INTRODUCTION TO JAVASCRIPT

DAY 2: JAVASCRIPT DATA, AJAX, AND APIS

QUESTIONS FROM YESTERDAY?

TODAY'S AGENDA

- Pimp Your Editor
- Review Objects
- Learn Debugging Techniques
- Manipulate the DOM
- Learn about callback functions and asynchronous JavaScript
- Learn what APIs are and how to use them with JavaScript
- Learn what AJAX is and how to use it to interact with APIs
- Learn different methods for authenticating to an API with JavaScript

NEW TOOLS

- Sublime Packages
- Postman
- JSONView (Chrome/Firefox Extension)

TWEAKING SUBLIME

- Package Control - https://packagecontrol.io/installation
- Tour of Command Pallete/Finder
- snippets
- Sidebar Enhancements
- BracketHighlighter
- View in Browser (edit settings)
- jQuery

OBJECTS VS ARRAYS QUIZ

- If you want to store a number of data elements in order, which do you use?
- True/False: All elements in an array must be the same type.
- What data type are all object keys?
- True/False: You must define all of an object's properties when you create it.
- If you had an object that represented a director, how would you store that director's movies?
- How would you store a bunch of director objects?

ERRORS & DEBUGGING

- Setting breakpoints
- `debugger` statement
- Watching variables
- Logging (`console.log(); console.warn(); console.error();`)

THROWING AND CATCHING EXCEPTIONS

```
try {
  //something that might explode
  throw new Error('welp');
}
catch(e) {
  //handle the error
}
finally {
  //optionally clean up
}
```

LINTING

- Linting is the practice of evaluating code for syntax, usage, and even style. There are linters for almost every language.
- You can use http://jshint.com/ online, or use a linter within your editor.
- Linting is great for preventing the most common bugs in interpreted languages - syntax issues

JAVASCRIPT ON THE WEB

JavaScript is still primarily used to manipulate web pages. It's often thought of as a "front-end" or "client-side" technology, though, with Node, it's a legitimate server-side language as well.

HTML/CSS REVIEW

```
<html>
<head>
  <title>A VERY simple page</title>
  <style>
    .someStyle {
      background-color: blue;
    }
  </style>
</head>
<body>
  <div id="uniqueId">I'm the Inner Text of Div uniqueid</div>
  <div class="someStyle">This is all Inner HTML of Body
    <a href="#">Link That Goes Nowhere!</a></div>
  <!-- "class" and "id" are ATTRIBUTES of their elements -->
  <!-- when possible load scripts at end of body
  so they don't block page rendering -->
  <script src="class.js"></script>
</body>
</html>
```

THE DOM

- DOM - Document Object Model
- Browser exposes *objects* that represent the HTML structure of the page.
- Document -> Node -> Properties
- Common Node Properties: parentNode, childNodes, prevSibling, nextSibling
- In Chrome/Firefox - Open Page -> Right Click -> Inspect Element

NAVIGATING THE DOM

```
console.log(document.body);
console.log(document.body.parentNode);
console.log(document);

for (var i = 0; i<document.body.childNodes.length; i++
  console.log(document.body.childNodes[i]);
}
```

SELECTORS

```html
<div id="findMe">Hi</div>
<div class="nav">Div 1</div>
<div class="nav">Div 2</div>
```

```javascript
var div = document.getElementById('findMe');
var divs = document.getElementsByTagName('div');
var navs = document.getElementsByClassName('nav');

for(let i=0;i<navs.length;i++){
  console.log(navs[i]);
}
```

EVENTS

- Most DOM elements have ways to let us know when a user has taken an action. These are called *events.*
- Common events are: `click`, `dblclick`, `mouseover`, `mouseout`, `focus`, and `blur`
- A function that responds to an event is called an *event listener (or handler)*.
- See MDN for more.

EVENT LISTENERS

```
<div id="findMe" onmouseover="changeDiv()">Hi</div>
```

```
//class.js
function changeDiv() {
  var el = document.getElementById('findMe');
  el.innerText = "Moused Over";
}
```

```
<div id="findMe" onmouseover="alert('hi');">Hi</div>
//the value on the event is legit JS code but generally don't.
```

UNOBTRUSIVE EVENT LISTENERS

```
<div class="nav" click="changeNav();">Div 1</div>
<div class="nav" click="changeNav();">Div 2</div>
```

```
function changeNav() {
  var idk = document.....????
}
```

```
<div class="nav">Div 1</div>
<div class="nav">Div 2</div>
```

```
//put code where it belongs
var navs = document.getElementsByClassName('nav');
for(var i=0; i<navs.length;i++) {
  navs[i].addEventListener('click', function() {
    alert(this.innerText);
  })
}
```

JQUERY

- Library providing functions that simplify working with the DOM and AJAX.
- "Write less, do more"
- The most popular JavaScript library (20 million sites)
- Probably the closest we ever got to full cross-browser standardization.

