Introduction to Computer Graphics (CS360A)

Instructor: Soumya Dutta

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur (IITK)

email: soumyad@cse.iitk.ac.in

Reaching Out to Me for Doubts/Queries?

- Please use my CSE email to communicate with me and get quick response of your queries/doubts
- My CSE Email is: soumyad@cse.iitk.ac.in

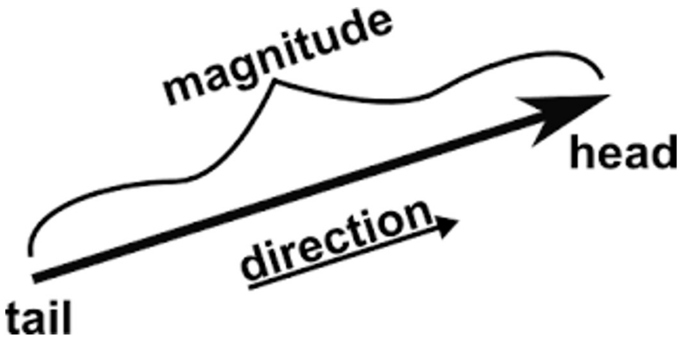
- **Off Topic:** Here is an interesting article for you:
- **How the Computer Graphics Industry Got Started at the University of Utah**
  - https://spectrum.ieee.org/history-of-computer-graphics-industry

Acknowledgements

- A subset of the slides that I will present throughout the course are adapted/inspired by excellent courses on Computer Graphics offered by Prof. Han-Wei Shen, Prof. Wojciech Matusik, Prof. Frédo Durand, Prof. Abe Davis**,** and Prof. Cem Yuksel

A Quick Mathematics Background

- Vectors
- Matrices



$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Vectors

- 1D: $[x]$
- 2D: $[x, y]$
- 3D: $[x, y, z]$
- 4D: $[x, y, z, w]$
- nD: $[x, y, z, w, \dots.]$

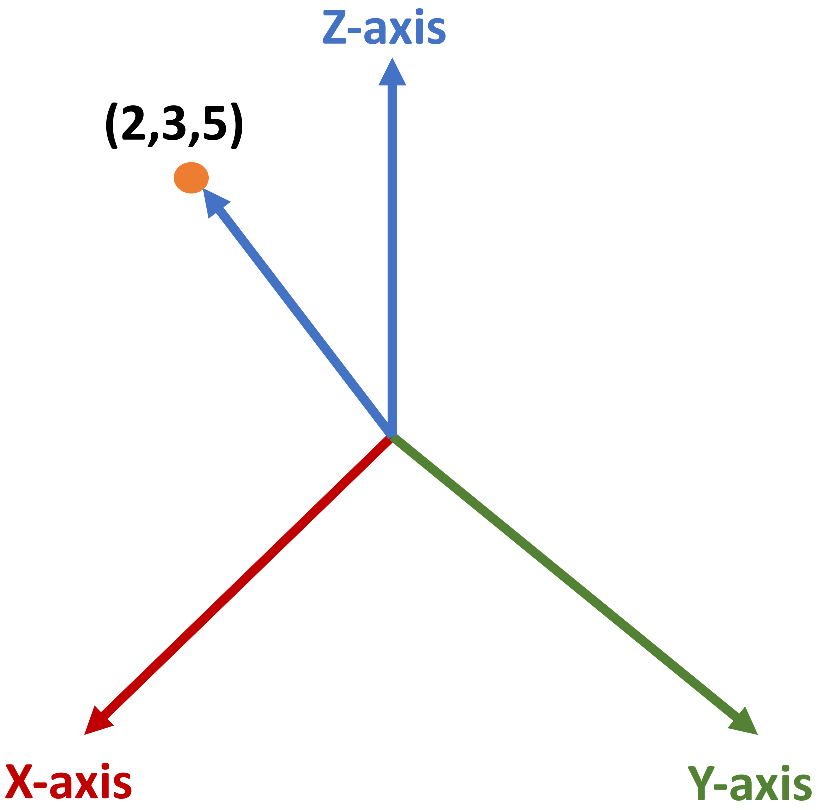
Typical notation:

- $\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [x\ y\ z]^{\mathrm{T}}$

- **Meaning:**
  - Position in 3D space
  - Direction (with length)

Vectors

- Meaning:
  - Position in 3D space
  - Direction (with length)
- **a** = $\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$



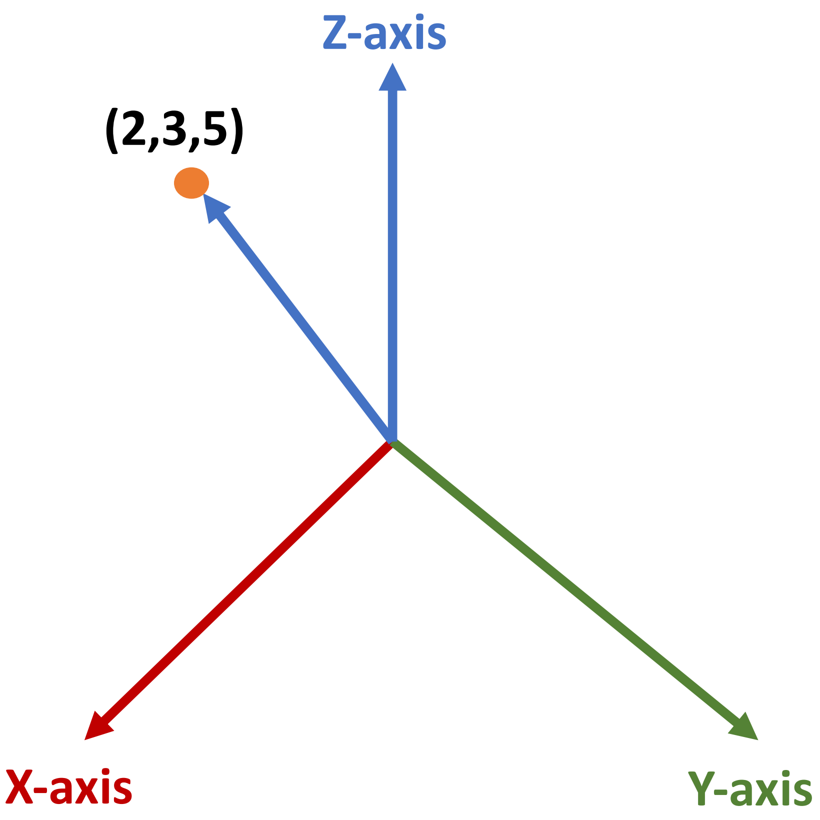
A vector without any context is meaningless, the context is often the coordinate axes

Vectors

Length of a vector:

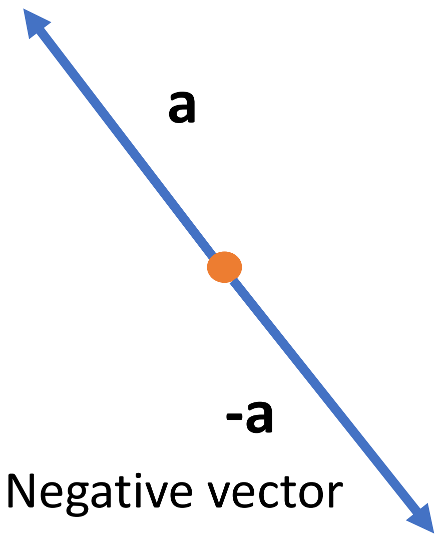
- |**a**| = $\sqrt{a_x{}^2 + a_y{}^2 + az^2}$

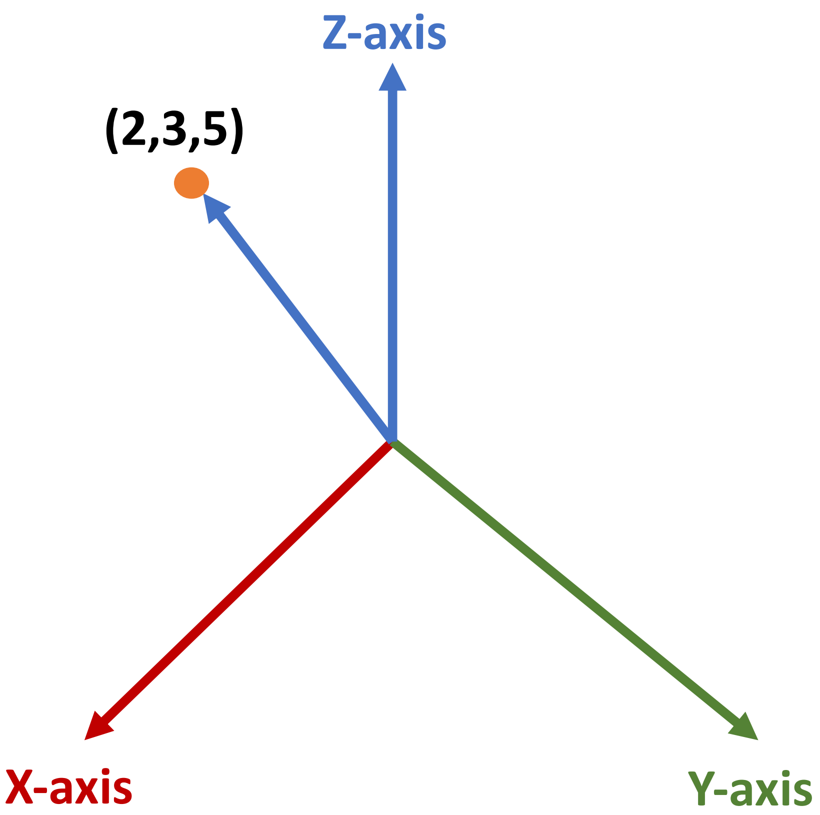
- Unit vector:
- |**a**| = 1



Vectors

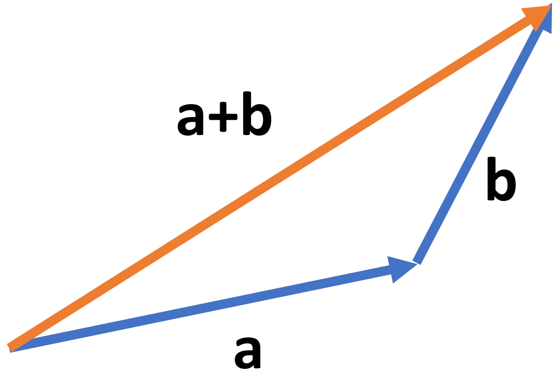
Operations: Negation



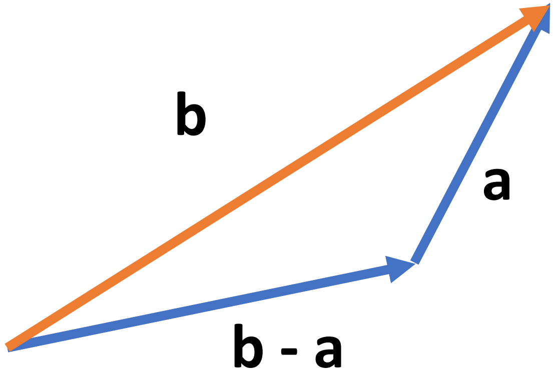


Vectors

Operations: Addition and Subtraction



Addition of two vectors

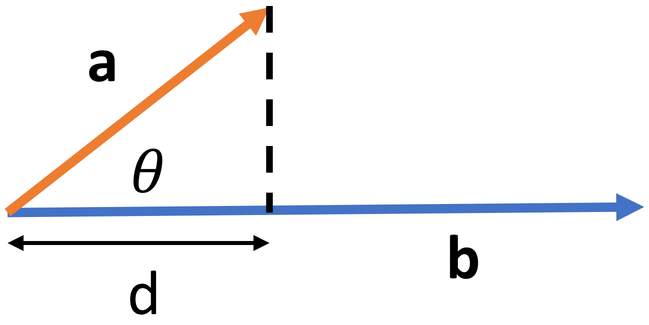


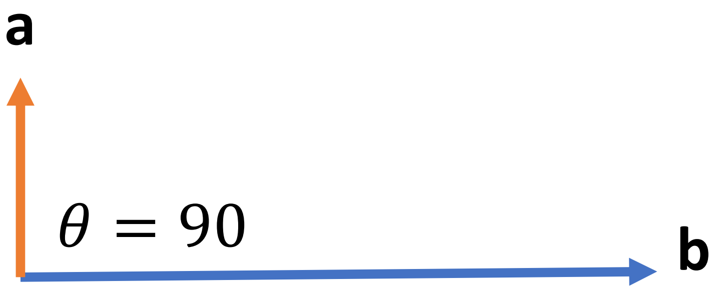
Subtraction of two vectors

Vectors

Operations: Multiplication

- Dot Product





$\mathbf{a} \bullet \mathbf{b} = a_x b_x + \mathrm{ay} b_y + \mathrm{az} b_z$

$\mathrm{d} = (\mathbf{a} \bullet \mathbf{b})/|\mathbf{b}|$

$\mathbf{a} \bullet \mathbf{b} = |\mathbf{a}|\,|\mathbf{b}|\cos\theta$

$\mathbf{a} \bullet \mathbf{b} = 0$

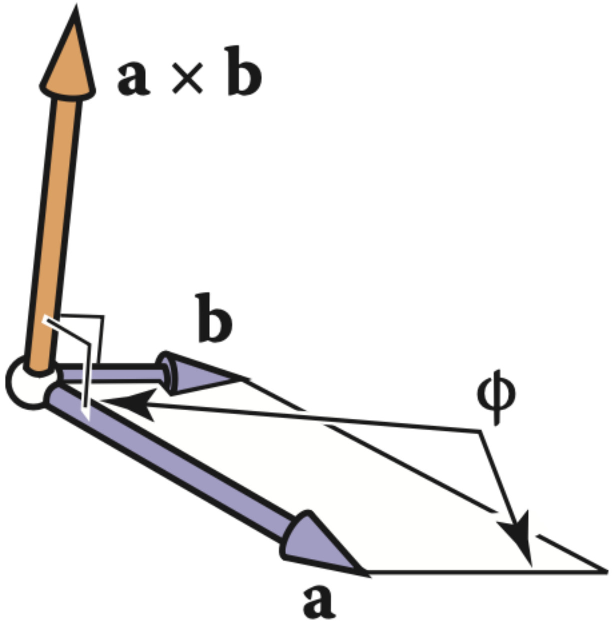
$\mathbf{a} \bullet \mathbf{b} = \mathbf{b} \bullet \mathbf{a}$

