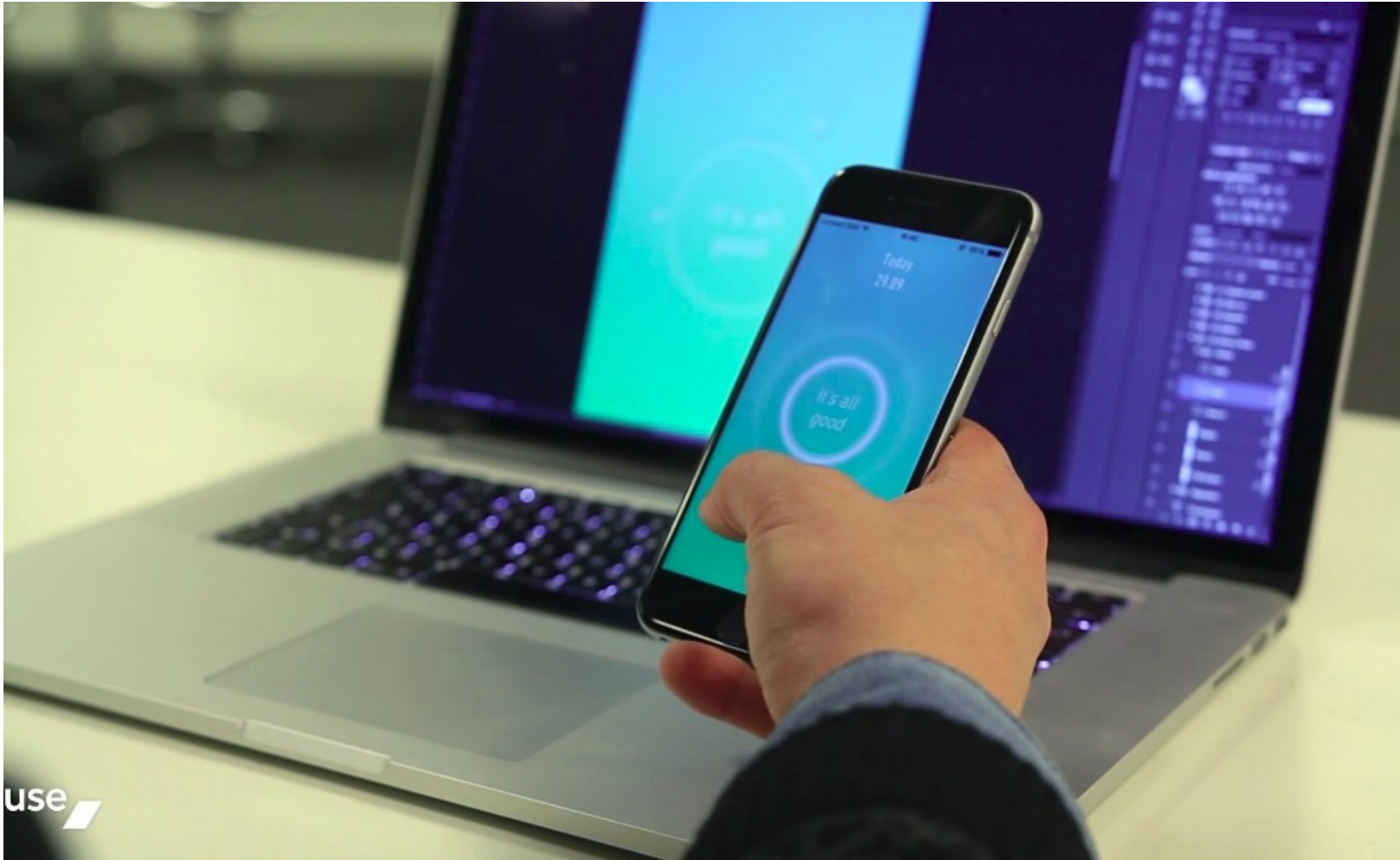


# CS385 Mobile Application Development (Lecture 2)



**Peter Mooney**

# Self Study Exercise – Follow the W3Schools Tutorial

A screenshot of the W3Schools website showing the 'HTML Introduction' page. The browser's address bar shows 'https://www.w3schools.com/html/html\_intro.asp'. The navigation menu at the top includes links for HTML, CSS, JavaScript, SQL, PHP, Bootstrap, How To, Python, W3.CSS, and JQuery. The left sidebar lists various HTML topics, with 'HTML Introduction' highlighted. The main content area is titled 'HTML Introduction' and includes a 'Previous' button. Below the title, it asks 'What is HTML?' and provides a definition: 'HTML is the standard markup language for creating Web pages.' It then lists several bullet points about HTML. Further down, it shows 'A Simple HTML Document' with a code snippet for a basic HTML document structure. A large blue arrow points from the text 'STOP HERE' to the 'HTML Id' item in the left sidebar.

