

JavaScript Frameworks

Day 1

What you will learn

- Introduction to React including
 - JSX
 - Components
 - Event Handling
 - Props and State
 - Routing
- How to Create an interactive web application using a REST API with React

Emmet with React Components in VS Code

- Emmet may not be enabled when typing HTML in React JSX components (more on JSX and components in upcoming lessons)
- To enable Emmet auto complete in your React JSX components add the following code to your “settings.json” file in VS Code
- **Note:** if you have other code in your “settings.json” file, then make sure to add a “,” after the last “}” before you add the “emmet.include...” code

```
"emmet.includeLanguages": {  
  "javascript": "javascriptreact"  
}
```

Access the settings.json File in VS Code

- To access the "settings.json" file in VS Code do the following:
 1. Open the Command Palette
 - CTRL+SHIFT+P Windows
 - CMD+SHIFT+P macOS
 - Via the menus by clicking View -> Command Palette
 2. With the Command Palette open type "settings.json"
 3. Click on "Preferences: Open Settings (JSON)"

React Developer Tools for Chrome

- To help with developing and troubleshooting your React apps, the people behind React have created a Chrome extension called React Developer Tools
- Navigate to the URL below from within Chrome to install the extension
 - <https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en>

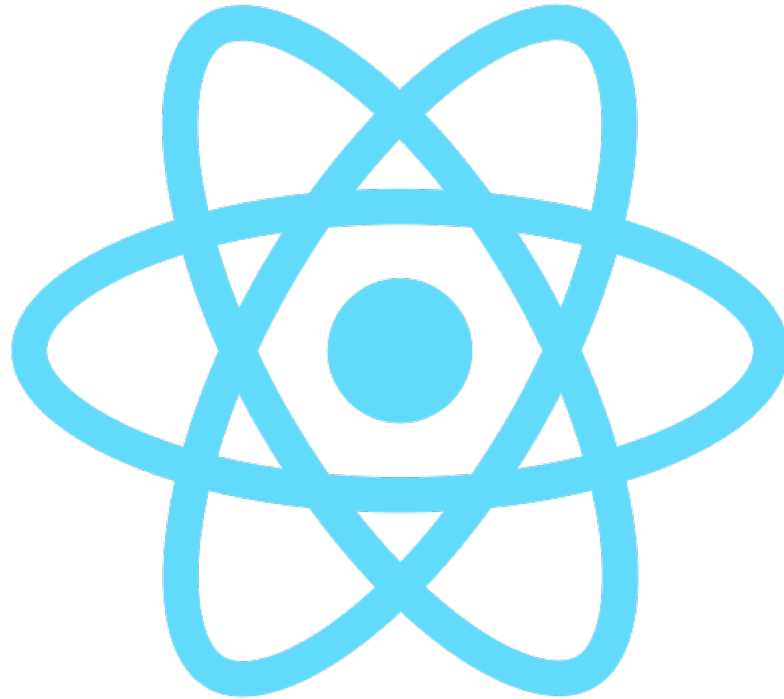
React Documentation

- Visit the official React JS web page for documentation on React and for some tutorials on how to use React
 - <https://reactjs.org/>

Lesson Demos

- For each day of the course you will be provided with lesson demo files
- Since React application development requires many Node dependencies, it is not practical to share demos with all the Node dependencies downloaded
- In order to run the code demos you will need to run “npm install” on any code demo folder that uses React
 - The command “npm install” tells the terminal to go to the internet and download and install all the Node dependencies that are required for the application
- We walk you through the process of how to do this together in class

Introduction to React



What is React

- React is a JavaScript library
- Developed by Facebook
- MIT licensed (free to use)
- Designed to facilitate easier development of interactive user interfaces
- React creates a virtual DOM that compares the states of elements and will only update the part or parts of the DOM that need updating
 - This makes DOM manipulation very performant when using React

Why use React?

