Javascript

What is Javascript

- JavaScript is interpreted
- JavaScript is asynchronous
- JavaScript is a Prototype-based scripting language with dynamic typing and first-class functions
- JavaScript was formalized in the EMCAScript language which was developed by Netscape
    - ActionScript (commonly used in Adobe Flash) is also an ancestor of EMCAScript and thus shares a lot of simlarities with JavaScript
- JavaScript is very liberal in what it allows a programmer to do. This is both good and bad
    - **Bad**: It makes finding errors in your code harder because the language does not point them out to you as well as a stricter language
    - **Good**: It gives you room to try some techniques that are not possible in a stricter language

Why the name JavaScript

- Despite the name, JavaScript has very little to do with Java
- The name, JavaScript, was inspired by marketing considerations because at the time of its inception in 1995, Java was rapidly gaining popularity.
- Someone at netscape thought it would be a good idea to ride the Java train despite the small similarities between the two languages

Introduction to the JavaScript Language

- Numbers
    - Numbers in JS are 64 bits
        - 1 bit is used to hold the sign
        - 11 bits are used to store the fractional part
        - This leaves 52 bits to hold the whole number
        - This means any number < 2^52 can safely be stored in a JS number
    - Operations on Numbers
        - Follow the order of operations when there are no parens (PEMDAS)
- Strings
    - "A string of characters"

    - Use \ to escape characters within quotes
        - "This is "quoted"": will not do as you expect
        - "This is \"quoted\"" will
    - \n: newline character. To print \n you need to escape it:
        - "A newline character is written like \"\n\"."
    - strings cannot be divided, multiplied, or subtracted but the '+' operator will concatenate them
- typeof
    - Prints a string representing the type of the operand passed to it.
    - e.g. typeof 4.5 = "number"
- booleans
    - true and false
    - "Aardvark" < "Zoroaster" = true
        - comparison of strings is more or less alphanumeric… Uppercase characters are always less than Lowercase characters
        - Why?
            - Unicode values
    - and: &&
    - or: ||
    - not: !
- Variables
    - Things used to catch and hold values
        - var caught = 5*5;
    - var is used to create a new variable
    - variables cannot include spaces, cannot start with a digit, and can include '$' and '_'
    - generally JS coders camelCase their variables
    - The collection of variables and their values at any given time is called the environment. When a program starts up, the environment is not empty.
        - e.g. When your browser loads a page, it creates a new environment and attaches the standard value to it
- Interacting with your browser's console
    - Type 'alert("Hello World!");' into your browsers console
    - Does it work if you forget the semicolon?
    - Eloquent JavaScript suggests using 'print' to print to the console. Does this work in our browser?
        - How should we print to the console?
    - confirm: allows us to ask yes/no in alert boxes
- Loops:

```
* while:
    var currentNumber = 0;
    while(currentNumber <=12){
        console.log(currentNumber);
        currentNumber = currentNumber + 2;
    }
* for:
    for(var number = 0; number <=12; number = number+2)
        console.log(number)
```

- Control Flow
  - if, else if, else: work like they do in Java or C++
- Truth Values:
  - JS has strange comparison rules for values of different types:
    - false == 0: true
    - "" == 0: true
    - "5" == 5: true
    - null == undefined
  - Generally what JS tries to do is to convert one the values to the type of the other.
  - Falsy values: 0, "", null, undefined, and false
  - Use === and !== when you do not want any automatic type conversion. This checks if the two values are exactly equal
    - null === undefined: false
    - false === 0: false
    - "" === 0: false
    - "5" === 5: false
- Implicit type conversion
  - null + "ify": "nullify"
    - adding a non-string to a string will convert the non-string into a string and then concat.
  - "5" * 5: multiplying a string by a number will try to convert the string into a number first