← → ↺ 🏠

📄 🔒 https://www.w3schools.com/html/html\_intro.asp

🏠 **HTML** CSS JAVASCRIPT SQL PHP BOOTSTRAP HOW TO PYTHON W3.CSS JQUERY

HTML5 Tutorial

HTML HOME

**HTML Introduction**

HTML Editors

HTML Basic

HTML Elements

HTML Attributes

HTML Headings

HTML Paragraphs

HTML Styles

HTML Formatting

HTML Quotations

HTML Comments

HTML Colors

HTML CSS

HTML Links

HTML Images

HTML Tables

HTML Lists

HTML Blocks

HTML Classes

HTML Id

HTML Iframes

HTML JavaScript

HTML File Paths

HTML Head

HTML Layout

HTML Responsive

HTML Computercode

HTML Entities

HTML Symbols

## HTML Introduction

← Previous

### What is HTML?

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph"
- Browsers do not display the HTML tags, but use them to render the content

### A Simple HTML Document

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

**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**


**See video on Moodle.**

# 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.

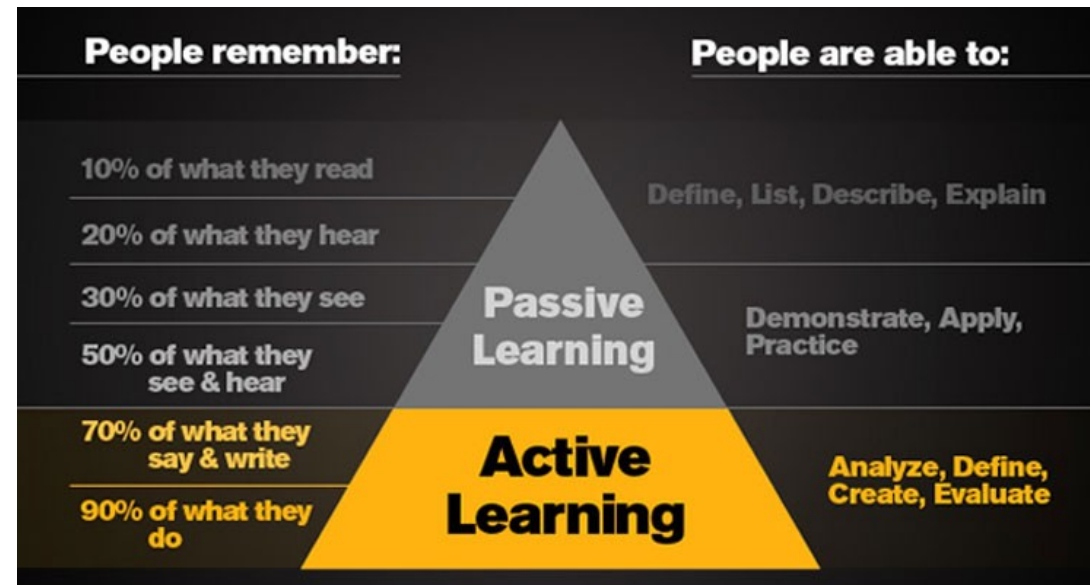
 [Using the w3schools tutorial on HTML for CS385](#)

 [Opening codesandbox.io for the first time](#)