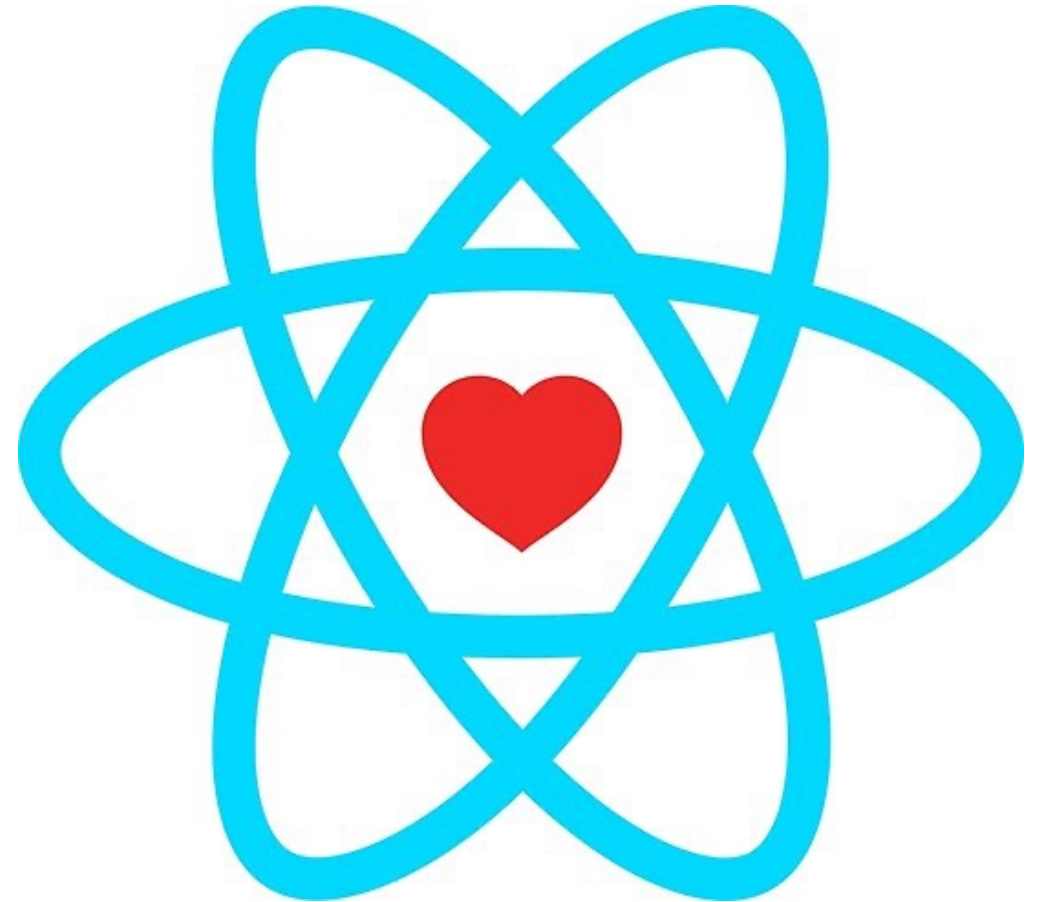
- Easy to get started in React (once your dev environment is setup)
 - If you know JavaScript, you can work in React
 - No new syntax (except for JSX)
 - React leans heavily on standard ES6 JavaScript Syntax
 - React uses JSX
 - JSX makes creating interactive HTML templates easy
 - JSX allows you to write HTML directly in JavaScript

Why use React?

- One of the most popular front-end frameworks
 - React has large and active community
 - Lots of free online tutorials on how to use React
- React has good documentation
- Developers who know how to use React are in demand

Why use React?

- It is one of the most loved frameworks as decided by you, developers!!!
- See the 2020 Stack Overflow Developer survey:
 - [Most Loved Web Frameworks](https://survey.stackoverflow.co/2020/#frameworks-most-loved)



How React Views the World

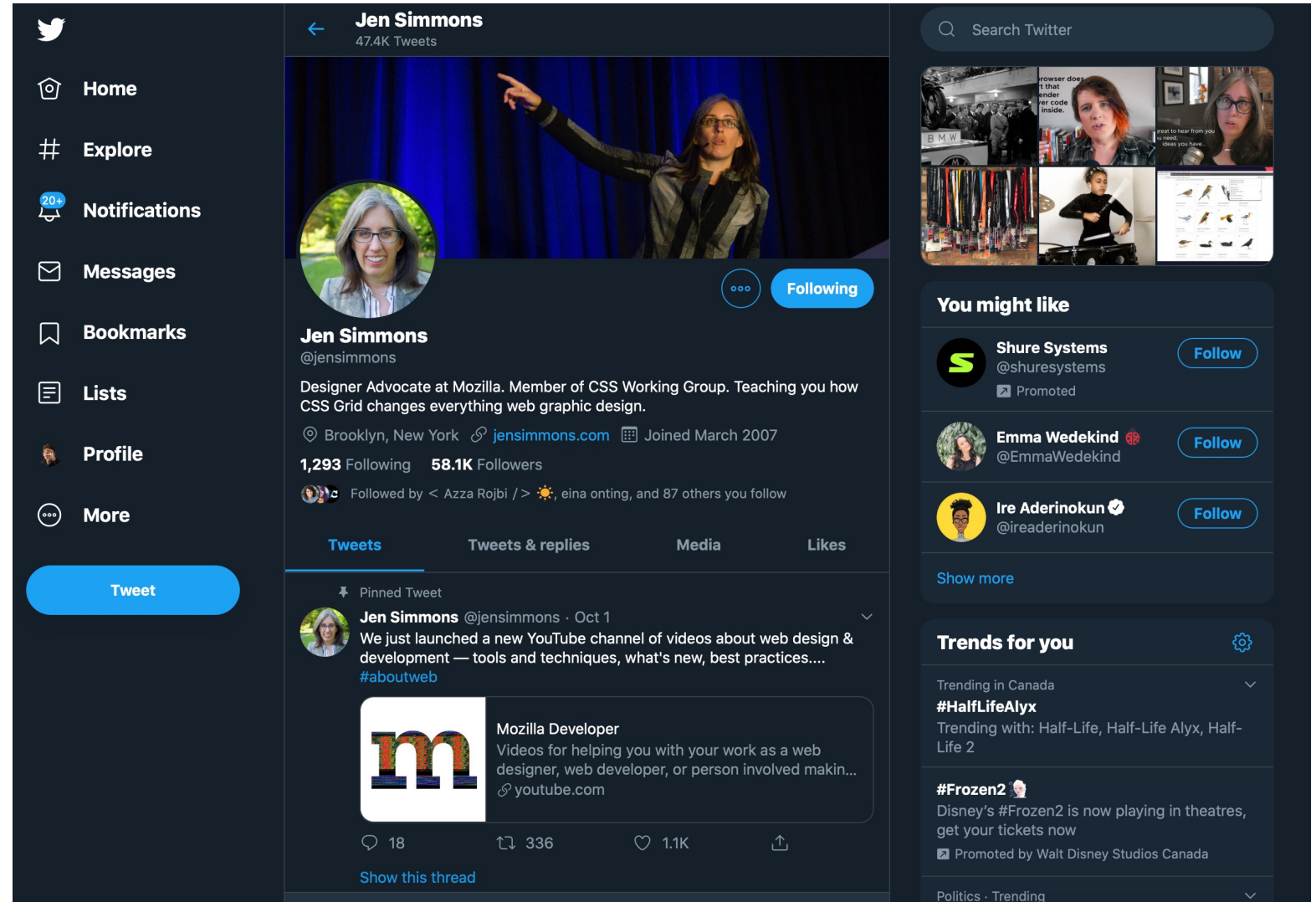
(Hint: It's all about the component)

