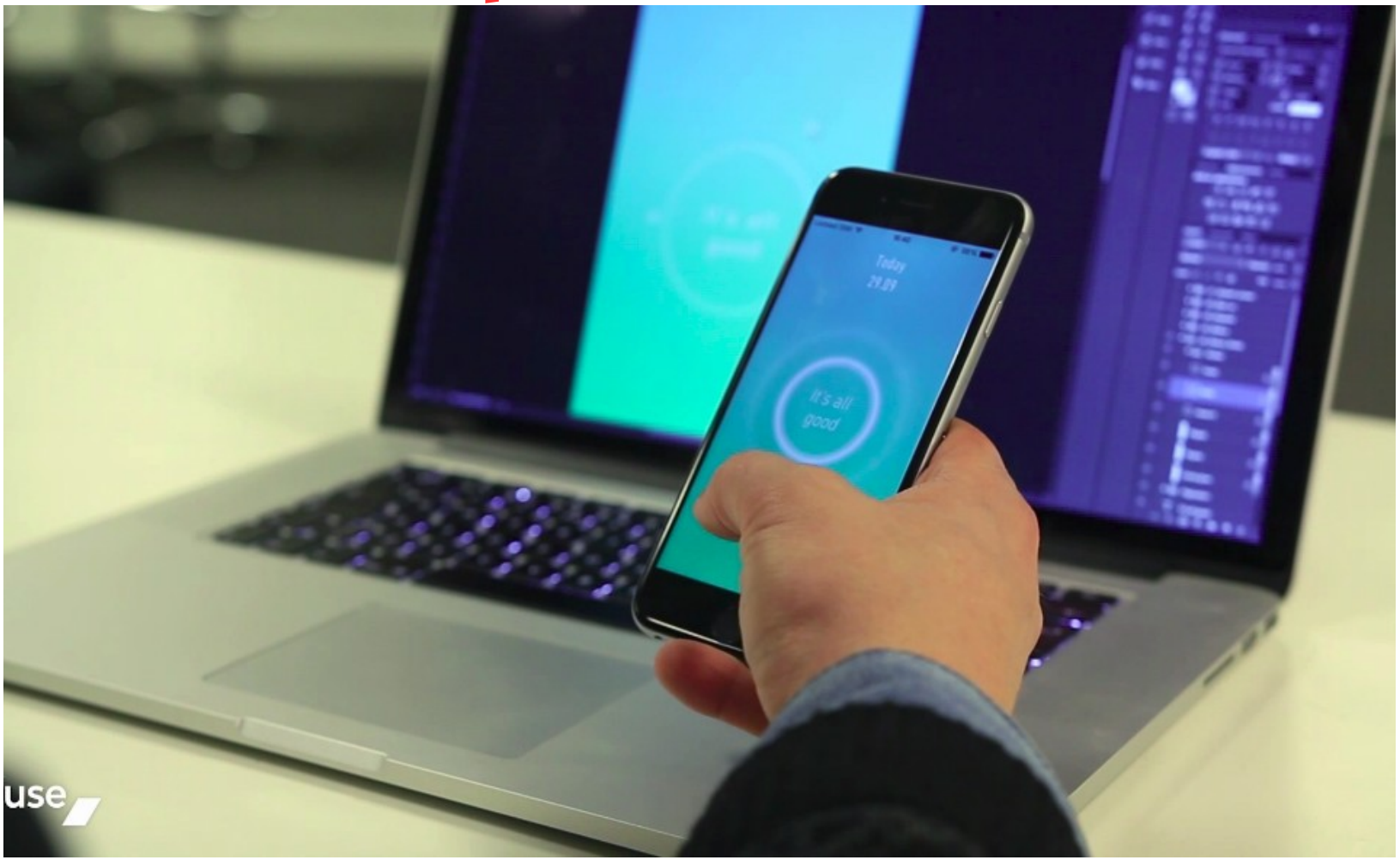
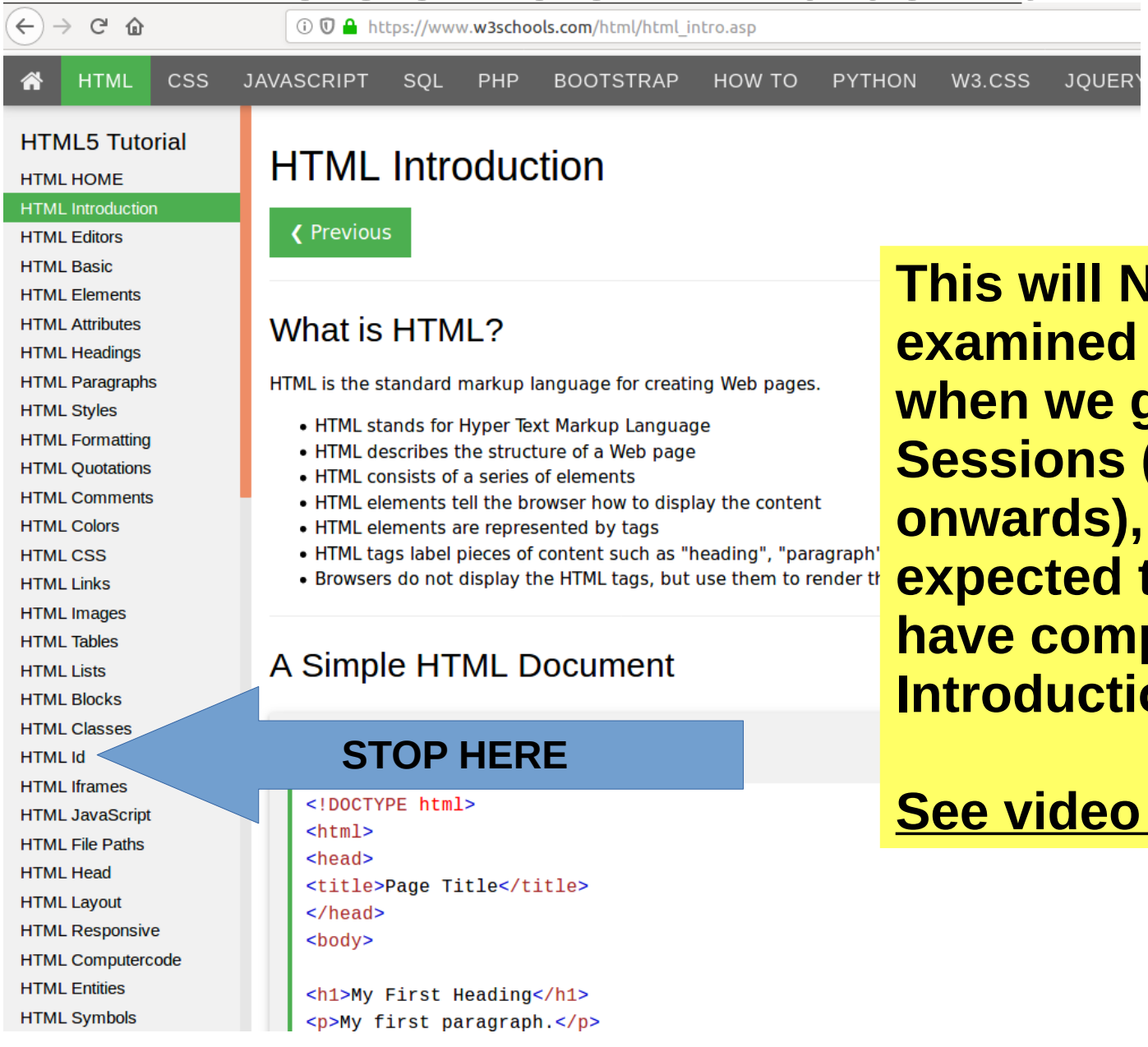
CS385 Mobile Application Development (Lecture 2)