# CS385 – Some assumptions



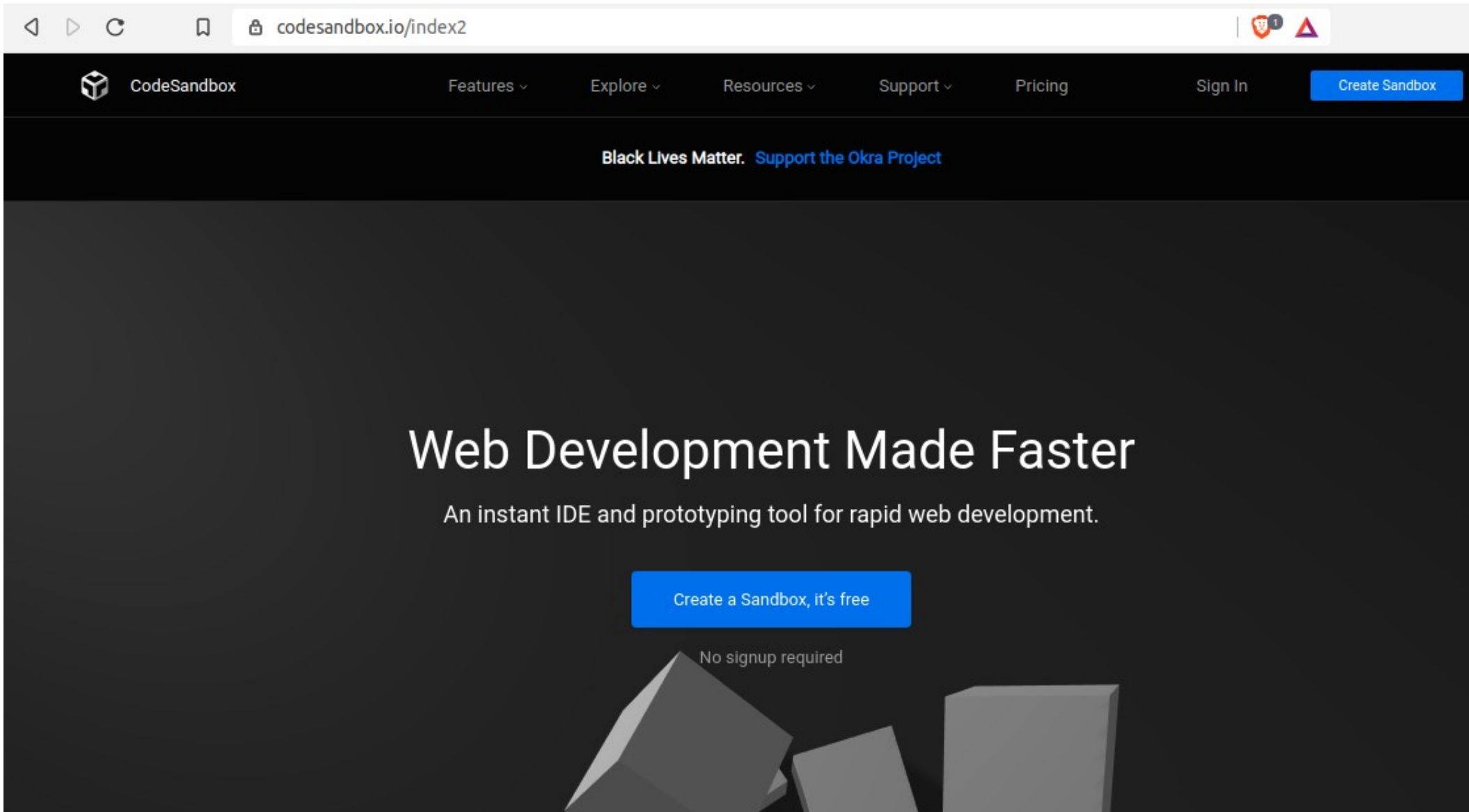
- You have programmed **before** (even if this is just your Introduction to Java) and have a **basic understanding of programming constructs**
- You will try out (run, practice) the code examples provided in lectures