$\mathrm{k}\mathbf{a} \bullet \mathbf{b} = \mathbf{a} \bullet \mathrm{k}\mathbf{b} = \mathrm{k}(\mathbf{a} \bullet \mathbf{b})$

Vectors

Operations: Multiplication

- Cross Product



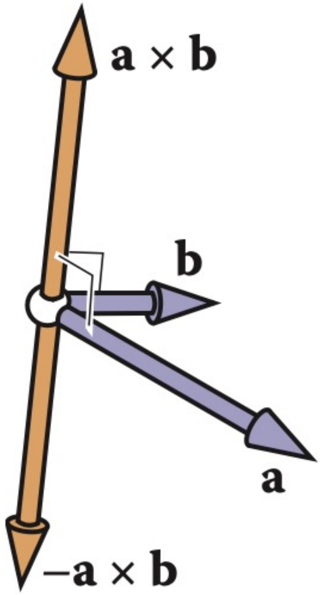
a x **b** = $(y_a z_b - z_a y_b, z_a x_b - x_a z_b, x_a y_b - y_a x_b)$

|**a** x **b**| = |**a**||**b**|$sin\emptyset$

a x **b** = -(**b** x **a**)

a x (**b** + **c**) = **a** x **b** + **a** x **c**

a x k**b** = k(**a** x **b**)



In 2D, the cross product reflects the area of the parallelogram

In 3D, length is reflected by the area with a notion of direction

Matrices

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Matrices

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

- We will heavily rely on Matrix Vector multiplications
- Matrix matrix multiplication
  - Mostly we will deal with square matrices up to [4x4]

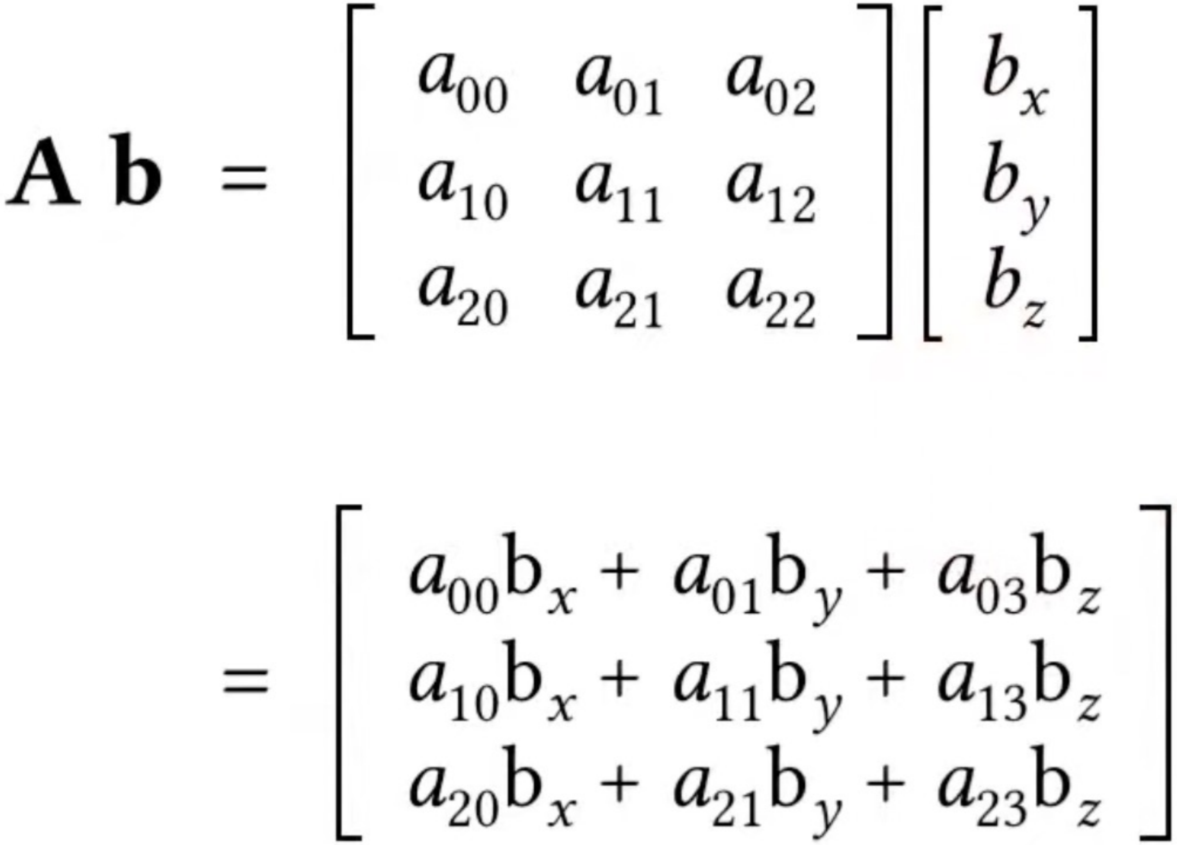
Matrices

- Matrix Vector multiplication

$$\mathbf{A}\,\mathbf{b} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

Matrices

- Matrix Vector multiplication



$$\mathbf{A}\,\mathbf{b} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

$$= \begin{bmatrix} a_{00}\mathrm{b}_x + a_{01}\mathrm{b}_y + a_{03}\mathrm{b}_z \\ a_{10}\mathrm{b}_x + a_{11}\mathrm{b}_y + a_{13}\mathrm{b}_z \\ a_{20}\mathrm{b}_x + a_{21}\mathrm{b}_y + a_{23}\mathrm{b}_z \end{bmatrix}$$

Matrices

- Matrix matrix multiplication

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

Matrices

- Matrix matrix multiplication

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

$$\mathbf{C} = \mathbf{AB} \qquad c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

$$\mathbf{AB} \neq \mathbf{BA}$$

Programming Language and APIs

Programming Language/APIs We Will Use

- **HTML + CSS**
- **JavaScript**

} Language of the Web

Programming Language/APIs We Will Use

- **HTML + CSS**
- **JavaScript**

Language of the Web

- **WebGL 2.0**
- **GLSL Shader Programming**

Graphics APIs + GPU Shader Programming Language

Programming Language/APIs We Will Use

- **HTML + CSS**
- **JavaScript**

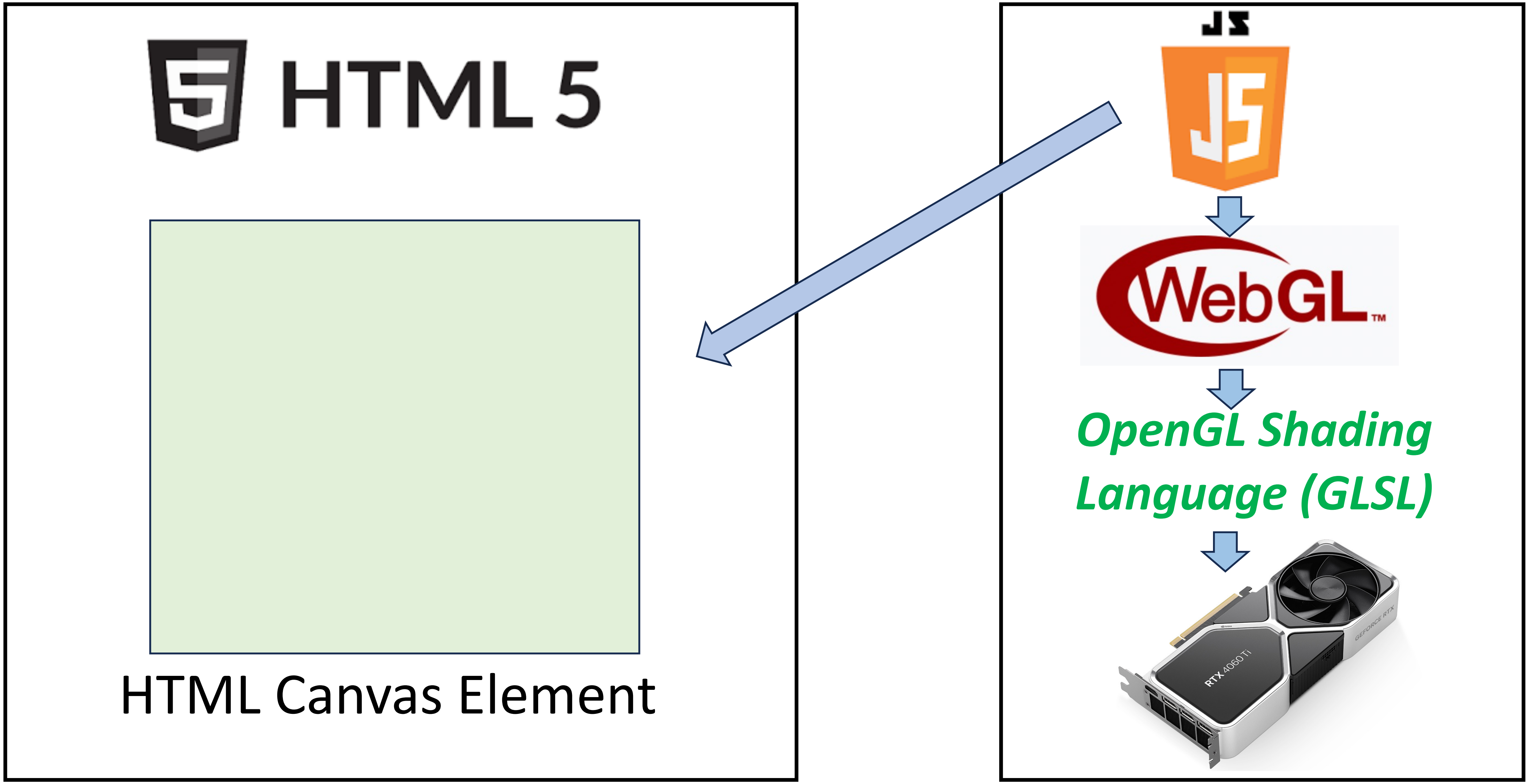
Language of the Web

- **WebGL 2.0**
- **GLSL Shader Programming**

Graphics APIs + GPU Shader Programming Language

- We will combine these to build a 2D/3D graphics rendering framework on the web
  - We will use local server to see the output
  - You can also open your HTML file in a web browser to see the results

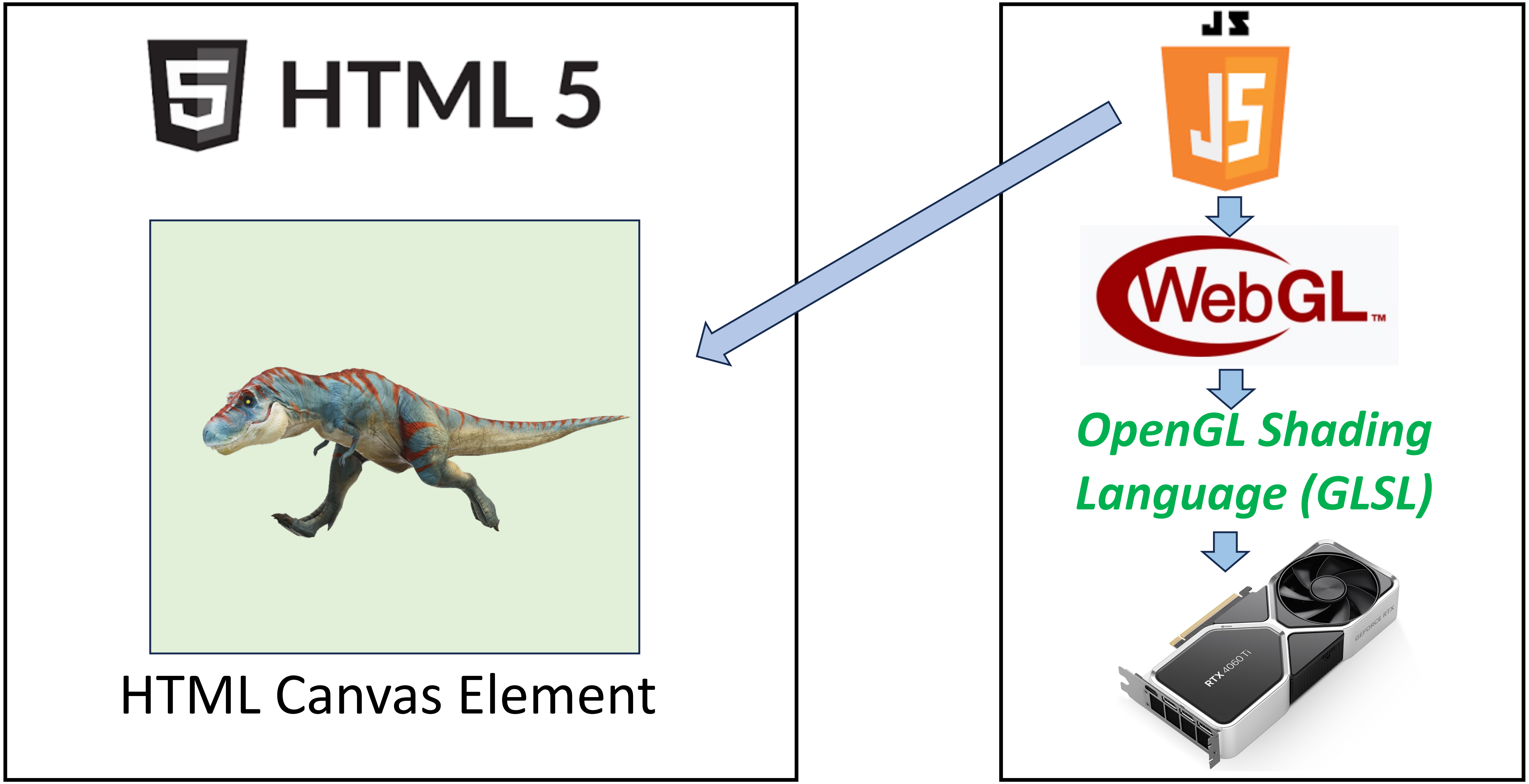
How The Framework Works



HTML Canvas Element

OpenGL Shading Language (GLSL)