Peter Mooney

Self Study Excercise – Follow the W3Schools Tutorial



This will NOT be examined – However, when we get to the Lab Sessions (Lab 2 onwards), it will be expected that you will have completed this Introduction to HTML

<u>See video on Moodle.</u>

Check out Moodle for some screencasts on using Codesandbox for the first time.

Topic 1 - Introduction to React Javascript, Introduction to the CS385 module

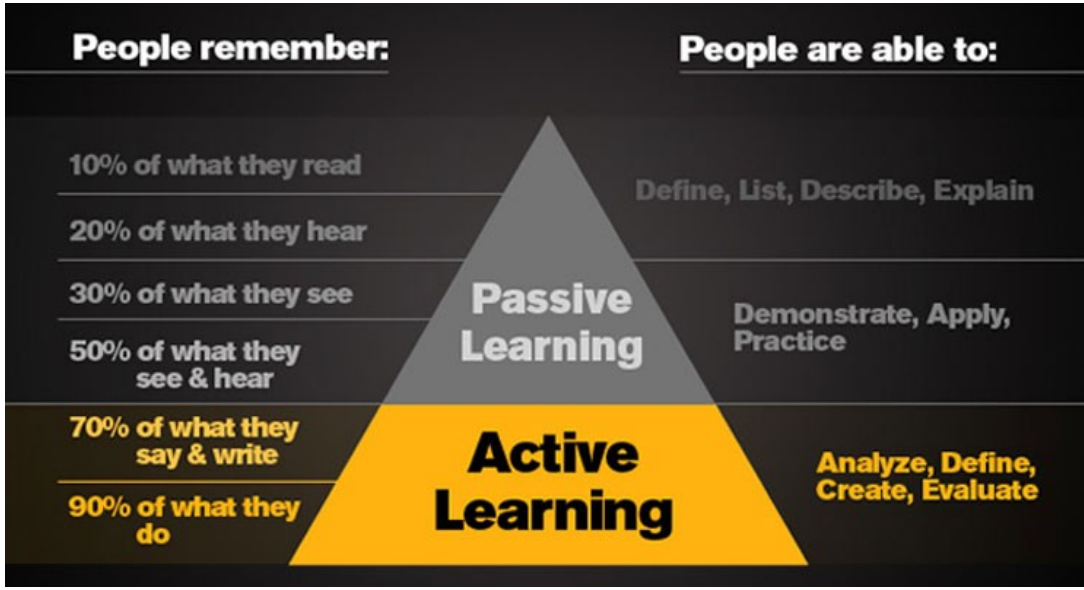
Please note - some of the screencasts may have recording dates which are one or two years old. Please ignore these as the content of the screencast videos is unchanged.

Using the w3schools tutorial on HTML for CS385

Opening codesandbox.io for the first time

CS385 – Some assumptions

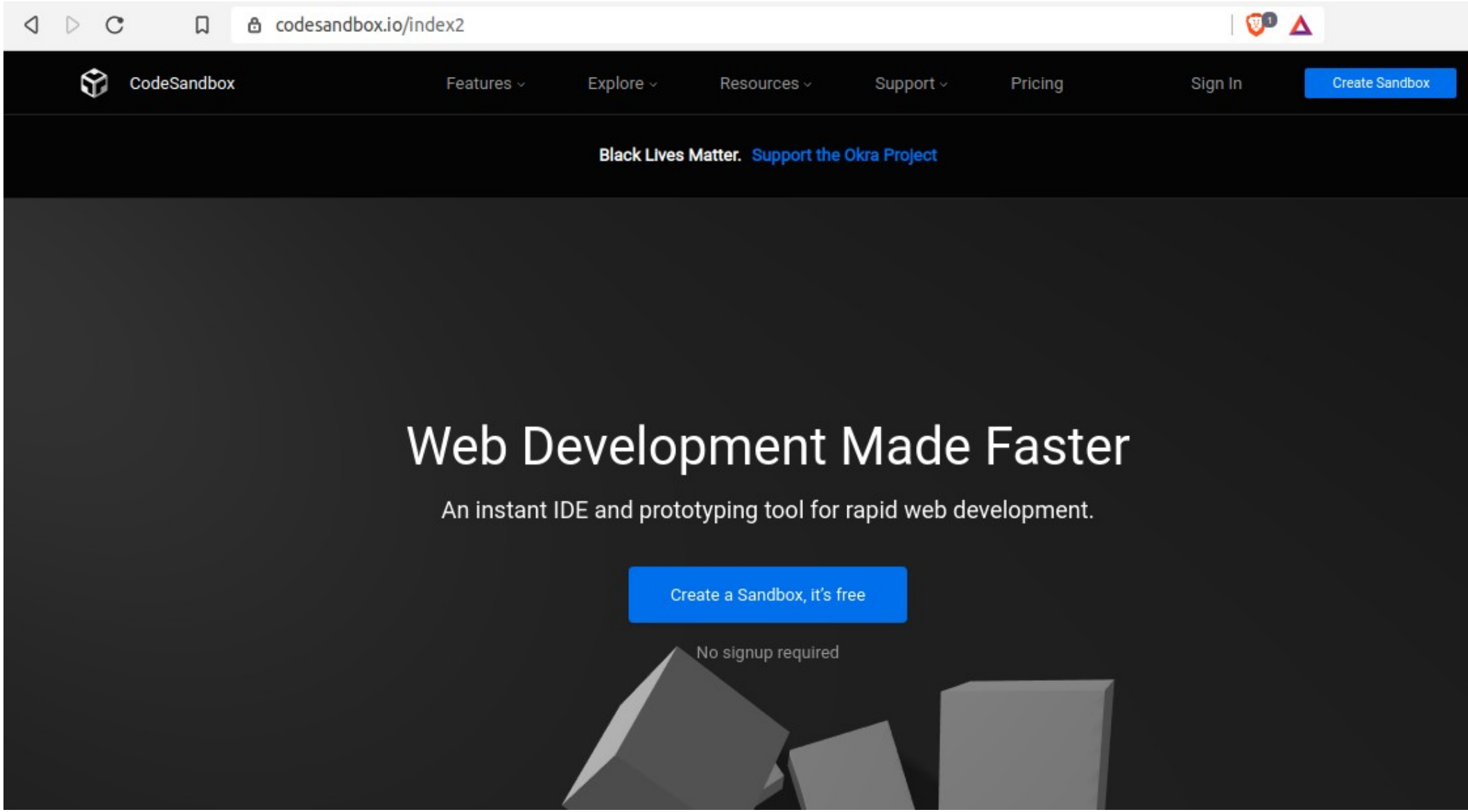
- **You have programmed before** (even if this is just your Introduction to Java) and have a **basic understanding of programming constructs**
- **You will try out (run, practice) the code examples provided in lectures**



