the Master Course

{CODENATION}

React JS

Introduction to React JS

{CODENATION}

{CODENATION}

Learning Objectives

To explore React JS and what it's used for

To discover components and be able to write your own

To be familiar with Props

What is React?

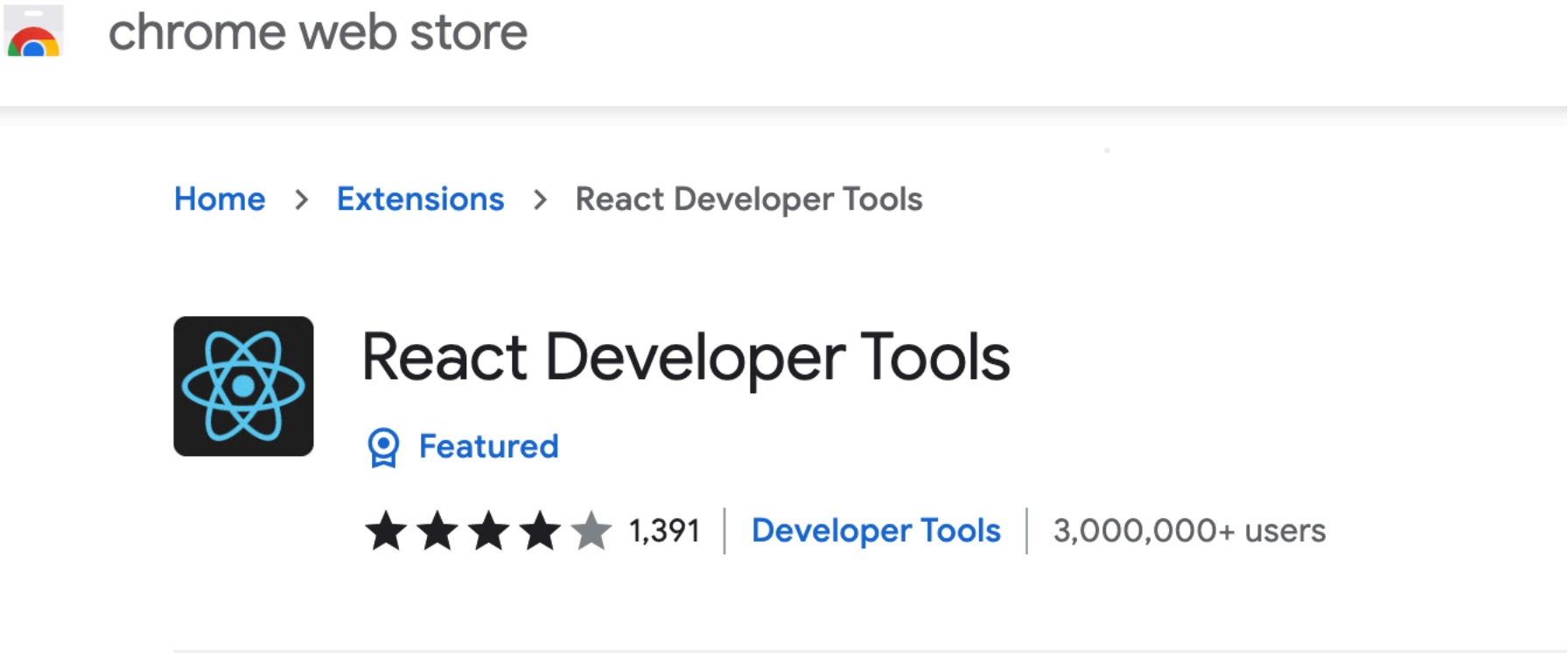
A JavaScript library for creating
interactive user interfaces (UI).

How does React work?

By building a user interface with **components.** These components can be reused anywhere in our **application**.

Why use React?

Building the UI with reusable components makes our code is **much easier** to manage and update.



https://chrome.google.com/webstore/detail/React-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en

Let's look at some web pages that use React and how they split the UI into components.

*BBC, Netflix, Airbnb, Udemy

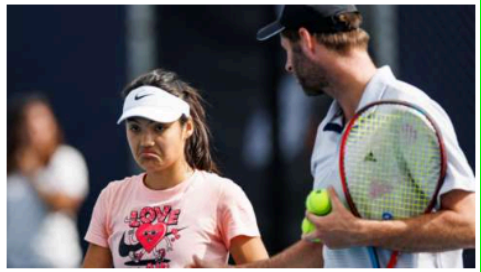
LIVE Watch: World Snooker Championship - Higgins and Trump leading

Watch live coverage from day 11 of the World Snooker Championship at The Crucible in Sheffield.

