit an inline style
$('#myButton').css({color: 'red'});

//Better...

//Add a CSS Class
$('#myButton').addClass('newClass');

//Remove a CSS Class
$('#myButton').removeClass('newClass');
```

Chaining

All jQuery commands return a jQuery object, which means you can call multiple functions in a row. This is called "chaining"

```
$( "#content" )
    .find( "h3" )
    .eq( 2 )
        .html( "new text for the third h3!" )
        .end() // Restores the selection to all h3s in #content
    .eq( 0 )
        .html( "new text for the first h3!" );
```

DOM Traversal

You can search for elements up and down the DOM tree. There are many methods to search for parents, siblings, and children that you can read about. A few examples:

```
<div class="grandparent">
    <div class="parent">
        <div class="child">
            <span class="subchild"></span>
        </div>
    </div>
    <div class="surrogateParent1"></div>
    <div class="surrogateParent2"></div>
</div>
```

```
// Parents - Selecting all the parents of an element that match a given selector:
// returns [ div.parent ]
$( "span.subchild" ).parents( "div.parent" );

// Siblings - Find the next or previous sibling
// returns [ div.surrogateParent1 ]
$( "div.parent" ).next();

// Children, Grandchildren - Search down the tree
// returns [ div.child, div.parent, div.surrogateParent1, div.surrogateParent2 ]
$( "div.grandparent" ).find( "div" );
```

Exercise - Form Validation

- Create a Form and add validations using jQuery
- Utilize the html and css provided below
- Create a function that will fire when the **submit** `<button>` is *clicked*
- Make sure the function loaded after the `DOM`
- Inside the function:
- create variables for each of the form inputs
- use jQuery to accomplish this task
- *(think about querySelector)*
- create an `array` named *required*
- store the variables you've **name**, **email**, and **phone number**
- Utilize a `for` loop to iterate of the entire **required** `array`
- Inside the `for` loop:
- use an `if` statement to check the **value** of each array item
- **if** the array item is equal to an empty string (`""`)
- populate the `message` `<p>` with the following text:
  "Please Fill Out Required Fields"
- add the **warning** `class` to this message
- Attach the **warning** `class` to the array item's `<label>`

note: You want the label to turn red, if the field is empty

- once the field has something *other* than a blank string. make sure the **warning** `class` is removed
- close out the `for` loop
- utilize another `if` statement
- verify that no `<label>`'s have the **warning** `class`
- if true
- remove the `form` from the `DOM`
- manipulate the `<h2>` to say: *"Thanks for your feedback!"*
- close out your function
- test it out!

html

```
<div id="pre-form">
  <h2>We'd love to hear your thoughts!</h2>
</div>
<div class="form-container">
  <form id="form" name="form">

    <h3>Contact Us Here:</h3>
    <p id="message"></p>

    <label>Name:</label>
    <input id="name" placeholder="Name" type="text"><br>
    <label>Email:</label>
    <input id="email" placeholder="Email" type="email"><br>
    <label>Number:</label>
    <input id="phone" placeholder="10 digit phone number." type="tel"><br>
    <label>Message:</label>
    <textarea id="message" placeholder="Your thoughts here :)"></textarea><br>
    <input id="submit" type="button" value="Send It!">

  </form>
</div>
```

CSS

```
* {font-family: sans-serif;}

#form {line-height: 2;}

#form input {margin-left: 25px;}

#form input[id="phone"] { margin-left: 10px;}
```

```
#form textarea {
  margin-left: 2px;
  margin-top: 10px;
}

.warning {
  color: red;
}
```

Exercise Answer

```
$(document).ready(function() {

  $("#submit").on('click', function() {

    var name = $("#name");
    var email = $("#email");
    var message = $("#message");
    var phone = $("#phone");

    var required = [name, email, phone];

    for (var i = 0; i < required.length; i++) {
      if (required[i].val() === "") {
        $(message).text("Please Fill Out Required Fields").addClass(
          'warning');
        required[i].prev().addClass('warning');
      } else {
        required[i].prev().removeClass('warning');
      }
    }

    if (!$("#form label").hasClass('warning')) {
      $("#form")[0].remove();
      $("#pre-form h2").text("Thanks for your feedback!")
    }

  });
});
```

Animations

jQuery has convenience methods for basic animation. Most of the time, you should do animations using CSS, but jQuery animations are used quite often.

html

```
<html>
  <head>
    <link id="" rel="stylesheet" href="main.css" media="screen" title="no title"
charset="utf-8">
  </head>
  <body>
      <div id="box"></div>
      <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
      <script src="main.js" charset="utf-8"></script>
  </body>
</html>
```

css

```
#box {
  background-color: aqua;
  width: 300px;
  height: 200px;
  margin: 200px auto;
}
```

```
//Animate fade out
$('#box').fadeOut();

//Animate fadeIn
$('#box').fadeIn();

//Slide and remove element
$('#box').slideUp('slow');

// Custom effects with .animate()
$( "#box" ).animate(

  //properties to animate
  {opacity: 0.25},

    // Duration
    300,
```

```
    // Callback to invoke when the animation is finished
    function() {
        console.log( "done!" );
    }
);
```

jQuery Plugins

jQuery lets you extend it
The road to hell is paved with jQuery plugins...

```
$.fn.greenify = function() {
    this.css( "color", "green" );
};

$( "a" ).greenify();
```

You can download and use many jQuery plugins. Problems:

- You don't know what they're actually doing.
- They don't know what each other are doing

jQueryUI

Is a library of plugins that let you create UI widgets

- Datepicker
- autocomplete
- menu
- progress bar
- etc.

You can install jQueryUI manually into your project by downloading it directly from the jQueryUI website
or
you can utilize a CDN (recommended)

Caveats

jQuery lets you do a lot with this DOM manipulation pattern, but it is not maintainable. It's fast, easy, and a one way road to hell.

- Only use jQuery for quick and dirty demos.
- Otherwise, use native DOM API as much as you can.
- If what you're doing is too hard with native DOM, bring in a real framework like Backbone. Don't use jQuery if you want it to be maintainable.

Lab - jQuery ToDo App

- Re-create your ToDo App with jQuery
- compare to your previous Vanilla JS ToDo App
- Add comments explaning what each jQuery method is accomplishing
- Push to our class GitHub using the naming convention: `jQuery_ToDo_YOUR_INITIALS_HERE`