CS 4220

- Current Trends in Web Design & Development -

Cydney Auman

AGENDA

01 About the Course

02 Review the Syllabus

03 Setup Dev Environment

04 Intro to JavaScript

About the Course

JavaScript

Learning and understanding the language. ES6 syntax.

Node.js

Building command line applications. Creating a node module. Designing a server.

APIs

Interacting with data from APIs using Node.js.

Etc...

Frameworks.
NoSQL Databases.
TypeScript.

Cydney Auman

Software Engineer at NASA's Jet Propulsion Laboratory (JPL).

Lead Backend Software Engineer and Database Architect working on several projects that use Node.js, Python, ArangoDB and Elasticsearch.

Taught CS4220 focusing on JavaScript and Node.js for the past 5 years.

Syllabus

github.com/cydneymikel/CS4220

Interactive Poll

vevox.app

code: **112 - 123 - 027**

What is JavaScript?

JavaScript is a programming language that was created in 1995 at Netscape as a scripting tool to manipulate web pages inside the Netscape Navigator browser. The language has since been adopted by all major web browsers. It has made modern websites dynamic and interactive.

Part of the Netscape business model was to sell Web Servers, which included a component called Netscape LiveWire. This allowed server-side JavaScript programming. Unfortunately, Netscape LiveWire wasn't very successful. Server-side JavaScript wasn't popularized until the introduction of Node.js in 2009.

It is important to note that JavaScript has almost nothing to do with the programming language Java. The similar name was inspired by marketing considerations rather than good judgment. When JavaScript was being introduced, the Java language was being heavily marketed and was gaining popularity. Someone thought it was a good idea to try to ride along on this success. Now we are stuck with the name.

JavaScript Standard

The ECMAScript Specification

What is ECMAScript Standard? And why is ES6 important?

ECMAScript Standard is the document that describes the way the JavaScript language should work within the browser and/or other various software that claimed to support JavaScript.

ES5 is what most people are familiar with when talking about JavaScript. ES5 was released in 2009 and with it came several enhancements to the language.

ES6 started the next generation of JavaScript. And with its release in 2015 came an abundance of new features. **ES6** is geared toward solving problems that developers actually face in the real world scenarios. It is now the standard used across almost all major companies when writing JavaScript code. ES6 also introduced the concept of moving forward with yearly releases of standard specs.

Java vs. JavaScript

Java

Java is a **compiled** programming language. It order to execute it is first compiled into bytecode and then runs on the JVM.

Java uses **static type checking**. The type of a variable is checked at compile-time. Variables must declared with a specific type (int, float, string, etc.).

In Java **functions are not first class**. Java 8 seems to partially support this concept, but it is not support in versions before Java 8.

Class Based Object Oriented Programming. Class and instance are distinct entities.

JavaScript

JavaScript is an **interpreted** scripting language. It can be interpreted directly by a browser and/or engine in the same syntax it is written

JavaScript use **dynamic type checking**. This type checking is done at run-time. Dynamic typing is flexible...which means it allows for variables to change types.

JavaScript has **first class functions**. This means a function can be passed as an argument to other functions, can be returned by another function and can be assigned as a value to a variable.

Prototype Based Object Oriented Programming. All objects can inherit from another object.

CORE JS

Contains a core set of objects, such as Array, Date, Math, etc. Also contains a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by providing it with additional objects.

CLIENT SIDE JS

Extends the core language by supplying objects to control the browser and its Document Object Model (DOM). For example, client-side extensions allow applications to interact with HTML forms, respond to events such as mouse clicks, and handle page navigation.

SERVER SIDE JS

Extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow applications to communicate with databases, access files on a server, and interact with the operating system.

Variable Declaration

Variables in standard JavaScript have no type attached, and any value can be stored in any variable. In ES5 variables were declared using the keyword **var**.

ES6 introduced **const** and **let**.

Using **const** makes variables a constant value. Variables defined using the keyword **const** cannot be changed through reassignment and it cannot be redeclared.

Using **let** is more similar to var in the sense that you can change the value assigned to it.

JavaScript Primitive Types

NUMBER

JavaScript does not define different types of numbers, like integers, short, long, float, etc. *They are always 64-bit Floating point.*

STRING

JavaScript strings can be created using single or double quotes.

BOOLEAN

true and false literals.

UNDEFINED

The value of "undefined" is assigned to all uninitialized variables. It is also returned when checking for object properties that do not exist.

NULL

Often set to indicate that something has been declared *BUT* has been defined to be empty.

Operators

The Equals Operator

`(==)` and `(!=)`

The **==** version of equality is quite liberal. Values may be considered equal even if they are different types. In JavaScript the operator will **force type coercion** of one or both into a single type before performing a comparison.

The Strict Equals Operator

`(===)` and `(!==)`

If the values are of different types the answer is always false. If they are of the same type an intuitive equality test is applied. Strings must contain identical character sets and other primitives must share the same value. Null and Undefined will **never** === another type.

Operators Continued...

Comparison Operations

Greater than (`>`)

Great than or equal (`>=`)

Less than (`<`)

Less than or equal (`<=`)

Unary Operators (typeof)

Operators that take one value. Not all operators are symbols, so can be written as words.

`typeof` - this returns the type of variable as a string.

Logical Operators

Javascript supports three logical operators.

- And (`&&`)
- Or (`||`)
- Not (`!`)

Structure and Indentation

Spaces are not required — the browser and/or engine will accept the program just fine without them. In fact, even the line breaks in programs are optional. You could write a program as a single long line.

The **role of this indentation** inside blocks is to make the structure of the code stand out. In code where new blocks are opened inside other blocks, it can become hard to see where one block ends and another begins if indentation is not done properly.

With proper indentation, the visual shape of a program corresponds to the shape of the blocks inside it.

Installation & Setup

- Go to **www.nodejs.org**.
  Download the LTS version (12.x.x)

- Open your terminal.
  Run the command: **node -v**
  The output should be: 12.x.x

- Go to **www.code.visualstudio.com**
  Download the version for your OS.

```
console.log(‘Week 01’);
console.log(‘Code Examples’);
```

Review and Prep

Review

- Review Slides
- Run Class Examples in local Terminal
- Read Eloquent Javascript - Chapters 1 & 2

Preparation for Next Week

- Read Eloquent Javascript Chapters 3 & 4