Snooker

All-English final or Spain to reign in Europe?

20h | European Football



Raducanu splits from coach after five months

3h | Tennis

Watch Robertson's brilliant maximum in 1min 47secs

17h | Snooker

Djokovic can defend Wimbledon title

3h | Tennis

LIVE Listen: IPL - Royal Challengers Bangalore v Rajasthan Royals

Cricket

Hunter to miss Grand Slam decider

3h | Rugby Union

What is a component?

A component is an independent reusable piece of code. They come in two types, a JavaScript **class component** or a **functional component**.

React renders our components to represent HTML elements (JSX)

But these elements are just JavaScript objects.

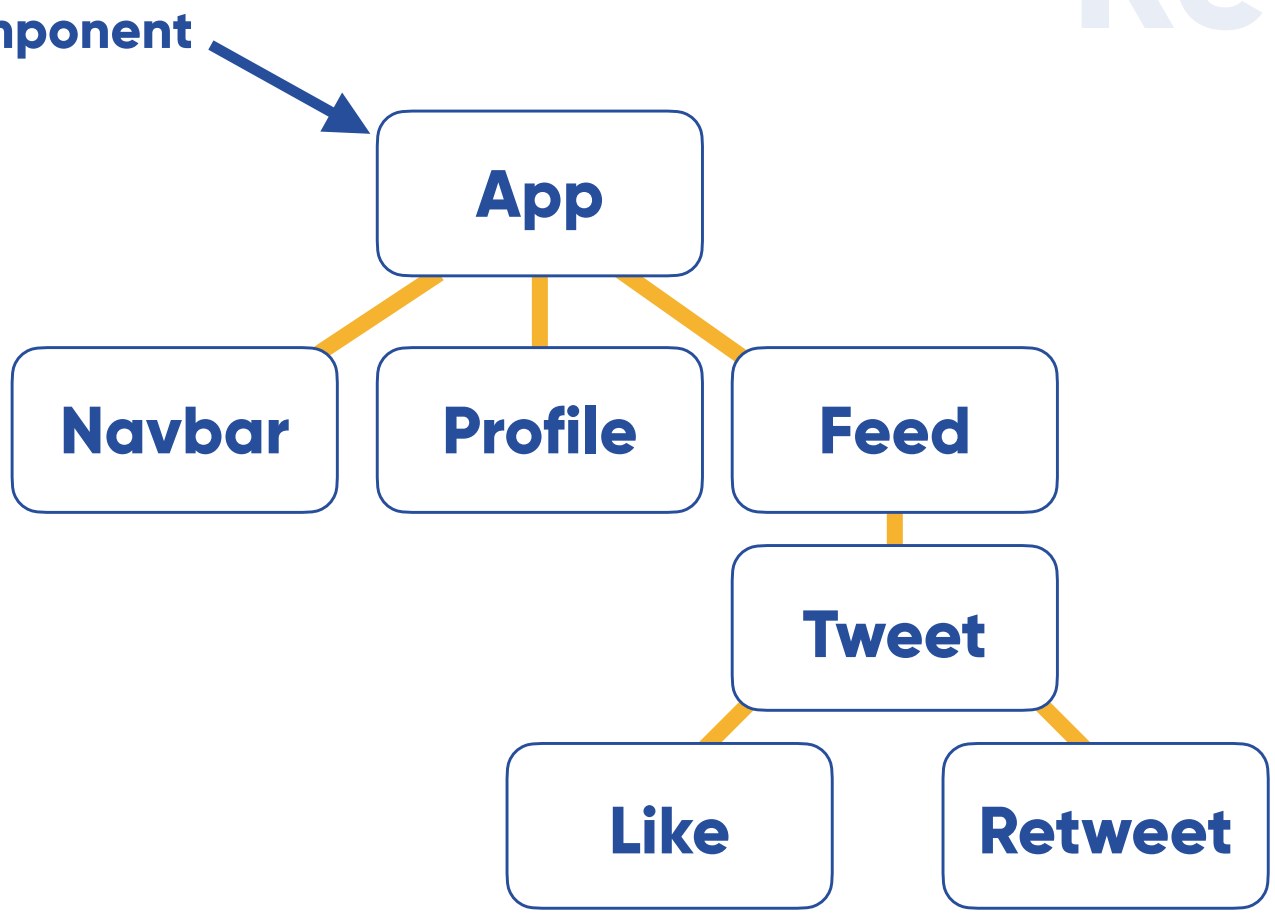
What is JSX?

