Introducing Assignment 0:
A JavaScript Crash Course

COS 426: Computer Graphics (Spring 2022)

Vivien Nguyen, Ethan Tseng

First Let's Motivate: Why JavaScript?

Traditional Graphics Education and Industry Programming is in C++

- Pros of C++:
  - Commonly used in industry for graphics programming
  - Fast execution; systems access for optimization (memory, threads, etc.)
  - Decades worth of libraries and support
- Cons of C++:
  - Steeper learning curve than JS; need to worry about manual memory management
  - Hard to debug and high debugging overhead with memory issues as well
  - 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):

- Pros of JS:
    - High demand for JS development experience
    - JS is more accessible and faster to debug and test
    - JS/WebGL can use the 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 by running directly in browsers
    - Assignments will give students the tools they need to develop beautiful 3D art demos that they can drop right into a personal website or publish to a github webpage.
- 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
    - People potentially interested in entering the graphics industry will eventually need to learn C++; however, they will likely take additional graphics courses such as COS 526 which covers C++.

First Let's Motivate: Why JavaScript?

https://www.benfrederickson.com/ranking-programming-languages-by-github-users/
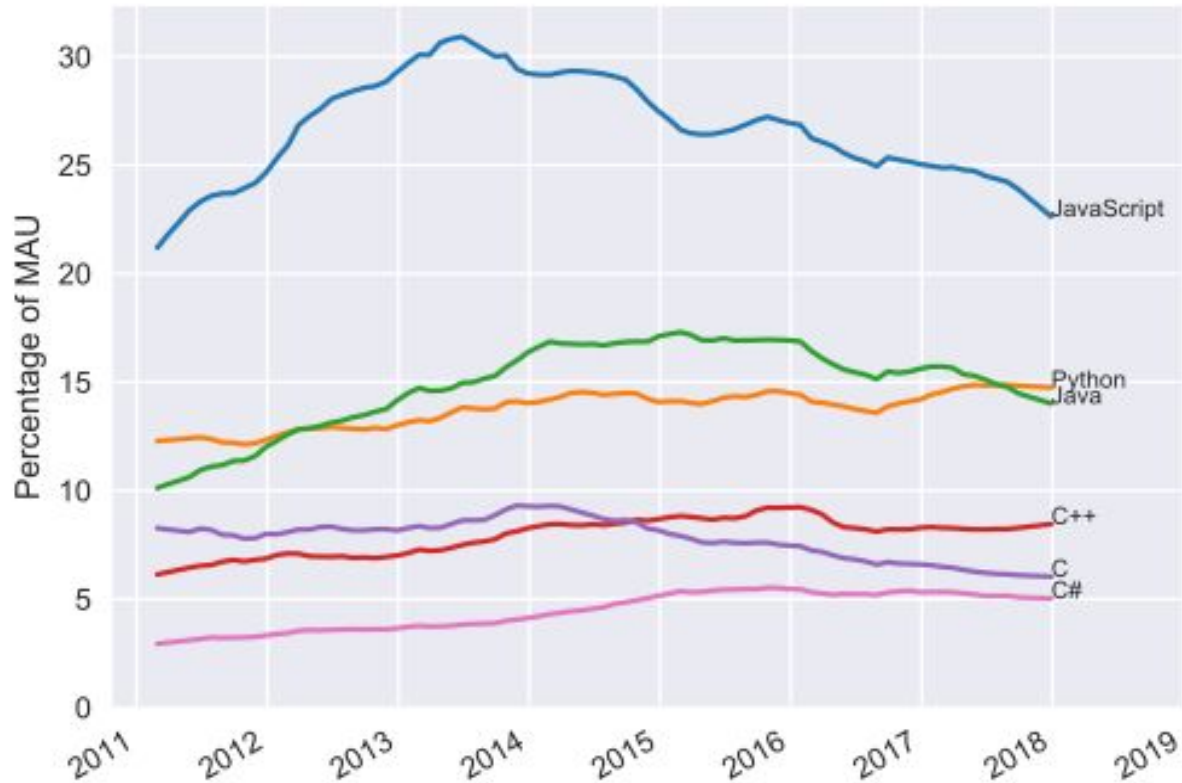
First Let's Motivate: Why JavaScript?

TLDR:

- We want students to do as **much** as possible, as **easily** as possible, for as many **people** as possible.