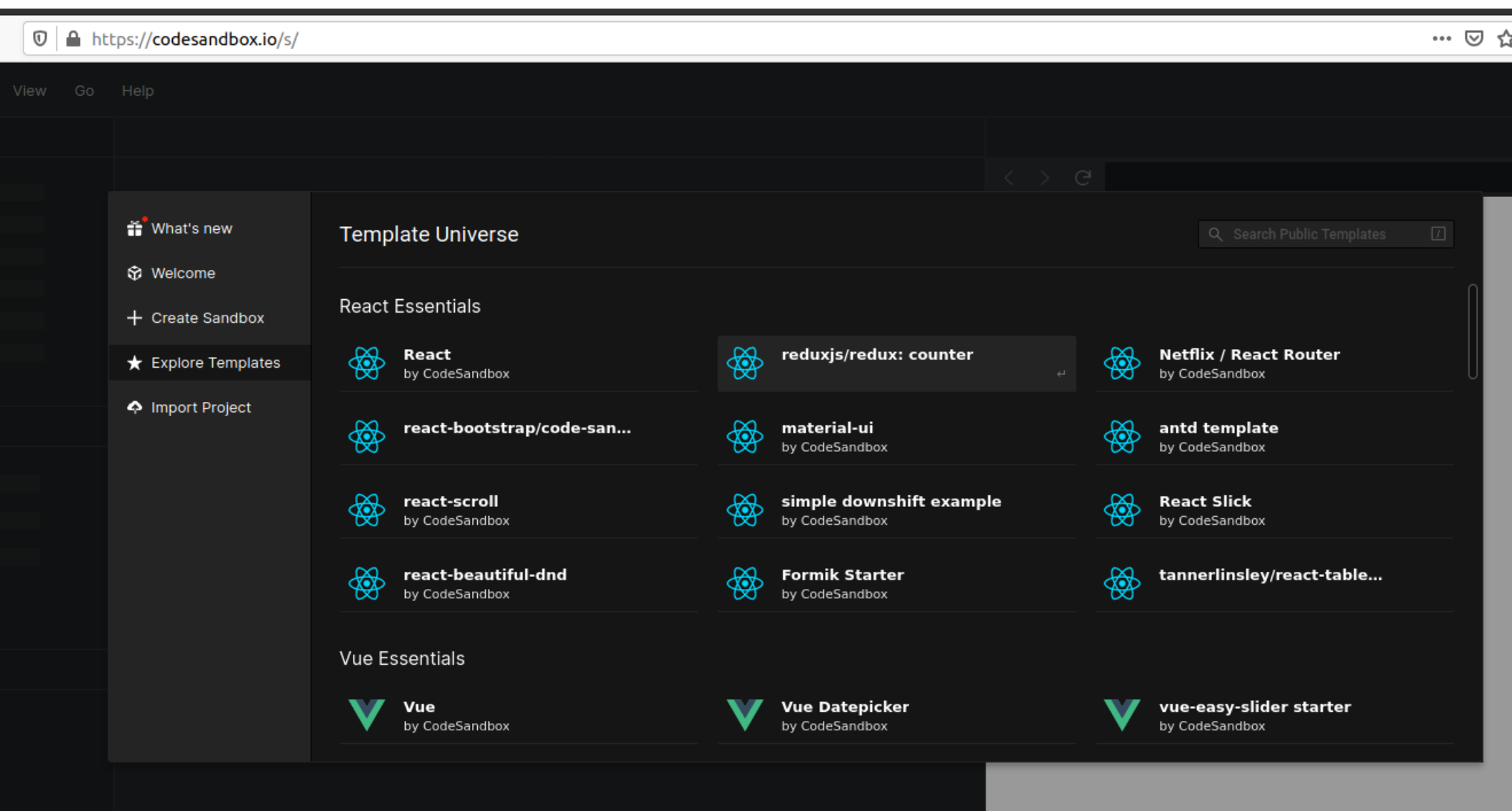
# Starting Javascript – let's look at some of the features of basic Javascript