JSX stands for **JavaScript XML**.
React uses a compiler called Babel, which turns our JSX back into vanilla JavaScript.

Babel Compiler

We can build our components in isolation, but they have to form a hierarchy with one root component when we use them.

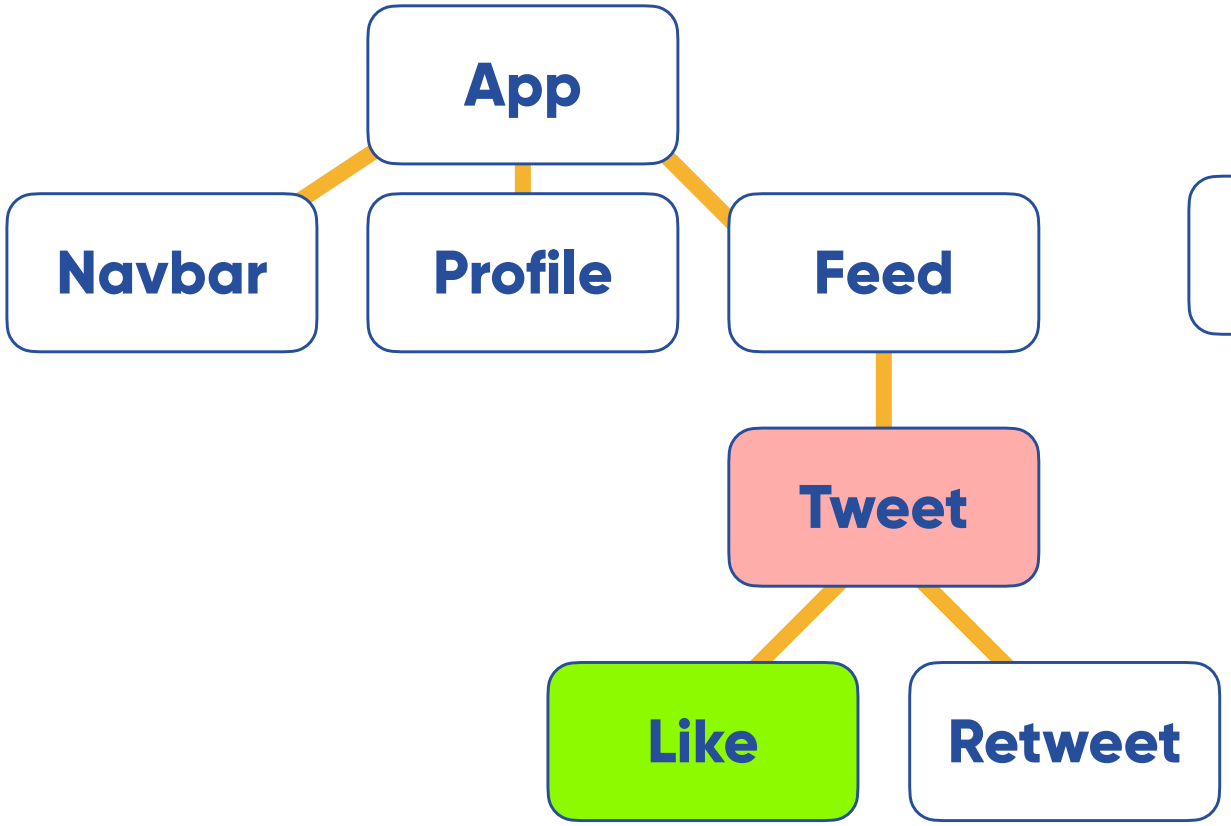
Component Tree



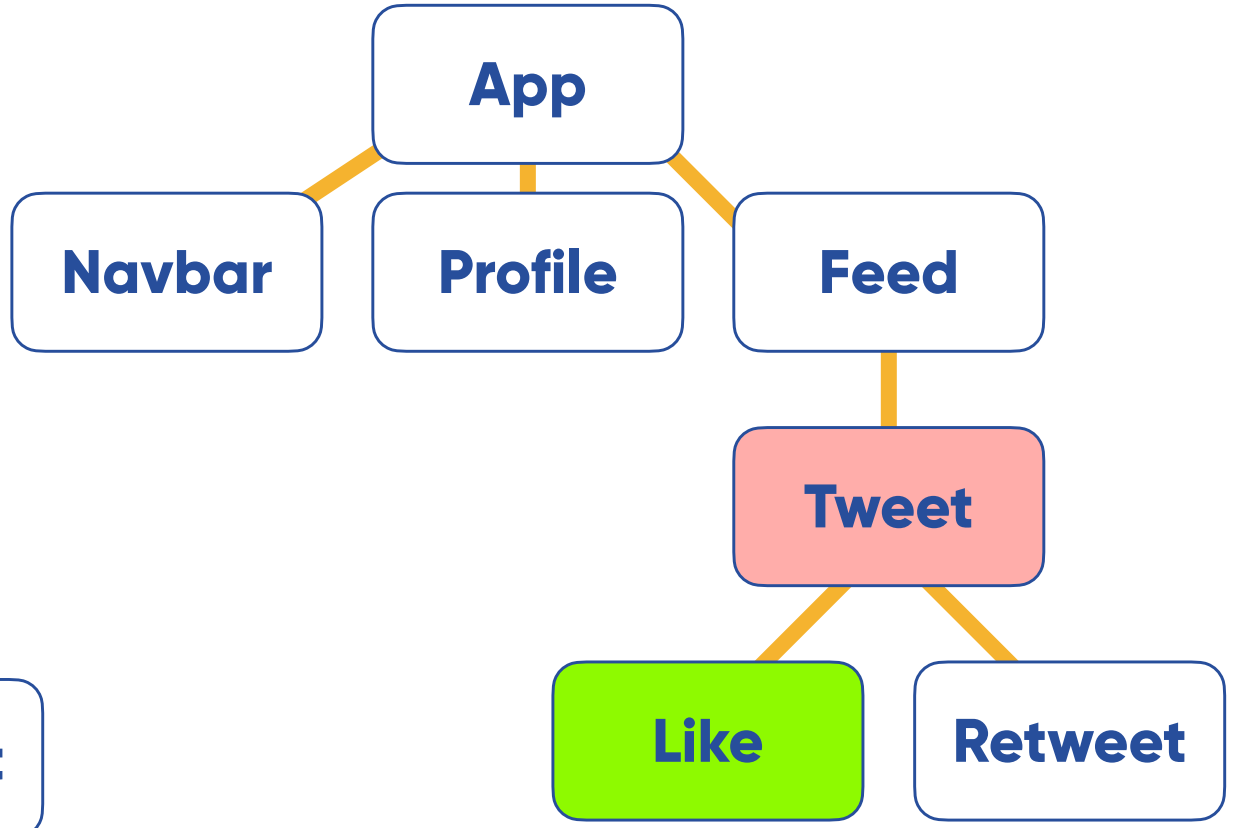
React creates a virtual DOM, a lightweight representation of the actual DOM, stored in memory.

When the state of our app changes, React compares the virtual DOM and the actual DOM. If there is a difference, the actual DOM is updated to keep it in sync.

Virtual DOM



Actual DOM



{CN}®

React JS

What does this mean?

We no longer have to work with the DOM API in the browser. So no more...

document.getElementBy.....

If we change our UI, React re-renders the necessary component that updates the DOM.

The React build workflow

What do we need to make a **React app?**

What we need to create a React app:

- npm
- Webpack
- Babel compiler
- React and React-dom
- Development server to use locally

Or use create-React-app

Using create-react-app

In the terminal, navigate to the directory where you want to create a new React package and run the **command**:

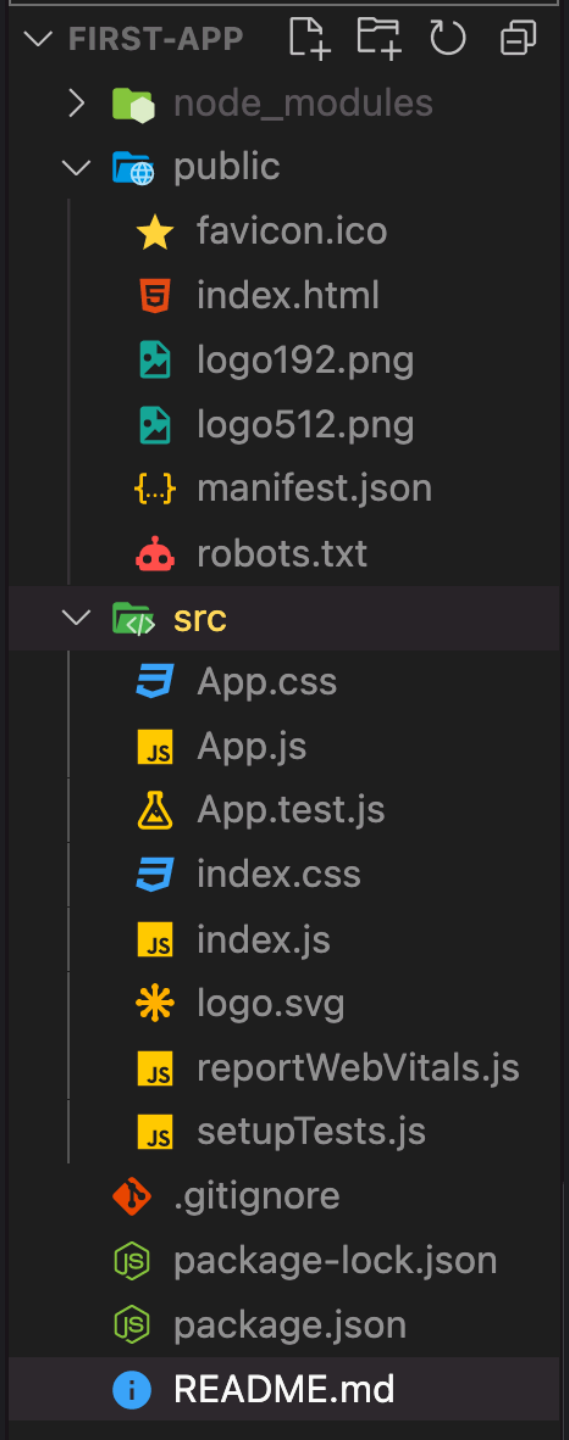
```
npx create-react-app appName
```

Open your new folder

To start the React live server.
In the terminal, type the command:

npm start

If you want to stop the server, ctrl/ command C.



React JS

Let's take a look at the React app file structure.

{CN}®

package.json


```
{
  "name": "first-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.4",
    "@testing-library/React": "^13.1.1",
    "@testing-library/user-event": "^13.5.0",
    "React": "^18.1.0",
    "React-dom": "^18.1.0",
    "React-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "React-scripts start",
    "build": "React-scripts build",
    "test": "React-scripts test",
    "eject": "React-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "React-app",
      "React-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

This file has the list of node dependencies which are needed.

Index.html


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-React-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

This will be the only HTML file in the project. All of the application components are loaded into this div.

Index.js

React JS

```
import React from 'React';
import ReactDOM from 'React-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

This is the JavaScript file that corresponds to the index.html. These lines tell us that the App component will be rendered into the div element in the index.html with the id of root.

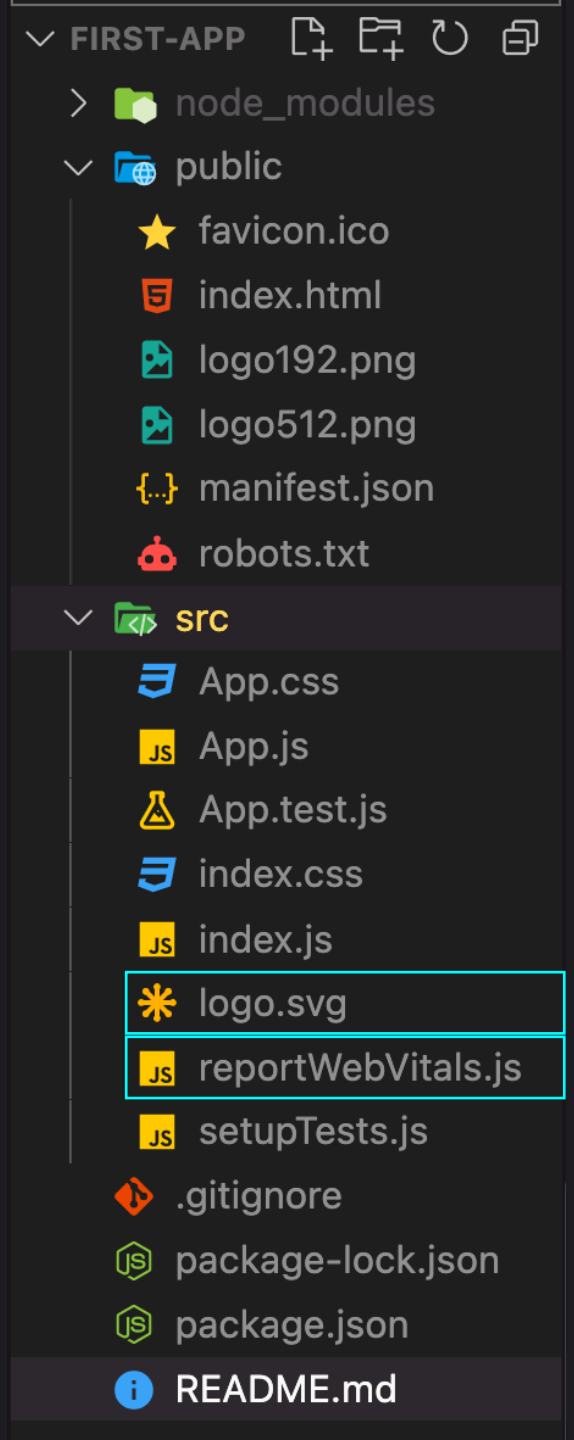
App.js


```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://Reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

Note **className** is used here, not class.

This is the file for the **App** component. **App** component is the main component in React which acts as a container for all other components.

Cleaning up before we get started



Delete the logo and reportWebVitals

Index.js