How The Framework Works



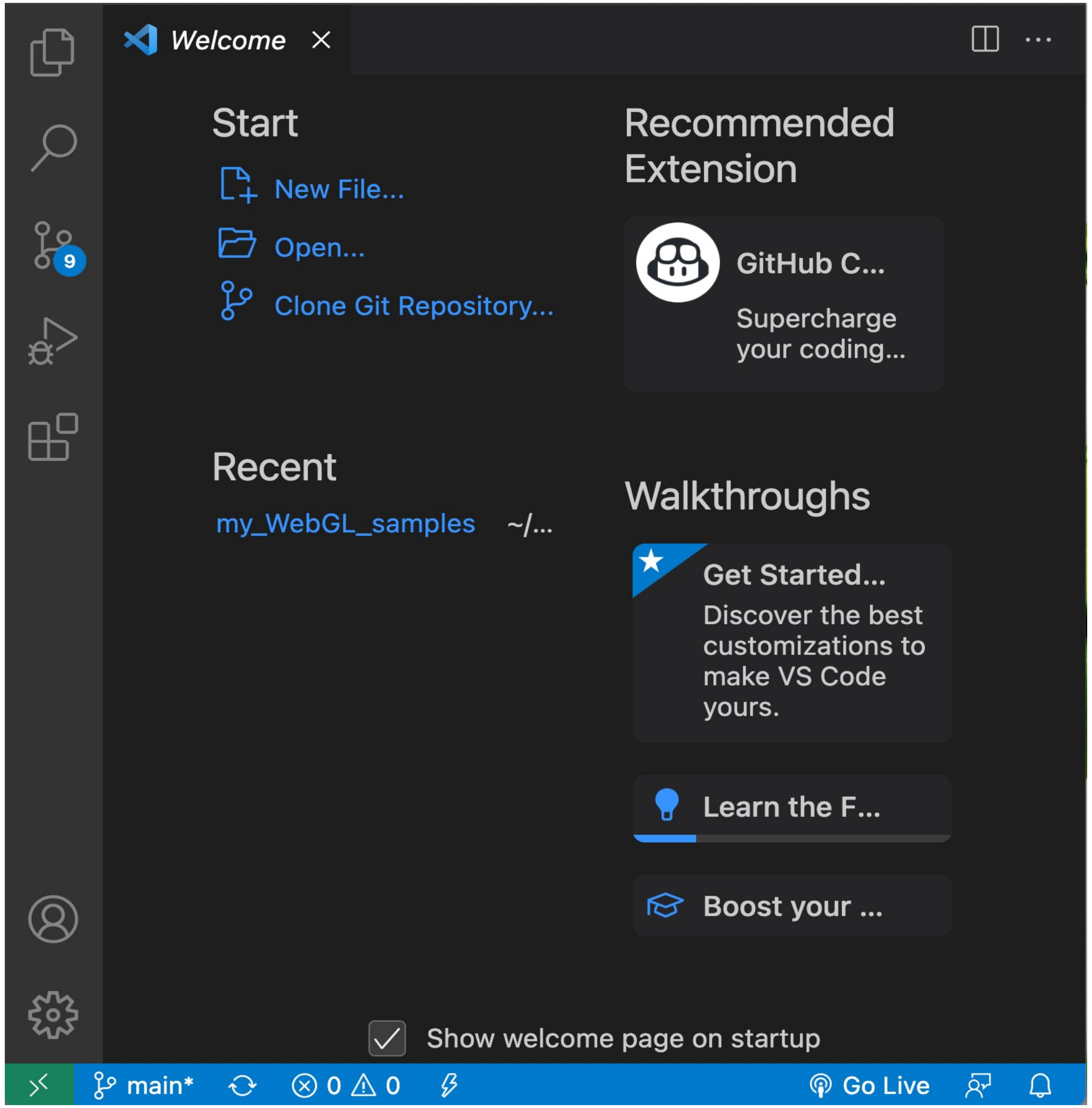
HTML Canvas Element

OpenGL Shading Language (GLSL)

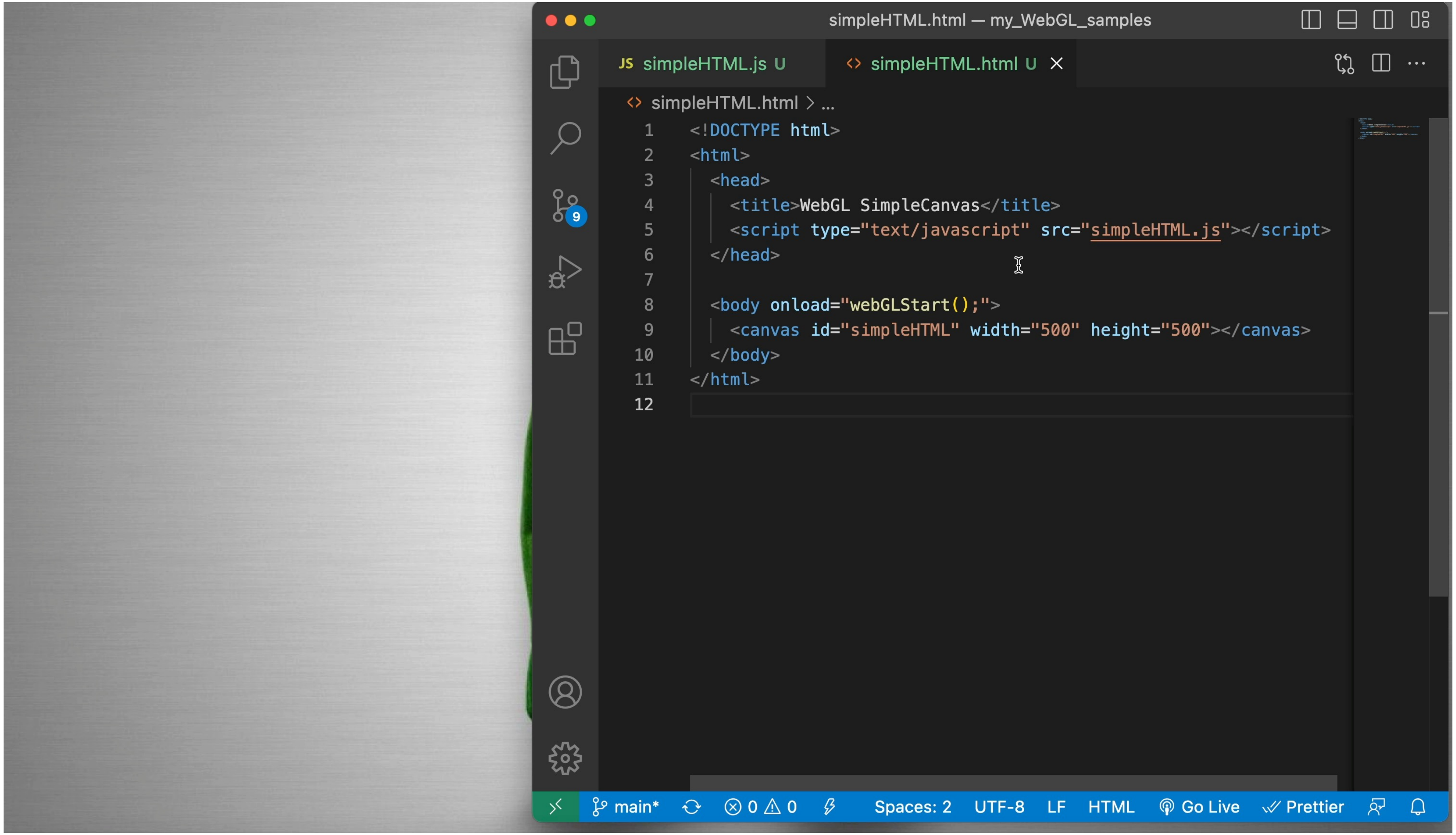
Development Environment Suggestion

- Microsoft Visual Studio Code

- “Live Server” extension
- “Prettier” extension for automatic code formatting



Development Environment Suggestion

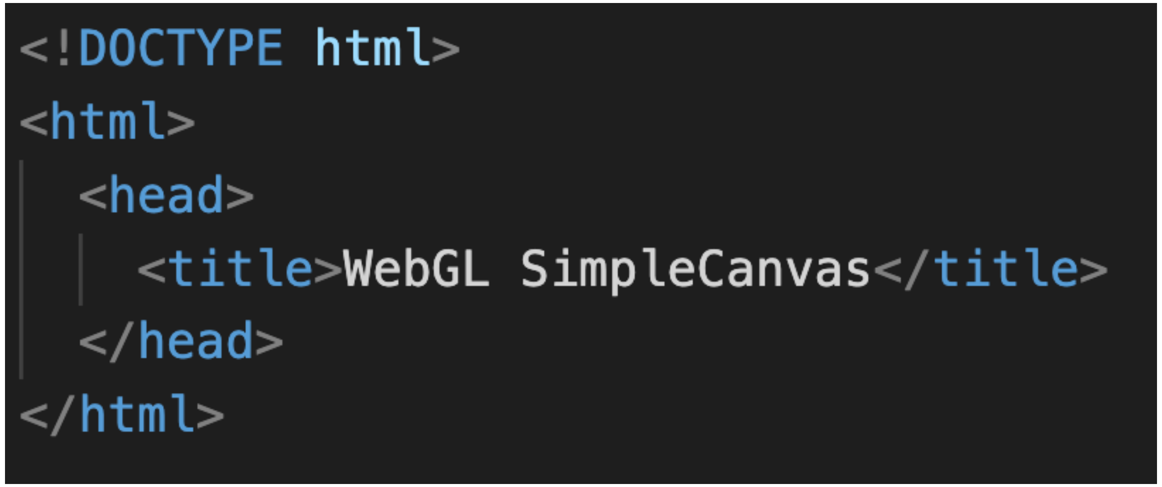


```
<!DOCTYPE html>
<html>
  <head>
    <title>WebGL SimpleCanvas</title>
    <script type="text/javascript" src="simpleHTML.js"></script>
  </head>

  <body onload="webGLStart();">
    <canvas id="simpleHTML" width="500" height="500"></canvas>
  </body>
</html>
```