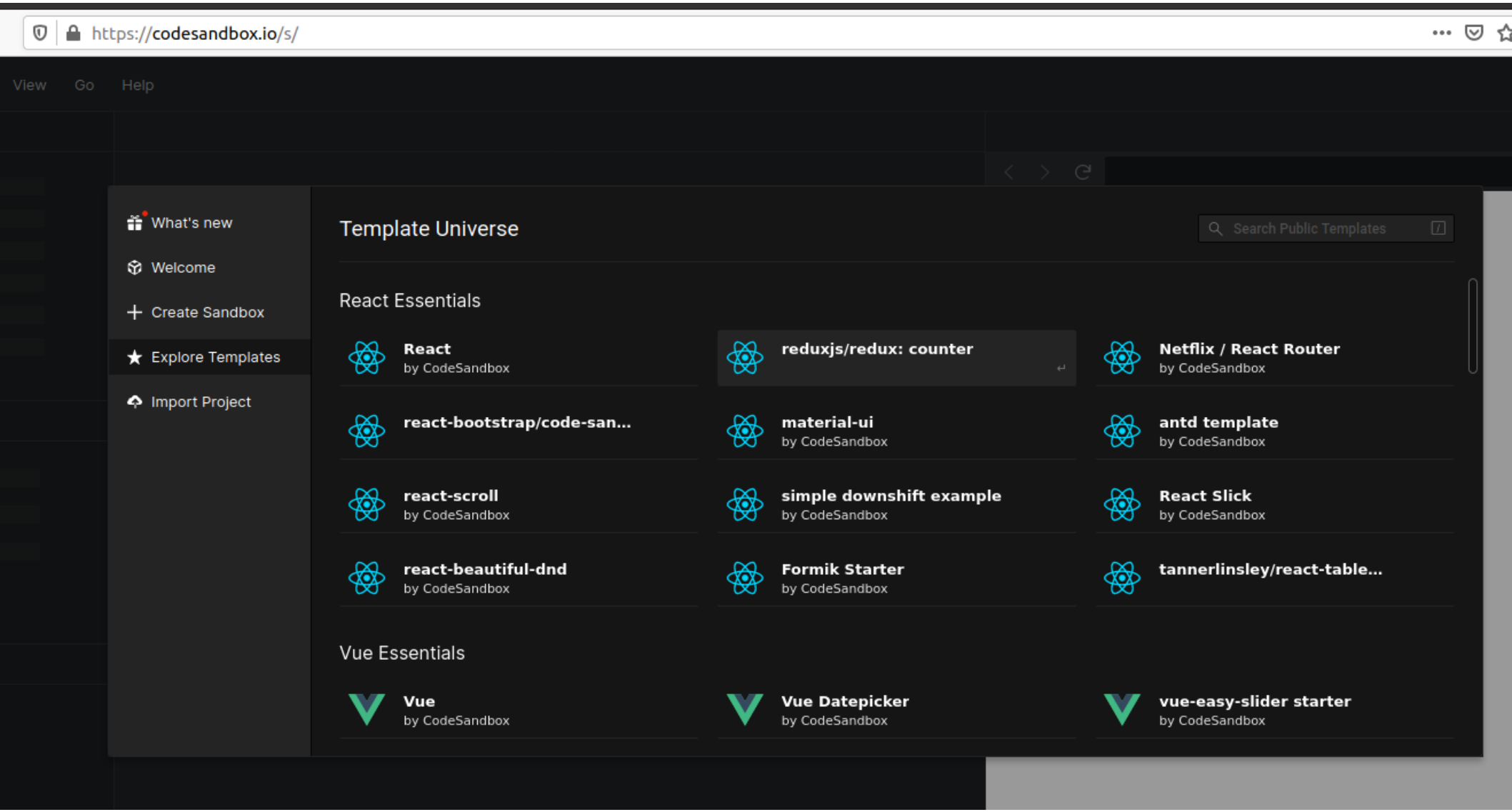
Starting Javascript – let's look at some of the features of basic Javascript

