JavaScript and WebGL

CS425: Computer Graphics I

Khairi Reda

Administrativia

- Sign up for Piazza! (Link/code posted on Blackboard)
- Only grades will be posted on Blackboard (most likely)
- We will still use **go.uic.edu/CS425** for slides/schedule/syllabus
- We will be using **GitHub Classroom** for assignments
  - I'll post assignment "invitation" in Piazza
  - Make sure you have a GitHub account ready in advance
  - Get familiar with using Git if you haven't already

Administrativia

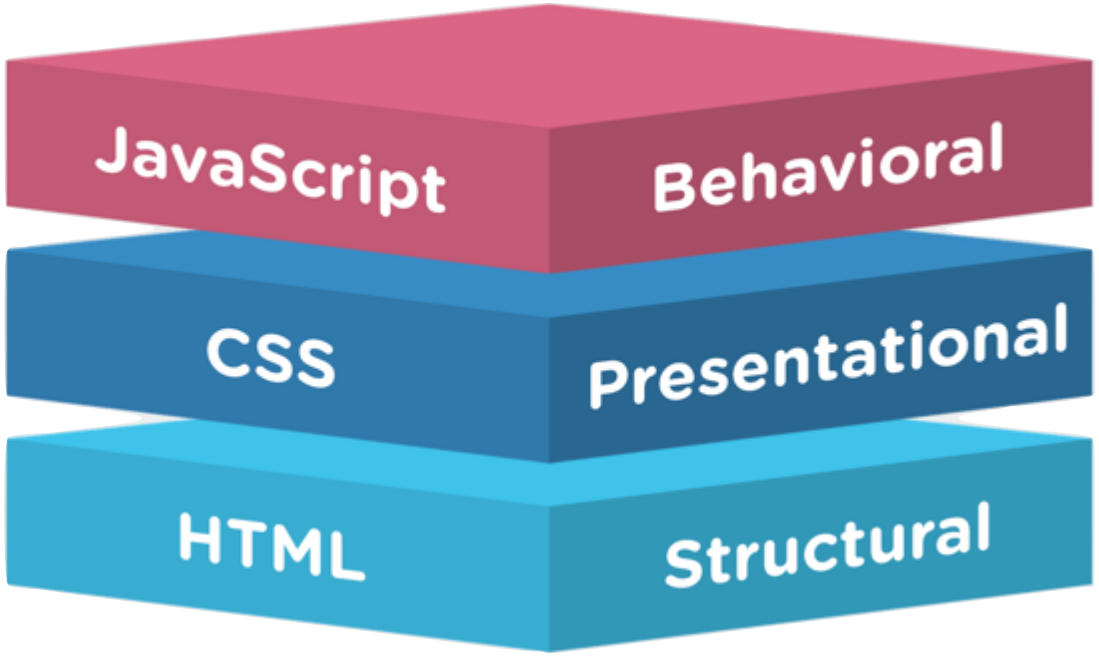
- For today, we will be working of examples under lab1 (Piazza -> Resources -> Lab1)
- Hands on exercises to get you started with JavaScript
- Culminates in setting up an environment for WebGL

Overview

- Web technologies
- Web environment
- JavaScript
- WebGL

Web technologies

Web technologies



- **JavaScript**: manage user interaction with the structure and presentation.
- **CSS**: manage presentation
- **HTML5**: mark-up language to structure the content of web pages.

HTML: First example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>First example</title>
</head>
<body>

Content here

</body>
</html>
```

Elements:
<element>
</element>

Examples:
html, head, body

HTML: First example

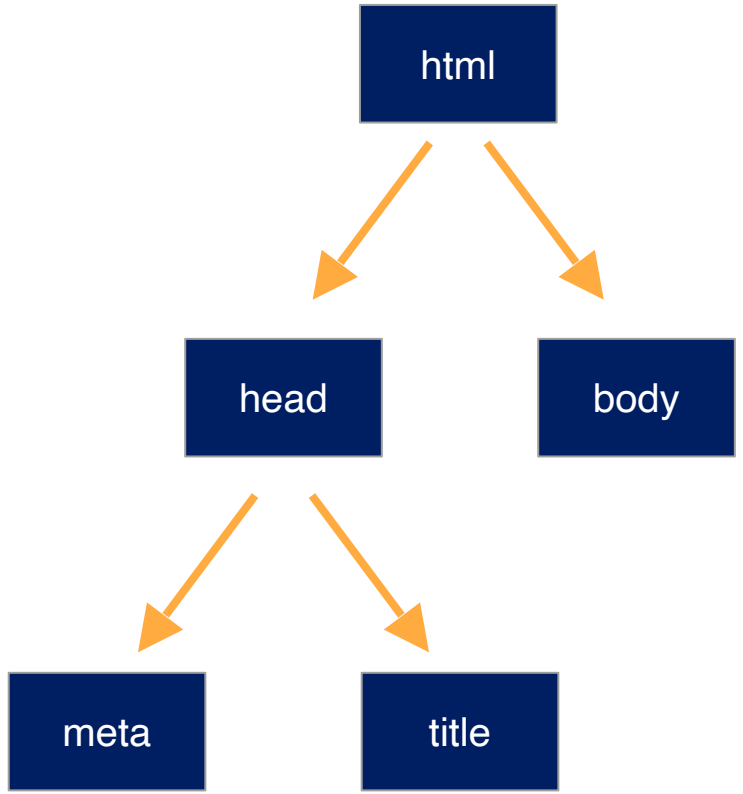
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>First example</title>
</head>
<body>

Content here

</body>
</html>
```

Attributes:
<… attribute=“”>

Examples:
lang, charset, id

HTML: First example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>First example</title>
</head>
<body>

Content here

</body>
</html>
```



HTML: Elements and attributes

Main root: <html>

Metadata: <head>, <link>, <meta>, <style>, <title>

Root body: <body>

Text content: , <figure>, <hr>,

Image and multimedia: <**canvas**>, <img>, <audio>, …, <video>

Scripting: <**script**>

…

Cascading Style Sheets (CSS)

```
body {
    margin: 10px;
    font-size: 20px;
}
```

Selectors:
selector {
    …
}

Examples:
body, #first, .special

Cascading Style Sheets (CSS)

Different selectors