INCLUDING JQUERY

You can download jQuery locally and include it, which is recommended for real projects. For this, you can get a link to a CDN-hosted jQuery and use that. Go to https://code.jquery.com

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"
  integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt
  crossorigin="anonymous"></script>
// don't forget to include jQuery ABOVE any other
// code that needs it!
```

EASIER DOM MANIPULATION

```
<div id="somediv"></div>
<div class="nav"></div>
```

```
let oldAndBusted = document.getElementById("somediv");
let newHotness = $('#somediv'); // # = id
let navDivs = $('.nav'); // . = class
```

```
var div = $('#somediv');
let innerHTML = div.html();
div.html('<p>HTML YAY</p>');
// var p = document.createElement('p');
// document.body.appendChild(p);
$('body').append('<p>');
```

```
div.css('width'); //read
div.css('width', '200px'); //write
div.attr('id'); //read
// how to write id?
```

DOCUMENT READY

So far, we've just been relying on our simple DOM to be loaded fast enough, but more complicated pages take time to render. We need a way to ensure the page has finished loading before we try to do something, like, attach event listeners.

```
$(document).ready(function() {
  //put any page init code here
  // like event handlers
});

$(function() {
  //shorthand version, same thing
})
```

EVENTS IN JQUERY

```html
<a href="#" id="link" class="clickable">Click Me</a>
<a href="#" id="link2" class="clickable">Click Me 2</a>
```

```javascript
$(function() {
  $('#link').on('click', function() {
    // $(this) = jQuery element that triggered event
    alert($(this).text());
  })
});
```

```javascript
$('.clickable').on('click', function() {
  alert($(this).text()); //no more loop
});
```

FORMS

HTML forms allow us to capture structured data

```
<form id="about">
  <input type="text" id="name" placeholder="Enter Your Name"/>
  <label>Favorite Dinosaur</label>
  <select id="dinosaur">
    <option value="velociraptor">Velociraptor</option>
    <option value="somewronganswer">Some Wrong Answer</option>
  </select>
  <input type="text" id="dinoName"/>
  <input type="submit" value="Go!"/>
</form>
```

```
$(function(){
  $('#about').submit(function(event)
    var name = $('#name').val(); //get value from form
    var dino = $('#dinosaur').val();
    $('#dinoName').val(name = " the " + dino); // set val
    return false; // prevent form from trying to "submit"
  });
})
```

DOM LAB

We're going to create a library inventory system!

- Create a form to add books to our library. It should have fields for book title and author at minumum.
- On submit, add to the "library" - append to a list of books and titles on the page.

REVIEW QUIZ

- What data type would you use to hold a list of items in order?
- What is the literal constructor for an object?
- Which loop always executes at least once?
- Which loop would you use to act on every member of a collection?
- What library provides common, cross-browser functions for manipulating the DOM?

QUIZ, CONTINUED

- When using jQuery selectors, how do you specify that you are looking for an id? A css class?
- How would you turn a comma-separated string into an array?
- What function do you use to wait for the DOM to fully load before executing?
- What is type coercion?
- Why do we use triple-equals?

INTRO TO APIS

An application programming interface (API) is a façade providing simplified access to more complex code.

We've already seen a bunch of APIs in action.

- The `console` object is an API wrapping complicated browser functions.
- The DOM libraries in JavaScript and jQuery.
- Technically, any set of functions you make that abstract logic from the caller constitutes an API.

WHY BUILD APIS?

- Separate Concerns for architecture and scale reasons (front-end/back-end, system boundaries, etc).
- Reuse across clients (iOS, Android, Web -> same back end)
- Abstract complex logic/data, and control access.

WEB APIS

Many web appliations provide APIs to subsets of their data and functionality, allowing developers the freedom to create new applications by combining available data. For some, it's their whole business model.

- Social APIs: Twitter, Facebook, Foursquare, Meetup, YouTube, Spotify.
- Utility APIs: Github, Google Maps, Twilio, Stripe, Mailchimp, Slack, Spark, Outlook.
- Authentication APIs: Google, Facebook, Twitter, Amazon, Tumblr, a forever amount more.

SOME INTERESTING APIS

- Twinword Sentiment Analysis: https://www.twinword.com/api/sentiment-analysis.php
- Star Wars API: http://swapi.co/api/
- Meme API: http://apimeme.com/meme?meme=Chemistry+Cat&top=Eat+bananas+for+potassium%3F&bottom=K

HTTP VERBS

- GET - retrieve resource data
- POST - create new resource
- PUT - Update/Replace resource
- PATCH - Update/Modify resource
- DELETE - Delete resource
- GET/POST are what you will deal with most often when consuming APIs

REST

- Representational State Transfer
- Using a predictable resource-focused URI to communicate with the server.
- `https://api.site.com/resource/action/filter`
- Resource: noun that represents a system object (e.g. user)
- Action: what to do to the resource (retrieve/create/update)
- Filter: any sort of query/filter on the data (e.g. id)
- Most API endpoints are a combination of a VERB + URI. GET /users = retrieve all users. POST /users = create new user (why /users?)

READING API DOCS

- https://developer.github.com/v3/
- https://developer.spotify.com/web-api/
- https://developer.foursquare.com/docs/

JSON

- JavaScript Object Notation
- Say it like it's a human name not a robot name (Jason, not Jay-Son)
- Plain Ol' Javascript Objects, mostly. Data only, no behavior.
- Uses JavaScript syntax, but is not necessarily "JavaScript". It's a *serialized* string.
- Modern format for API data/communication. Has *largely* replaced XML as the standard.

JSON STRUCTURE