- Some of the concepts will be familiar to you from your Java programming
- Some of the concepts will be familiar to you from any other programming language you have used before.
- **What is important to remember is that Javascript is a fully featured, powerful, programming language. Don't be fooled by its simplistic style.**

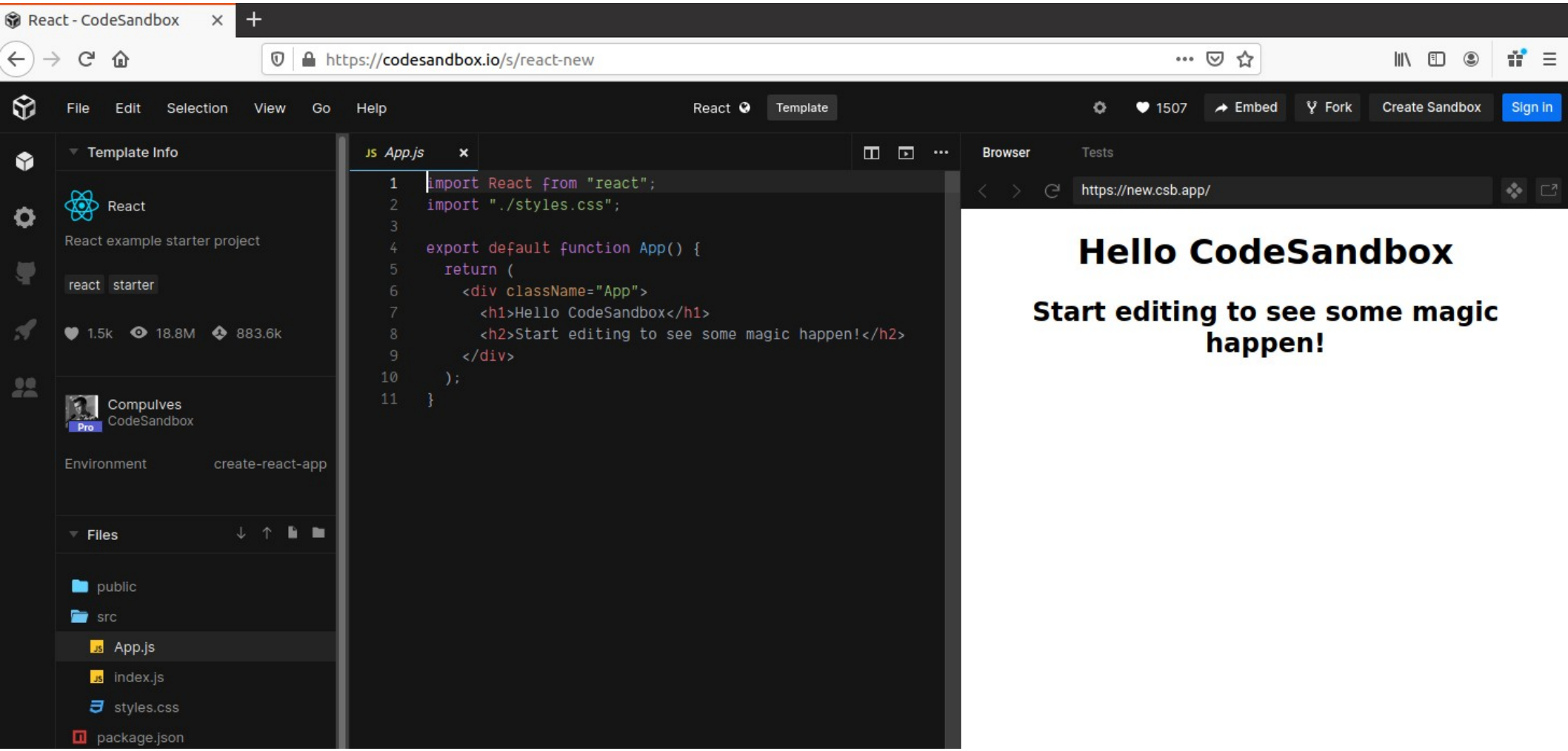
We will use codesandbox.io



In codesandbox choose a react template to start your work



Default starting template



Replace all of the code in `App.js` with the code in `Lecture2ex1.txt`

petermooney

Drafts / blissful-nobel-fxkqlt

Share

Files

- public
- src
  - App.js
  - index.js
- package.json

Dependencies

Add Dependency

loader-utils	3.2.1
react	18.2.0
react-dom	18.2.0
react-scripts	5.0.1

External resources

App.js

```
import React from "react";

/**
 * The App component in React.
 * App is just a common name for a component that
 * wraps the rest of your app.
 * There's nothing particularly special about it.
 * Here is one of the simplest React applications we could have
 */

function App() {
  return (
    <>
      <h1>Hello and welcome to CS385</h1>
      <p>This is Mobile App Development</p>
    </>
  );
}

/** Essentially your app is contained within the return
 *  statement in the App() function. This is the same for the simplest
 * applications and really complex production release apps
 */
export default App;
```

Browser Tests Terminal

https://fxkqlt.csb.app/

Hello and welcome to CS385

This is Mobile App Development

This is a very simple fully working React application

```
import React from "react";

/**
 * The App component in React.
 * App is just a common name for a component that
 * wraps the rest of your app.
 * There's nothing particularly special about it.
 * Here is one of the simplest React applications we could have
 */

function App() {
  return (
    <>
      <h1>Hello and welcome to CS385</h1>
