PROGRAMMING FOR NON-PROGRAMMERS BOOTCAMP — DAY 2

Warmup:

Deployment and git.

45 minutes

Let's learn JavaScript and jQuery!

A LITTLE HISTORY LESSON

---

15 minutes

The web was originally created by Tim Berners Lee while he was working with CERN in Switzerland. It was mainly intended to make it easier for academics to share research and link between them.

Websites started out like that. A piece of paper with links. Once the page loaded, that was it!

Brendan Eich, an engineer at Netscape, was tasked with creating the first fully functional programming language that can be executed in browsers. He did this over the course of 10 days, and some of the 'gotchas' of the language have to do with that. The rest involves a Game of Thrones-like power struggle amongst Silicon Valley giants who didn't do a great job of cooperating. The result was chaos and incompaitble implementations for years. This makes it a bit more confusing to learn, but this is just how things are right now.

Between 2009 and 2015, we had one specification — ECMAScript 5 — which is still implemented by modern browsers, but now we have ES6, which is WAAAY nicer and easier to work with. Most of that is implemented already by modern broswers.

By the way, when you hear "modern browser" just think "Not Internet Explorer or something old as hell"

We will be learning the Es6 syntax where applicable today.

As I mentioned yesterday, JavaScript is going through a cambrian explosion of tools and frameworks and it's going to move faster and faster.

There is an endless and endlessly expanding universe of frameworks for JavaScript that make it more powerful and easier to write. For example, we will experiment with jQuery the most popular framework for front-end JS.

What can I do with JavaScript?

- Breathe life into web pages, let users control stuff on the page
- Animation
- Fetch data from servers asynchronously
- Everything

JQUERY SLICK LAB

60 minutes

What does jQuery do?

jQuery makes it MUCH easier to manipulate the stuff on the page. It's practically as easy as CSS!

Now to be clear, you don't *need* jQuery to do everything that jQuery helps you do. It just makes your life a lot easier.

Where do I put my JavaScript files?

```
NOTE the <script> tag

This is a new tag we have never seen before;
remember that the <link> tag is for CSS files and
the <script> tag is for javascript files (for now).
the src attribute is what we use to link to the
external js file
remember to CLOSE your script tag, unlike the <link>
tag, <script> is NOT self closing!
```

Check out the HTML in the slick demo folder.

Explain Semantic UI and walk through the HTML.

So this is going to be a slideshow.

How do I get jQuery?

Point them to the CDN. Which is....?

place CDN link in the bottom to get a hold of jquery and demonstrate in the console.... which is???

So we want to make a slideshow. That could take all day! Fortunately there are like a million jQuery plugins for everything under the sun. We just have to find the tools, read the docs, and figure out how to use them for our purposes.

This bears repeating. Look for answers and open source tools, read the manual.

Go to Slick `https://kenwheeler.github.io/slick/`

throw this after semantic ui css

```
<link rel="stylesheet" type="text/css"
href="https://cdn.jsdelivr.net/jquery.slick/1.6.0/sl
ick.css"/>
```

and after jquery at the bottom

```html
<script type="text/javascript"
src="//cdn.jsdelivr.net/jquery.slick/1.6.0/slick.min
.js"></script>
```

```javascript
1  $('#slideshow').slick();
2
3  //make images 100% width to solve one problem
4  //then solve the height problem by looking at the
   docs.
5
6  $('#slideshow').slick({
7    adaptiveHeight: true
8  });
```

Wow!

 FOR LUNCH

WRITING OUR OWN

JAVASCRIPT

Let's open up Atom and install the `script` package that will let us

REVIEW OF DECLARATIONS, EXPRESSIONS AND STATEMENTS.

Declaring variables in JavaScript

3 ways:

```javascript
1  // hi! I am a comment
2
3  const pi = 3.141592654;
4  // the = sign means assignment, not equality.
5  // You can get away with no semicolon, but
   Santa Clause is watching.
6  // const means the variable can never be
   reassigned.
7
8  let x = 42;
9  // this is the new way of doing it.
10
11 var p = 55;
12 // you will see tons of tutorials still using
   var instead of let. I won't get into the
   difference.
13
14 //you can declare something without setting it.
15
16 let n;
17 // not defined vs undefined
```

```
18
19  //Concatenation and addition (polymorphism)
20
21  let x = 5;
22  let y = 3;
23  let answer = x + y; // => 8
24  alert(answer);
25  let x = "five";
26  let y = "three";
27  alert(answer);
28
29  //strings and numbers
30
31  typeof 'imastring';
32  typeof 4;
33
34
```

You do:

https://codepen.io/trivett/pen/yVvPaq

`/js-practice/declarations.js`

Try it in atom, then try it in the console of your browser (doesn't matter which site you are on)

Types of data

A variable type is a way to classify the different kinds of data we can save to a variable. There are exactly 6 types of variables:

Primitives

- `undefined`
- `null`
- boolean
- number
- string

Non Primitive

- Object

Primitives

A Primitive type is a most basic bit of information that you can store. For example, a number is a primitive because it cannot be made up of any of the other types of variables

Alternate definition: Think of this as an atom -- atoms are atoms because we cannot break them down into any more basic bits, same goes for primitives

`undefined`

Undefined is the default state of any variable. Basically means the variable is empty or has not yet been assigned a value, primitive or otherwise

`null`

The null variable is different from the `undefined` type, but only subtly so.