```
{
  "star": "Sun",
  "planets": [
    {
      "name": "Mercury",
      "position": 1,
      "age": 4503000000
    },
    {
      "name": "Earth",
      "position": 3,
      "age": 4543000000,
      "satellites": [
        {
          "name": "Moon"
        }
      ]
    }
  ]
```

READING JSON

```
//from string
var planet = '{"name":"Jupiter","rings":true,"moons":["Europa","Ganymede","Io","Callisto"]}';
planet.name; // nope
var j = JSON.parse(planet);
j.name; // "Jupiter"
j.moons;
```

```
var nested = '{"name": "Tom", "kids":[{"name": "Sally"},{"name": "Tina"}]'
var tom = JSON.parse(nested);
tom.kids[0].name;
tom["kids"][0]["name"];
tom.kids.length;
```

WRITING JSON

```
var obj = {name: "Jupiter", moons: ["Europa", "Io"]};
var json = JSON.stringify(obj);
console.log(json);
console.log(obj);

obj.sayMoons = function() {
  for(let i=0;i<this.moons.length;i++) {
    console.log(this.moons[i]);
  }
}
obj.sayMoons();
json = JSON.stringify(obj);
console.log(json); //no methods
```

PARSING XML

You may find yourself needing to interact with XML from time to time. It's less fun than JSON, but XML has a DOM just like an HTML web page (in fact, HTML is a specialized XML).

```
var text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";

var parser = new DOMParser();
var xmlDoc = parser.parseFromString(text,"text/xml");
console.log(xmlDoc.getElementsByTagName('title')[0].innerHTML)
```

JSON PARSING LAB

Use the Star Wars API (http://swapi.co/api/) to practice parsing JSON. We will build on these functions in a future exercise! Each function should take a string argument for the JSON input.

- Create a function to parse the JSON from the Films endpoint (http://swapi.co/api/films/) for a single film and log the values for the title, opening crawl, characters, and planets to the console.
- Create a function to parse the JSON from the People endpoint for a specific character and output the name to the console.
- Create a function to parse the JSON from the Planets endpoint for a given planet, outputting the name and terrain.

AJAX

- Asynchronous JavaScript and XML.
- But not really XML that much anymore.
- Communicate to server/API without blocking the client.
- Allows interactivity and client updates without full refreshes/stopping user action.
- Consider a fully synchronous Twitter. Nightmare!!

JQUERY AJAX

jQuery provides very easy ways to do AJAX with a common interface.

```
var uri = 'https://api.github.com/users/octocat/repos'
$.get(uri, function(data) {
  //what kind of function are we in?
  console.log(data);
})
//what if there's an error? or a lot of code?
//what about Separation of Concerns/Single Responsibility Princ
```

JQUERY AJAX HANDLERS

```
var uri = 'https://api.github.com/users/octocat/repos'
$.get(uri)
  .done(function(data) {
    console.log(data);
    console.log("success!");
  })
  .fail(function(err) {
    console.log(err);
    console.log('oops!');
  })
```

EVEN MORE JQUERY AJAX

jQuery.get() is a *wrapper* for jQuery.ajax(), simplifying the call. But if you need more control, you can call `$.ajax()` directly.