```
h1, h2   {
     ...
}

.classe p {
     ...
}
```

```
div > p   {
     ...
}

#id1 ~ #id2 {
     ...
}
```

```
div[attr]   {
     ...
}

div[attr="val"] {
     ...
}
```

Cascading Style Sheets (CSS)

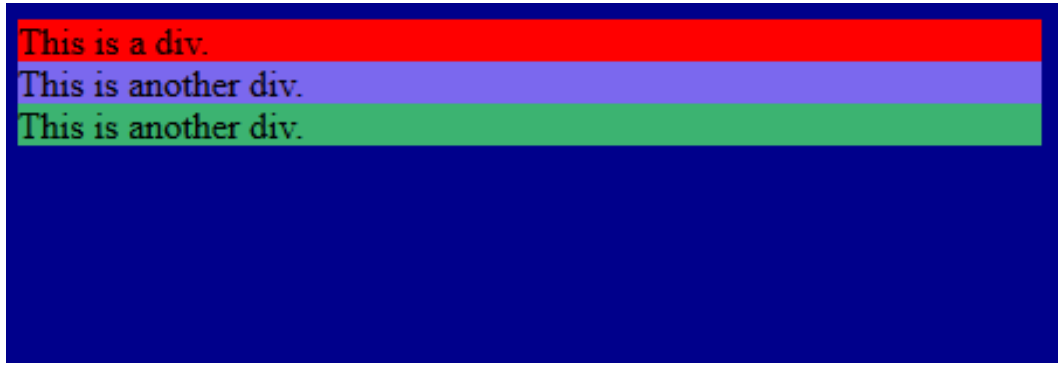
```
body {
    margin: 10px;
    font-size: 20px;
}
```

Properties:
key: value;

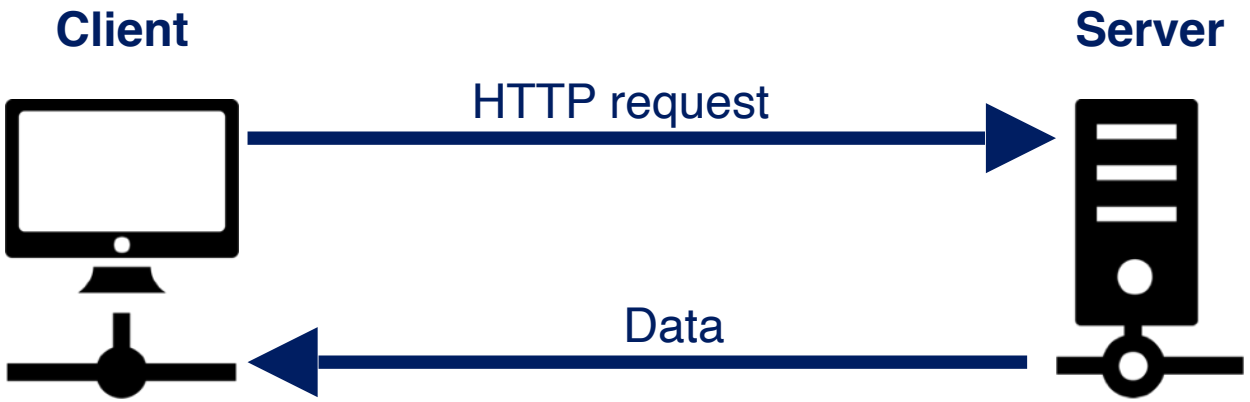
Examples:
margin, color, font-size

Putting it all together

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>First example</title>
</head>
<style>
    body { background-color: darkblue;}
    div { background-color: mediumslateblue;}
    #mydiv { background-color: red;}
    .myclass { background-
color: mediumseagreen;}
</style>
<body>
    <div id="mydiv">This is a div.</div>
    <div id="myseconddiv">This is another div.</
div>
    <div class="myclass">This is another div.</
div>
</body>
</html>
```



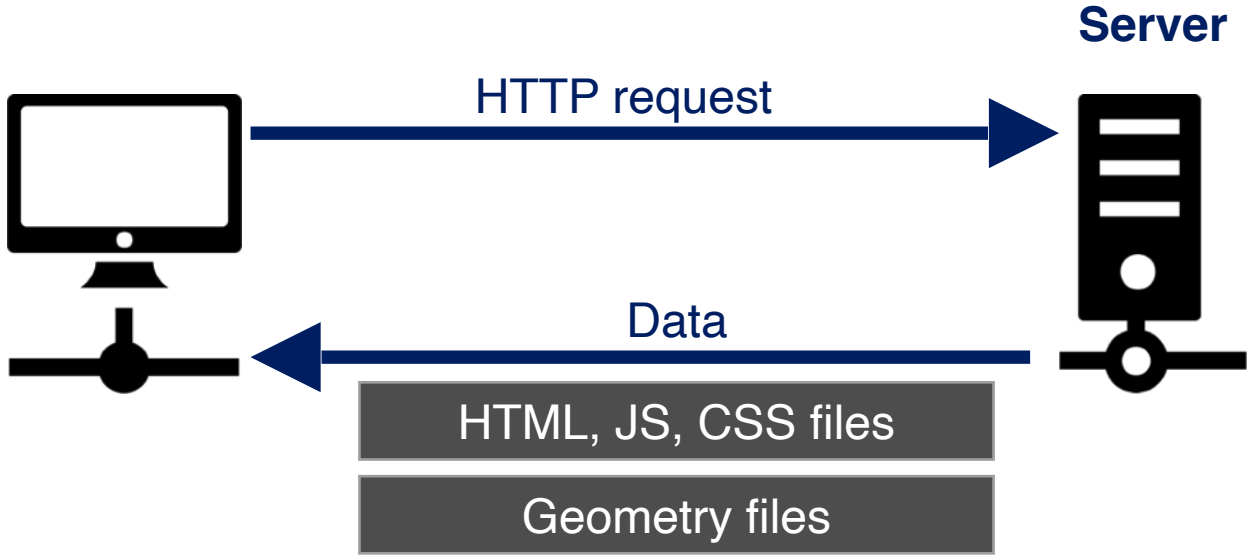
Client and server



WebGL model

Client side (Web browser)