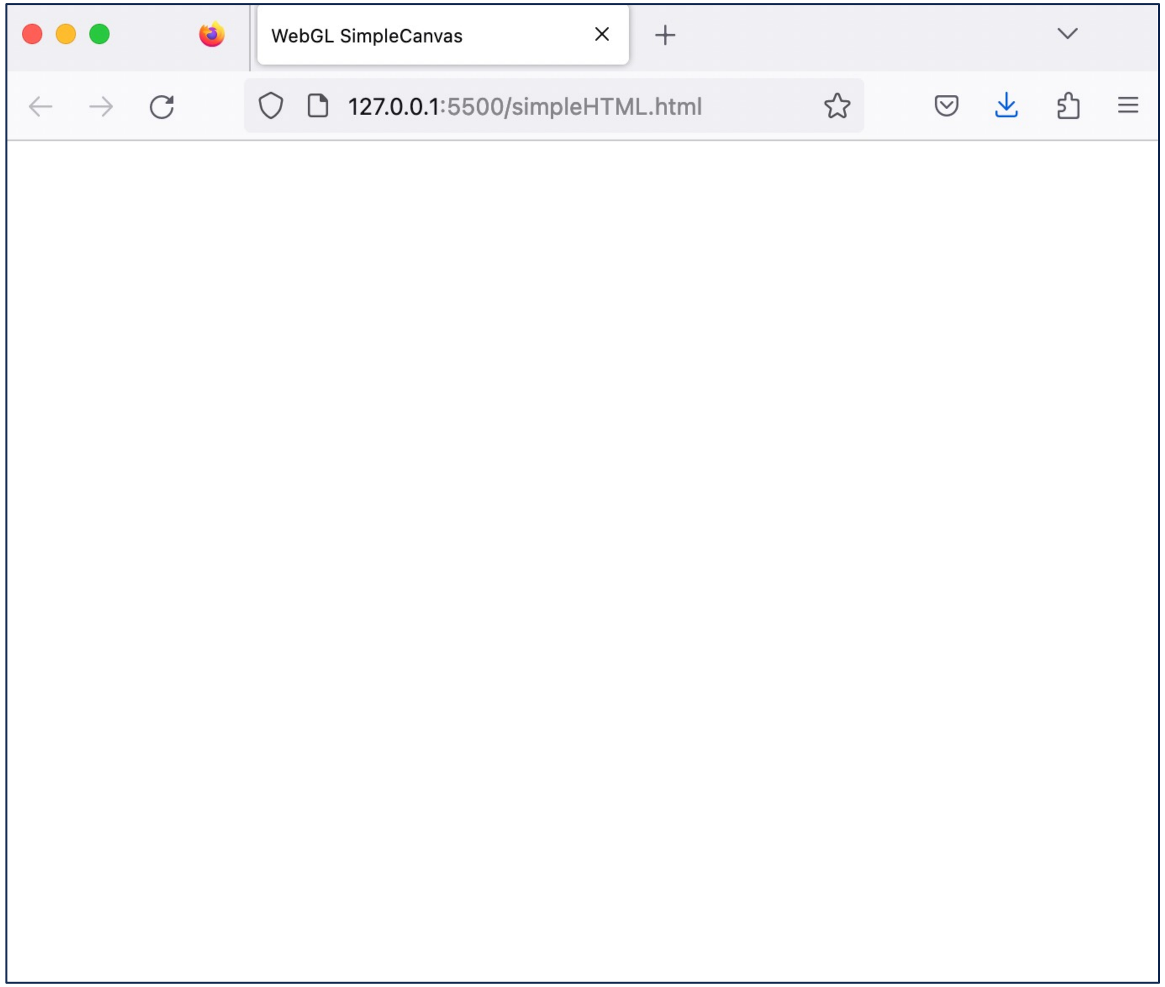
A Minimalistic Overview of HTML

HTML

- Hyper Text Markup Language

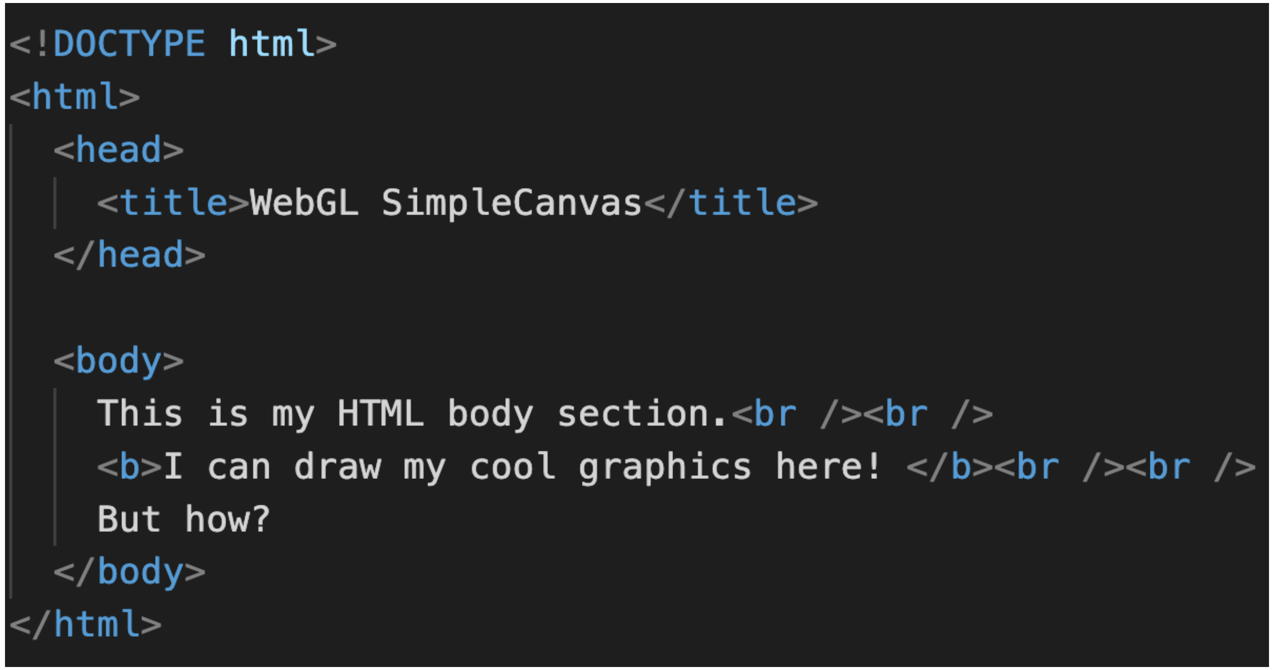


```html
<!DOCTYPE html>
<html>
  <head>
    <title>WebGL SimpleCanvas</title>
  </head>
</html>
```

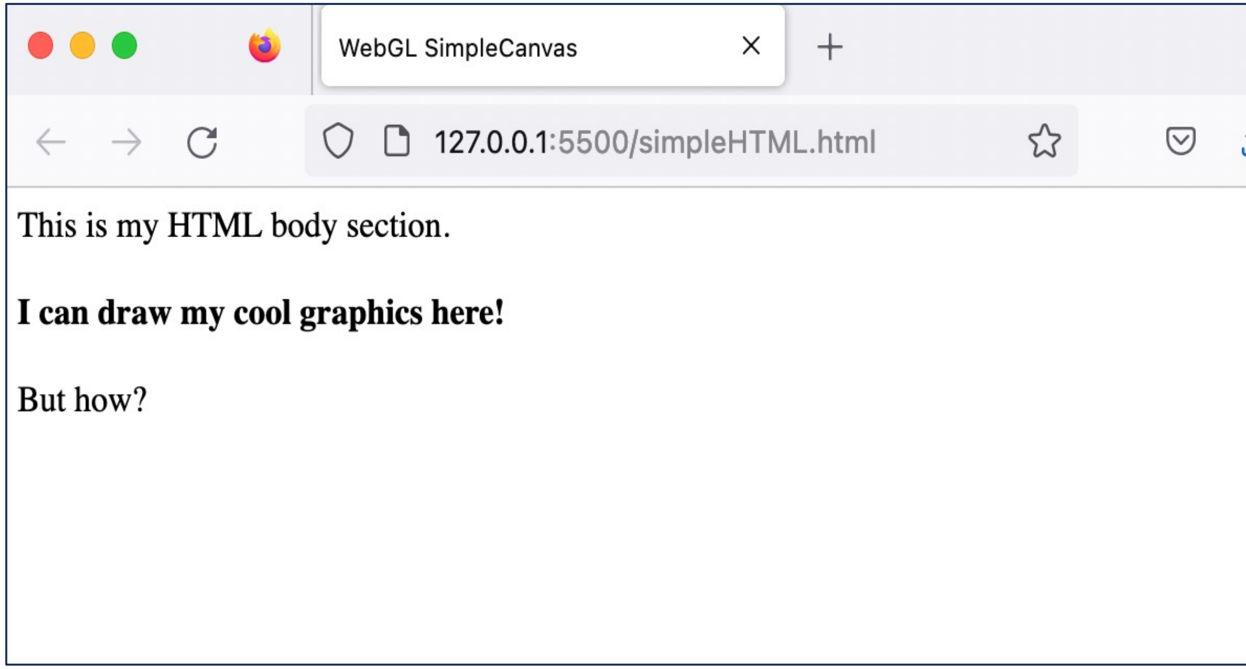


HTML

- Hyper Text Markup Language



```
<!DOCTYPE html>
<html>
  <head>
    <title>WebGL SimpleCanvas</title>
  </head>

  <body>
    This is my HTML body section.<br /><br />
    <b>I can draw my cool graphics here! </b><br /><br />
    But how?
  </body>
</html>
```



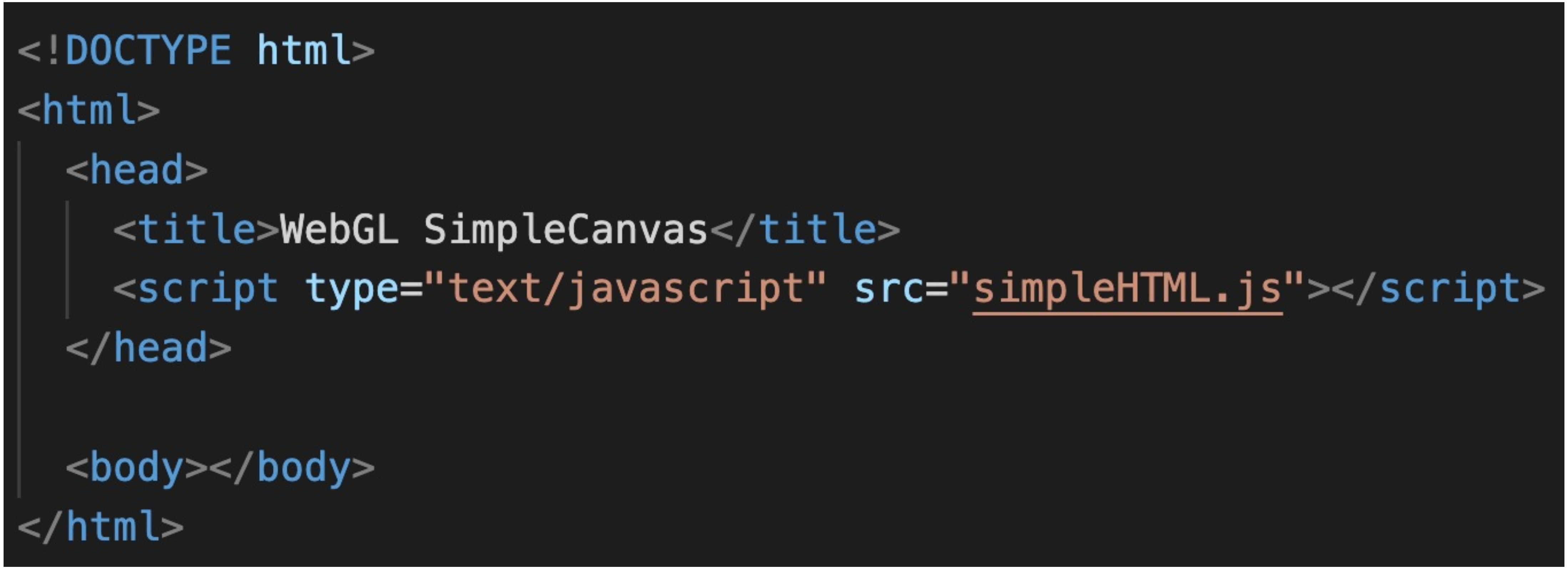
This is my HTML body section.

I can draw my cool graphics here!

But how?

HTML

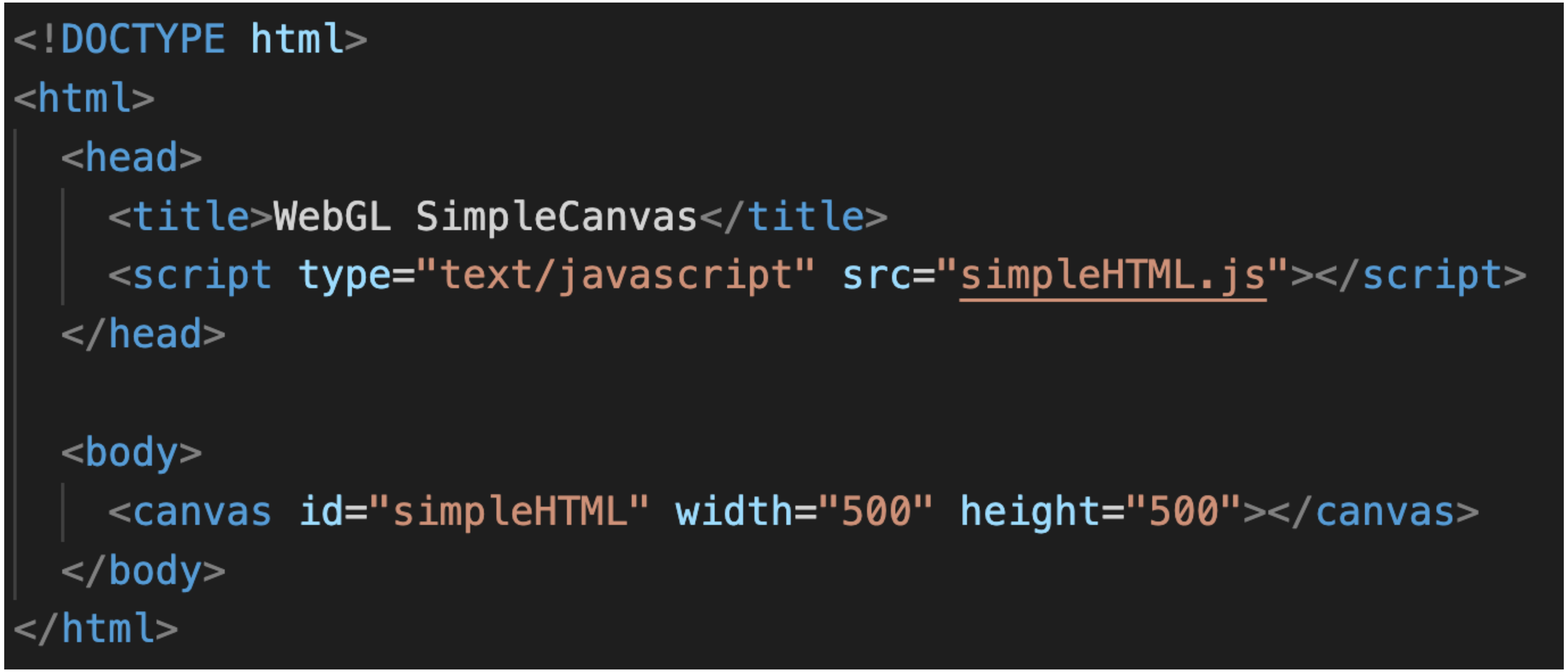
- How We add JavaScript



```
<!DOCTYPE html>
<html>
  <head>
    <title>WebGL SimpleCanvas</title>
    <script type="text/javascript" src="simpleHTML.js"></script>
  </head>

  <body></body>
</html>
```