Everything is a Component

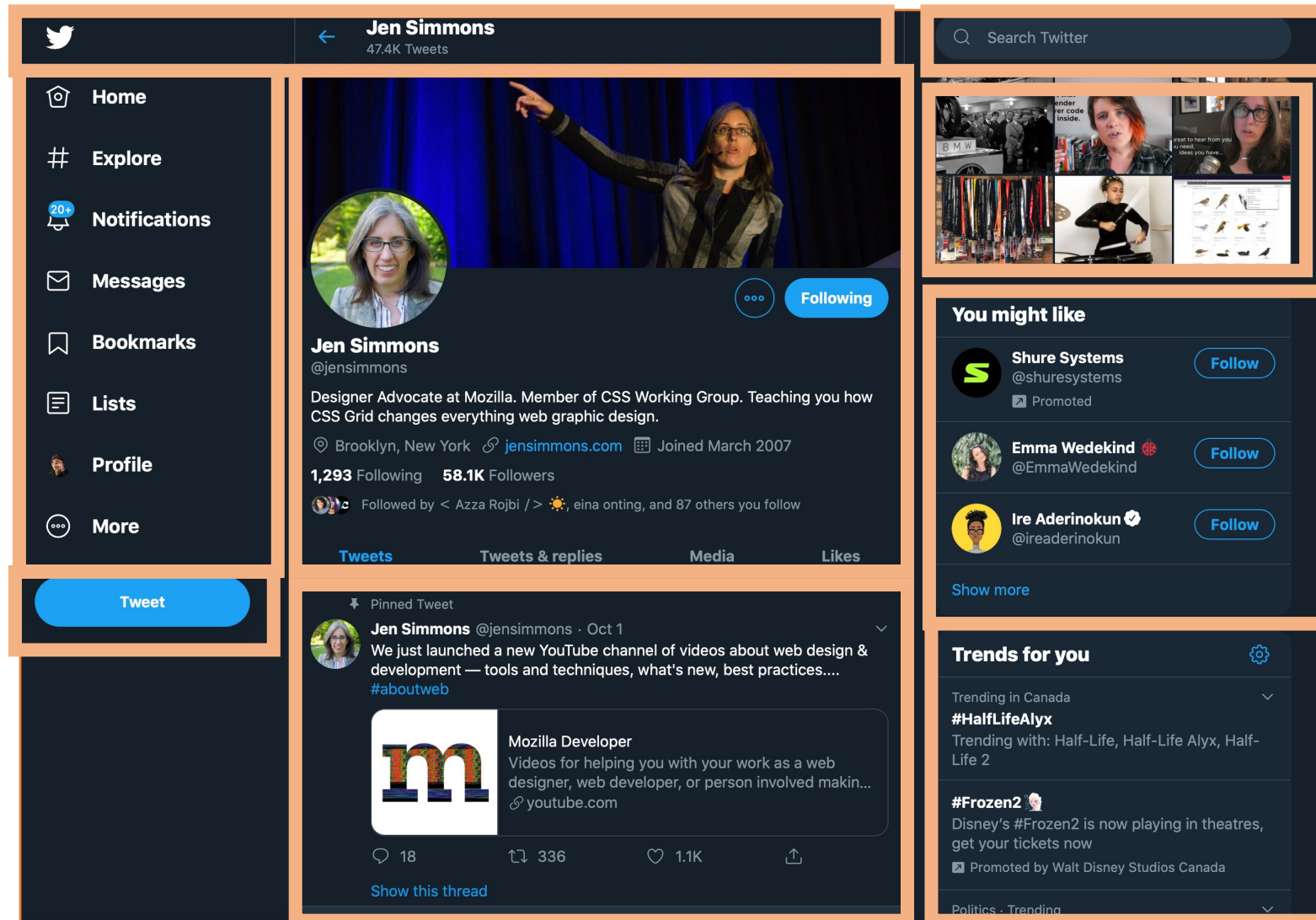
- When building a React web application, it is best to think about everything in your application as a component
 - A component could be:
 - A static header or footer element
 - An interactive log in form element
 - An interactive shopping cart element
 - A blog post
 - A search component
 - A component can have child components

