Module 3-6

Intro to JavaScript

Some Quick Facts

JavaScript is an interpreted scripting language that runs on internet browsers.

- It is not really related to Java.
  - Though it does share similar language syntax.
- Came into being in the mid-1990’s.
- In recent times, JavaScript libraries and frameworks have greatly extended the language’s capabilities.

The Three Pillars of The World Wide Web

- **HTML**: The content being presented.
- **CSS**: How that content is formatted.
- **JavaScript**: Any actions or behaviors the content can provide.

JavaScript: Where does It Go?

JavaScript can be incorporated directly into a HTML page:

```
<html>
<head>
     <script>
          window.alert('Hello World.');
     </script>
</head>
<body>Helpful Content.</body>
</html>
```

A block of JavaScript code is enclosed in a set of <script> tags.

JavaScript: Where does It Go? (Preferred Method)

It is recommended that JavaScript logic be placed in a separate file and “included” in the HTML file.

```
<html>
<head>
<script src="thisScript.js"></script>
</head>
<body>Helpful Content.</body>
</html>
```

```
window.alert('Hello World.')
```

This is preferred over the first method we discussed.

Loosely Typed

- In terms of data types, JavaScript is loosely typed, meaning we do not explicitly tell JavaScript what data type a variable will hold.
- These are the data types a variable can take on: **String**, **Number**, **Boolean**, **Arrays**, and **Objects**.

Declaring Variables

- Declaring variables in JavaScript takes on the following form:

  let <<variable name>> = <<initial value>>;

```
let myStrVariable = 'hammer';
let myNumVariable = 3;
let myOtherNumVariable = 3.14
let myBoolean = true;
```

- In older texts you will see variables declared using var, i.e. var myBoolean = true. This should be avoided at all costs, **always use let**.
- Values that do not change are declared using **const**.

typeof

- We can use typeof to ascertain the data type of a variable.

```
let myStrVariable = 'hammer';
console.log(typeof myStrVariable); // string
let myNumVariable = 3;
console.log(typeof myNumVariable); // number
let myOtherNumVariable = 3.14
console.log(typeof myNumVariable); // number
let myBoolean = true;
console.log(typeof myBoolean); // boolean
```

Declaring An Array

Here are a few examples of array declarations:

```
//Declaring an array with three strings:
let myArray = ['Fiat Chrysler', 'Ford', 'GM'];

//An empty array:
let myEmptyArray = [];
```

Iterating Through an Array

Our loop friends are back:

```
let myArray = ['Fiat Chrysler', 'Ford', 'GM'];
for (i=0; i < myArray.length; i++) {

   console.log(myArray[i]);
}
// Prints out Fiat Chrysler Ford GM
```

- Note that the for-loop is structurally similar to its Java counterpart.

- We access individual elements of an array in a similar way: myArray[0] for the first, myArray[1] for the second, etc.

- We can access the length of an array with the .length property.

Let’s declare some variables.

Conditional Statements and Comparisons

These should also look familiar:

```
let x = -3;
let positive = (x > 0);
console.log(positive);
// Prints false

if (x <0) {
   console.log(x + ' is a negative number.');
}
// Prints -3 is a negative number
```

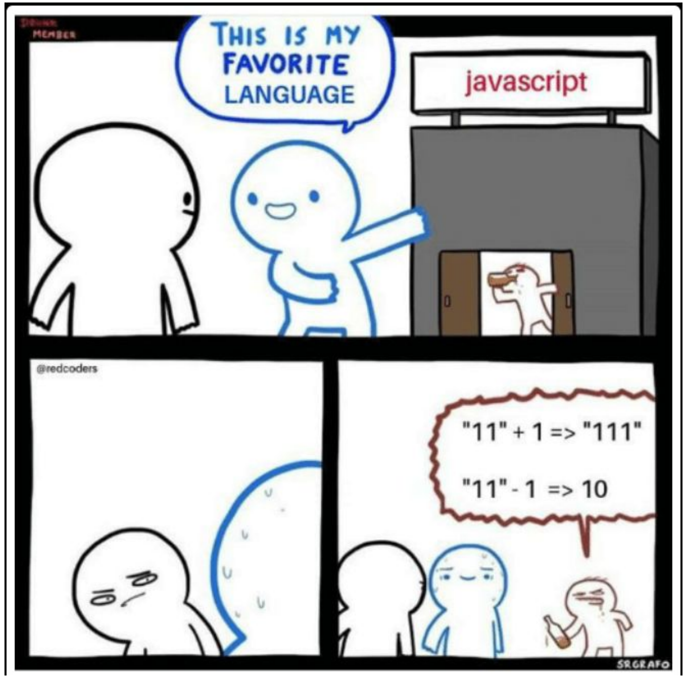
Conditional Statements and Comparisons

We can also apply AND / OR / XOR statements:

```
let x = -3;
let y = -4
let positive = (x > 0);

if (x <0 && y <0) {
   console.log('Both numbers are
negative.');
}
else if ( x < 0 ^ y < 0) {
   console.log('Only one is negative.');
}
else {
   console.log('Both are positive');
}
```

Truthy and Falsy (1/3)



If you are coming from a strictly typed language like Java, there are some unusual things to consider with regards to data types, one of these is the idea of “truthy” and “falsy.”

Truthy and Falsy (2/3)

- These rules can sometimes strike one as bizarre, but we can derive an intuitive understanding of what’s going on. Here is a good site with a more in-depth explanation: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality_comparisons_and_sameness
- For now, consider the following code:

```
let i ='1';
let j = 1
console.log(i == j); // true
console.log(i === j); // false
```

- The triple equals is to evaluate “strict equality” - meaning that not only do the values have to be the same, but the types must equal as well.

Truthy and Falsy (3/3)

All values, regardless of data-type have an inherent true or false value:

- **false**, **0**, **''**, **null**, **undefined**, and **NaN** are inherently false
- **Everything else** is inherently true

Let’s experiment with truthy and falsy

Objects

- JavaScript is not generally considered an object oriented language, it is instead a functional language (one that is based on functions).
  - Over time though, some OO features have been added to the language.

- JavaScript objects follow JSON notation, with the object itself surrounded by curly braces, and the object properties listed in comma delimited key-value pairs:

  ```
  { prop1: <<prop1Value>>, prop2: <<prop2Value>>}
  ```

Objects Example

Let’s look at a concrete example:

```
let crewMember = {
firstName: 'James',
lastName: 'Kirk',
rank: 'Captain'
};

console.log(crewMember.firstName);
console.log(crewMember.lastName);
console.log(crewMember.rank);
crewMember.rank = 'Admiral';
console.log(crewMember.rank);
```

Let’s practice creating some objects (does the format look familiar? It should, think JSON.)