HTML

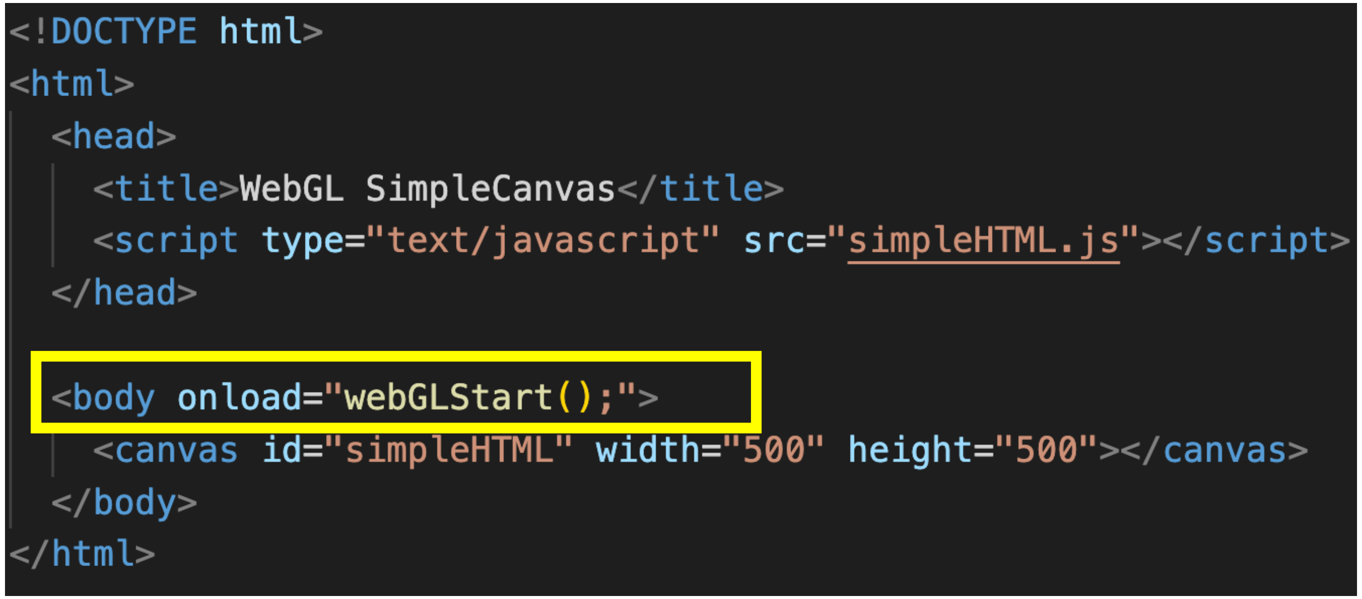
- How We add a Canvas for drawing



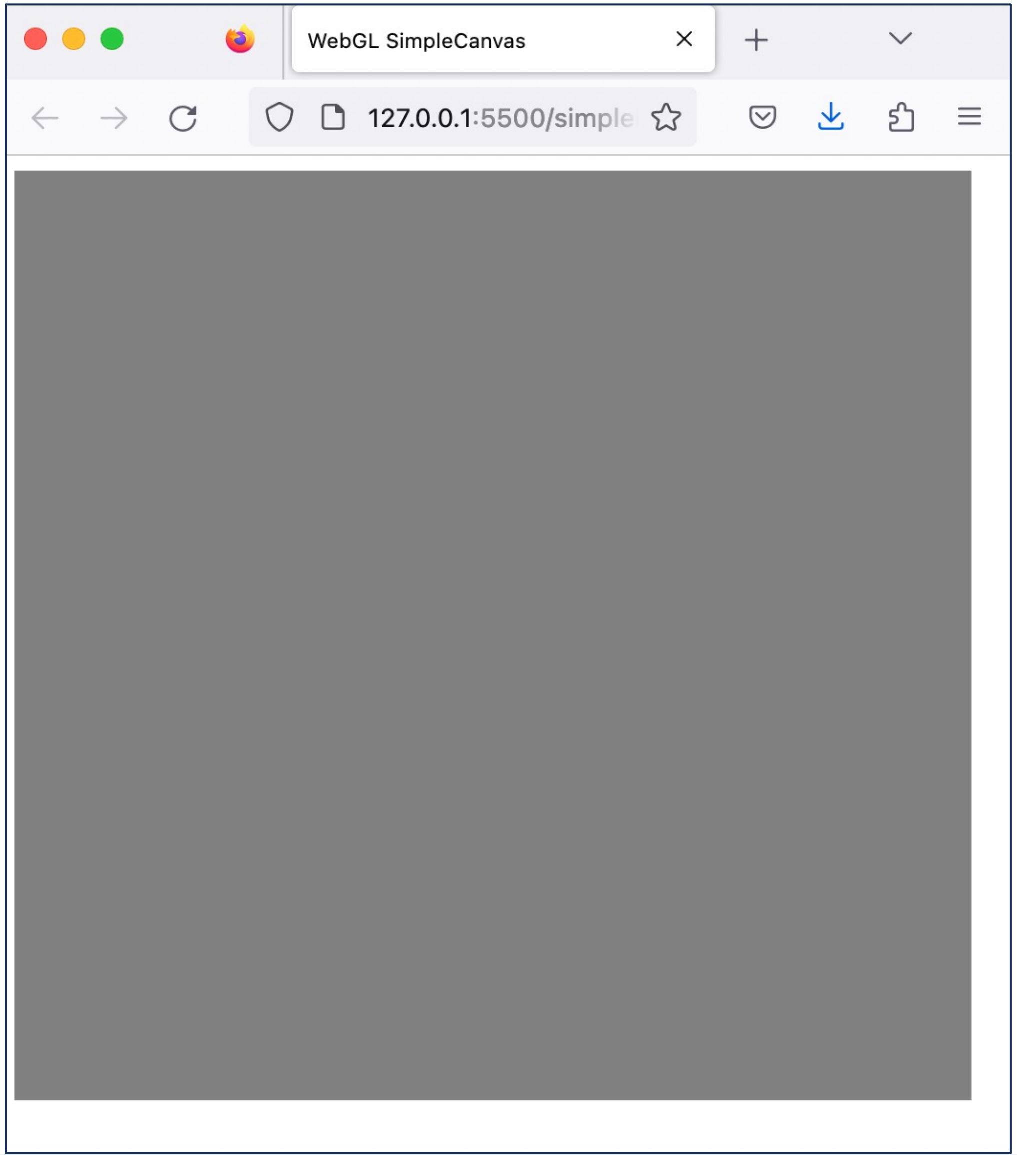
```
<!DOCTYPE html>
<html>
  <head>
    <title>WebGL SimpleCanvas</title>
    <script type="text/javascript" src="simpleHTML.js"></script>
  </head>

  <body>
    <canvas id="simpleHTML" width="500" height="500"></canvas>
  </body>
</html>
```

HTML

- Hyper Text Markup Language

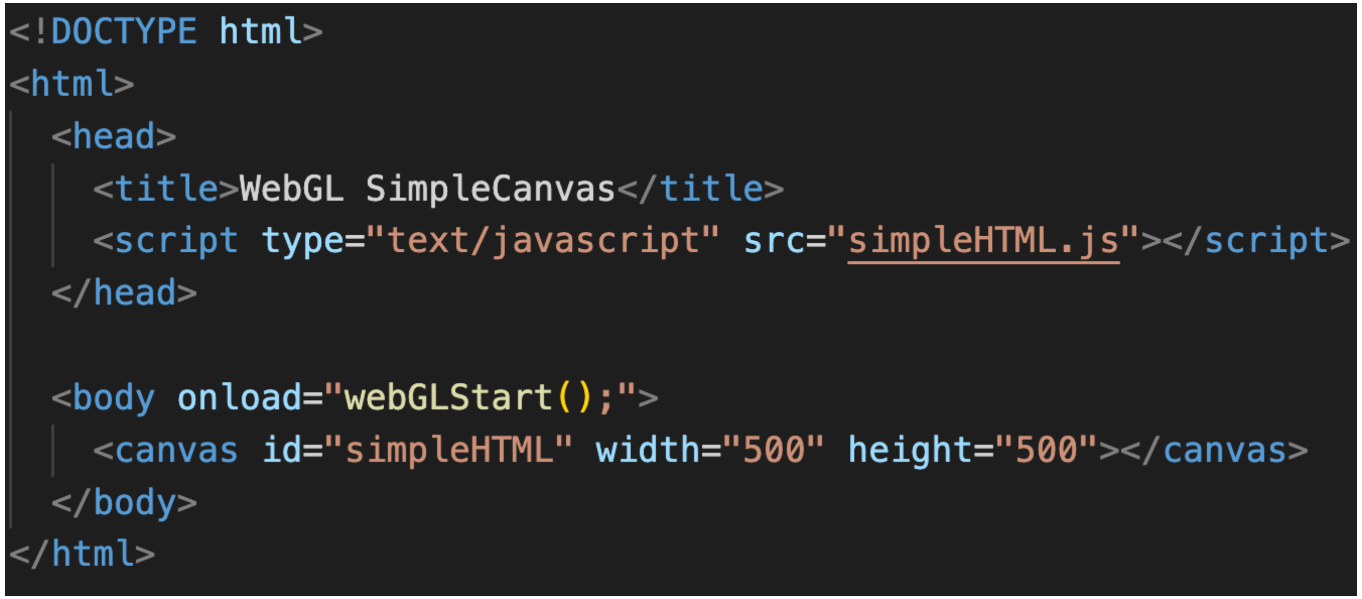


```
<!DOCTYPE html>
<html>
  <head>
    <title>WebGL SimpleCanvas</title>
    <script type="text/javascript" src="simpleHTML.js"></script>
  </head>

  <body onload="webGLStart();">
    <canvas id="simpleHTML" width="500" height="500"></canvas>
  </body>
</html>
```