- 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**

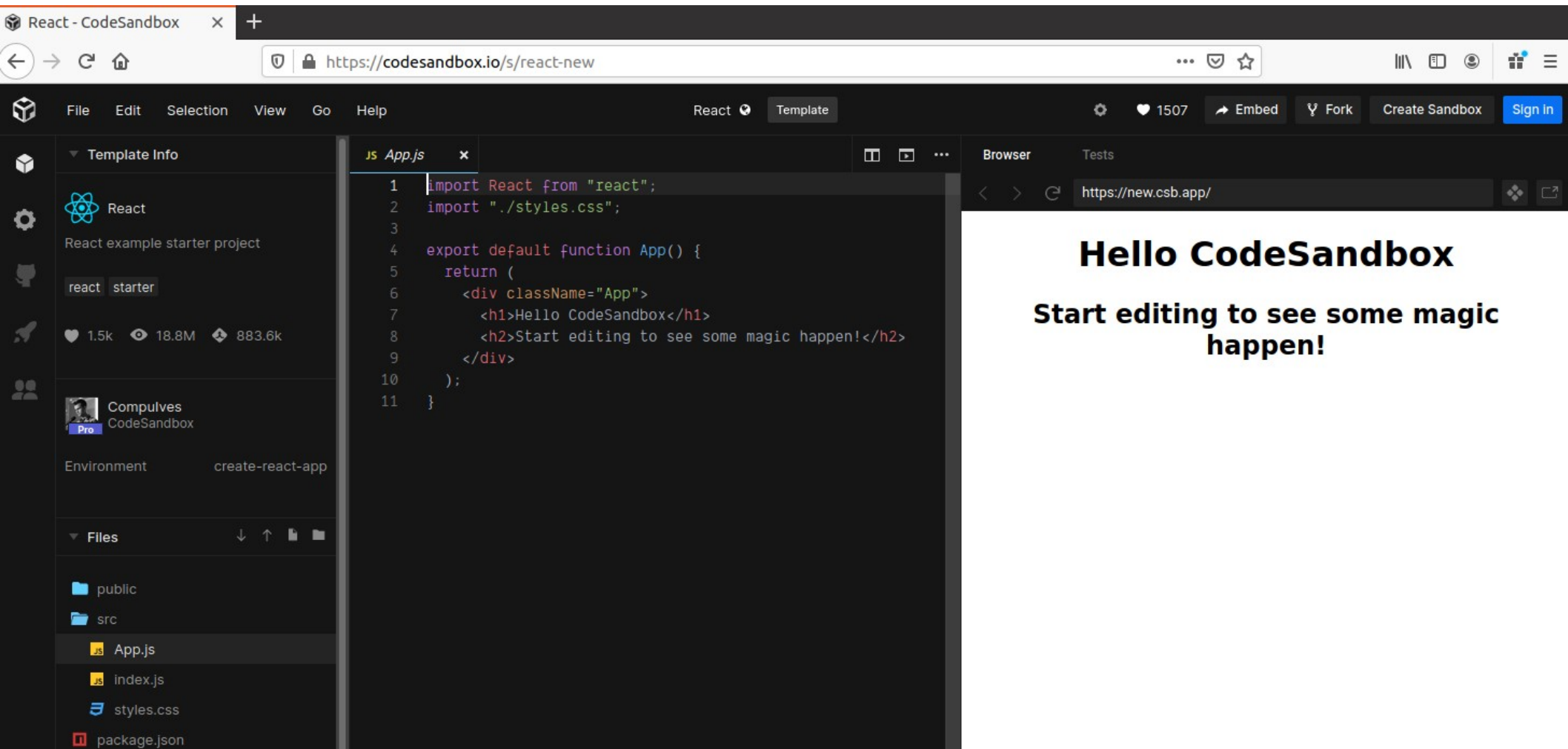


# In **codesandbox** choose a react template to start your work





# Default starting template



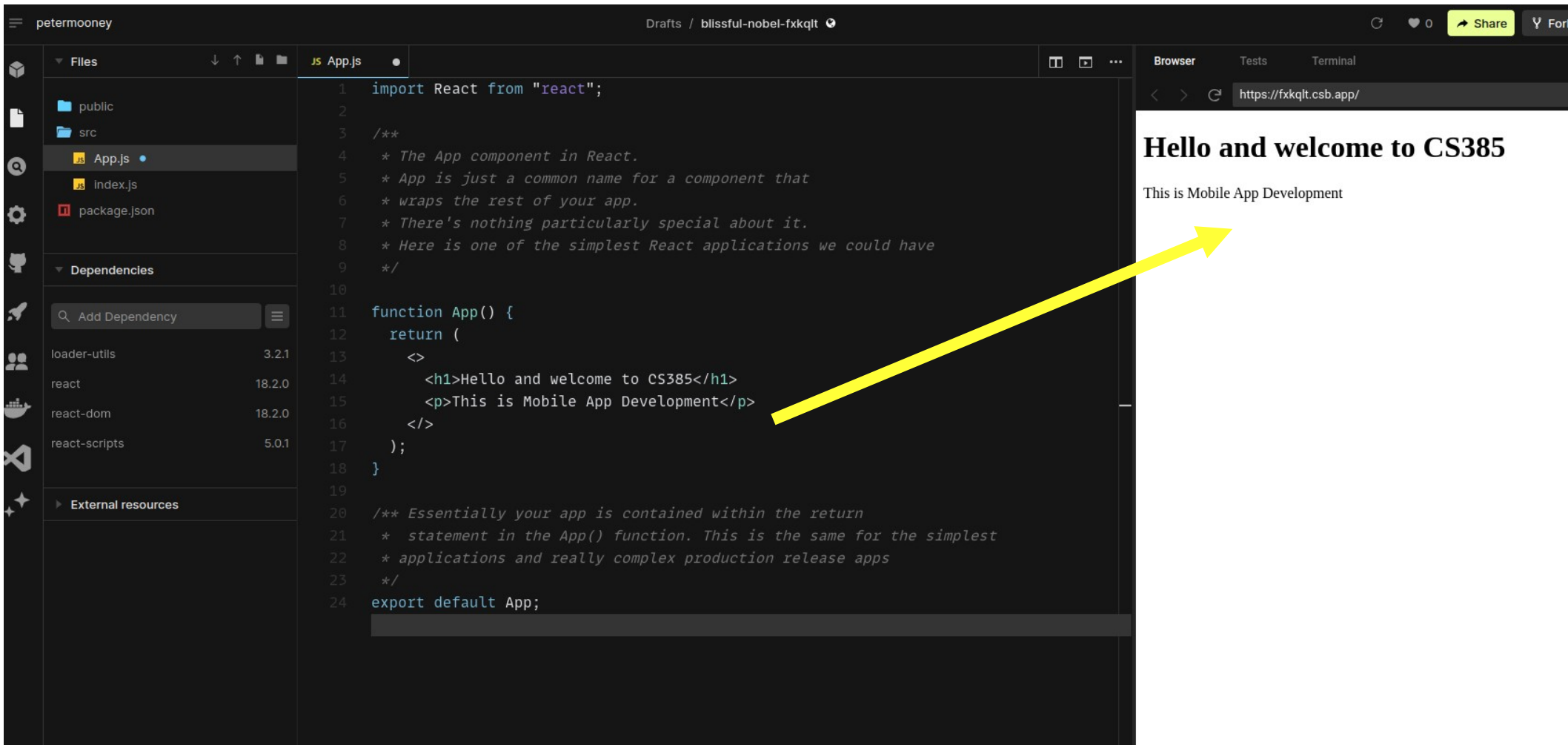
The screenshot displays the CodeSandbox web interface for a new React project. The browser address bar shows <https://codesandbox.io/s/react-new>. The interface includes a top navigation bar with 'File', 'Edit', 'Selection', 'View', 'Go', and 'Help' menus, along with a 'Template' button. On the left, the 'Template Info' panel shows the 'React' template, described as a 'React example starter project', with 1.5k likes and 18.8M views. Below this, the 'Files' panel lists the project structure: 'public', 'src', 'App.js', 'index.js', 'styles.css', and 'package.json'. The main editor area shows the 'App.js' file with the following code:

```
1 import React from "react";
2 import "../styles.css";
3
4 export default function App() {
5   return (
6     <div className="App">
7       <h1>Hello CodeSandbox</h1>
8       <h2>Start editing to see some magic happen!</h2>
9     </div>
10   );
11 }
```

