Intro to Javascript

JS for the Non-Programmer

About Me

- Work for Campus Labs going on 5 years
- Primary languages of choice are C#, Javascript and SQL, worked on projects with VB.NET (ugh), PHP
- Recently got involved with mobile application development - developed actively working on a few hybrid applications using javascript, having a lot of fun learning Swift
- Writing javascript for around 7 years

Course Overview

- Begin learning cross-language programming skills
- Understand javascript's expressive and somewhat crazy features
- Make server side data requests using Ajax
- Simplify DOM manipulation code with jQuery
- Simplify complex spaghetti-like jQuery code with frameworks like Angular.js / Ember.js

Course Overview

1. Begin to understand and read code
2. Write your own code
3. Write complex and useful code
4. Turn useful code into overly complex, terrible and useless code
5. Fix that complex code
6. Write great code from scratch

This Part - Beginners

We’re going to approach this course as though you are learning to program for the first time.

If you’re new to programming, get ready. This is a crash course but I will attempt to hand-hold you through it.

Veteran programmers, the workshops will have exercises that challenge everyone, if the first 35-45 min or so are boring, hang in there.

Today's Outline

1. Why start with Javascript?
2. Why it's a terrible idea to start with Javascript
3. History of the Language
4. Programmer Mindset Fundamentals
5. Basics of the Language
6. Types - Strings, Numbers and Boolean
7. Variables
8. Conditional Statements

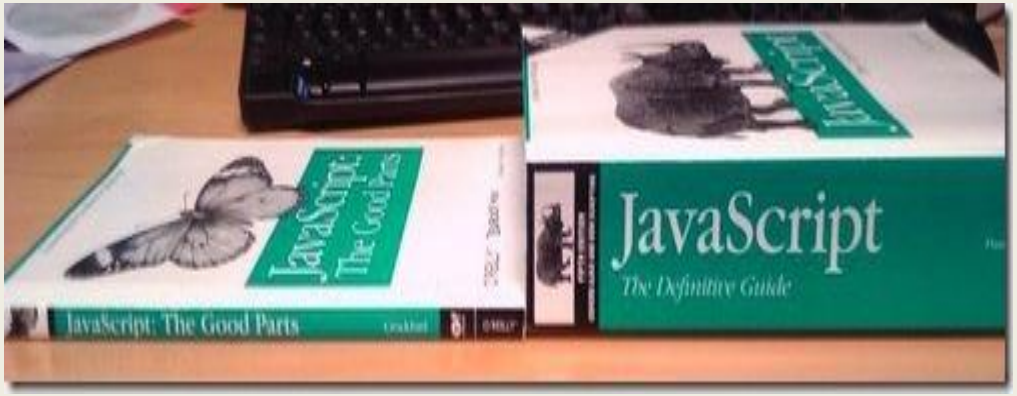
Why Start With Javascript?

- Incredibly easy to pick up and begin coding
- Very forgiving
- Relatively small language compared to most *
- Advent of Angular.js frameworks and Node.js means it’s becoming a real answer to programming problems
- More than swapping images or adding UI flair, entire applications are written client-side in javascript

Javascript: The Bad Parts

- Very forgiving, oftentimes you can’t eye out mistakes
- Imperfectly designed language (the use of ‘this’)
- Global variables
- Incredibly complex, advanced techniques for code reuse (prototypal inheritance)
- Scope of variables can be misleading
- Requires strict formatting (or “Linting”) to be assured of correct standards

Javascript ?


JavaScript: The Good Parts
JavaScript
The Definitive Guide

Brief History of JS

- Created in 10 days for Netscape in 1995 by Brendan Eich
- Originally referred to as Mocha, then LiveScript
- Finally called Javascript with trademark licensing from Sun, who created the programming language Java
- Java and Javascript are not even close to the same, more like long lost relatives
- Long period (1997 - 2004) of stagnation, ECMA tries to make standards for implementing client-side scripting languages, no thanks to IE
- jQuery is released in 2006, attempts to aid in cross-browser compatibility
- Javascript is now run in pretty much every device, Netflix app on PS4, Samsung Smart TVs

Javascript

Javascript is a prototype-based scripting language with dynamic typing and first-class functions. The mix of features allows it to support a mix of object-oriented, functional and imperative styles of programming.