Think in Components

- I don't know if Twitter uses React, but we could easily build a Twitter clone with React



Think in Components



Create React App

- Create React App provides a simple one click install that installs all the basic tools you need for creating a React web application including:
 - React
 - JSX Transpiler
 - ES6 Support
 - A development web server
 - A build tool for production

Create React App – How to Setup

- Easy...
- Make sure your version of npm is at least at version 5.2 (for an easier install experience)
- CD (change directory) to the parent directory of where you want to install your application from within Terminal or CMD prompt or PowerShell
- Run the install command (details on next slide)
- CD into your applications folder (details on next slide)
- Type: npm start

Create React App – Terminal Install

- Installation

Change directory into the parent directory of where you want to install your React app

```
cd /Users/joe/Documents/Apps
```

Run the command below to install the react starter app

```
npx create-react-app my-app
```

This is the folder name where you want to install the React app. It can be called whatever you like (use web safe file names to be safe)

```
cd my-app
```

Change directory into the directory where the React app was installed into

```
npm start
```

This command will fire up a web server. To view your app in the browser, use this URL:
<http://localhost:3000/>

Create React App Install Error

- If you get the error:
 - You are running `create-react-app` [old version number], which is behind the latest release...
- Then run the modified install command below:

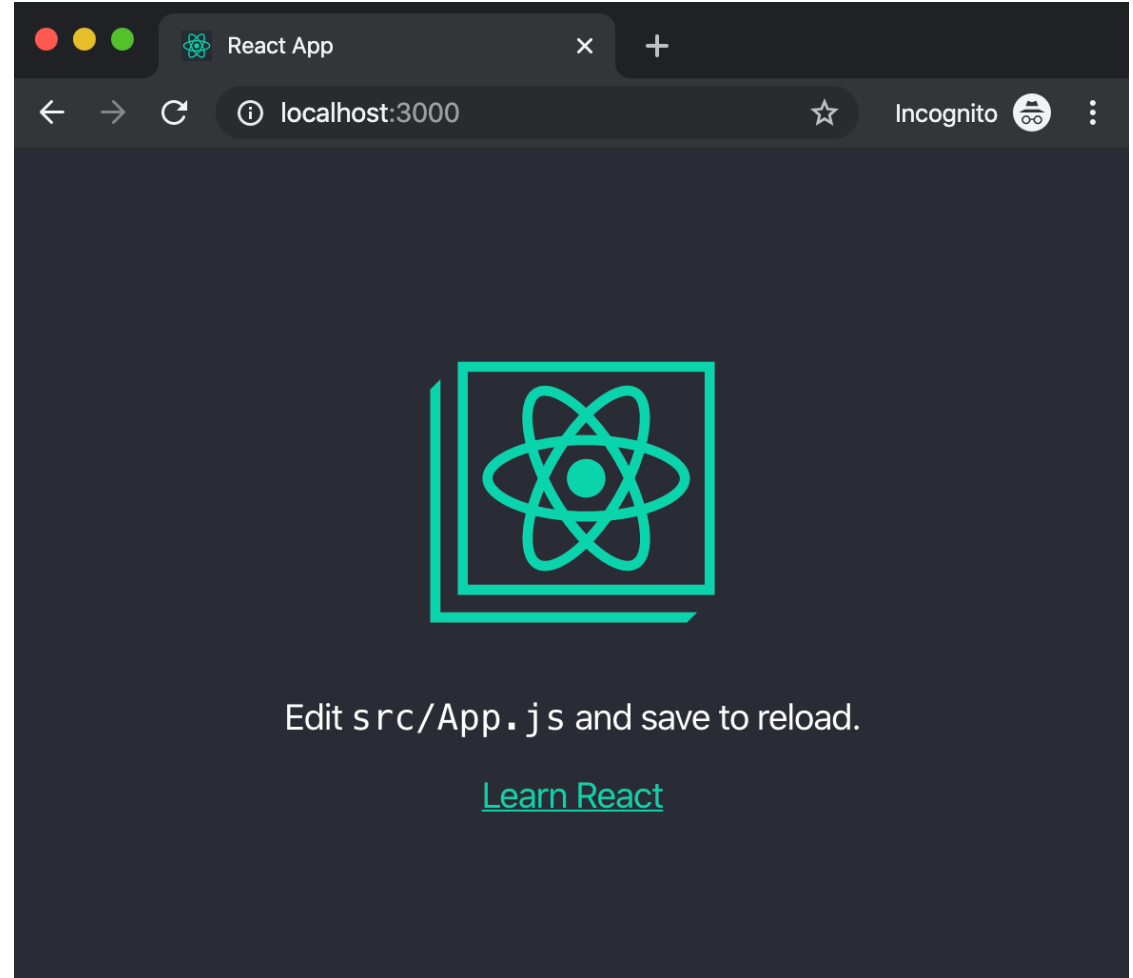
```
npx create-react-app@latest my-app
```