- Most 426 students will not continue into the graphics industry, but the skills they learn in this class will still be extremely useful
  - Mathematical concepts in graphics are broadly applicable across the sciences
  - JS is common in both front-end & back-end development (ReactJS, Node.js)
  - Final project is good way to practice building a large-scale project

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://www.shadertoy.com/
- https://dreamworld-426.github.io/dreamworld/ <- S20 Final Project!
- https://oliverschwartz.github.io/going-viral/ <- S20 Final Project!
- https://beckybarber18.github.io/coloring/ <– S19 Final Project!
- https://collideoscope.github.io/ <- S19 Final Project!
- https://jbechara.github.io/Singularity/ <– A3 Art Project!

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

A Crash Course in JavaScript

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 exploded. Modern websites are now written entirely in JavaScript (ex. React).
- The runtime has also improved to match its modern demands:
  - Google's V8 interpreter compiles JS to assembly during execution.
  - Syntax has improved following the ES6 standards.

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.

- Like Python, JavaScript is a dynamically typed, interpreted language.

- Like Java, JavaScript requires brackets and variables must be declared (semicolons are optional)

  - Recommend installing the add-on Prettier, a code formatter, in your code editor to keep your code nice and neat!

- "Try translating a Python script to Java, but then give up halfway through. That's pretty much JavaScript"
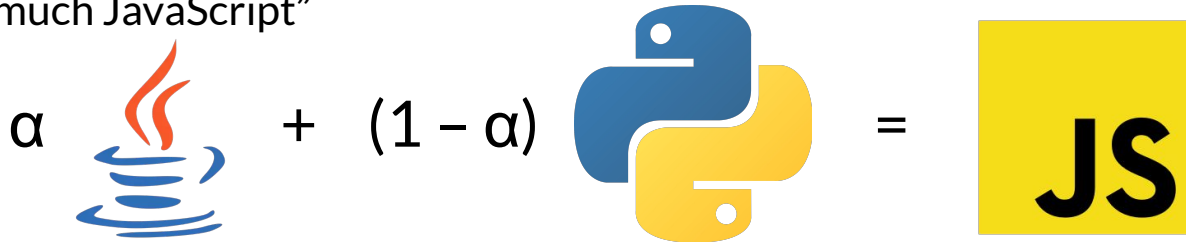
$$\alpha \quad + \quad (1 - \alpha) \quad = \quad \text{JS}$$

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:

```javascript
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:

```javascript
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 with **const** inside a scope. Note that **const** variables cannot be changed:

```javascript
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, do not use **var** in your assignment code to avoid bugs! Instead use **let** for mutable variables, and **const** for immutable variables
  - Our assignment code is not great about this at the moment, but it will be changing

```javascript
function myFunction() {
    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*:
  - Numbers (there is **no distinction** between integers and floats)
  - Strings (use '' or ""; use `back tick` for multiline)
  - Booleans: true/false
  - Arrays: [1,2,3]
  - Objects (including **null**)
  - Functions
  - Undefined

* https://medium.com/better-programming/everything-in-javascript-is-an-object-except-for-when-it-isnt-305bc65a3410

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:

```javascript
let arr = [];
for ( let x = 0; x < 10; x++ ) {
  arr.push(x);
}
console.log(arr, arr[5]);
// prints: [0, 1, ..., 9] 5

let [x, y, ...rest] = arr; // destructuring an array
console.log(x, y, rest);
// prints: 0, 1, [2, … , 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};
let x = (a, b) => a * b; // implicitly returns result in this form
```

A Crash Course in JavaScript

Objects in JS

- Objects are declared similar to Python dictionaries / Java maps
- 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
x.hairColor = "black"; // This adds the property hairColor to x and person

const {firstName, lastName} = person; // destructuring an object
console.log(firstName, lastName);
// prints: John Doe
```

A Crash Course in JavaScript

Classes in JS

- Classes can be defined as a function

```javascript
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;
  };
}
```

- The **this** keyword is not available in an arrow function

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. (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

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

let me = new Person("Jane", "Doe", 20, "Brown");
console.log(me.name())
// prints "Jane Doe"
```

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-Assignment-0.zip && cd cos426-Assignment-0
```

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

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

- 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 longer for those with no experience
  - 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