- Wikipedia

What Makes a Programmer

- Two-way communicator - conduit from machine to human
- Strong desire to understand
- Practice, Practice, Practice
- Lots of Reading Code
- Standards
- Immediate Feedback
- Coding is only part of the toolbox - need domain expertise
- Stack Overflow (?)

Basics

Helpful debugging tools

document.write() - please don’t use in production.

console.log()

window.alert()

window.prompt()

// comments

Basics - Built In Types

- String
- Number
- Boolean
- Null
- Undefined
- Object (Not a Primitive Type)

Strings - Creation

- A string is any text inside single or double quotes

“Hello World”

‘Hello World’

- No special type for a single “character” in javascript.

Strings - Manipulation

- Find out the length

"Hello World".length

- Put two strings together

"Hello World".concat(" and the universe.")

- Make every letter uppercase

"Hello World".toUpperCase()

Strings - Manipulation

- Replace a portion

“Hello World”.replace(“World”, “Buffalo”)

- Return a portion with substring between indices

“Hello World”.slice(0, 5)

- Remove all whitespace from the beginning and end

“ Hello World ”.trim()

Strings - Other Methods

- indexOf(“Other string”) - Returns the numerical index of the string parameter or -1 if the string wasn’t found.
- substr(0, 6) - Similar to “slice”, except the second parameter is a count of how many characters you want to return
- toLowerCase() - exactly what you would expect.

Find more at http://www.w3schools.com/jsref/jsref_obj_string.asp

Numbers

Javascript numbers can come in the following forms

4	Positive Integer
-4	Negative Integer
4.89	Decimals
489e2	Scientific Notation
0xFF	Hexadecimal -> 255

For programmers -> Numbers are always 64-bit floating point in javascript

Numbers - Operators

Addition	+	3 + 4	// 7
Subtraction	-	2 - 5	// -3
Multiplication	*	10 * 10	// 100
Division	/	5 / 2	// 2.5
Remainder	%	5 % 2	// 1

Numbers - Operations

Order Of Operations - PEMDAS!

1. Parenthesis
2. Multiplication, Division and Remainders, left to right
3. Addition and Subtraction, left to right

$$5 + 3 * 4 - 7$$

$$(5 + 3) * (4 - 7)$$

Strings and Numbers

Because javascript is a dynamically-typed language, strings and numbers can be manipulated together, conversion is implicit.

Adding a string and a number together results in a string.

```
"3" + 4        // "34"
```

Be careful when doing math with user input, it may or may not be a number. Check your framework or HTML standards, convert as necessary.

Strings and Numbers

String to numeric conversion - the parseInt() function

parseInt() takes two parameters, the string to convert and a radix.

Radix defaults to 10 (typical decimal).

parseInt(“3px”) - Stops on the first non-numeric value

To do math with strings then, use the following…

```
parseInt(“3”) + 4    // 7
```

Strings and Numbers Demo

NaN - Not a Number

Strings that cannot be converted return the value NaN.

```
parseInt( “,asd” ) // NaN
```

Combine “NaN” with the javascript function isNaN() to tell whether you can successfully use a string in a mathematical operation.

```
isNaN( parseInt( “,kh” ) ) // true
```

Boolean

True or False Values

An expression or value that evaluates to TRUE or FALSE

When we use boolean values...

Need a yes or no answer

Turn part of the program “on” or “off”

Branching code paths

Boolean - Examples

The following operators, with strings and numbers, create boolean values

```
"Hello World".length > 7          // true

4 + 5 === 9                       // true

parseInt("3px") < 1               // false

8 + 7 >= 9 + 6                    // true
```

Boolean - Comparison

===	equal value and type	5 === 5
!==	not equal value or not equal type	6 !== 5
>	greater than	6 > 5
>=	greater than or equal	7 >= 3
<	less than	200 < 1000
<=	less than or equal	200 <= 200

Boolean - Evil Twins

There are two more evil twin comparison operators == and !=.

