Before we begin...

- Videos On!

Welcome

Agenda

- THREE.js
- Hosting with Vercel
- Wrapping Up
- Surveys

THREE.js

Documentation and Examples

What is THREE.js?

What is THREE.js?

A project that aims to create lightweight 3D library with the lowest level of complexity

More or less, it is an attempt to make 3D stuff easier in browsers

Who made THREE.js?

A fellow named Ricardo Cabello. Here is his:

- GitHub
- Website
- Twitter

What is it built with?

It's built on top of WebGL, which is a JavaScript API

It uses a renderer to show the 3D environment

How do we install it?

There are many ways to install it, but we will be using NPM (as well as Parcel). As well as THREE.js, we will also be using a few addons

```
npm install --save three three-orbitcontrols dat.gui
```

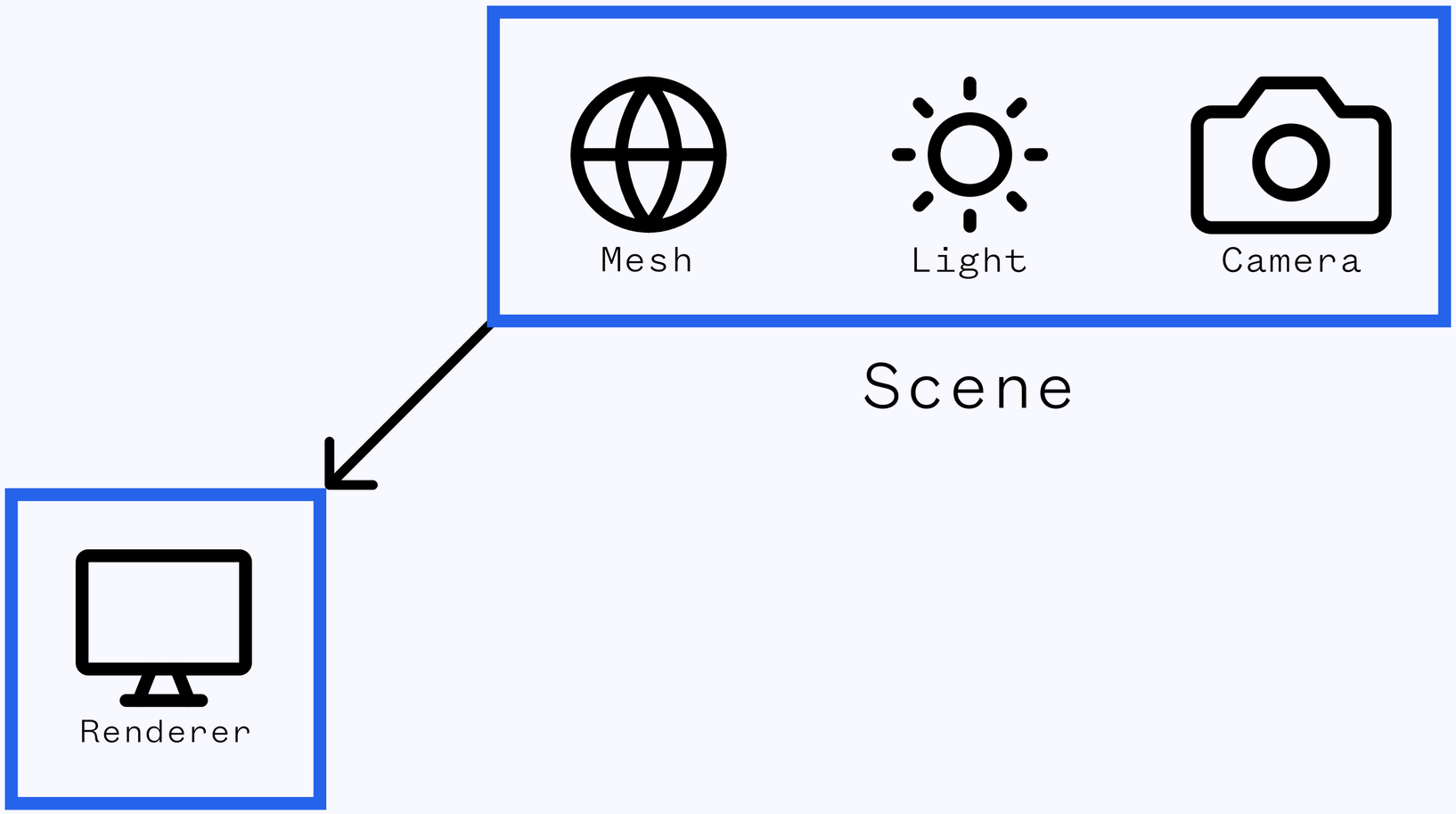
OrbitControls will allow us to change the perspective and dat.gui will allow us to change the speed of animations