On the right, the 'Browser' panel shows the rendered output of the application, which displays the text 'Hello CodeSandbox' and 'Start editing to see some magic happen!'.



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



The screenshot shows a code editor interface with a file explorer on the left, a code editor in the center, and a browser preview on the right. The file explorer shows a project structure with 'public' and 'src' folders, and files 'App.js', 'Index.js', and 'package.json'. The code editor shows the content of 'App.js' with the following code:

```
1 import React from "react";
2
3 /**
4  * The App component in React.
5  * App is just a common name for a component that
6  * wraps the rest of your app.
7  * There's nothing particularly special about it.
8  * Here is one of the simplest React applications we could have
9  */
10
11 function App() {
12   return (
13     <>
14     <h1>Hello and welcome to CS385</h1>
15     <p>This is Mobile App Development</p>
16     </>
17   );
18 }
19
20 /** Essentially your app is contained within the return
21  * statement in the App() function. This is the same for the simplest
22  * applications and really complex production release apps
23  */
24 export default App;
```

The browser preview on the right shows the rendered output of the code, displaying the text "Hello and welcome to CS385" and "This is Mobile App Development". A yellow arrow points from the code in the code editor to the rendered output in the browser.

# This is a very simple fully working React application

```
1  import React from "react";
2
3  /**
4   * The App component in React.
5   * App is just a common name for a component that
6   * wraps the rest of your app.
7   * There's nothing particularly special about it.
8   * Here is one of the simplest React applications we could have
9   */
10
11  function App() {
12    return (
13      <>
14        <h1>Hello and welcome to CS385</h1>
15        <p>This is Mobile App Development</p>
16      </>
17    );
18  }
19
20  /** Essentially your app is contained within the return
21   * statement in the App() function. This is the same for the simplest
22   * applications and really complex production release apps
23   */
24  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

```
1  import React from "react";
2
3  /**
4   * The App component in React.
5   * App is just a common name for a component that
6   * wraps the rest of your app.
7   * There's nothing particularly special about it.
8   * Here is one of the simplest React applications we could have
9   */
10
11 function App() {
12   return (
13     <>
14       <h1>Hello and welcome to CS385</h1>
15       <p>This is Mobile App Development</p>
16     </>
17   );
18 }
19
20 /** Essentially your app is contained within the return
21  * statement in the App() function. This is the same for the simplest
22  * applications and really complex production release apps
23  */
24 export default App;
```

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

JS App.js

```
1 import React from "react";
2
3 /**
4  * The App component in React.
5  * App is just a common name for a component that
6  * wraps the rest of your app.
7  * There's nothing particularly special about it.
8  * Here is one of the simplest React applications we could have
9  */
10
11 function App() {
12   return (
13     <>
14       <h1>Hello and welcome to CS385</h1>
15       <p>This is Mobile App Development</p>
16       <h2>Watch as your app renders immediately</h2>
17     </>
18   );
19 }
20
21 /** Essentially your app is contained within the return
22  * statement in the App() function. This is the same for the simplest
23  * applications and really complex production release apps
24  */
25 export default App;
26
```

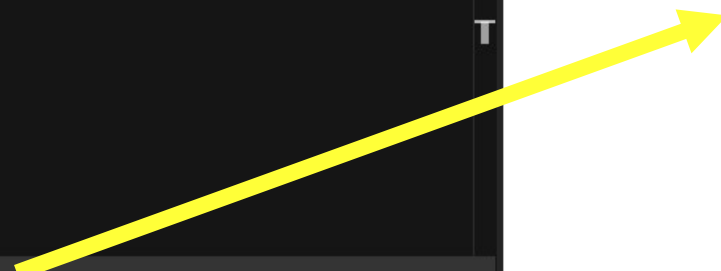
Browser Tests Terminal

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