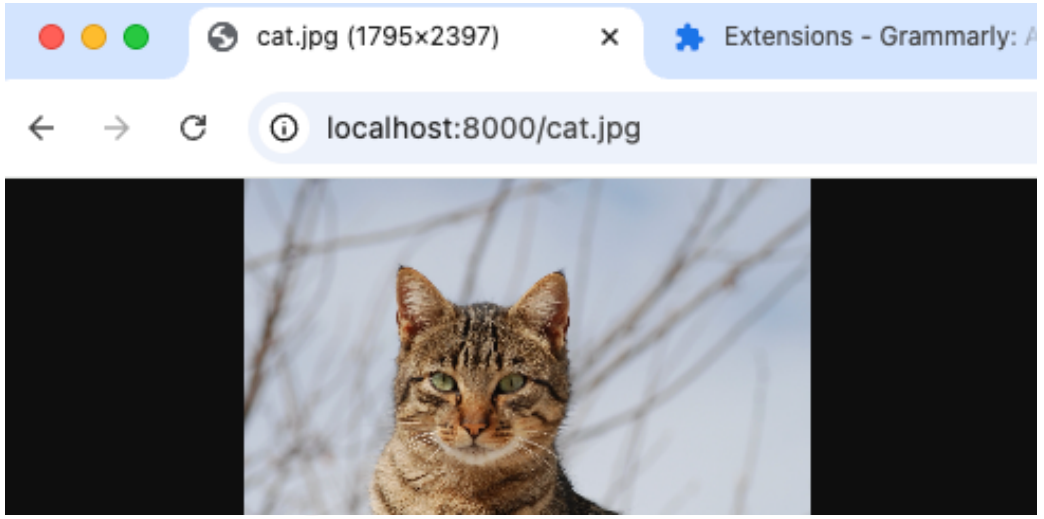
- Rendering
- Interaction
- Light-weight aggregation
- Filtering



CS425: Client and server run on the same computer

Installing Python web-server: **go.uic.edu/webserver**

```
[ redak khairis-mbp Desktop ] cd webserver
[ redak khairis-mbp webserver ] ls
cat.jpg
[ redak khairis-mbp webserver ] python3 -m http.server
```

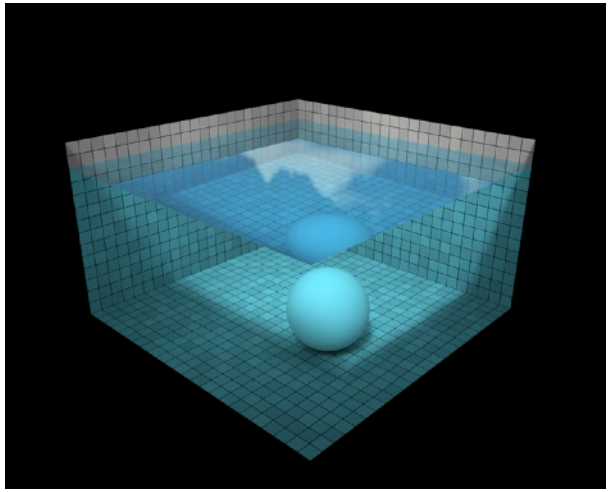


JavaScript: a client-side programming language

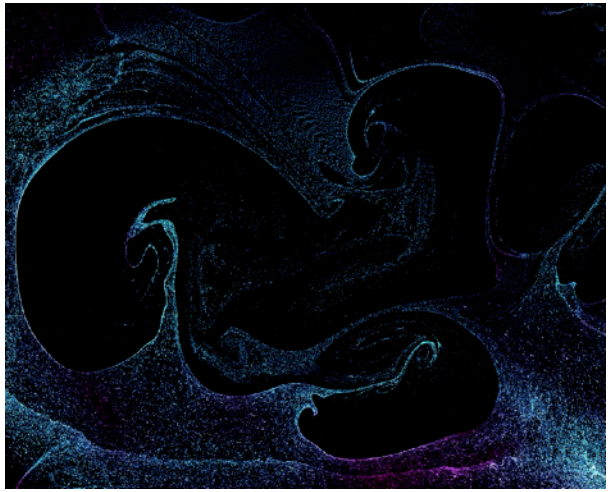
- Interpreted object-oriented language.
- Loosely typed language.
  - Does not require a variable type to be specified.
- Add, delete, and modify nodes from the HTML document tree.

JavaScript: a client-side programming language

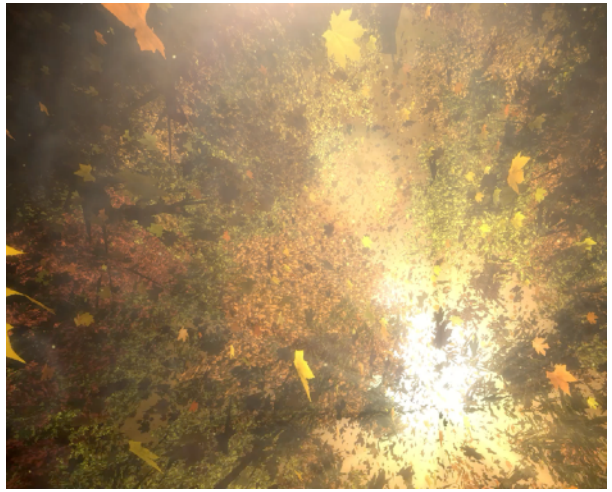
- Is JavaScript slow?
  - JavaScript engines in browsers are getting much faster.
  - Not an issue for graphics, since we transfer the data to the GPU with WebGL.



http://madebyevan.com/webgl-water/



https://haxiomic.github.io/projects/webgl-fluid-and-particles/



http://oos.moxiecode.com/js_webgl/autumn/

JavaScript basics

- Two scopes:
  - Local
  - Global
- Variable created inside a function with ‘var’ keyword: local to function.
  - Created and destroyed every time function is called.
  - BUT: variables declared without ‘var’ keyword are always global.
- Variable created outside a function: global

JavaScript basics

- Inserting JavaScript code in a web page:
  - Inside an HTML tag script.
  - In an external file.
  - As an HTML attribute value.

```
<script type="text/javascript">
    alert("Here is an example.");
</script>
```

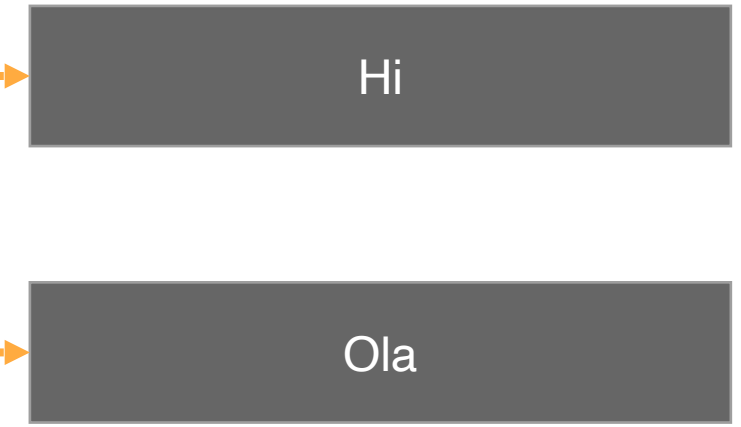
```
<script type="text/
javascript" src="file.js"></ script>
```