Add @latest to the install command
to fix the error listed above

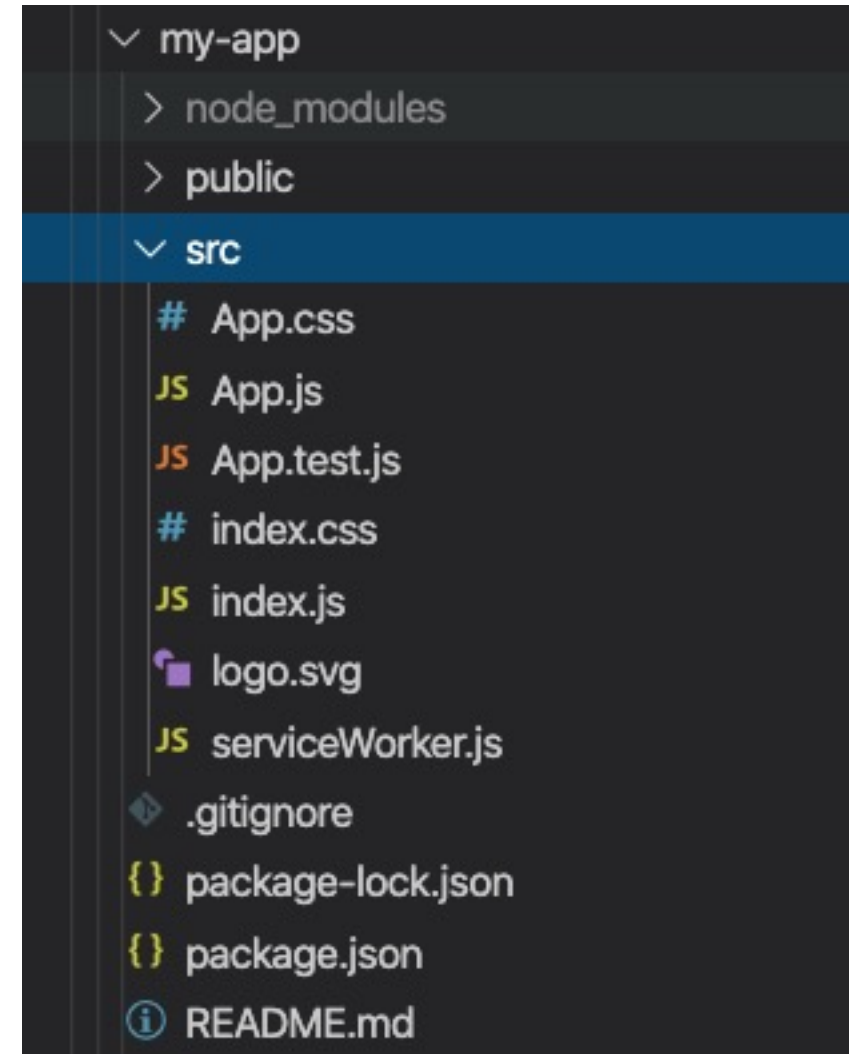
Create React App – Initial Page Load

- The first time you run the app you will see this default screen



Create React App – Folder Structure

- You will do your developing in the "src" folder
 - Edit the JavaScript in "App.js"
 - App styles can be edited in "App.css"
 - More on styling your app in a future lesson
 - The main entry point and render of your app is done in the "index.js" file



Modern Alternatives to Create React App

- Create React App has been deprecated by React
 - Create React App still works and can still be used for new applications
 - React is pushing developers that are creating new applications to use modern frameworks which include many of the features needed when building a full-stack web application
 - Some modern alternative frameworks
 - NextJS
 - Remix
 - Gatsby
- This course will not cover other frameworks that use React, but it will teach you the core concepts of React. Understanding these core concepts will enable you to quickly learn any existing or future frameworks that use React.
- You're welcome to use any modern framework for any of your upcoming projects in this course
 - If using a modern framework for another course, be sure to ask your instructor for approval first

Alternative React Tooling System

- Vite is not a framework but it offers an alternative way to start and build a React-based web application
- If you want to learn more about Vite, you can visit vitejs.dev
- You're welcome to use Vite on any future projects for this course
 - If using a Vite for another course, be sure to ask your instructor for approval first

Why Are We Still Learning Create React App?

