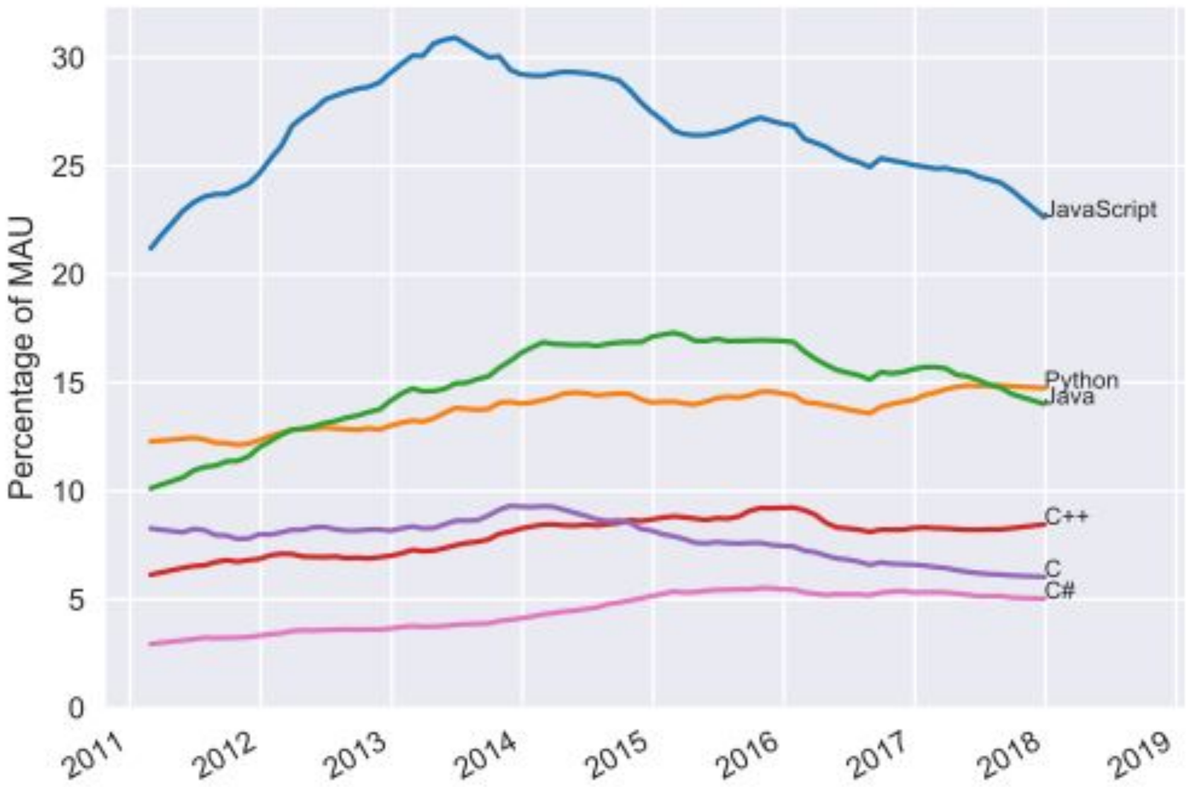
Introducing Assignment 0: A JavaScript Crash Course

COS 426: Computer Graphics (Spring 2020)

Reilly Bova

First Let’s Motivate: Why JavaScript?



First Let’s Motivate: Why JavaScript?

Traditional Graphics Education and Industry Programming is in C++

- Pros of C++:
  - Great if you are going into industry
  - Fast execution; systems access for optimization (memory, threads, etc.)
  - Decades worth of libraries and support
- Cons of C++:
  - Steep learning curve; need to worry about syntax
  - Hard to debug and high debugging overhead (need to recompile following any change)
  - Not always portable, which makes both development and grading somewhat harder
  - Difficult to share live C++ graphics demos, since users would need to download and compile
  - Showing its age, and generally considered a messy/poorly designed language

First Let's Motivate: Why JavaScript?

Our Assignments are written in JavaScript (and GLSL). **Hear us out!**

- Pros of JS:
  - High demand for JS development experience (this is great for you!)
  - JS is far more accessible and far easier to debug and test (no compiling overhead!)
  - Reduced overhead speeds up your development time a ton. We can assign more interesting tasks
  - JS/WebGL can harness GPU; powerful enough to run realistic 3D games at high FPS
  - Excellent JS graphics libraries (e.g. ThreeJS) with modern support/documentation
  - Extremely portable and easy to share (can run directly in modern browsers)
  - Assignments will give students the tools they to develop beautiful 3D art demos that they can drop right into a personal website or publish to a github webpage.
    - **Great for impressing friends, family, and future employers :)**
- Cons of JS:
  - Slower than C++, but not noticeably so within the use-cases of assignments
  - Limited memory/threading, but these are not needed for assignments
  - The portion of the class potentially interested in entering the graphics industry will eventually need to learn C++; however, they will likely take additional graphics courses (covering C++) anyways

First Let’s Motivate: Why JavaScript?