Statements, comments, and variables

- Statements: separated by new line or semicolon.
- Comment:
  - Single line: // here is a comment
  - Multi line: /* here is a comment */
- Loops and iteration:
  - for, for…in, for…of, do…while, while.
- Variables:
  - Assignment operator (=) to assign values.
- **Similarity to C/C++**

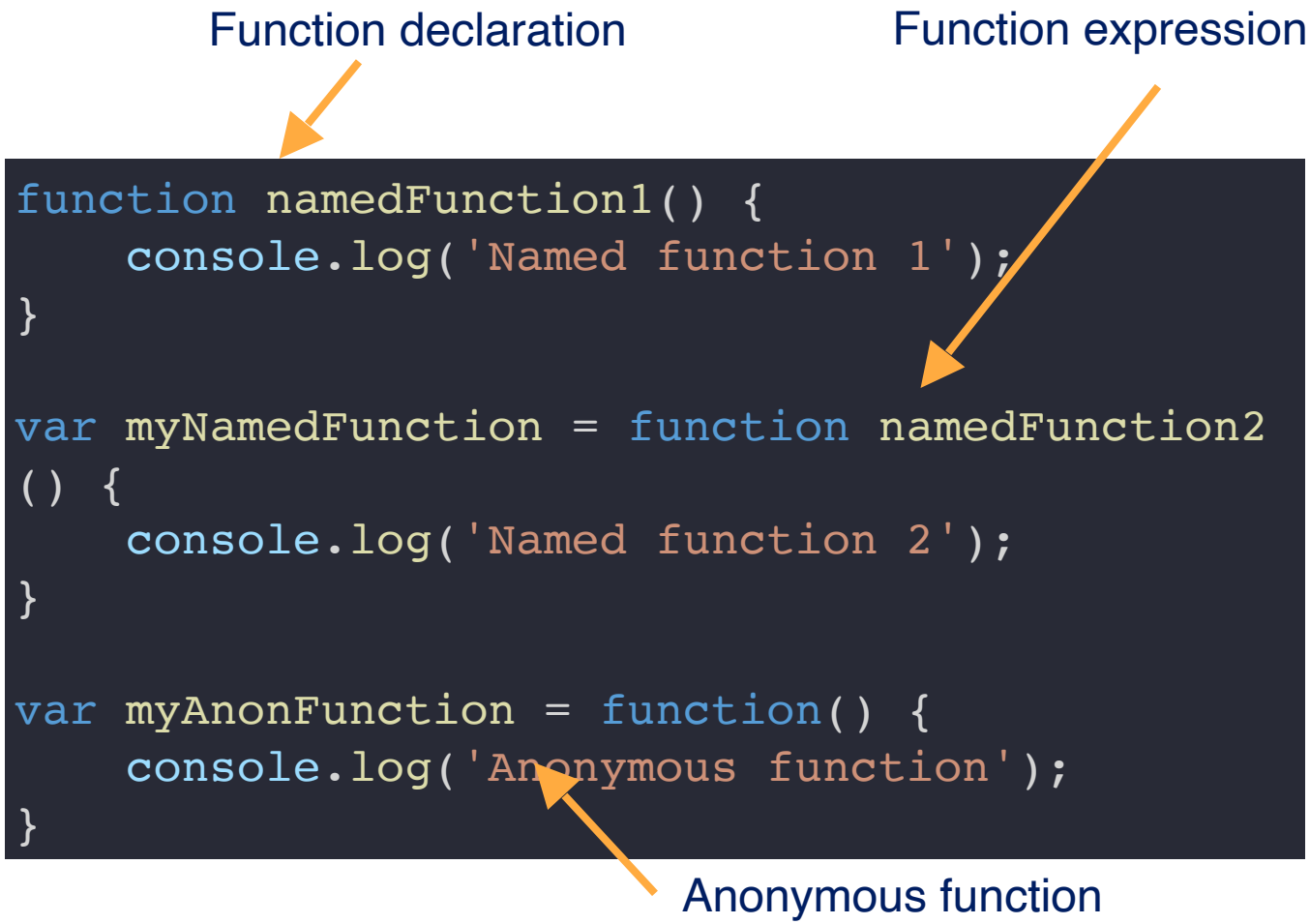
Variable scope

```
var message = 'Hi';

function modify1(){
    var message = 'Hello';
}

function modify2(){
    message = 'Ola';
}

modify1();
console.log(message);

modify2();
console.log(message);
```



Functions

- Different ways to define functions:
  - Named
  - Anonymous
- Function expressions cannot be used before they appear in the code.



Functions

- Function declarations load before any code is executed, while function expressions load only when the interpreter reaches that line.
- Function expressions: closures, arguments to other functions

```
alert(foo());
function foo() { return 5; }
```

Function declaration: alerts 5. Declarations are loaded before any code can run.

```
alert(foo());
var foo = function() { return 5; }
```

Function expression: error in this case, as foo wasn't loaded yet.

Functions

- Functions are first-class objects:
  - Supports passing functions to other functions.
  - Returning them as values from other functions.
  - Assigning them to variables or data structures.
- Closure:
  - Function that maintains the local variables of a function, after the function has returned.

Closure example

```
function sayHi(name){
    var whatToSay = 'Hi '+name;

   return function(){
        console.log(whatToSay);
    }
}

var say = sayHi('Bob');
say();
```

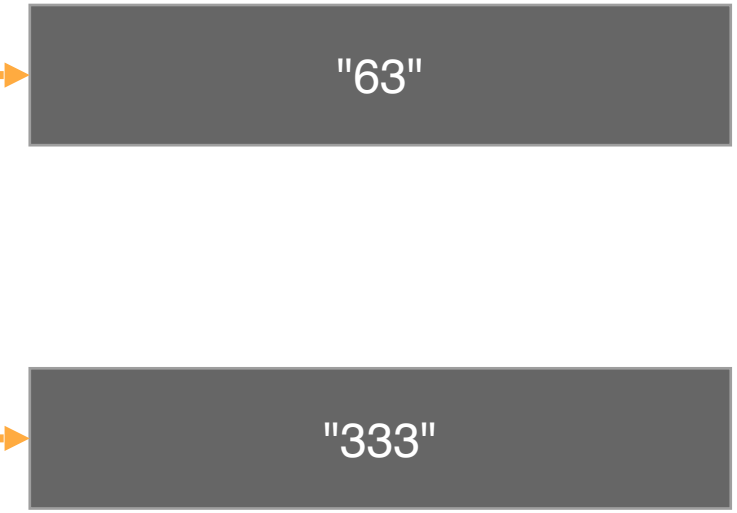
A closure: a function inside a function

No matter where it is executed, closure function will always remember variables from sayHi.