      <p>This is Mobile App Development</p>
    </>
  );
}

/** Essentially your app is contained within the return
 *  statement in the App() function. This is the same for the simplest
 * applications and really complex production release apps
 */
export default App;
```

HTML Tags

React Fragment

It is important to understand the structure of this React App

- The `function App()` is a 'component' (more later)
- It has a `return` statement which returns **JSX** (HTML and Javascript)
- This JSX is rendered by React and the Browser to become your application

```
import React from "react";

/**
 * The App component in React.
 * App is just a common name for a component that
 * wraps the rest of your app.
 * There's nothing particularly special about it.
 * Here is one of the simplest React applications we could have
 */

function App() {
  return (
    <>
      <h1>Hello and welcome to CS385</h1>
      <p>This is Mobile App Development</p>
    </>
  );
}

/** Essentially your app is contained within the return
 *  statement in the App() function. This is the same for the simplest
 * applications and really complex production release apps
 */
export default App;
```

As your App or return statement changes, the app is re-rendered immediately

JS App.js

```
import React from "react";

/**
 * The App component in React.
 * App is just a common name for a component that
 * wraps the rest of your app.
 * There's nothing particularly special about it.
 * Here is one of the simplest React applications we could have
 */

function App() {
  return (
    <>
      <h1>Hello and welcome to CS385</h1>
      <p>This is Mobile App Development</p>
      <h2>Watch as your app renders immediately</h2>
    </>
  );
}

/** Essentially your app is contained within the return
 *  statement in the App() function. This is the same for the simplest
 * applications and really complex production release apps
 */
export default App;
```