1. the `null` type is assigned to a variable, but its "value" is empty.
2. the `undefined` type is by default the value of each variable that is declared but not defined

NUMBERS

```
1  let myNumber = 1;
2  let pi = 3.14159; // ...approximately
3  // all the rules of math apply
4  // show modular division in action
5  // ++
6  // --
```

YOUR TURN

Let's write the code to actually implement the Farenheit to Celsius conversion. We can do that already with our JavaScript knowledge! Write the code to convert 212 degrees F into Celsius. Then try it with some more inputs of your own. Remember to `console.log` the result!

```
1  let farenheit = 100;
2  let celsius = (farenheit - 32) * 1.8;
3  console.log(celsius)
```

STRINGS

Escape `'` with `\'`

If you start with `"` you have to end with it.

Concatenation

BOOLEANS

True or false. Basically.

Booleans are super handy for control flow, which we will get to in a bit.

```
1  let myBooleanValue = true; // true
2  let myBooleanValueThatIsFalse = false; // false
3  console.log( typeof myBooleanValue );
```

FUNCTIONS

Imagine having to write the same stuff or edit code every time you need to change inputs.

Functions (just like proper math functions) let us reuse code.

In football, when the quarterback is shouting out orders, he doesn't say "Hey, I'm going to throw the ball in that direction. You run up the middle and do a buttonhook to catch it, oh and you, please act like you are going to catch it off to the left. Thank you!"

Nope. The QB has to use some kind of shorthand for a series of instructions that the players already know. Perhaps the QB might say `OMAHA, OMAHA` and the other players know that the running back will then execute the Omaha move (running up the middle.)

These are functions. Blocks of code that you can reuse. You define them, then you call them, or execute them.

```
1  function greet(){
2    //this function doesn't take any input, or
   arguments as we would put it
3    console.log("why hello!")
4  }
5
6  //this is the same thing
7
```

```javascript
 8  let greet = function(){
 9    console.log("why hello!")
10  }
11
12  //nothing happened.
13
14  //Right! gotta call the function
15
16  greet()
17
18  //Let's give it an argument
19
20  function greet(name){
21    let response = "Why hello, " + name +"."
22    console.log(response);
23  }
24
25
26
27  //return value vs console.log rigth now these
    functions are returning undefined
28
29  //you go and bake your cake, then you clean the
    kitchen and make a new cake.
30
31
32  //in the function, its like a separate
    environment. this is called scope.
33
34
```

YOU DO

- concatenate strings
- area of a circle
- hypoteneuse
- create a function for temperature conversion that lets you convert any temperature.

```
1
2  function myConcatenate( firstStr, secondStr,
   thirdStr ) {
3      let answer = firstStr + " " + secondStr + " "
   + thirdStr;
4      console.log(answer);
5  }
6
7  myConcatenate('I', 'am', 'iron man'); // 'I am
   iron man'
```

COLLECTIONS

ARRAYS

An array is an ordered list of stuff. That's it.

Some Array methods:

1. push
2. pop
3. shift
4. unshift
5. forEach

```
1  a = ["a", "b", "c"];
2
3  a.forEach(function(element) {
4      console.log(element);
5  });
```

OBJECTS

```
1  // just primitives
2  const name = "Vincent;
3  let hasHadAllHisShots = true;
4  let likesFootball = false;
5  let age = 32;
6  let friends = ["Kejal", "Ben", "Phillip",
   "Paola"]
7  let introduceSelf = function(){
8    console.log("Hi. I am " + name +".")
9  }
10
11
12 //imagine doing this for more than one person!
```

```javascript
13
14  // objects are basically key-value pairs
15  // the values are primitives.
16  // Objects are the bedrock of most web APIs
    that deliver data in the form of JSON (Java
    Script Object Notation)
17  // super important to get used to!
18  let vincent = {
19      name: "Vincent",
20      hasHadAllHisShots: true,
21      likesFootball: false,
22      age: 32,
23      friends: ["Kejal", "Ben", "Phillip",
    "Paola"]
24      introduceSelf: function(){
25          console.log("Hi. I am " + name +".")
26      }
27  }
28
29  //use dot notation to access these values
30
31  console.log(vincent.name);
32  console.log(vincent.friends[0])
```

CONDITIONALS AND

OPERATORS

JavaScript Statements use a set of operators

- `===`
- `!==`
- `>`
- `<`
- `>=`
- `<=`
- `&&`
- `||`
- `!`

FOR AND WHILE LOOPS

There is a handy way to loop through arbitrary data.

for loops

```
1  for (var i = 0; i < 9; i++) {
2      console.log(i);
3      // more statements
4  }
```

while loops

```
1  var n = 0;
2  var x = 0;
3
4  while (n < 3) {
5    n++;
6    x += n;
7  }
8  console.log(x);
```

Avoid creating infinite loops!

Check for understanding

Fizzbuzz exercise

```
hoping that it is about 3:30 p.m. at this point . if
its only like 2 then do the madlibs thing
```

WEB APIS

What is an API exactly?

Application Programming Interface. It's a set of functions and objects etc that work over HTTP to let other programmers use other systems .

When people talk about APIs they mainly mean web apis, but there are tons. Like for example, a menu is like an API for users to send instructions to a kitchen (the back end!) and receive food.

Tons of web apps use third-party services like Twitter, Google Maps etc.

First step to work with an API is to look at the docs. RTFM!

GRAND FINALE, CODE ALONG OF MOVIE SEARCH ENGINE

First, have the students read over the code that is already written. Without running it in your browser, what do you think some of these things do?

Explore the OMDB api. Look at the options it gives you and what kind of result you can expect. Explain what JSON is.

Work with a partner and explore the jQuery documentation. what does `.show()` `.hide()` and `.on("submit", function …. )` mean in jQuery? What do you think each function that I wrote means? Look at the CSS file included. What is that doing? Why?

Walk through the anatomy of the entire file.