Data types: numbers and strings

- Numbers: a primitive data type (32-bit float).
- String: sequence of characters.
- Booleans.

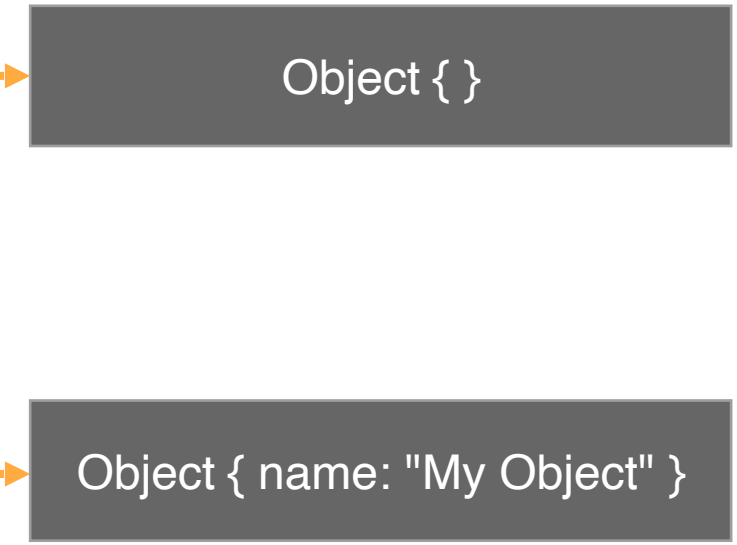
```
var aux1 = 3.0;
var aux2 = 3;
var aux3 = '3';

console.log(aux1+aux2+aux3);
console.log(aux3+aux2+aux1);
```



Objects

- In JavaScript, objects are a collection of properties with a name and a value.

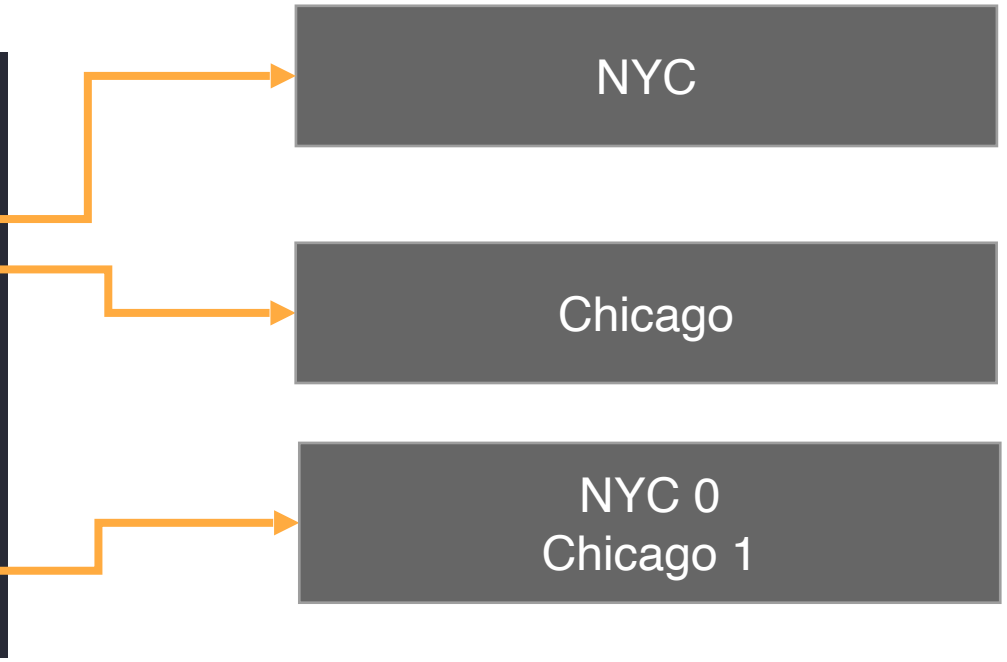
```
var myObject = new Object();
console.log(myObject);

myObject.name = "My Object";
console.log(myObject);
```



Arrays

- List-like objects.

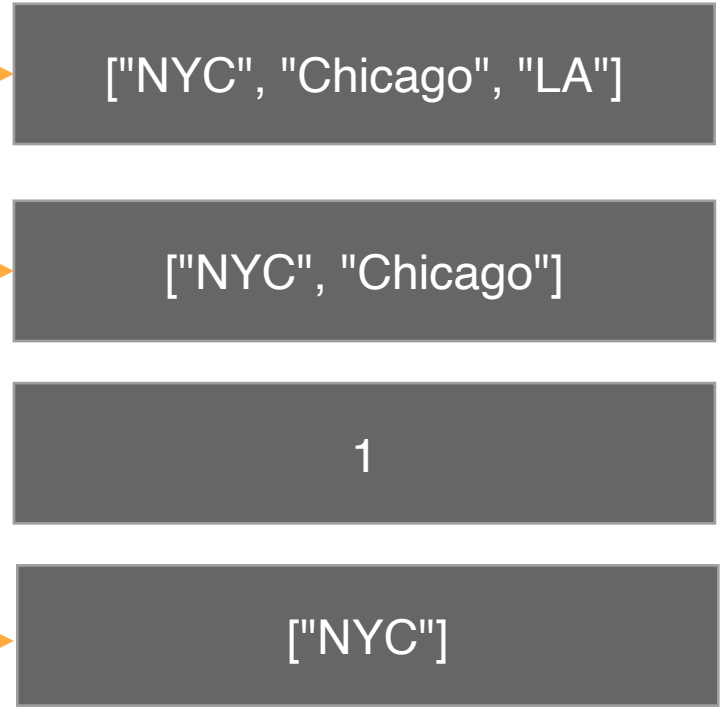
```
var cities = ['NYC', 'Chicago'];

console.log(cities[0]);
console.log(cities[cities.length-1])
;

cities.forEach(function(item, index)
) {
    console.log(item, index);
}
```



Arrays

- List-like objects.

```
cities.push('LA'); // in place
console.log(cities);

cities.pop(); // in place
console.log(cities);

var pos = cities.indexOf('Chicago');
console.log(pos);

cities.splice(pos, 1);
console.log(cities);
```



Example: map

```
var a = [1, 2, 3];

for(var i=0; i<a.length; i++){
    a[i] = a[i] * 2;
}

for(var i=0; i<a.length; i++){
    console.log(a[i]);
}
```

```
var a = [1, 2, 3];

function map(f, a){
    for(var i=0; i<a.length; i++){
        a[i] = f(a[i]);
    }
}

map(function(x){return x * 2;}, a);
map(alert, a);
```