Key Components

Key Components

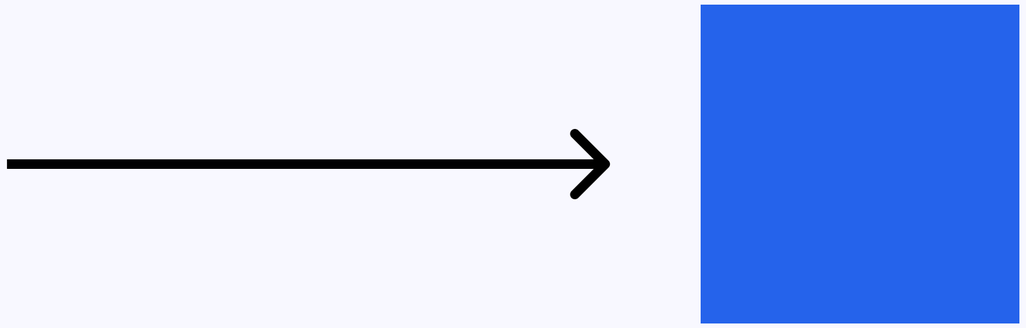
We extend things that THREE.js provides to create:

- Camera
- Scene
- Renderer
- Light(s)
- Mesh(es)

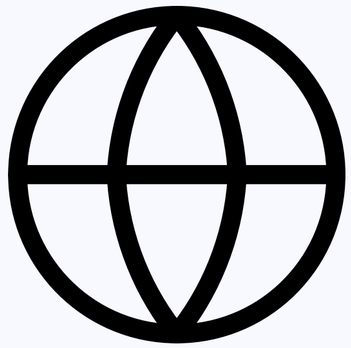

Mesh
Light
Camera
Scene
Renderer

What is a Mesh?

Material



Geometry



Mesh

What tools will we be using?

- THREE.js
- OrbitControls
- dat.gui

Let's see some code!

Creating a Renderer

```
function createRenderer() {
  let renderer = new WebGLRenderer({
    antialias: true,
  });

  renderer.setSize(window.innerWidth, window.innerHeight);
  renderer.setClearColor("#16161d");
  renderer.setPixelRatio(window.devicePixelRatio);

  let output = document.querySelector("#output");
  output.appendChild(renderer.domElement);

  return renderer;
}
```

Creating a Scene

```
function createScene() {
  return new Scene();
}
```

Creating a Camera

```
function createCamera() {
  let camera = new PerspectiveCamera(
    45, // Field of View
    window.innerWidth / window.innerHeight, // Ratio
    0.1, // Near
    1000, // Far
  );
  camera.position.x = -30;
  camera.position.y = 40;
  camera.position.z = 30;
  camera.lookAt(0, 0, 0);
  return camera;
}
```

Creating an AxesHelper

```
function createAxesHelper() {
  let axesHelper = new AxesHelper(40);
  return axesHelper;
}
```

Creating a Cube

```
function createCube() {
  let geometry = new BoxGeometry(4, 4, 4);
  let material = new MeshLambertMaterial({
    color: "dodgerblue",
  });
  let mesh = new Mesh(geometry, material);
  return mesh;
}
```

Adding Everything to the Scene

```
let renderer = createRenderer();
let scene = createScene();
let camera = createCamera();
let axesHelper = createAxesHelper();
let cube = createCube();

scene.add(axesHelper, cube);
renderer.render(scene, camera);
```

Creating a Light

```
function createLight() {
  let light = new PointLight("#ffffff", 2);
  light.position.set(10, 18, 10);
  return light;
}
```

Creating a Light Helper

```
function createLightHelper(light) {
  return new PointLightHelper(light);
}
```

requestAnimationFrame

What is `requestAnimationFrame`?