- **Foundation for React Development:** Create-React App (CRA) is a widely used tool. By learning CRA, you'll gain a solid understanding of React project structure, which is essential for any developer starting with React.
- **Community Support:** CRA is a widely adopted tool, which means you'll find a wealth of resources, documentation, and support from the React community. This makes troubleshooting and learning much easier.
- **Transferable Skills:** The knowledge you gain from working with CRA can be transferred to other tools and frameworks, as the concepts of project setup, build process, and dependency management are common across modern web development.
- **Future-Proof:** Even as new tools and frameworks emerge, the skills and knowledge you gain from learning CRA will remain valuable, as it continues to be a popular choice for developers and businesses alike.
- **Legacy Applications:** Create React App (CRA) has been around for a long time and has been used by many developers to create React applications. As a result, there are many legacy React applications that were built using CRA. If you're working on one of these applications, it might make sense to continue using CRA for consistency and ease of maintenance.

JSX 101

An Introduction to JSX Syntax

What is JSX

- Writing HTML markup in JavaScript is a pain
 - Making strings like this:
 - `'<div><h1>React</h1><p>Hello from React</p></div>';`
 - ...is no fun...hard to read...and hard to maintain
- JSX allows us to write HTML code in our JavaScript using regular tags and without crazy string concatenation or using hard to read template strings
- JSX allows us to easily output variable values and write JavaScript expressions directly in our template HTML code

Template String vs JSX


Regular JavaScript Template String

```
const content =  
`<div class="special">  
  <h1>${title}</h1>  
  <p>Hello ${fName(fn, ln)}</p>  
</div>`;
```

VS

JSX

```
const content = (  
  <div className="special">  
    <h1>{title}</h1>  
    <p>{formatName(fn, ln)}</p>  
  </div>  
);
```



This is JSX. HTML in JavaScript. Much easier to write and to read

One Minor Downside to JSX

- Native JavaScript does not understand JSX syntax
- JSX syntax will cause an error in native JavaScript
- This problem is easily solved by using a transpiler to convert our JavaScript code with JSX to native JavaScript code that the browser can understand

What is a transpiler?

- A transpiler converts code from one format or language into another without altering the functionality of the code

How a Transpiler Works?

- You send the code through a transpiler program
- The transpiler converts your code into a format that your output device can understand...



Create React App Includes a JSX Transpiler

- The good news is that if you use Create React App to build your React application then you are all set as Create React App includes a JSX transpiler
- If you prefer to roll your own build system than you will need to add a JSX transpiler into your build system
 - Babel is a popular JavaScript Transpiler
 - <https://babeljs.io>

JSX Syntax Rules and Best Practices

- JSX must always have a single parent element


```
const template = <h1>Welcome</h1><p>This is some information.</p>;
```



This is invalid.
Only a single
parent element
can be output in a
JSX statement

To fix this error, simply wrap
your HTML in a container
element (usually a plain div
works, but any valid HTML
element is fine)***

```
const template = (  
  <div>  
    <h1>Welcome</h1>  
    <p>This is some  
    information.</p>  
  </div>  
)
```



*** Alternatively you can use React Fragment to output multiple child elements. Give the React document article on Fragments a read for more details: <https://reactjs.org/docs/fragments.html>

JSX Syntax Rules and Best Practices

- Wrap your templates in parenthesis "(" ")" characters
 - Not a requirement, but a best practice
 - Can help with eliminating the chance of a JavaScript parser performing auto semi-colon insertion and breaking your code

```
const template = (  
  <div>  
    <h1>Welcome</h1>  
    <p>This is some  
    information.</p>  
  </div>  
) ;
```

Parenthesis

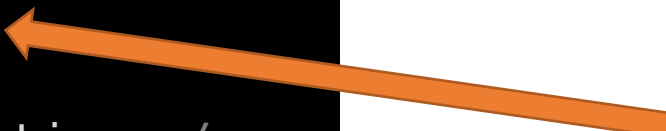


Parenthesis

JSX Syntax Rules and Best Practices

- Watch out for the "class" HTML attribute and other JavaScript reserved words in JSX
 - The word "class" is a reserved word in JavaScript and if used in JSX it will break the code
 - To fix this, simply change all mentions of the "class" attribute to "className"

```
const template = (  
  <div className="special">  
    <h1>Welcome</h1>  
    <p>This is some information.</p>  
  </div>  
);
```