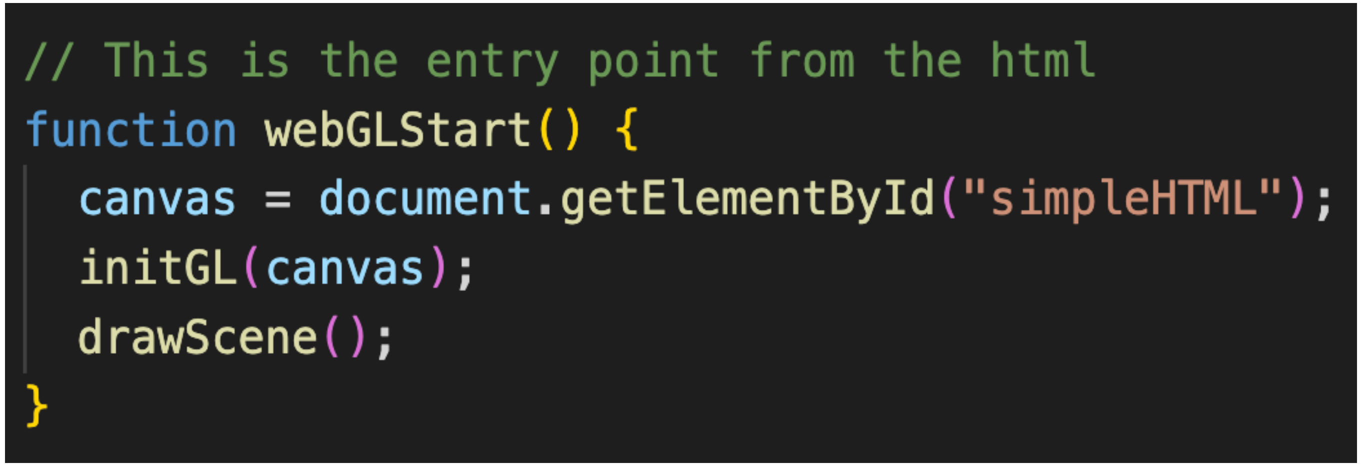
HTML

HTML

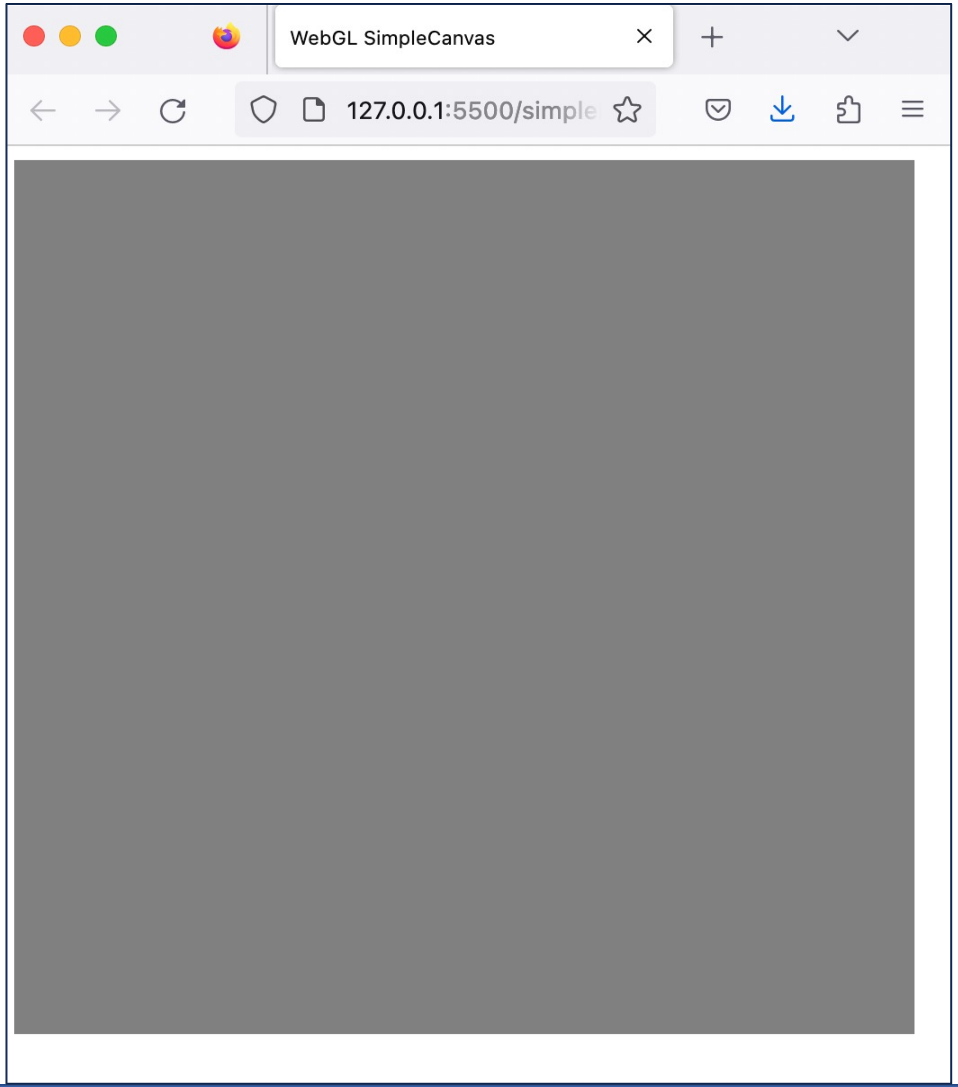


```html
<!DOCTYPE html>
<html>
  <head>
    <title>WebGL SimpleCanvas</title>
    <script type="text/javascript" src="simpleHTML.js"></script>
  </head>

  <body onload="webGLStart();">
    <canvas id="simpleHTML" width="500" height="500"></canvas>
  </body>
</html>
```

JavaScript



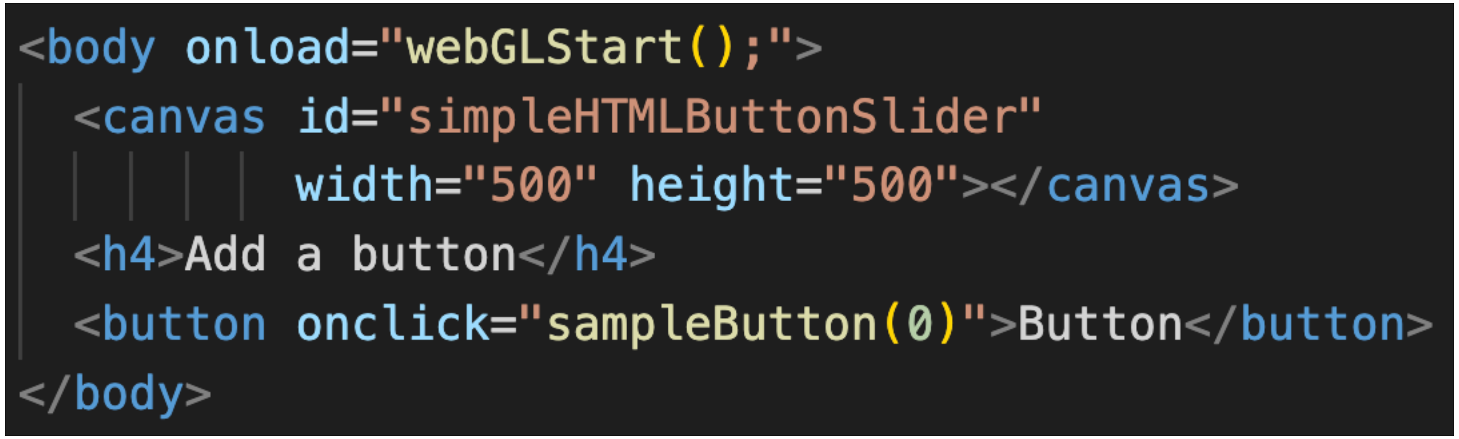
```javascript
// This is the entry point from the html
function webGLStart() {
    canvas = document.getElementById("simpleHTML");
    initGL(canvas);
    drawScene();
}
```



HTML

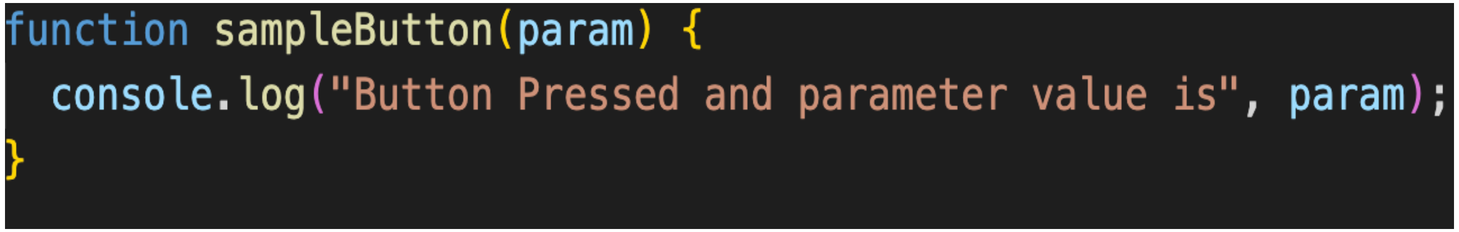
Add a button

HTML

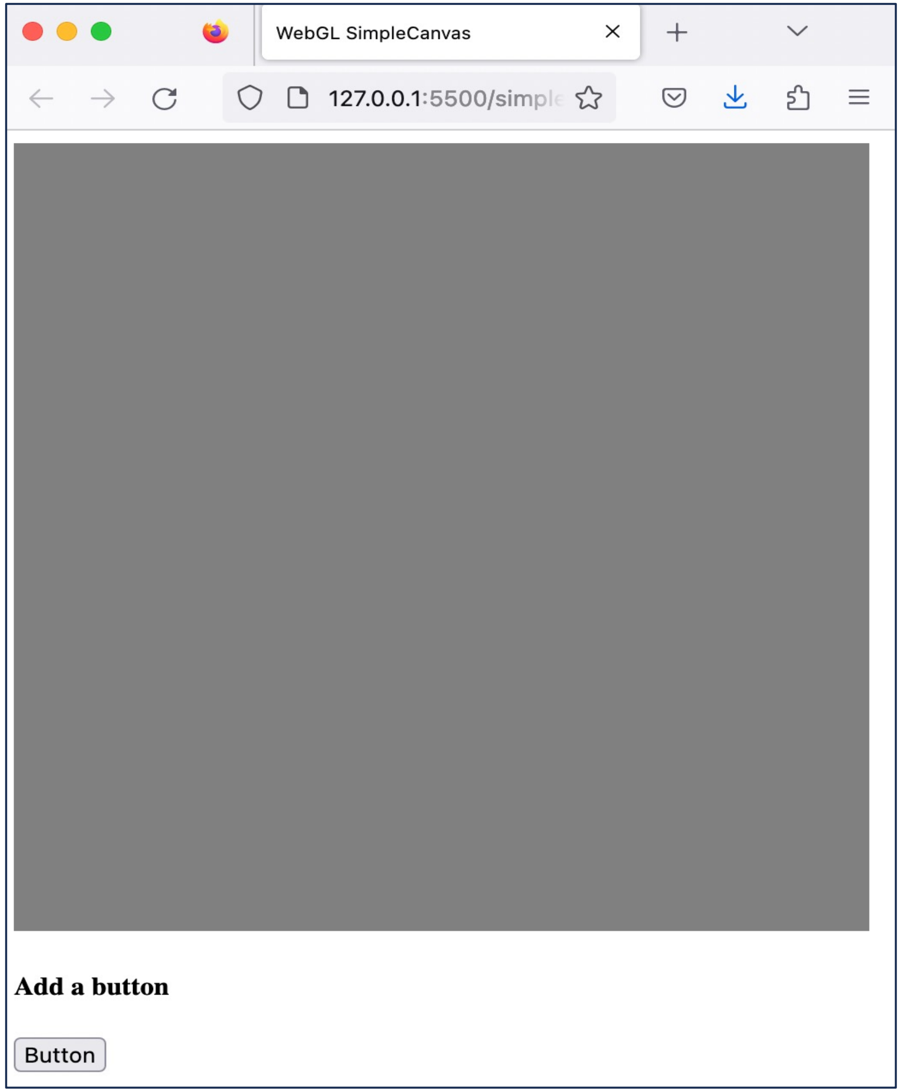


```
<body onload="webGLStart();">
  <canvas id="simpleHTMLButtonSlider"
          width="500" height="500"></canvas>
  <h4>Add a button</h4>
  <button onclick="sampleButton(0)">Button</button>
</body>
```

JavaScript



```
function sampleButton(param) {
  console.log("Button Pressed and parameter value is", param);
}
```