Example: reduce

```
var nums = [1, 2, 3, 4];

function sum(a){
    var sum = 0;
    for(var i=0; i<a.length; i++)
        sum += a[i];
    return sum;
}

function mult(a){
    var mult=1;
    for(var i=0; i<a.length; i++)
        mult *= a[i];
    return mult;
}

console.log(sum(nums));
console.log(mult(nums));
```

```
var nums = [1, 2, 3, 4];

function reduce(f, a, init){
    var s = init;
    for(var i=0; i<a.length; i++)
        s = f(s, a[i]);
    return s;
}

function add(a, b){
    return a+b;
}
function mult(a, b){
    return a*b;
}

console.log(reduce(add, nums, 0));
console.log(reduce(mult, nums, 1));
```

Manipulating documents

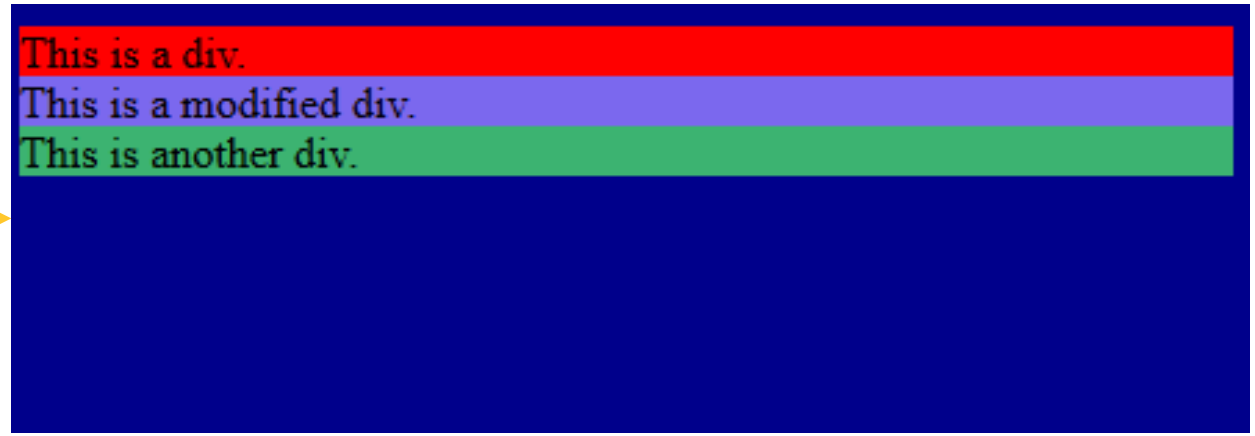
- So far: HTML, CSS, JavaScript.
- But how can we use JavaScript to modify the HTML page itself?
- Answer: **document object model (DOM)**.
- When an HTML document is loaded by a browser, it becomes a **document object**, containing the root node of the HTML document.

Document object

```
» document
← ▼ HTMLDocument https://www.google.com/
    URL: "https://www.google.com/"
  ▶ __wizdispatcher: Object { La: trigger(c) ↱≡ , Fa: {…}, Aa: false, … }
  ▶ __wizmanager: Object { w0: false, JN: (1) […], Ha: 10, … }
  ▶ activeElement: <body id="gsr" class="hp vasq big" jsmodel="TvHxbe" jsaction="VM8bg:.CLIENT;hWT9Jb:.CL…:.CLIENT;kWlxhc:.CLIENT"> 🔲
    alinkColor: ""
  ▶ all: HTMLAllCollection { 0: html 🔲 , 1: head 🔲 , 2: meta 🔲 , … }
  ▶ anchors: HTMLCollection { length: 0 }
  ▶ applets: HTMLCollection { length: 0 }
    baseURI: "https://www.google.com/"
    bgColor: ""
  ▶ body: <body id="gsr" class="hp vasq big" jsmodel="TvHxbe" jsaction="VM8bg:.CLIENT;hWT9Jb:.CL…:.CLIENT;kWlxhc:.CLIENT"> 🔲
    characterSet: "UTF-8"
    charset: "UTF-8"
    childElementCount: 1
```

DOM elements using selectors

```
var allDivs = document.querySelector('div');
var myDiv = document.querySelector('#mydiv');
var mySecondDiv = document.querySelector('#myseconddiv
');
var myClass = document.querySelector('.myclass');
mySecondDiv.textContent = 'This is a modified div.';
```



DOM elements using selectors

```
var newDiv = document.createElement('div');
newDiv.textContent = 'This is a new div.';
newDiv.className = 'myclass';
document.querySelector('body').appendChild(newDiv);
```

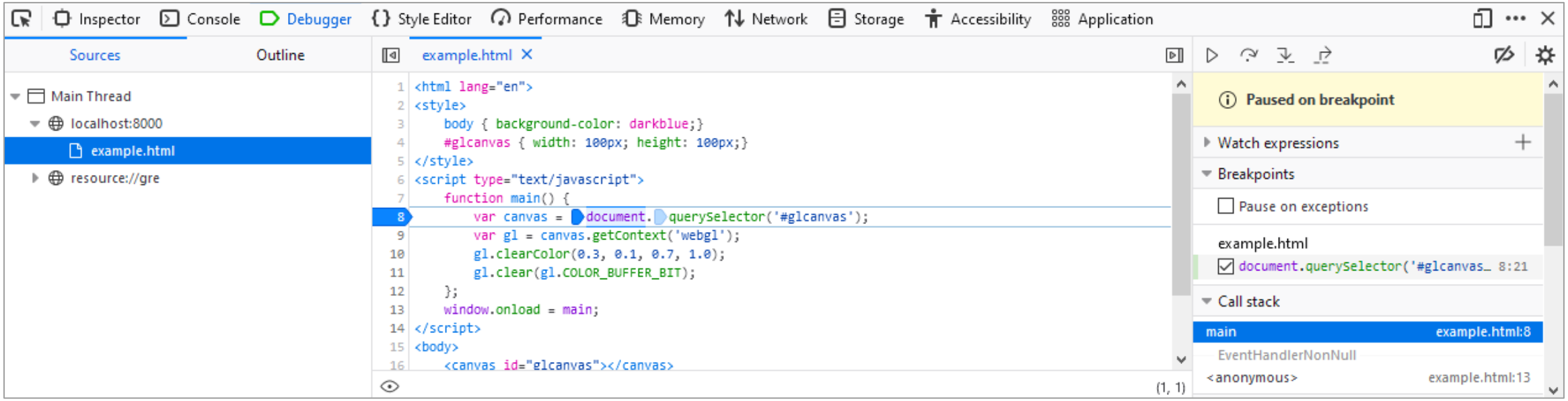
Event handlers

- Events are actions like being clicked, pressed keys, getting focus, etc.
- Different ways to specify handlers for a particular event:

```
<button onclick="handleClick()">
```

```
document.querySelector("button").onclick = function(event) {
}
```