```
import React from 'React';
import ReactDOM from 'React-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Delete the reportWebVitals import and function call.

App.js

```
import './App.css';

const App = () => {
  return (
    <div className="App">
      <div>
        <h1>My React Project</h1>
      </div>
    </div>
  );
};

export default App;
```

App.css

```
.App {
  text-align: center;
}
```

Clear the App.js, create a functional component and clean up the App.css.

As mentioned earlier, a component is either a JavaScript function or a JavaScript class.

Let's take a look....

{CN}®

Functional Component


```
const Person = () => {
    return (
        <div>
            <h1>I'm a functional component</h1>
        </div>
    );
};
```

This component is a function which returns some JSX. It looks like HTML but, it's not. It is converted to JavaScript.

Functional Component

```
const Person = () => {
    return (
        <div>
            <h1>I'm a functional component</h1>
        </div>
    );
};
```

Note that the return statement is wrapped in normal brackets. This is standard in JavaScript when our return statement is written over multiple lines.

Functional Component

```
const Person = () => {
    return (
        <div>
            <h1>I'm a functional component</h1>
        </div>
    );
};
```

It is best practice to use a CAPITAL letter when naming our functional components.

Functional Component

React JS

```
const Person = () => {
    return (
        <div>
            <h1>I'm a functional component</h1>
        </div>
    );
};
```

When returning JSX, there must be ONE parent element.

Class Component


```
class App extends React.Component {
    render() {
        return (
            <div>
                <h1>I'm a class Component</h1>
            </div>
        );
    }
}
```

In React there is a class called component, which uses the **extends** keyword.

Class Component


```
class App extends React.Component {
    render() {
        return (
            <div>
                <h1>I'm a class Component</h1>
            </div>
        );
    }
}
```

Class based components use the render() method. Inside the render method, we also have the return statement.

Class vs Functional


```
class App extends React.Component {
    render() {
        return (
            <div>
                <h1>I'm a class Component</h1>
            </div>
        );
    }
}
```

```
const Person = () => {
    return (
        <div>
            <h1>I'm a functional component</h1>
        </div>
    );
};
```

Let's create a functional component

called Person, and nest it into our main App component

Functional component

```
const Person = () => {
    return (
        <div>
            <h1>I am a functional component</h1>
        </div>
    );
};

export default Person;
```

Make sure you export your functional component

Nesting child component


```
import './App.css';
import Person from './Person';

const App = () => {
  return (
    <div className="App">
      <div>
        <h1>My React Project</h1>
        <Person />
      </div>
    </div>
  );
};

export default App;
```

A **custom** HTML element with a self closing tag. You can also have not self closing <person></person>.

Ensure that you import your functional component

Exporting and Importing


```
export const Person = () => {
    return (
        <div>
            <h1>I am a functional component</h1>
        </div>
    );
};
```

We can also export like this!

Exporting and Importing

```
import './App.css';
import {Person} from './Person';

const App = () => {
  return (
    <div className="App">
      <div>
        <h1>My React Project</h1>
        <Person />
      </div>
    </div>
  );
};

export default App;
```

Just ensure curly brackets surround your component on the import.