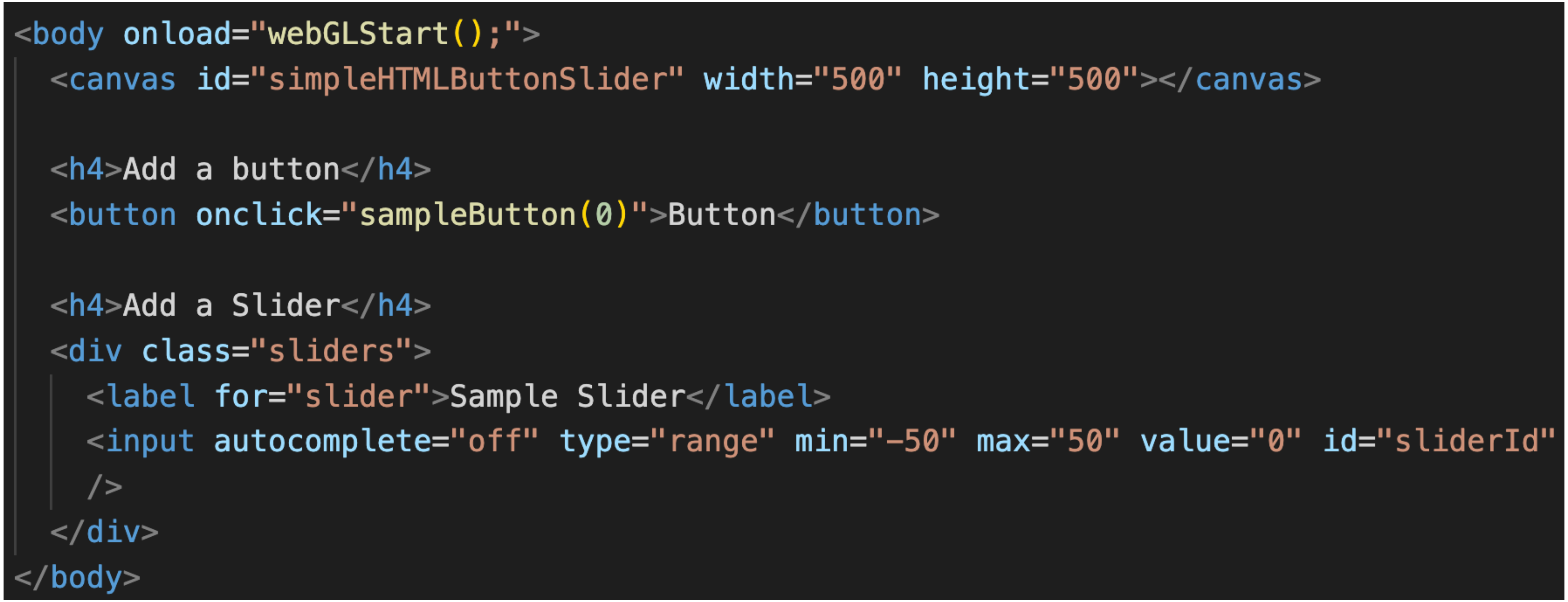


Add a button

HTML

Add a Slider

HTML

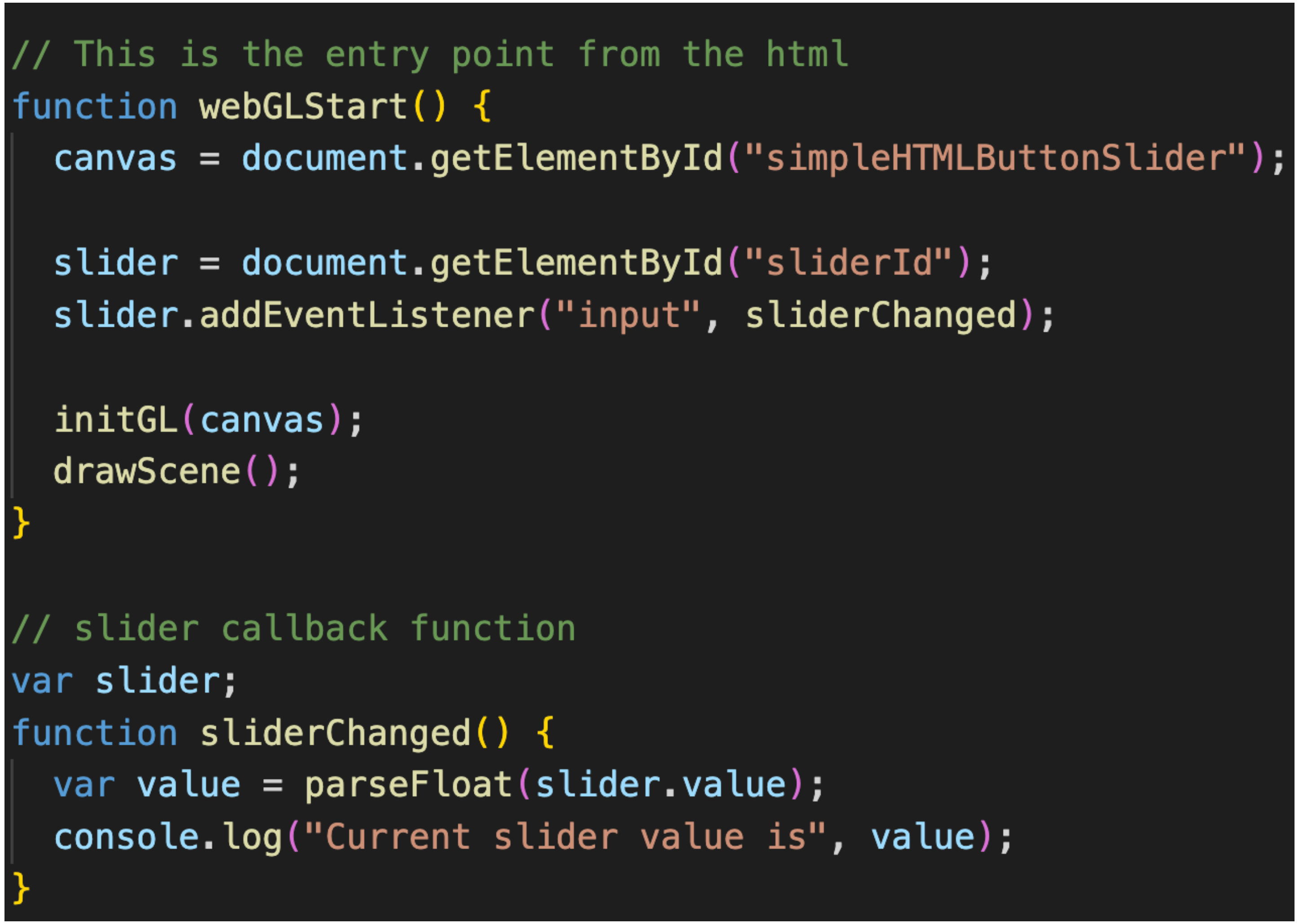


```
<body onload="webGLStart();">
  <canvas id="simpleHTMLButtonSlider" width="500" height="500"></canvas>

  <h4>Add a button</h4>
  <button onclick="sampleButton(0)">Button</button>

  <h4>Add a Slider</h4>
  <div class="sliders">
    <label for="slider">Sample Slider</label>
    <input autocomplete="off" type="range" min="-50" max="50" value="0" id="sliderId"
    />
  </div>
</body>
```

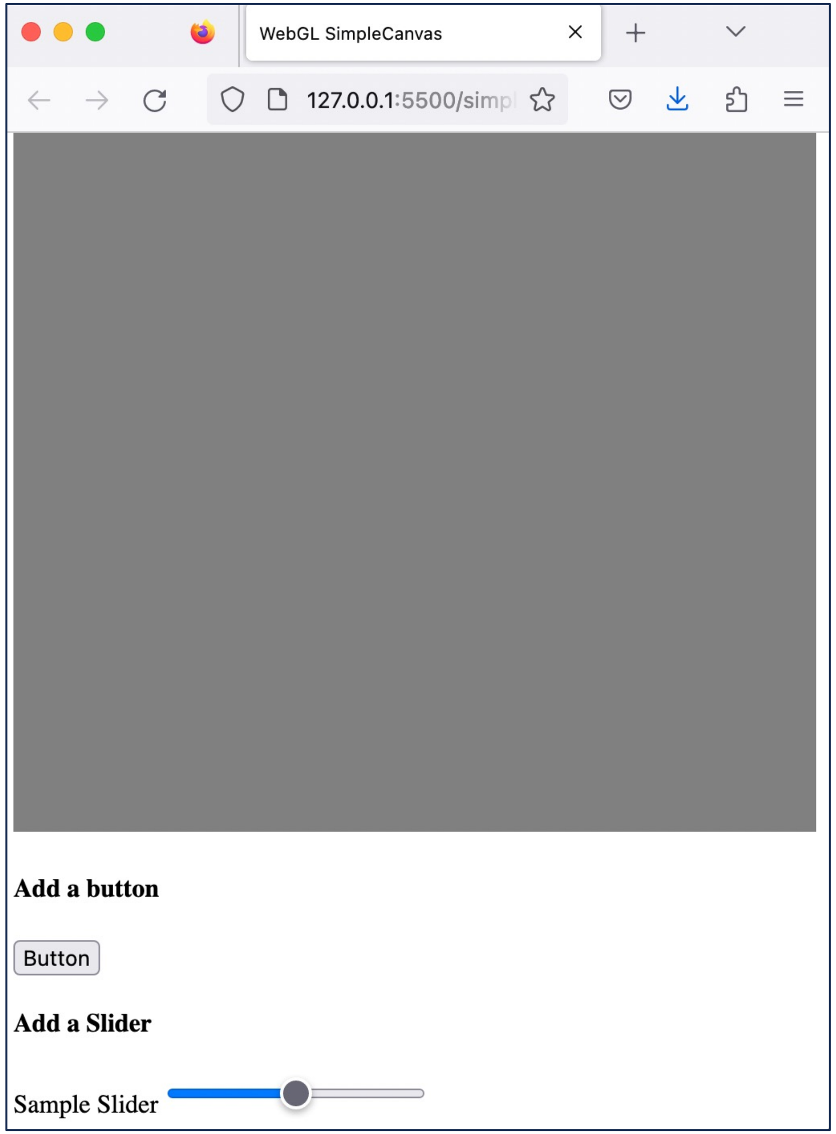
HTML

Add a Slider

JavaScript



```
// This is the entry point from the html
function webGLStart() {
  canvas = document.getElementById("simpleHTMLButtonSlider");

  slider = document.getElementById("sliderId");
  slider.addEventListener("input", sliderChanged);

  initGL(canvas);
  drawScene();
}

// slider callback function
var slider;
function sliderChanged() {
  var value = parseFloat(slider.value);
  console.log("Current slider value is", value);
}
```



Add a button

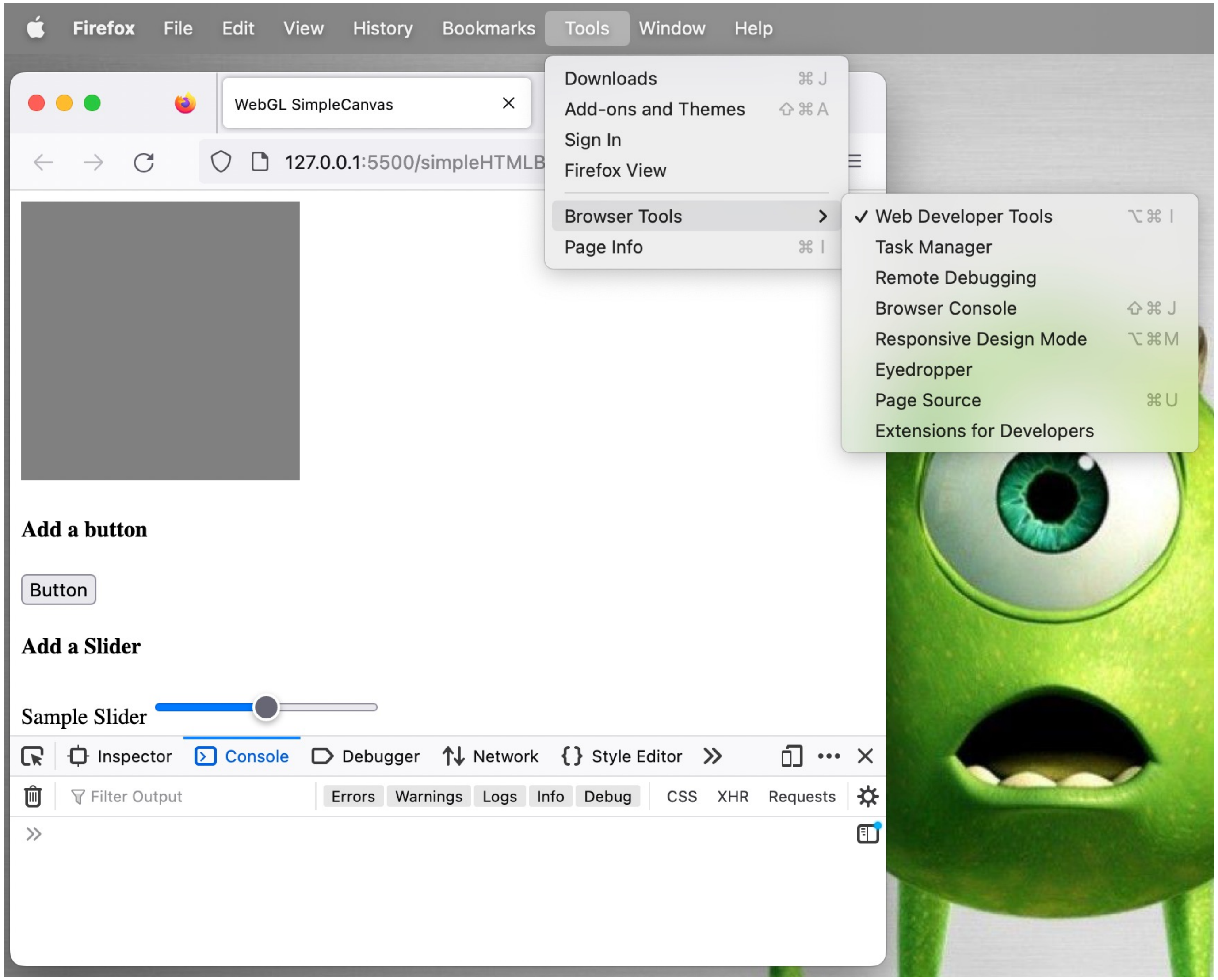
Button

Add a Slider

Sample Slider

HTML: How Do We See Print Output?