Debugging JavaScript



Finally drawing something

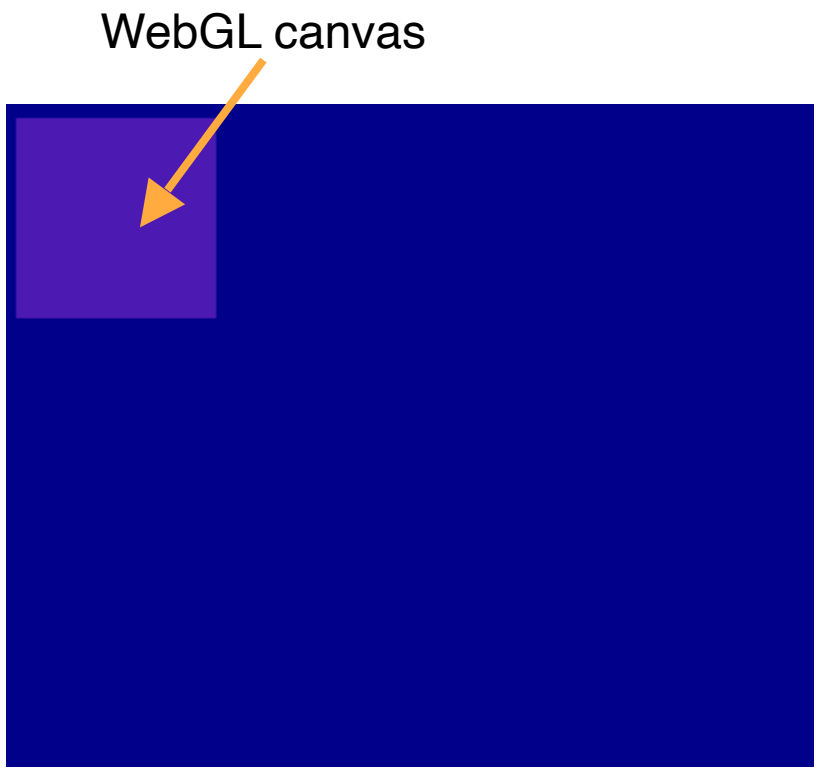
- Several ways to draw graphics on the web:
  - **Canvas**
    - HTML element.
    - Allows you to draw primitives like lines and rectangles.
    - Bitmap-style interactions
  - **WebGL**
    - Complex 3D geometries.
    - Uses rendering pipeline.
    - Hardware acceleration.

WebGL: a bird's-eye view

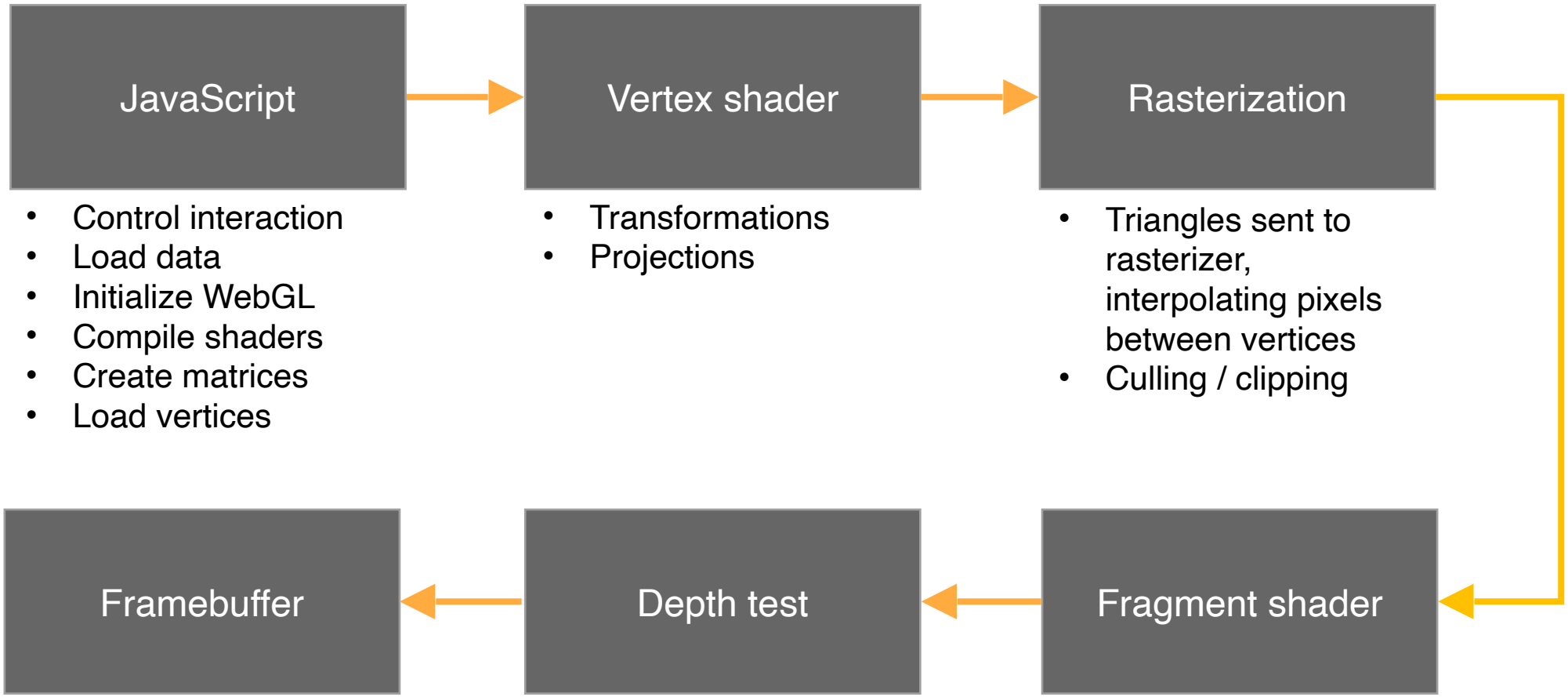
- API for rendering graphics within a web browser without plug-ins.
- Hardware accelerated.
- Shader based (no fixed-function API).
  - Fixed function pipeline: set of calls for matrix transformation, lighting.
  - Programmable pipeline: shaders for vertex and fragment processing.
- WebGL 2.0 based on OpenGL ES 3.0.