Render the functional component

three times in our main App component

Example

```
import './App.css';
import Person from './Person';

const App = () => {
  return (
    <div className="App">
      <div>
        <h1>My React Project</h1>
        <Person />
        <Person />
        <Person />
      </div>
    </div>
  );
};

export default App;
```

Whenever we see an HTML tag in React, it is just a React method in disguise.

React.createElement()

The method takes three arguments.

```
React.createElement(arg1, arg2, arg3)
```

```
React.createElement(
type of element,
{an object of properties},
any children)
```

JS

```
<Element property="value">
<p> I'm a child element</p>
</Element>
```

JSX

```
React.createElement(
Person,
{name: “Dave”, age: “33”},
React.createElement (‘p’,null, “I’m a child Element”))
```

JS

```
<Person name=“Dave”, age=“33”>
<p> I’m a child element</p>
</Person>
```

JSX

Props (Passing Data)

React JS

BBC | Sign in | Home | News | Sport | Weather | iPlayer | Sounds | CBBC | CBeebies | More | Search

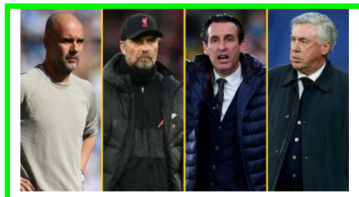
SPORT

Home | Football | Cricket | Formula 1 | Rugby U | Rugby L | Tennis | Golf | Boxing | Athletics | More

LIVE Watch: World Snooker Championship - Higgins and Trump leading

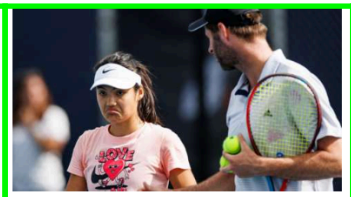
Watch live coverage from day 11 of the World Snooker Championship at The Crucible in Sheffield.

Snooker



All-English final or Spain to reign in Europe?

20h | European Football



Raducanu splits from coach after five months

3h | Tennis

Watch Robertson's brilliant maximum in 1min 47secs

17h | Snooker

Djokovic can defend Wimbledon title

3h | Tennis

LIVE Listen: IPL - Royal Challengers Bangalore v Rajasthan Royals

Cricket

Hunter to miss Grand Slam decider

3h | Rugby Union

These have the same components being repeated.

But they have different images and text passed to them.

The core component is the same, the data being passed to it is different.

{CODENATION}

Example App.js

```
import './App.css';
import Person from './Person';

const App = () => {
  return (
    <div className="App">
      <div>
        <h1>My React Project</h1>
        <Person name="Dave" age="33" />
        <Person name="Karen" age="45" />
        <Person name="Steve" age="40" />
      </div>
    </div>
  );
};

export default App;
```

When React renders the JSX and turns it into standard JS, it turns the attributes on our custom HTML elements into a JS object.

Example Functional Component


```
const Person = (props) => {
    return (
        <div>
            <h1>
                My name is {props.name} and my age is {props.age}
            </h1>
        </div>
    );
};

export default Person;
```

We refer to these objects as **props**. The **props object** is passed to our component as a function **argument**.

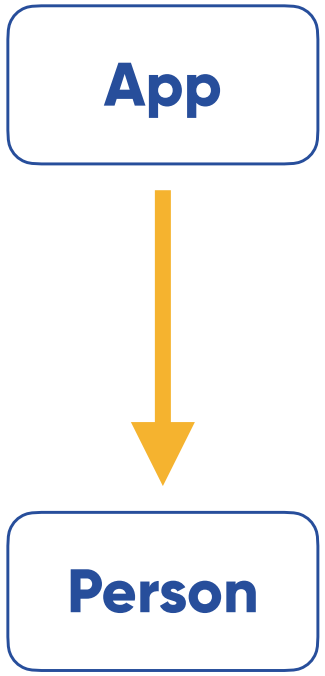
Destructuring props

```
const Person = ({name, age}) => {
    return (
        <div>
            <h1>
                My name is {name} and my age is {age}
            </h1>
        </div>
    );
};

export default Person;
```

Destructuring our **variables** means that we do not have to use the **props** keyword, making our code cleaner and more readable.

Passing props is one of the ways we pass data down the hierarchy of components.

Passing Props



Data flows down the component tree.

Task 1

React JS

Create a **new functional component** called job that returns some text (a h2 for example) “I’m a chef.”

Nest your **custom component** into the App component. Then save and check the browser that it has rendered correctly.