Firefox browser



In Google Chrome, View --> Developer → Developer Tools

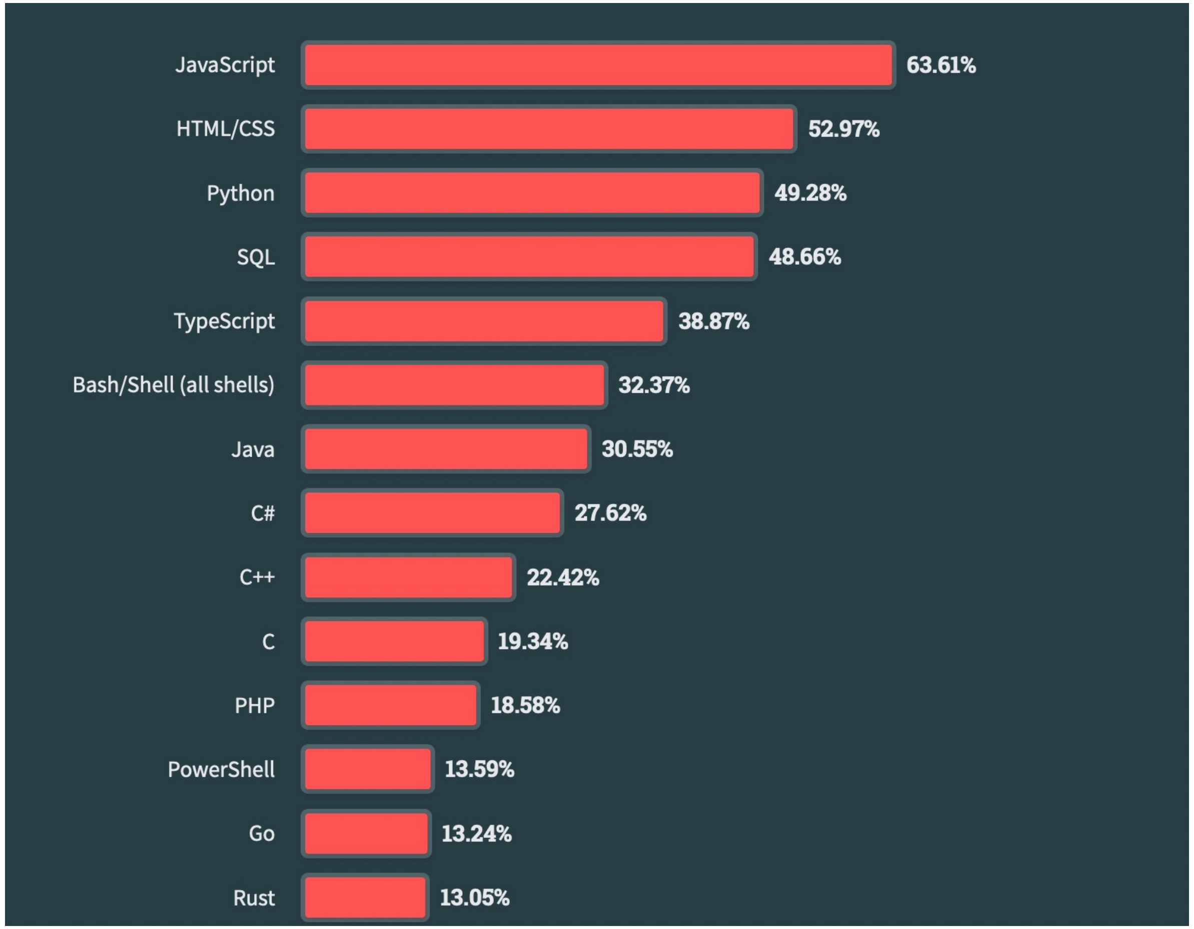
HTML Demo

- simpleHTML.html, simpleHTML.js
- simpleHTMLButtonSlider.html, simpleHTMLButtonSlider.js
- A quick crash course on HTML: https://www.youtube.com/watch?v=qz0aGYrrlhU

A Brief Minimalistic Overview of JavaScript

JavaScript

- Created by Netscape in 1995



https://survey.stackoverflow.co/2023/#most-popular-technologies-language

JavaScript

- JavaScript is the programming language of the Web
- You will primarily use JavaScript as the base language for your assignments/projects
- You will use JavaScript combined with WebGL APIs to control and command GPU for rendering scenes
- Code that will be executed in GPU will be written using another programming language called OpenGL Shading Language (GLSL)
  - GLSL code is very very very similar to C code
- We will only need basic functionalities of JavaScript in this course
  - Fancy/complicated features are not required

JavaScript

- A client-side scripting language
  - Client is the browser
- Interpreted on-the-fly by the client, not compiled
  - Each line is processed as it loads in the browser
- Make interactive webpages
  - Operates on DOM (Document Object Model)
- Follows ECMAScript
  - A standard for scripting languages
  - Current version ECMAScript 2022

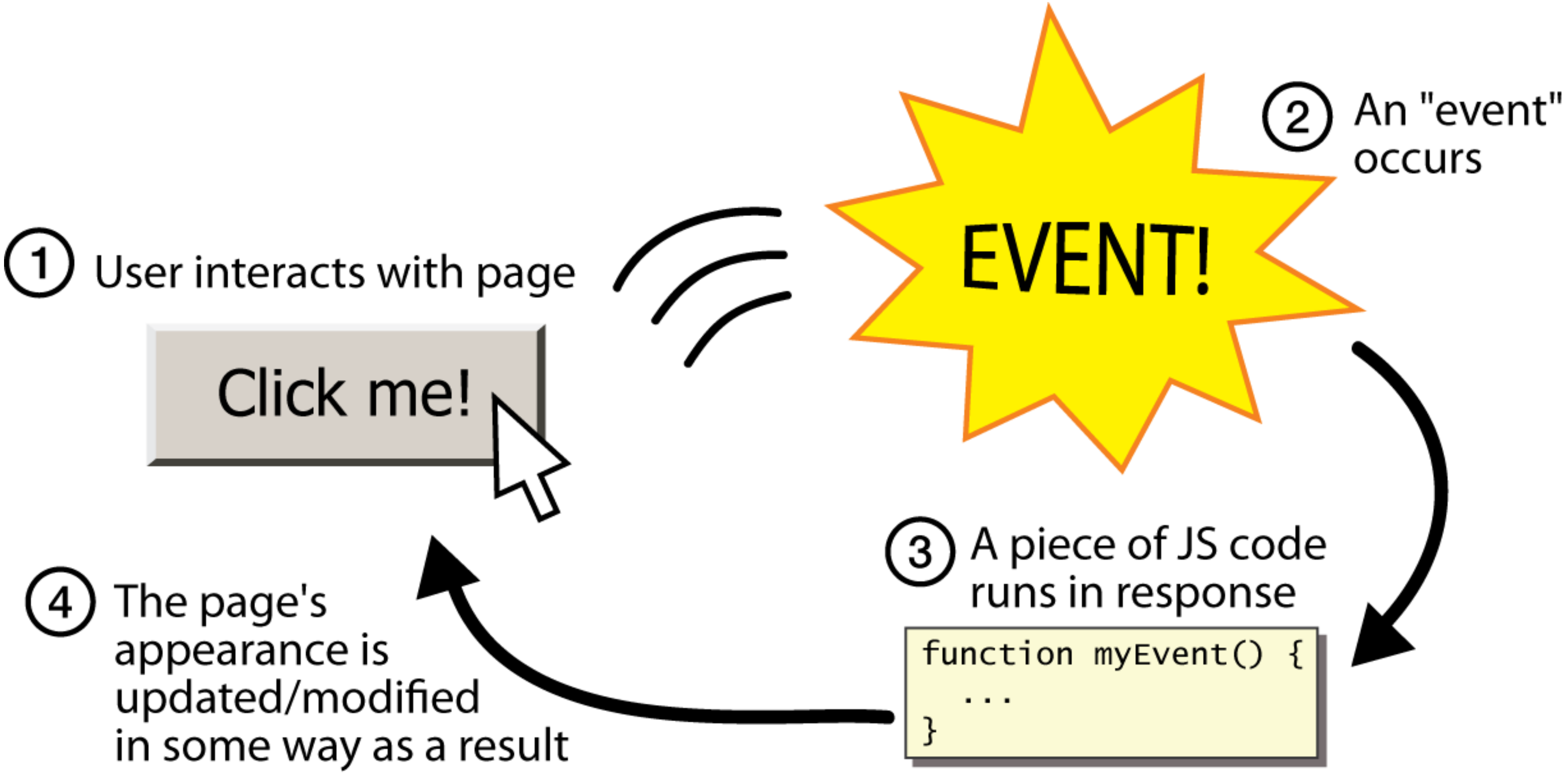
https://survey.stackoverflow.co/2023/#most-popular-technologies-language

JavaScript

- We can run JavaScript outside of browser as well, which makes it very powerful and versatile
- You need to use Node.js
  - Cross platform opensource server environment
  - A back-end JavaScript runtime environment
  - Runs on the Chrome V8 JavaScript Engine
  - Not required for this class

https://survey.stackoverflow.co/2023/#most-popular-technologies-language

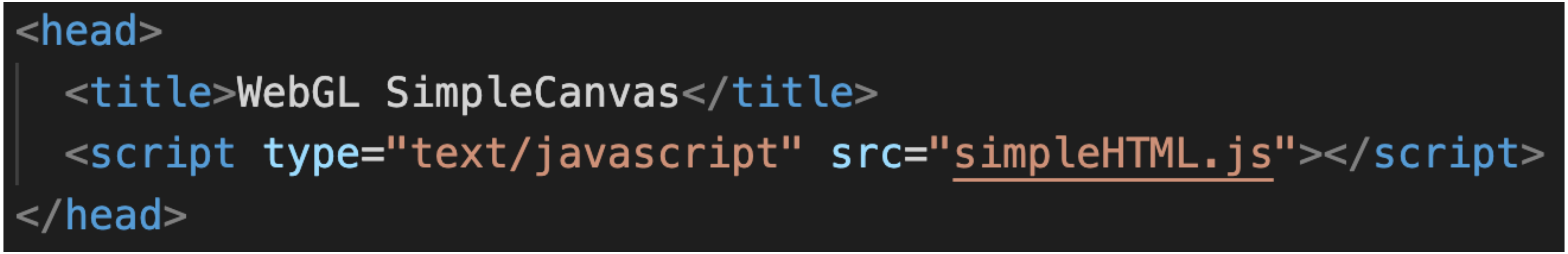
Event Driven Programming


① User interacts with page
Click me!
EVENT!
② An "event" occurs
③ A piece of JS code runs in response
④ The page's appearance is updated/modified in some way as a result

```
function myEvent() {
  ...
}
```

Including JavaScript in HTML

- Use the *<script>…</script>* tag
- Include the script in an external file and import that file in HTML code



```
<head>
  <title>WebGL SimpleCanvas</title>
  <script type="text/javascript" src="simpleHTML.js"></script>
</head>
```

Comments in JavaScript

- Two types of comments
- Single line
  - Uses two forward slashes (i.e., **//**)
- Multiple line
  - Uses **/*** and ***/**

Prompt Message

- console.log(“show this text in console”);
  - Equivalent to printf in C / cout in C++
- alert(“*open a pop up and show this text*”);

Variables and Types

- Variables are declared with the var keyword (case sensitive)
- var *name* = *expression*;
- Examples:
  - var age = 32;
  - var weight = 127.4;
  - var name = "Connie Client";
- Types are not specified, but JS does have types ("loosely typed")
  - Number, Boolean, String, Array, Object, Function, Null, Undefined
  - Can find the type of a variable by calling “typeof”

Variables: Number Type

- var enrollment = 99;
- var grade = 2.8;
- var credits = 5 + 4 + (2 * 3);
- Integers and real numbers are the same type
  - No int vs. float vs. double
- Operators: + - * / % ++ -- = += -= *= /= %=

For Loop

```
for (initialization; condition; update) {
        statements;
}
```

- Example:

```
var sum = 0;
for (var i = 0; i < 100; i++) {
        sum = sum + i;
}
```

Logical Operators

> < >= <= && || ! == != === !==

- Most logical operators automatically convert types:
  - 5 < "7" is true
  - 42 == 42.0 is true
  - "5.0" == 5 is true

- === , !== are strict equality tests; checks type *and* value
  - "5.0" === 5 is false

If/Else Conditional Statements

```
if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

While Loops

```
while (condition) {
    statements;
}

do {
    statements;
} while (condition);
```

- Can use break and continue keywords to go out of the loop

Objects

- Objects are a collection of properties
  - each has a name and a specific value
- Often used to structure information compactly
- We can use dot notation to access properties of an object
  - Obj.prop_name

Math Object in JavaScript

Math.E	*e*, base of natural logarithms: 2.718...
Math.LN10, Math.LN2, Math.LOG2E, Math.LOG10E	natural logarithm of 10 and 2; logarithm of *e* in base 2 and base 10
Math.PI	π, circle's circumference/diameter: 3.14159...
Math.SQRT1_2, Math.SQRT2	square roots of $^1/_2$ and 2
Math.abs(***n***)	absolute value
Math.acos/asin/atan(***n***)	arc-sin/cosine/tangent of angle in radians
Math.ceil(***n***)	ceiling (rounds a real number up)
Math.cos/sin/tan(***n***)	sin/cosine/tangent of angle in radians
Math.exp(***n***)	e^n, *e* raised to the *n*th power
Math.floor(***n***)	floor (rounds a real number down)
Math.log(***n***)	natural logarithm (base *e*)
Math.max/min(***a***, ***b***...)	largest/smallest of 2 or more numbers
Math.pow(***x***, ***y***)	x^y, *x* raised to the *y*th power
Math.random()	random real number *k* in range $0 \le k < 1$
Math.round(***n***)	round number to nearest whole number
Math.sqrt(***n***)	square root

Arrays

- var *name* = []; // empty array
- var *name* = [*value*, *value*, ..., *value*]; // pre-filled
- *name*[*index*] = *value*; // store element
- Array serves as many data structures such as **list, queue, stack**, ...

Functions

function ***name****(****paramName***$_1$*, ..., **paramName***$_n$*) {*

    statements*;*

}

- Example

```
function myFunction(name) {
    console.log(“this is a print statement in javascript”
   console.log(“you can see the output in console of javascript in browser”
}
```

- Functions are ***first-class*** citizens in JavaScript!
    - Can be stored as variables, passed as parameters, returned, ...

KeyWords in JavaScript

```
break   case     catch  continue debugger
default delete   do     else     finally
for     function if     in       instanceof
new     return   switch this     throw
try     typeof   var    void     while
with
```