```
1  import React from "react";
2  /**
3   * The App component in React. Here we have a very simple application.
4   * Notice the return statement:
5   * Each piece of Javascript is enclosed in curly brackets.
6   * Try refreshing the browser page and see how the time changes.
7   * The app gives a different message depending on the value of y
8   */
9
10 function App() {
11   // This is the App function or "component"
12   let x = 10;
13   let y = Math.random() * 10; // Javascript generates a random number
14   // we get the current date and time from Javascript
15   let currentTime = new Date().toLocaleString();
16   let message = "Blank"; // message is a string
17   if (y >= 5) {
18     message = "Goodbye, CS385";
19     x = y;
20   } else {
21     message = "Hello, CS385";
22     x = y;
23   }
```



SCAN ME

# Variables and rendering (example given in class)

```
JS App.js
22   x = y;
23   }
24   // return - this is where the JSX or your Javascript and HTML
25   // is returned to the calling environment and rendered as your 'app'
26   return (
27     <>
28       <p>
29         <h1>{message}</h1> Rendered at {currentTime}
30       </p>
31       <p>To render or print the value of x we write {x}</p>
32       <p>
33         To render or print the value of an expression such as 10 times x we
34         write {10 * x}
35       </p>
36       <p>Value of 4 times x plus 100 is {4 * x + 100}</p>
37       <p>
38         We can use Javascript functions inside curly brackets such as rounding
39         {Math.round(4 * x + 100)}
40       </p>
41       <p>
42         We can use String functions inside curly brackets such as
43         <b>{message.toUpperCase()}</b> or <b>{message.toLowerCase()}</b>
44       </p>
45     </>
46   );
47 }
48 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 brackets such as rounding134

We can use String functions inside curly brackets such asGOODBYE, CS385 or goodbye, cs385

Console Problems React DevTools

No problems!

# The curly braces or brackets { } in React Javascript

- We can evaluate ANY valid piece of Javascript inside the curly braces or brackets
- 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

Share

JS App.js

## Our array of objects

```
1 import React from "react";
2
3 function App() {
4   // Here we have an array called companies with 5 JSON objects
5   // representing companies. Each object has four properties.
6   const companies = [
7     { cid: 1, company: "Cassin Inc", employees: 105, material: "Wood" },
8     { cid: 2, company: "Koch Inc", employees: 62, material: "Wood" },
9     { cid: 3, company: "Legros-Haley", employees: 18, material: "Tile" },
10    { cid: 4, company: "Grant Wisoky", employees: 61, material: "Steel" },
11    { cid: 5, company: "Hills LLC", employees: 89, material: "Glass" }
12  ];
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  ));
20  // Finally, the return statement - the JSX generated is then returned
21  // and RENDERED as your 'app'.
22
23  return <>{companiesJSX}</>;
24 }
25 export default App; // notice how this matches the function App()
26
```

## 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

Problems

React DevTools

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

```
4 // Here we have an array called companies with 5 JSON objects
5 // representing companies. Each object has four properties.
6 const companies = [
7   { cid: 1, company: "Cassin Inc", employees: 105, material: "Wood" },
8   { cid: 2, company: "Koch Inc", employees: 62, material: "Wood" },
9   { cid: 3, company: "Legros-Haley", employees: 18, material: "Tile" },
10  { cid: 4, company: "Grant Wisoky", employees: 61, material: "Steel" },
11  { cid: 5, company: "Hills LLC", employees: 89, material: "Glass" }
12 ];
```