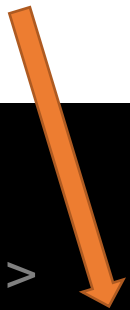


Use "className" instead of "class" in your templates

Commenting in JSX

- Standard HTML and JavaScript comments (single and multi-line) do **NOT** work in JSX
- To write a comment in JSX you must wrap the comment in "{ }"
- Multi-line comments work best inside the "{ }"
- Comments must go inside the parent element

JSX comment



```
const template = (  
  <div className="special">  
    { /* This is a JSX comment... */ }  
    <h1>Welcome</h1>  
    <p>This is some information.</p>  
  </div>  
) ;
```

JSX Expressions

- Like native template strings in JavaScript, JSX allows you to output JavaScript expressions inside your template
- You can output variable values that are strings and numbers
- You can call a function in JSX that outputs data that is a string or a number

JSX Expressions

- JSX Template with JavaScript expressions
- JavaScript expressions are always wrapped in "{ }"

```
const template = (  
  <div>  
    <h1>{app.title}</h1>  
    <h2>{app.subtitle}</h2>  
    <ol>  
      <li>Item 01</li>  
      <li>Item 02</li>  
      <li>Item 03</li>  
      <li>Item 04</li>  
      <li>Item 05</li>  
    </ol>  
    <h3>Selected Item</h3>  
    <p>{ selectItem(app.items) }</p>  
  </div>  
) ;
```

Conditional Rendering with JSX

- Although you cannot use an "if" statement in JSX as an "if" statement is a JavaScript statement and not a JavaScript expression, we still have several other options to render content conditionally in JSX
- We can render conditional content in JSX using:
 - A function call with conditional code inside of it that returns data
 - Ternary operators
 - Logical && (and) operator and logical || (or) operators

Conditional Rendering with JSX

- JSX Template with conditional rendering using a ternary operator, a logical && (and) operator and a function call

```
const template = (  
  <div>  
    <h2>About You</h2>  
    { /* Using a ternary operator to render different values  
      * Line below will either render the value of user.name  
      * if it exists, otherwise it will render the text  
      * "Anonymous" */ }  
    <h2>{user.name ? user.name : 'Anonymous'}</h2>  
    { /* Using the logical && to conditionally render elements.  
      * Line below will only render if user.age exists and  
      * user.age has an age greater than 18 */ }  
    {(user.age && user.age >= 18) && <p>Age: {user.age}</p>}  
    { /* Using a function call to conditionally render elements  
      * Line below will only render if location exists and the  
      * the location is not listed as unknown */ }  
    {getLocation(user.location)}  
  </div>  
)  
);
```



Arrays in JSX

- You can use regular array methods to parse arrays and render values as JSX
- One common error is forgetting to output a unique key for each list item
 - The key attribute is how React keeps track of elements for rendering and updating the virtual DOM
 - Each key attribute must be unique for each list item
 - Usually we obtain the "key" value from a database key, for our early examples we will just use the array index value
 - Using array index values can have side effects, so it is preferable to use a key from a database entry when working on real applications

Arrays in JSX

- A JavaScript function that returns an unordered list using JSX from an array

You must set a key attribute for each list item for React to keep track of each list item. Each key value must be unique. The key value is normally obtained from the "key" value of data coming from a database



```
function makeList(arr) {  
  const listItems = arr.map((item, i) => <li key={i}>{item}</li>);  
  return <ul>{listItems}</ul>;  
}
```

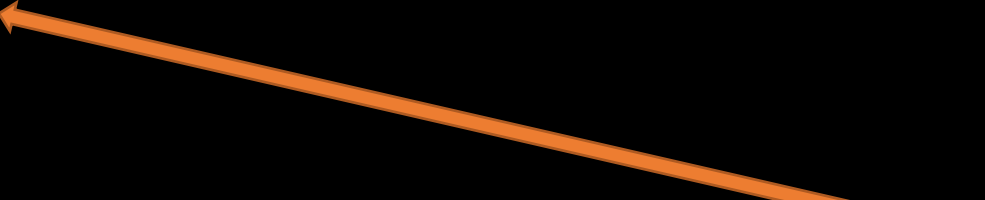
Events in React

React Component Events

- You can call events via an `on+[The Event]` attribute on the JSX component element
- See syntax on the next page

React Component Events Syntax

```
function Main() {  
  
  function sayRandomNumber(){  
    const ranNum = Math.floor(Math.random() * 1000) + 1;  
    alert(ranNum);  
  }  
  
  return (  
    <main>  
      <div className="button-container">  
        <button onClick={sayRandomNumber}>Random Number</button>  
      </div>  
    </main>  
  );  
}
```