Pass the Job component some **props.** For example job=“junior developer”. Then inside the job component, replace the word “chef” with the props that you have passed to that component.

Task 1 - Job.js

```
const Job = ({ job }) => {
    return (
        <div>
            <h2>I am a {job}!</h2>
        </div>
    );
};

export default Job;
```

Task 1 - App.js

```
import './App.css';
import Job from './Job';
import Person from './Person';

const App = () => {
    return (
        <div className="App">
            <div>
                <h1>My React Project</h1>
                <Person name="Dave" age="33" />
                <Person name="Karen" age="45" />
                <Person name="Steve" age="40" />
                <Job job="junior developer" />
            </div>
        </div>
    );
};

export default App;
```

Task 2

Create a new **functional component** called Pet, this component should return some text (a h2 tag for example) “Hi my name is Ben”

Render your **Pet component** in the Person component. Then save and check on the browser that it has rendered correctly.

Then pass some **props** from a Person in the **App component: petsName** and type. Give them any values you want.

Inside the **Person component,** create new props for your Pet component. Then finally, inside the Pet component, change the sentence to use the values.

Task 2 - Pet.js

```
const Pet = ({ petsName, type }) => {
    return (
        <div>
            <h2>
                My pets name is {petsName} and it is a {type}
            </h2>
        </div>
    );
};

export default Pet;
```

Task 2 - Person.js

```
import Pet from './Pet';

const Person = ({ name, age, petsName, type }) => {
    return (
        <div>
            <h1>
                My name is {name} and my age is {age}
            </h1>
            <Pet petsName={petsName} type={type} />
        </div>
    );
};

export default Person;
```

Task 2 - App.js

```
import './App.css';
import Job from './Job';
import Person from './Person';

const App = () => {
    return (
        <div className="App">
            <div>
                <h1>My React Project</h1>
                <Person name="Dave" age="33" petsName="Garfield" type="Cat" />
                <Person name="Karen" age="45" petsName="Polly" type="Budgie" />
                <Person name="Steve" age="40" petsName="Stan" type="Dog" />
                <Job job="junior developer" />
            </div>
        </div>
    );
};

export default App;
```

Passing Props

Data flows down the component tree, and we can keep passing props along.

```jsx
const App = () => {
    return (
        <div className="App">
            <div>
                <h1>My React Project</h1>
                <Person name="Dave" age="33" petsName="Garfield" type="Cat" />
                <Person name="Karen" age="45" petsName="Polly" type="Budgie" />
                <Person name="Steve" age="40" petsName="Stan" type="Dog" />
                <Job job="junior developer" />
            </div>
        </div>
    );
};
const Person = ({ name, age, petsName, type }) => {
    return (
        <div>
            <h1>
                My name is {name} and my age is {age}
            </h1>
            <Pet petsName={petsName} type={type} />
        </div>
    );
};
const Pet = ({ petsName, type }) => {
    return (
        <div>
            <h2>
                My pets name is {petsName} and it is a {type}
            </h2>
        </div>
    );
};
```

We can use CSS in React like we would in HTML.

However, we need to use the className attribute instead of class. We can also use tags and a unique id.

We must remember to import the CSS file at the top of our JS file.

Activity

The following slide shows some **screenshots** of some websites we can recreate.

You need to decide how you may break the **images** into the components and then put your **components** together to match the image.

More info
https://create-React-app.dev/docs/adding-images-fonts-and-files/

Don't think pandemic is over, Whitty warns

Unlocking too quickly would lead to a substantial surge in infection, UK chief medical adviser says.

2h | Health |

Pay rise was set to be 2.1% - NHS chief

NHS England boss backs ministers over pay dispute but does not rule out a one-off bonus for workers.

1h | UK Politics

Charity criticises Morgan's comments about Meghan

Mental health charity Mind says it is "disappointed" by comments Piers Morgan made on Monday.

2h | Entertainment & Arts

Unilever drops word 'normal' from beauty products

The owner of Dove and Vaseline will remove the word from about 200 products in a push for inclusivity.

10m | Business

Entire homes

Unique stays

Cabins and cottages

Pets allowed

{CODENATION}

Learning Objectives

To explore React JS and what it's used for

To discover components and be able to write your own

To be familiar with Props

For later...

... take a look at the **useState hook.**

https://Reactjs.org/docs/hooks-state.html

https://www.youtube.com/watch?v=O6P86uwfdR0

Can you **import the useState hook**?
How can you **change the state of a variable**?