```
$.ajax({
  method: "POST",
  url: "some.php",
  dataType: "xml", // how the "data" param knows what it is
  headers: {"Custom-Header": 'header-value'},
  data: { name: "John", location: "Boston" }
})
  .done(function( msg ) {
    alert( "Data Saved: " + msg );
  });
```

JQUERY AJAX EXERCISE

Set up an HTML page that uses jQuery to interact with the Github API

- List all public repositories for user octocat in an HTML list element on the page.
- List the first five commits for each repository within a sub-list.

DATA ATTRIBUTES

When building dynamic HTML elements, we sometimes need to store useful bits of data so that we can retrieve them with JavaScript. For instance, in our last exercise, we might want easy access to the user id of a repository owner that we can access without parsing the text of the element.

```
//not super easy to access the data
<li>User Name: octocat</li>

//not semantically accurate and fails if more than one
//repo from octocat
<li id="octocat">User Name: octocat</li>

//here we go
<li data-user="octocat">User Name: octocat</li>
```

USING DATA ATTRIBUTES

```
<li class="user" data-user="octocat">User Name: octocat</li>

//what is this function?
$(document).ready(function() {
  $('.user').on('click', function() {
    var id = $(this).data("user");
    $.get('https://github/repos/' + id, function(data) {
      //do something
    })
  })
})

//plain javascript
el = document.getElementsByClassName('user')[0]
el.dataset.user
```

API LAB

Go back to the Star Wars API from earlier and let's put this all together.

- Create an HTML page. Add a link to get the list of films. Use jQuery AJAX and display just the list of film titles as links. Think about data attributes!
- When you click the link for a film, retrieve the film's planet and character data and display it on the page.

WRITING DATA

So far we've just been performing GETs to read data from an API, but we often need to write data as well. Usually, this is done with a POST.

```
$.post( "test.php", { name: "John", time: "2pm" })
  .done(function( data ) {
    alert( "Data Loaded: " + data );
  });
  //easy enough but there's other concerns
```

AUTHORIZATION/AUTHENTICATION

Most APIs will require some form of authentication to use write (and often, read) API endpoints. Common auth methods are:

- HTTP Auth with Username/Password (less common, not great)
- User-based API access token (OAuth/other)
- Application-based API id/key pair
- Combination!

SETTING UP AN API ID/KEY PAIR

An API key/id pair is just a way to identify an app and is not tied to your user login.

- Go to https://foursquare.com/developers/apps (sign in or register).
- Create new app. Use http://localhost for the APP URL.
- Copy and save CLIENT_ID and CLIENT_SECRET somewhere so we can use it.
- Make sure you know which is the ID and which is the secret!

SETTING UP AN OAUTH TOKEN

An OAuth token is tied to your user and is essentially a login.

- Log in to (or sign up for) Github.
- Go to https://github.com/settings/tokens and Generate New Token. Name it "Javascript Class" and check "repo" scope.
- Copy and past the token somewhere, because you won't see it again (this is a common security practice).

AUTHENTICATING TO APIS

Let's check out authentication with an API call using the Foursquare API. Go to https://developer.foursquare.com/docs/ and let's try the venues endpoint.

Now let's try Github

EXPLORING APIS WITH POSTMAN

Postman is an app that makes interacting with APIs easy. https://www.getpostman.com/

AJAX WITH AUTH

GET

```
let your_token = "YOUR TOKEN";
$.ajax( {
  url: 'https://api.github.com/user/repos',
  headers: {
    "authorization": `token ${your_token}`
  }
})
.done(function(data) {
  console.log(data);
})
```

AJAX WITH AUTH

POST

```
let your_token = "YOUR TOKEN";
$.ajax( {
  url: 'https://api.github.com/user/repos',
  method: 'POST',
  headers: {
    "authorization": `token ${your_token}`
  },
  data: '{"name": "scr-gh-post-test"}'
})
.done(function(data) {
  console.log(data);
})
```

EASY CODE SNIPPETS WITH POSTMAN

You can use Postman to generate your API access code for you!

- Set up an API call like normal
- Click "Code" and select language

AJAX LAB

- Pick an API that interests you. Explore it. Create an HTML page that utilizes its functions. Use jQuery or fetch()
- Bonus: Use data from two or more APIs to create a mashup!

Some APIs of interest: Twitter, Facebook, Foursquare, Darksky, IBM Watson, Github, Google, Meetup, Twilio, Spotify, News API, Battle.net, and a billion more.