Browser Tests Terminal

https://fxkqlt.csb.app/

Hello and welcome to CS385

This is Mobile App Development

Watch as your app renders immediately

SCAN ME

Variables - Lecture2ex2.txt

JS App.js

```javascript
import React from "react";
/**
 * The App component in React. Here we have a very simple application.
 * Notice the return statement:
 * Each piece of Javascript is enclosed in curly brackets.
 * Try refreshing the browser page and see how the time changes.
 * The app gives a different message depending on the value of y
 */

function App() {
  // This is the App function or "component"
  let x = 10;
  let y = Math.random() * 10; // Javascript generates a random number
  // we get the current date and time from Javascript
  let currentTime = new Date().toLocaleString();
  let message = "Blank"; // message is a string
  if (y >= 5) {
    message = "Goodbye, CS385";
    x = y;
  } else {
    message = "Hello, CS385";
    x = y;
  }
```

SCAN ME

Variables and rendering (example given in class)

App.js

```
    x = y;
  }
  // return - this is where the JSX or your Javascript and HTML
  // is returned to the calling environment and rendered as your 'app'
  return (
    <>
      <p>
        <h1>{message}</h1> Rendered at {currentTime}
      </p>
      <p>To render or print the value of x we write {x}</p>
      <p>
        To render or print the value of an expression such as 10 times x we
        write {10 * x}
      </p>
      <p>Value of 4 times x plus 100 is {4 * x + 100}</p>
      <p>
        We can use Javascript functions inside curly backets such as rounding
        {Math.round(4 * x + 100)}
      </p>
      <p>
        We can use String functions inside curly backets such as
        <b>{message.toUpperCase()}</b> or <b>{message.toLowerCase()}</b>
      </p>
    </>
  );
}
export default App; // notice how this matches the function App()
```

Browser Tests Terminal

https://fxkqlt.csb.app/

Goodbye, CS385

Rendered at 9/20/2023, 2:56:02 PM

To render or print the value of x we write 8.505990490403194

To render or print the value of an expression such as 10 times x we write 85.05990490403194

Value of 4 times x plus 100 is 134.02396196161277

We can use Javascript functions inside curly backets such as rounding134

We can use String functions inside curly backets such as**GOODBYE, CS385** or **goodbye, cs38**

Console 1 Problems 0 React DevTools 0

No problems!

The curly braces or brackets { } in React Javascript

- We can evaluate ANY valid piece of Javascript inside the curly braces or backets
- Most commonly you will use the { } to print out the values of variables, display user interface elements, change the user interface, and so on.

A first look at Objects and arrays in Javascript

- Javascript is natively capable of working with objects (like those you have seen in Java)
- Indeed, Javascript allows us to very easily define objects and manipulate them.
- **One very important concept in Mobile Application Development is working with ARRAYS of Objects. This will become more obvious in a few weeks time.**
- So let's look at some objects and then move to arrays.

Arrays of objects in Javascript/React

- **One of the most important features of Javascript is the ability to easily work with arrays of objects.**
- Javascript provides many functions to allow the manipulation and management of arrays containing objects.
- This type of functionality will be very important in our mobile development software code.
- Let's have a look at an example – `Lecture2Ex3.js` is the file containing the source code.

An app to render an array of objects

Drafts / blissful-nobel-fxkqlt

0 Share

App.js

Our array of objects

```
import React from "react";

function App() {
  // Here we have an array called companies with 5 JSON objects
  // representing companies. Each object has four properties.
  const companies = [
    { cid: 1, company: "Cassin Inc", employees: 105, material: "Wood" },
    { cid: 2, company: "Koch Inc", employees: 62, material: "Wood" },
    { cid: 3, company: "Legros-Haley", employees: 18, material: "Tile" },
    { cid: 4, company: "Grant Wisoky",employees: 61,material: "Steel"},
    { cid: 5, company: "Hills LLC", employees: 89, material: "Glass" }
  ];
  // Here we use a map function (applied to the companies array)
  // we use an index to create a UNIQUE KEY or value for each <p>
  const companiesJSX = companies.map((c, index) => (
    <p key={index}>
      <b>{c.company}</b>,Emp: {c.employees}, {c.material}
    </p>
  ));
  // Finally, the return statement - the JSX generated is then returned
  // and RENDERED as your 'app'.

  return <>{companiesJSX}</>;
}
export default App; // notice how this matches the function App()
```

Rendering the array

Return statement

Browser Tests Terminal

https://fxkqlt.csb.app/

Cassin Inc,Emp: 105, Wood

Koch Inc,Emp: 62, Wood

Legros-Haley,Emp: 18, Tile

Grant Wisoky,Emp: 61, Steel

Hills LLC,Emp: 89, Glass

Console 1 Problems 0 React DevTools 0

No problems!