WebGL: a bird's-eye view

```
<html lang="en">
<style>
    body { background-color: darkblue;}
    #glcanvas { width: 100px; height: 100px;}
</style>
<script type="text/javascript">
    function main() {
        var canvas = document.querySelector('#glcanvas
');
        var gl = canvas.getContext('webgl2');
        gl.clearColor(0.3, 0.1, 0.7, 1.0);
        gl.clear(gl.COLOR_BUFFER_BIT);
    };
    window.onload = main;
</script>
<body>
    <canvas id="glcanvas"></canvas>
</body>
</html>
```



WebGL: a bird's eye view



- Control interaction
- Load data
- Initialize WebGL
- Compile shaders
- Create matrices
- Load vertices

- Transformations
- Projections

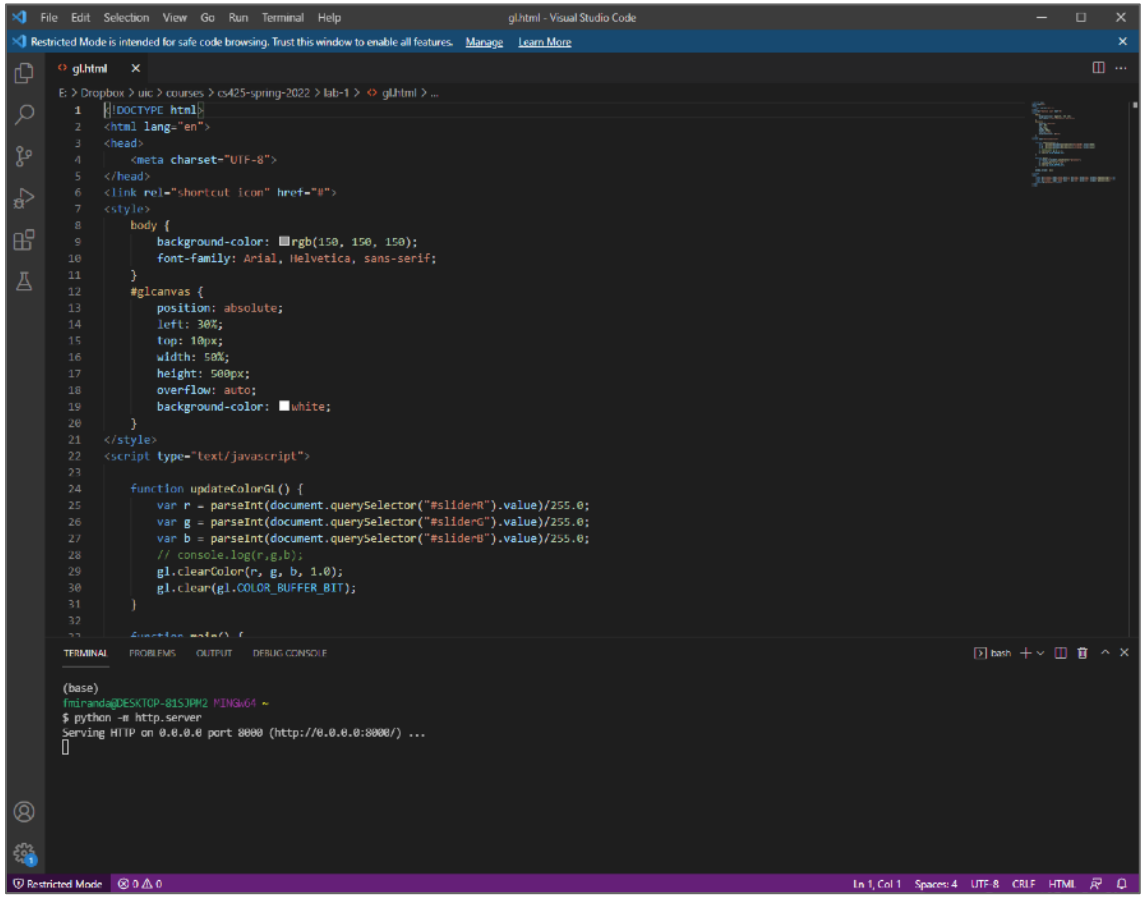
- Triangles sent to rasterizer, interpolating pixels between vertices
- Culling / clipping

- **Final image**

- Executed for each fragmented outputted by rasterization

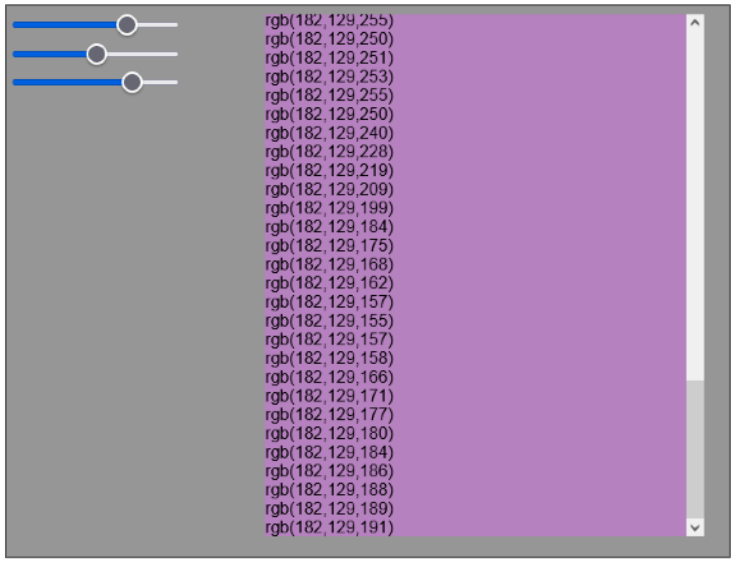
Lab

- **Web server:**
  - Python: go.uic.edu/webserver
  - Windows: https://gitforwindows.org/
- **Text editor:**
  - Visual Studio Code: https://code.visualstudio.com/
  - Atom: https://atom.io/
  - Sublime: https://www.sublimetext.com/
- Double check if your browser supports WebGL: **https://get.webgl.org/**



Lab

1. Create a JavaScript program that adds an array of numbers (1,…, 5) to a `div`.
2. Add to a div the RGB color defined by 3 sliders. Then change the background color of a `div` according to the RGB color.
3. Instead of changing the background color of a `div`, use a GL canvas.



Useful links

https://developer.mozilla.org/en-US/docs/Web/HTML

https://developer.mozilla.org/en-US/docs/Web/CSS

https://developer.mozilla.org/en-US/docs/Web/JavaScript