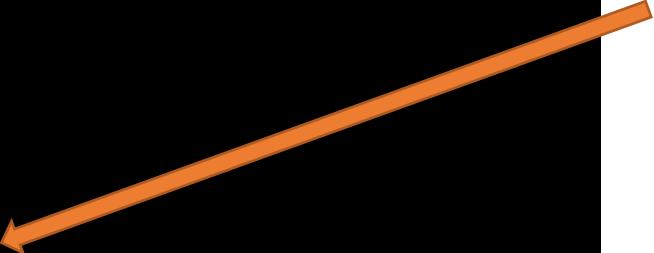


Call the event handler function via an “on+[The event type]” attribute on the component

Calling a React Event Handler with a Parameter

```
function Main({ user }) {  
  
  function sayHello(username){  
    alert(`Hello ${username}!`);  
  }  
  
  return (  
    <main>  
      <div className="button-container">  
        <button onClick={() => sayHello(user)}>Say Hello</button>  
      </div>  
    </main>  
  );  
}
```

To call an event handler that has a parameter, simply wrap the event handler function in an anonymous arrow function



Data Binding & State

Data Binding

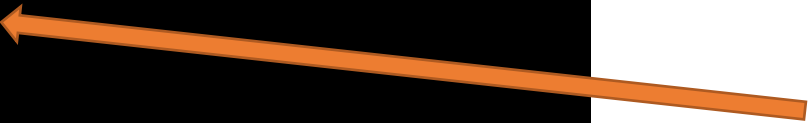
- JSX does not have built in data binding
- Data binding means connecting our templates to our data, so that when our data changes our template changes
- In React we can use the “useState” hook to sync our data to our template’s output

React State

- State in a react component is data that when changes causes the component to update with the new data
- React keeps track of a components state through one or more state objects set on the functional component
- You create a state objects via a React `useState()` hook
 - More on React hooks in a future lesson
- Never change a state variable directly, always use a state setting function to change state

Creating state in a React Component

```
import { useState } from 'react';  
const Main = () => {  
  const [num, setNum] = useState(1);  
  function add(){  
    setNum(num + 1);  
  }  
  return (  
    <main>  
      <div className="messages-container">  
        <p>The current number is: {num}</p>  
      </div>  
      <div className="button-container">  
        <button onClick={add}>Add to Number</button>  
      </div>  
    </main>  
  );  
}
```



In order to use the “useState” hook in our component we have to make sure we import the “useState” function from React


Creating state in a React Component

```
import { useState } from 'react';

const Main = () => {
  const [num, setNum] = useState(1);
  function add(){
    setNum(num + 1);
  }

  return (
    <main>
      <div className="messages-container">
        <p>The current number is: {num}</p>
      </div>
      <div className="button-container">
        <button onClick={add}>Add to Number</button>
      </div>
    </main>
  );
}
```

We set state by calling the React “useState” method. The value we pass into “useState” sets the initial value of this state object



Creating state in a React Component

```
import { useState } from 'react';

const Main = () => {
  const [num, setNum] = useState(1);
  function add(){
    setNum(num + 1);
  }

  return (
    <main>
      <div className="messages-container">
        <p>The current number is: {num}</p>
      </div>
      <div className="button-container">
        <button onClick={add}>Add to Number</button>
      </div>
    </main>
  );
}
```

The “useState” returns an array with two values:

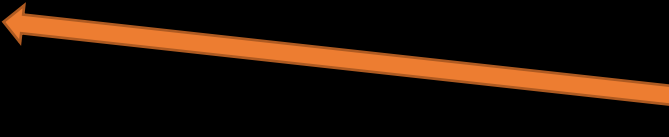
1. The initial state value
2. A function to update the state's value

Creating state in a React Component

```
import { useState } from 'react';

const Main = () => {
  const [num, setNum] = useState(1);
  function add(){
    setNum(num + 1);
  }

  return (
    <main>
      <div className="messages-container">
        <p>The current number is: {num}</p>
      </div>
      <div className="button-container">
        <button onClick={add}>Add to Number</button>
      </div>
    </main>
  );
}
```



We use destructuring* to create two variables:

1. The state's value
2. The update function that we use to update this state's value

* Give this MDN doc a read for more information on how destructuring works: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

Creating state in a React Component

```
import { useState } from 'react';

const Main = () => {

  const [num, setNum] = useState(1);

  function add(){
    setNum(num + 1);
  }

  return (
    <main>
      <div className="messages-container">
        <p>The current number is: {num}</p>
      </div>
      <div className="button-container">
        <button onClick={add}>Add to Number</button>
      </div>
    </main>
  );
}
```

We use our state updater function to update the value of the corresponding state value

When we update the value of state and our component uses that state in its returned JSX, React will automatically update the component with the new state value

Thou Shalt Never Update State Directly

***** WARNING *****

NEVER, EVER update the value of state directly. ALWAYS use your defined state updater function for updating state