== compares to values, not types

Javascript will try and do Type Coercion

```
"5" == 5          // true, thanks I guess
"" == '0'         // false
"" == 0           // true
0 == '0'          // true
false == 'false'  // false
false == '0'      // true ???
```

Boolean - Combinations

&&	AND	5 > 3 && 6 > 2	// true
||	OR	6 > 4 || 1 > 7	// true
!	NOT	!(6 > 4)	// false

Variables

Variables are used to store information you want to use later in the program.

Declare variables with the keyword var.

```
var myName = “Kyle”;
```

Five Parts to a Variable

```
var uniqueName = VALUE;
```

You can also declare a variable without an assignment.

```
var myName;
```

Variables - Rules

- Can't use any of the reserved words for javascript. i.e. (new, delete, for, in)

```
var for = "Whoops";
```

- Can't contain any whitespace

```
var space in name = "Wrong";
```

- Must start with a letter or underscore

```
var 12dec = "Really wrong";
```

Variables - More Rules

- You can declare more than one variable at a time with a comma.

```
var myName, myAge, myLocation;
```

- Variables are case-sensitive

```
var person = 3;
var PERSON = 4;

console.log(person === PERSON)        // false
```

Variables - Demo

Variables - Guidelines

- Never declare a variable without using the var keyword.

```
name = “Kyle”;
```

- Don’t begin a variable name with uppercase

```
var MyVar = ‘’;
```

- Take great care in naming your variables.

Variable Naming - Bad

```
var myF = function (k) {
   var i = 21;

   return k >= i;
}
```

Variable Naming - Good

```
var canDrink = function (personsAge) {
  var legalLimit = 21;

  return personsAge >= legalLimit;
}
```

If - Statements

If-statement (conditional)

Used to run different code depending on the results of the statement.

```
if ( condition ) {
        // block of code
}
```

The block of code will execute if the condition is true.

Conditional Statements

Usage

```
if ( 7 > 3 ) {
        console.log("Seven is greater than three");
}

var superhero = "spiderman";
if ( superhero.length < 5 ) {
        console.log("This will not print");
}
```

Conditional Statements

Practical Examples

Demo

Else Statement

```
if ( condition ) {
        // block of code
} else {
        // block of code that runs instead
}
```

Conditional Statements

```
if ( 10 > 11 ) {
        console.log('Nothing in this block will run.');
} else {
        console.log('This is the code that will execute.');
}
```

Conditional Statements

When do we use this?

Demo

If-ElseIf Statements

```
if ( condition ) {
        // block of code
} else if ( another condition ) {
        // another block of code
} else {
        // even more potential code
}
```

You can chain “else-if” as many times as you like.

Conditional Statements

```
if ( 10 > 11 ) {
        console.log('Nothing in this block will run.');
} else if (12 > 10) {
        console.log('This is the code that will execute.');
}
```

Conditional Statements

Practical Examples

Demo

Workshop

What you'll need

- Text Editor, Sublime Text, Web Storm, Atom
- Browser - preferably Chrome, though Firefox is fast becoming hipster
- Alternatively, you can play with JsFiddle, JsBin, any online editor

Guidelines

- Try your hardest not to google the answer, it is out there.
- Break the problems down into the smallest possible level. The workshops are supersets of the material covered.

Workshop

Code is located

https://github.com/kylepace/JavascriptClass

Citations

https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

http://en.wikipedia.org/wiki/Ajax_%28programming%29

Javascript: The Good Parts - Douglas Crockford

http://www.w3schools.com/jsref/default.asp

Helpful Question Links

https://developers.google.com/webmasters/ajax-crawling/docs/specification