# CS 4220

## - Current Trends in Web Development -

Application Design and Development with Node.js

Prof. Cydney Auman

# AGENDA

**01** About the Course

**02** Review the Syllabus

**03** A Look at JavaScript

**04** Dev Setup + JavaScript Intro

# Course Description

This course focuses on application design and development utilizing Node.js, a cross-platform runtime environment. Using the JavaScript language with Node.js this course will have students build back-end applications such as command line interfaces as well as building a web server.

In order for students to be successful working with Node.js, this course begins with an emphasis on JavaScript by covering the fundamentals of the JavaScript programming language .

Throughout the course there will also be a focus on interacting with Third Party Web APIs to make requests to retrieve JSON data. As well as, utilizing MongoDB to store and retrieve information.

# About the Course

## JavaScript

- Proper Code Formatting, Debugging
- Syntax, Variable Declaration, Types, Operators
- Working with Objects, Arrays and Functions
  - Methods on Objects and Arrays
  - Spread Operator, Destructuring
  - Arrow Functions, Default Parameters
- Callbacks and Promises
  - Event Loop
  - Call Stack
  - Async/Await

## Node.js + Web APIs + Mongo

- Working with Modules
  - Using Node Modules
  - Building a Node Module
- Building a Command Line Applications
- Building a Node.js Server
- Working with Web APIs
  - What is a Web API?
  - Integrating with an API and JSON
  - Handling and manipulating data structures
- Data Stores
  - Working with MongoDB
  - Mongo Query Language

# Syllabus

Visit Canvas for CS4220

# 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** language.  It can be interpreted directly (browser/node) in the same syntax it is written

JavaScript is  **dynamically typed** and uses **dynamic type checking**.  Checking is done at run-time.  The language allows for variables to change types.

JavaScript has  **first class functions**.  This means a function can be passed as arguments, 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.

# JavaScript History

1993

First web browser released by the name of Mosaic.

1994-1995

Netscape surpasses Mosaic in popularity.  Wants to integrate dynamic and interactive elements into the browser.  Brendan Eich is hired and works on a project with the code name Mocha.  This name is later changed to LiveScript and in December of 1995 to JavaScript.

1996

Microsoft reverse engineered the JavaScript interpreter and releases JScript. However, this works differently than standard JavaScript

1997

It is now obvious that a standard is needed for the JavaScript language.  Every involved agrees to use ECMA International to create a standardization and specification.  The name of this standardization and specification is "ECMAScript" or "ES" and was a compromise between the organizations involved.

# JavaScript History

1997 to 1999
ES1 of JavaScript released with ECMA in ‘97.  And ES2 in ‘98.  And ES3 in ‘99.

1995 to 2017
ES4 never released.  Browser Wars get even bigger.  First - (1995 to 2001) Explorer versus Netscape.
Second - (2004 to 2017) Explorer, Mozilla, Opera, Safari and Chrome…etc

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

2015
ES6 released and started the next generation of JavaScript.  ES6 came an abundance of new features and is geared toward solving problems that developers face in the real world scenarios. ES6 also introduced the concept of moving forward with yearly releases of standard specs.

PRESENT
As of June 2022 ES13 was published and released.  "ES Next" is always the code name for the next version.

## 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.

# Why JavaScript and Node.js?

JavaScript is the most popular programming language in the world according to many sources including GitHub, StackOverflow and site setup to scan job listings.

Many of the largest and most significant tech companies such as Microsoft, Meta, Netflix, eBay, PayPal, LinkedIn, Spotify, Uber and more use the JavaScript language.  In fact, you it would be difficult to find any company that deals with the development of some kind that doesn't have JavaScript in its tech stacks.

JavaScript has achieved it's popularity because - it is user-friendly, vast community with resources, performant, runs almost everywhere and speeds up delivery times.

Node.js has become popular because it uses the JavaScript language but also sits on its own laurels - it is fast, can be hosted anywhere, easy to maintain and scales.  Plus, it is cross platform and support mobile dev.

**See Resource Slide with links to references.*

# Installation & Setup

- Go to **www.nodejs.org**.
  Download the LTS version
  (18.x.x or higher)

- Open your terminal.
  Run the command: `node -v`
  The output should be the
  version number.

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

- Download the demo file.
  Go to the dir where the demo
  file was downloaded.
  **Run the file with:** `node demo.js`

**Output:** `Welcome to Spring 2023 CS4220`

# Formatting and Linting


Formatting and Linting  are two separate tools.  Formatting is useful to enforce coding style and code conventions .  While Linting is useful to detect and alert the user about potential bugs or errors.