In order to show changes in a THREE.js scene, we need to re-render it. `setInterval` used to be the way we do that, but we will be using `requestAnimationFrame`. Why?

- It runs as quickly as possible (higher FPS)
- It stops when the tab or window is no longer active (better battery life)
- It times the calls for optimum performance
- It is often hardware-boosted

requestAnimationFrame

```
function animate() {
  // Make sure to render your THREE.js scene here!
  console.log("This was called");
  requestAnimationFrame(animate);
}

animate();
```

Hosting with Vercel

Change your `package.json`

Firstly, remove the `"main"` key and value

Secondly, add a `"build"` script:

```
"build": "parcel build src/index.html",
```

Steps for Deploying

- Upload all of your code to GitHub (remember to use a Personal Access Token as your password)
- Sign up for Vercel using your GitHub account
- Import the repository to Vercel
- Customize your build settings (make sure you toggle the "Override" button!):
  - First box: `npm run build`
  - Second box: `dist`
  - Third box: `npm install`

Final Projects!

Here is the brief

What's next?

Moving Forward

- The projects will be due on the **14th of January**
- Between now and then, there will be one on ones with me
  - I'll also still be available to message

It's all over*

But not really

This course was all about:

- Skills that will last
- New approaches to learning
- Preparing you for jobs
- Putting you in a room with driven and intelligent people

Progress over Perfection

A Quick Rewind

What did we cover on the first night?

- What JavaScript is and does
- Data Types
- `console.log`

Think about how far you have come in just 60 hours

Maybe, at some point in this course, you have doubted yourself...

You all know enough

- You will be useful JavaScript developers in the workplace
- But you do have so much more to learn
- Forever...
- We all do
- That is a great thing

Most Important Things we Taught

How to think like a programmer

I probably sounded like a broken record but the concepts are, by far, the most important thing

Most Important Things we Taught

How to debug

- Everything comes with practice, but practice is easier when you can debug:
  - Well
  - And patiently
  - And calmly
  - And cheerfully (enjoy those little wins)
- Fixing other people's code is incredibly rewarding

What now?

You (kind of) never have to see me again!

- Don't stop coding: practice
- Learning new languages and frameworks makes you better at other ones
- Participate! Meetups, conferences, open source, blogs etc.
- Stay in touch with each other
- Enjoy! It's going to be easier that way
- **Get great: blow my mind!**

Practice

- Exercism
- Project Euler
- Rosetta Code
- CodeWars
- CoderByte
- Hacker Rank
- Code Signal
- Leet Code
- Code Kata

Passive Consumption

- Twitter
- Newsletters
- GitHub
- Web Inspiration
- Podcasts

Go to:

- Meetups
- Conferences (many are remote and many post videos of talks)
- Hackathons
- Digital Meetups
- Join Slack channels and Discords

Books

- Eloquent JavaScript
- Speaking JavaScript
- JavaScript: The Definitive Guide
- JavaScript: The Good Parts
- The You Don't Know JS Series
- Understanding ES6
- Exploring ES6
- Functional Light
- Essential JS Design Patterns

Getting great at programming

- Keep Learning
- Read books about code and other people's code
- Look for parallels between everything
- Have a plan (such as this roadmap or this roadmap)
- Teach and help others with their code
- Don't get too caught up in the code, as well
  - Focus on "real life" and the "user"
- Don't be afraid to take a break
- Focus on the language itself
- Don't forget to enjoy it!

Getting great at programming

- Watch videos about coding
  - FrontEnd Masters
  - Egghead
  - Dash
  - Codecademy
  - Plural Sight
  - Udacity
  - Udemy

For all of you

- You have time and skills
- Keep pushing
- Keep celebrating the little wins
- Remember that it is a marathon, not a sprint
- Figure out what suits you
- Remember to enjoy it
- Be proud of how far you have come in an exceptionally short period of time

Now a little bit about me

Why do I teach?

- It's the most rewarding thing I can ever imagine
- Every day is inspiring:
  - The effort
  - The projects
  - The people

Please, please, please stay in touch

- Slack
- jack.jeffress@generalassemb.ly
- Instagram
- Twitter

Thank You!

Really, for everything!

You have made this incredibly enjoyable

Survey Time

One last bit of feedback

That's all!

Thank You!