TLDR:

- We want our students to do as **much** as possible, as **easily** as possible, for as many **people** as possible.
- 426 is just an introduction to Computer Graphics.
  - Students who want to go into the graphics industry will need to take additional graphics courses (e.g. COS 526) anyways, and these advanced courses will use C++
- Most 426 students will not continue into the graphics industry, but the skills they learn in this class will still be extremely useful

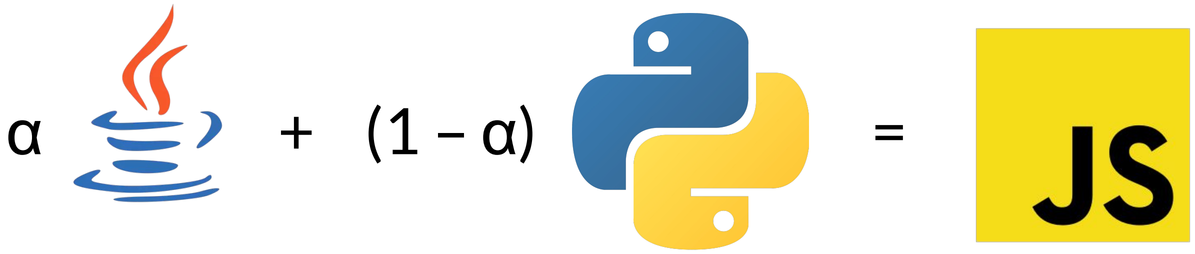
Some Cool Demos

- https://tympanus.net/Tutorials/TheAviator/
- https://paperplanes.world/
- https://www.foosballworldcup18.com/
- http://playdoh-lagaleriedesespeces.com/en/
- https://threejs.org/examples/?q=rea#webgl_postprocessing_unreal_bloom
- https://threejs.org/examples/?q=ocea#webgl_shaders_ocean
- https://phoboslab.org/xibalba/
- https://jbechara.github.io/Singularity/ <– A3 Art Project!
- https://beckybarber18.github.io/coloring/ <– S19 Final Project!

Rmk: No downloading required! The 3D viewer loads right into your browser!

A Crash Course in JavaScript

- JavaScript syntax is somewhere in between Java and Python. If you know one (or both) of these languages, you should be in good shape. **Learning a new programming language is really like learning a new dialect/accent!**
- Like Python, JavaScript is not compiled, but *interpreted*.
- Like Java, JavaScript requires brackets (although semicolons are optional) and variables must be declared
- “Try translating a Python script to Java, but then give up halfway through. That’s pretty much JavaScript”



A Crash Course in JavaScript

Ultra-brief History

- JS started at Netscape in the 1990s. Back then it was just meant to be used for quick-&-dirty web scripts. JS bears no relation to "Java". That's just marketing.
- Because of the informal use-case, JS is highly flexible — there are many ways to accomplish the same thing.
- Over the past decade or so, JavaScript has really exploded. Modern websites are now written entirely in JavaScript (thanks to frameworks like React).
- The language has also improved to match its modern demands:
  - Google's V8 interpreter compiles JS to assembly during execution.
  - Syntax has improved following the ES6 (aka ECMA2015) standards.

A Crash Course in JavaScript

Variable Scope in JS

- The scope of a JavaScript variable depends on how it was declared
- There are three scopes: **global**, **function**, and **block**
- As of JS ES6 , there are three declaration keywords: **var**, **const**, and **let**
- A variable has **global scope** if it was declared as a **var** outside of any function:

```
var carName = "Volvo";

// code here can use carName

function myFunction() {
  // code here can also use carName
}
```

A Crash Course in JavaScript

Variable Scope in JS

- The scope of a JavaScript variable depends on how it was declared
- There are three scopes: **global**, **function**, and **block**
- As of JS ES6 , there are three declaration keywords: **var**, **const**, and **let**
- A variable has **global scope** by default if it was declared without a keyword:

```
myFunction();

// code here can use carName

function myFunction() {
  carName = "Volvo";
}
```

A Crash Course in JavaScript

Variable Scope in JS

- The scope of a JavaScript variable depends on how it was declared
- There are three scopes: **global**, **function**, and **block**
- As of JS ES6 , there are three declaration keywords: **var**, **const**, and **let**
- A variable has **function scope** (like Python variables) if it was declared as a **var** inside a function:

```
// code here can NOT use carName

function myFunction() {
  var carName = "Volvo";
  // code here CAN use carName
}
```

A Crash Course in JavaScript

Variable Scope in JS

- The scope of a JavaScript variable depends on how it was declared
- There are three scopes: **global**, **function**, and **block**
- As of JS ES6 , there are three declaration keywords: **var**, **const**, and **let**
- A variable has **block scope** (like Java variables) if it was declared as a **let** inside a function:

```
var x = 10;
// Here x is 10
{
  let x = 2;
  // Here x is 2
}
// Here x is 10
```

A Crash Course in JavaScript

Variable Scope in JS

- The scope of a JavaScript variable depends on how it was declared
- There are three scopes: **global**, **function**, and **block**
- As of JS ES6 , there are three declaration keywords: **var**, **const**, and **let**
- A variable has **block scope** (like Java variables) if it was declared as a **const** inside a function. Note that **const** variables cannot be changed:

```
var x = 10;
// Here x is 10
{
  const x = 2;
  // Here x is 2
}
// Here x is 10
```