This is our array – containing JSON objects

- **Each object is contained within curly brackets**
- Each object has **properties** – these **`cid`, `company`, `employees`, `material`**
- Property names should NEVER change within arrays of objects


```
// Here we have an array called companies with 5 JSON objects
// representing companies. Each object has four properties.
const companies = [
  { cid: 1, company: "Cassin Inc", employees: 105, material: "Wood" },
  { cid: 2, company: "Koch Inc", employees: 62, material: "Wood" },
  { cid: 3, company: "Legros-Haley", employees: 18, material: "Tile" },
  { cid: 4, company: "Grant Wisoky",employees: 61,material: "Steel"},
  { cid: 5, company: "Hills LLC", employees: 89, material: "Glass" }
];
```

(1) The `map function`

- Probably one of the most important Javascript functions you will (ever) learn.
- We APPLY the map function to the companies array.
- Notice the map function generates JSX (Javascript and HTML)

```
13    // Here we use a map function (applied to the companies array)
14    // we use an index to create a UNIQUE KEY or value for each <p>
15    const companiesJSX = companies.map((c, index) => (
16      <p key={index}>
17        <b>{c.company}</b>,Emp: {c.employees}, {c.material}
18      </p>
19    ));
```

(2) The `map function`

- We'll learn in later lectures why we need the '**`index`**' variable – for now let's just include it as good practice.
- The variable **`c`** is for iterating across the entire **`companies`** array.
- **Note how we use `{ }` and the property names**

```
13    // Here we use a map function (applied to the companies array)
14    // we use an index to create a UNIQUE KEY or value for each <p>
15    const companiesJSX = companies.map((c, index) => (
16      <p key={index}>
17        <b>{c.company}</b>,Emp: {c.employees}, {c.material}
18      </p>
19    ));
```

The map function is used to RENDER or print the array

- The **map function** will work REGARDLESS of the size of the array – 0 elements, 1 element, …. 10,000 elements…

JS App.js

```
import React from "react";

function App() {
  // Here we have an array called companies with 5 JSON objects
  // representing companies. Each object has four properties.
  const companies = [
    { cid: 1, company: "Cassin Inc", employees: 105, material: "Wood" },
    { cid: 2, company: "Koch Inc", employees: 62, material: "Wood" },
    { cid: 3, company: "Legros-Haley", employees: 18, material: "Tile" },
    { cid: 4, company: "Grant Wisoky",employees: 61,material: "Steel"},
    { cid: 5, company: "Hills LLC", employees: 89, material: "Glass" }
  ];
  // Here we use a map function (applied to the companies array)
  // we use an index to create a UNIQUE KEY or value for each <p>
  const companiesJSX = companies.map((c, index) => (
    <p key={index}>
      <b>{c.company}</b>,Emp: {c.employees}, {c.material}
    </p>
  ));
  // Finally, the return statement - the JSX generated is then returned
  // and RENDERED as your 'app'.

  return <>{companiesJSX}</>;
```

Browser Tests Terminal

https://fxkqlt.csb.app/

Cassin Inc,Emp: 105, Wood

Koch Inc,Emp: 62, Wood

Legros-Haley,Emp: 18, Tile

Grant Wisoky,Emp: 61, Steel

Hills LLC,Emp: 89, Glass

In-class demo

- The map function and an array of Javascript objects

SCAN ME

Let’s try a quick POPUP QUIZ

- No C/A marks – this is for your own learning.
- You can work together – use your smartphone or laptop

SCAN ME

CS385

Moodle Page

Topic 1

Look for “Topic-1-Popup-Quiz”

The map function and arrays

- If you can work to understand the previous example then you're well on your way to grasping one of the most useful concepts in Javascript/React.
- The map function will work regardless of the number of objects in the array.
- We, as the programmer, decides how we print out the values of each property (in our example we put the name property value in bold text)

See you on Tuesday 3rd October for Lecture 3 + 4 (16:00 – 18:00)

There is NO laboratory session this week

All content available on Moodle