# (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 ));
```

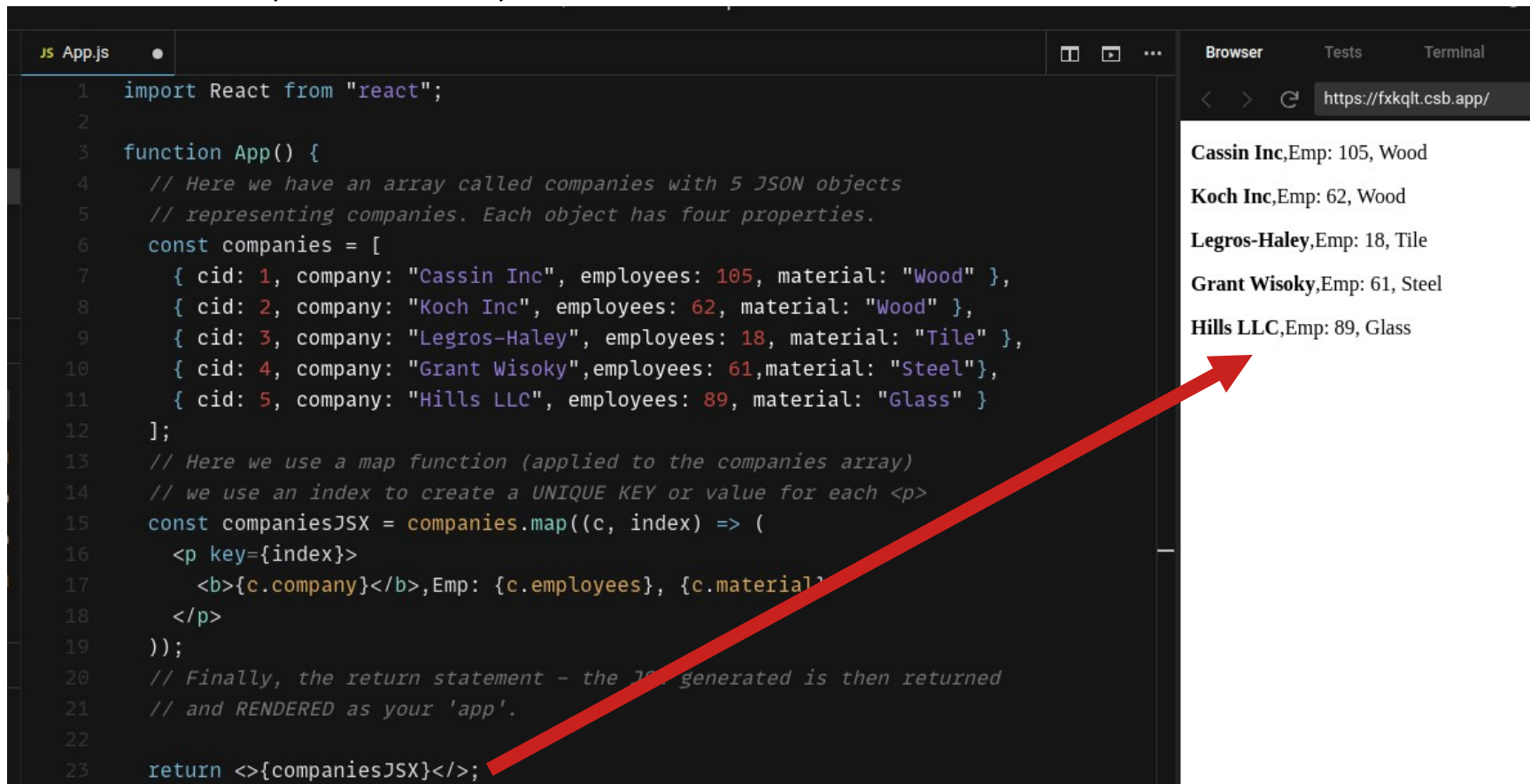
## (2) The **map** function

- 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...



The screenshot shows a code editor with a file named 'App.js'. The code defines a function 'App()' that uses the 'map' function to render an array of company objects. A red arrow points from the 'map' function call in the code to the rendered output in the browser.

```
1 import React from "react";
2
3 function App() {
4   // Here we have an array called companies with 5 JSON objects
5   // representing companies. Each object has four properties.
6   const companies = [
7     { cid: 1, company: "Cassin Inc", employees: 105, material: "Wood" },
8     { cid: 2, company: "Koch Inc", employees: 62, material: "Wood" },
9     { cid: 3, company: "Legros-Haley", employees: 18, material: "Tile" },
10    { cid: 4, company: "Grant Wisoky", employees: 61, material: "Steel" },
11    { cid: 5, company: "Hills LLC", employees: 89, material: "Glass" }
12  ];
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  ));
20  // Finally, the return statement - the JSX generated is then returned
21  // and RENDERED as your 'app'.
22
23  return <>{companiesJSX}</>;
```

Browser: <https://fxxkqlt.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



# 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



**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 3<sup>rd</sup> October  
for Lecture 3 + 4 (16:00 – 18:00)**

**There is NO laboratory session  
this week**

**All content available on Moodle**