By using VS Code Extensions and configuration files we can control and enforce these styles and conventions.

Make a top level **CS4220** directory.  Underneath you can create and nest as many sub-directories desired.

We will then add our formatting and linting configuration files under the top level directory.

# Formatting - Prettier

Prettier (https://prettier.io/) is a code formatter with support for JavaScript, TypeScript, CSS, HTML, and much more.

Prettier ensures that all outputted code conforms to a consistent style. It enforces a consistent code style (i.e. code formatting) across your entire codebase. Prettier supports a `.prettierrc.json` configuration file where users can override built in defaults and customize this uniform styling.

The Prettier Extension can be installed into VS Code and configured to format on save.

1. Install Prettier VS Code Extension
2. Download `prettierrc.json` file from Canvas
3. Rename to `.prettierrc.json` (prefixed with dot) and add the file in top level directory.

# Linting - ESLint

ESLint (https://eslint.org/) is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code, with the goal of making code more consistent and avoiding bugs.

ESLint supports a `.eslintrc.json` configuration file where users can override built in defaults and customize the way JavaScript coding is enforced.

ESLint requires 3 steps in order to be properly integrated into VS Code.

1 Install the ESLint VS Code Extension
2. Run this command in your terminal: `npm install -g eslint`
3. Download the `eslintrc.json` file from Canvas
4. Rename to `.eslintrc.json` (prefixed with a dot) and add the file in top level directory.

# Detect and Fix Errors

When running your JavaScript you will inevitably encounter an error at some point.

There is no need to get frustrated.  When executing the files, the **error feedback** is particularly detailed and designed to help you **locate** your error.

```
$ node examples.js <- command to execute the file
/Users/auman/CSULA/Spring2023/CS4220/Week01/errors.js:5          <- full path of file executed
xyz; <- a snippet of the code causing the error
^

ReferenceError: xyz is not defined <- the name of the error
   at Object.<anonymous>
      (/Users/auman/CSULA/Spring2023/CS4220/Week01/errors.js:5:1)   <- line number where the
                                                                       error is located
```

# Printing/Logging to the Console

**`console.log()`** is a function that will print/log any variables or messages to the console. This is function is useful for printing out data and/or debugging code. Additionally, **`console.log()`** can accept multiple argument separated by a comma.

```
console.log('Hello World !');
// Hello World !


console.log('Hello', 'World', '!');
// Hello World !
```

# 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`**. We will **not** be using `var` as this is dated and no longer a best practice.

**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*. But it can be mutated (Objects and Arrays).

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

**See VS Code Extensions Slide to assist with proper variable declarations*

# 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**

Is used when a variable has been declared *BUT* is intentionally being defined to be empty (null).

# JavaScript Non-Primitive Types

**OBJECT**

A collections of key-value pairs.  Similar to dictionaries in Python or HashMaps in Java. Unlike objects in statically typed languages, objects in JavaScript do not have fixed shapes - properties can be added, deleted, re-ordered, mutated, or dynamically queried at any time.

**ARRAY**

Arrays in JavaScript are actually a special type of object.  The Array object, as with arrays in other programming languages, enables storing a collection of multiple items.  Arrays also has methods for performing common array operations.  Arrays can have any elements and can grow or shrink arbitrarily.

**FUNCTION**

In JavaScript, every function is actually a Function object.  A JavaScript function can take 0 or more arguments. The function body can contain as many statements as you like and can declare its own variables which are local to that function. The return statement can be used to return a value at any time, terminating the function. If no return statement is used (or an empty return with no value), JavaScript returns undefined.

# Resources

**JavaScript History**

- ECMA
  - https://www.ecma-international.org/
- Archived NetScape
  - http://www.mcom.com/home/welcome.html
- Conversation with Brendan Eich
  - https://www.youtube.com/watch?v=krB0enBeSiE

**JavaScript Popularity and Usage**

- https://octoverse.github.com/2022/top-programming-languages
- https://survey.stackoverflow.co/2022/#technology
- https://www.devjobsscanner.com/blog/top-8-most-demanded-languages-in-2022/

**VS Code**

- https://code.visualstudio.com/docs/languages/javascript
- https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint
- https://github.com/SimonSiefke/prettier-vscode

# Review and Prep

**Review**

- Review Slides
- Ensure eslint and prettier are properly setup
- Run the demo.js file
- Run the error.js file

**Preparation for Thursday**

- Read Eloquent Javascript - Chapter 1
- Explore and get familiar with VS Code