A Crash Course in JavaScript

Variable Scope in JS

- In general, it is best practice to avoid **var** altogether (our assignment code is not great about this at the moment, but it will be changing).

```
for ( var x = 0; x < 10; x++ ) {
  console.log(x);
  // prints 0, 1, ..., 9
}
console.log(x);
// prints “10” because x is still within function scope!
```

A Crash Course in JavaScript

Data Types in JS

- JavaScript variables are **dynamic**; a variable that holds a number can be redefined as a string, function, etc.
- There are seven main data types in JavaScript **(you don't need to know this)**:
  - Numbers (Rmk: there is **no distinction** between integers and floats)
  - Strings (Rmk: use single or double quotes; use ` (back tick) for multiline)
  - Booleans (Rmk: lowercase): true/false
  - Arrays
  - Objects (including **null**)
  - Functions
  - Undefined

A Crash Course in JavaScript

Arrays in JS

- Arrays in JavaScript work just like lists in Python
- You can append to arrays using the .push() function:

```
let arr = [];
for ( let x = 0; x < 10; x++ ) {
  arr.push(x);
}
console.log(arr);
// prints [0, 1, ..., 9]
```

- Further useful Array operations (like sorting, mapping, and iteration) can be found here.

A Crash Course in JavaScript

Functions in JS

- There are three main ways to declare functions in JavaScript
- Version 1:

```
function myFunction(a, b="default value") {
  return a + b;
}
```

- Version 2:

```
const x = function (a, b="default value") {return a + b};
```

- Version 3 (arrow function; good for one-liners):

```
const x = (a, b="default value") => {return a + b};
```

A Crash Course in JavaScript

Objects in JS

- Objects are declared similar to Python dictionaries, but function more like a Java Class (although they can still be used like Python dictionaries)
- You can add and overwrite object properties as you go
- Objects can contain functions

```
let person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}

let x = person;
x.age = 10;            // This will change both x.age and person.age
```

A Crash Course in JavaScript

Object Constructors in JS

- Objects can be declared as functions, which serve as constructors

```
function Person(firstName, lastName, age, eyeColor) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
  this.eyeColor = eyeColor;
  this.changeName = function (name) {
    this.lastName = name;
  };
}
```

A Crash Course in JavaScript

Instancing Objects in JS

- You can instance objects (as you would instance a class in Java) using the **new** keyword. (NB: no need to free later; JS has garbage collection)
- If you wish to add additional instance variables or methods outside of the constructor, use Object.prototype

```
Person.prototype.name = function() {
  return this.firstName + " " + this.lastName;
};

let me = new Person(“Reilly”, “Bova”, 20, “Brown);
console.log(me.name())
// prints “Reilly Bova”
```

An Introduction to Assignment 0

Getting Started

1. Visit the assignment 0 page.
2. Download the zip file.

Starting the Server

1. Extract the files.

```
$ unzip cos426-assign0.zip && cd cos426-assign0
```

2. Start the server with any of the following commands:

```
$ python3 -m http.server
$ python -m SimpleHTTPServer
$ php -S localhost:8000
```

An Introduction to Assignment 0

Who Are You?

1. Open "student.js"[1] using your favorite editor. We recommend either:
   - VSCode
   - Atom
2. Edit 'Student Name' and 'NetID'
3. Open the server and check that it worked! Visit[2]

   ```
   http://localhost:8000
   ```

[1] Look in the directory named js
[2] We recommend Google Chrome for its developer tools, but Safari and Firefox are okay too.

An Introduction to Assignment 0

“Implement” the Fill Tool

1. Now open “filters.js”
2. Uncomment the “setPixel” line
3. Verify that it works:
   - Refresh `http://localhost:8000`
   - Click the Fill button
   - Disable cache by leaving the Developer Tools window open
   - You may need to “Force Reload” (CMD+Shift+R)

An Introduction to Assignment 0

Debugging Tip

- **Amateur**: Trace statements that print into the browser's developer console
  - E.g. "console.log(`Color is ${pixel.r} ${pixel.g} ${pixel.b}`);"

- **Professional**: Use the browser's built-in debugger
  - Just add the line "debugger;"

An Introduction to Assignment 0

Final Note

- This assignment is designed to be an easy warm-up! It may take 15 min for students familiar with JavaScript, and 3+ hours for those with no experience
  - Rmk: The idea here is to make sure everyone has some JS experience going into A1
  - Please style and comment your code so that it is readable.
- The **Art Project** is optional, but most students submit something. Instructors award bonus points to the top few submissions. We encourage:
  - Visually pleasing submissions (“Look at my work of art!”)
  - Intellectually stimulating submissions (“Look at this extra feature I made”!)
  - Funny submissions (“Look at my buggy output!”)
- Have fun!

An Introduction to Assignment 0

Learn JavaScript

- Mozilla JavaScript Guide
  - Mozilla is one of the developers of, and contributor to, many web standards
- Wikibooks JavaScript "Book"
  - structured as a book, but available completely online
  - great reference for quickly finding syntax