Module 02

Purwadhika
Digital Technology School

Fullstack Web Development

Intro to React JS

Job Connector Program

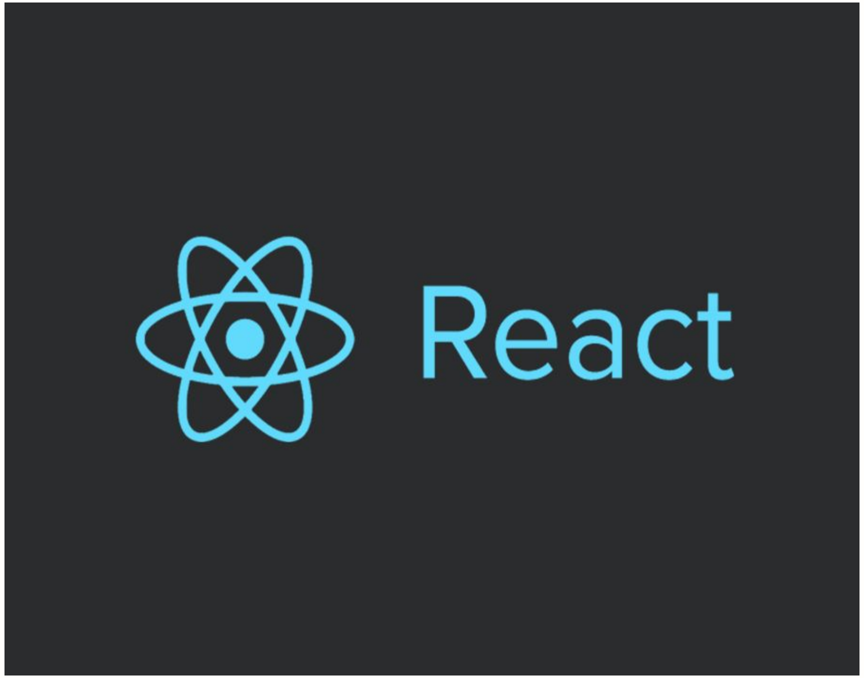
Outline

- Intro to ReactJs
- Component & props
- Class based component vs functional component
- Routing

Introduction

What is React?

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”.



Create Simple App Using “Create React App”

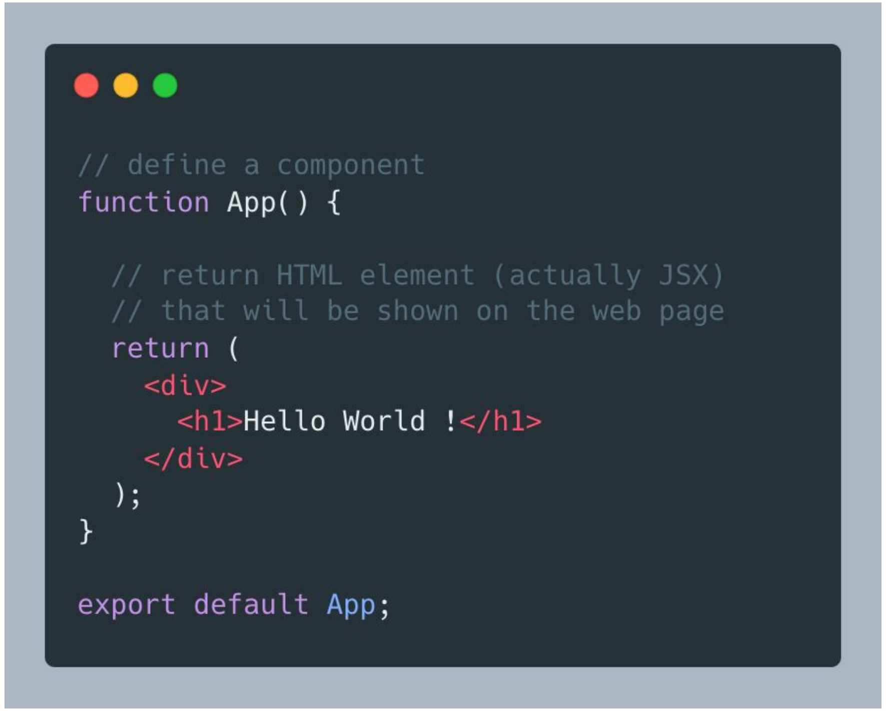
Purwadhika
Digital Technology School

Create React App is a comfortable environment for **learning React**, and is the best way to start building **a new single-page application** in React.

It sets up your development environment so that you can use the latest JavaScript features, provides a nice developer experience, and optimizes your app for production.

```
npx create-react-app my-app
cd my-app
npm start
```

Code Structure



```
// define a component
function App( ) {

  // return HTML element (actually JSX)
  // that will be shown on the web page
  return (
    <div>
      <h1>Hello World !</h1>
    </div>
  );
}

export default App;
```

In general, this is the code structure of the React JS project.

About JSX

JSX stands for JavaScript XML. It is simply a syntax extension of JavaScript. It allows us to directly write HTML in React (within JavaScript code).

Characteristics of JSX

- JSX is not mandatory to use there are other ways to achieve the same thing but using JSX makes it easier to develop react application.
- JSX allows writing expression in { }. The expression can be any JS expression or React variable.
- To insert a large block of HTML we have to write it in a parenthesis i.e, ().
- JSX produces react elements.
- JSX follows XML rule.
- After compilation, JSX expressions become regular JavaScript function calls.
- JSX uses camelcase notation for naming HTML attributes. For example, tabindex in HTML is used as tabIndex in JSX.

Components

Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.

Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called "props") and return React elements describing what should appear on the screen.

Components come in two types, **Class components** and **Function components**.

Class Based Component VS Function Component

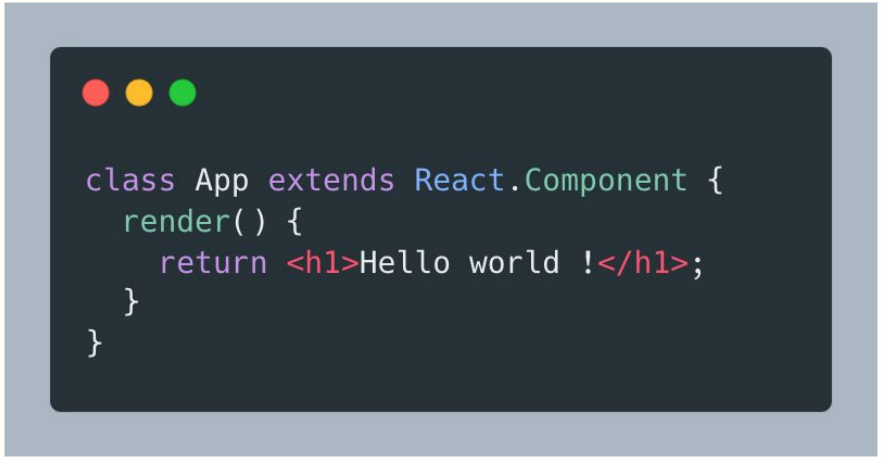
Class Component

- Class components make use of ES6 class and extend the Component class in React.
- Sometimes called “smart” or “stateful” components as they tend to implement logic and state.
- React lifecycle methods can be used inside class components (for example, componentDidMount).
- You pass props down to class components and access them with this.props

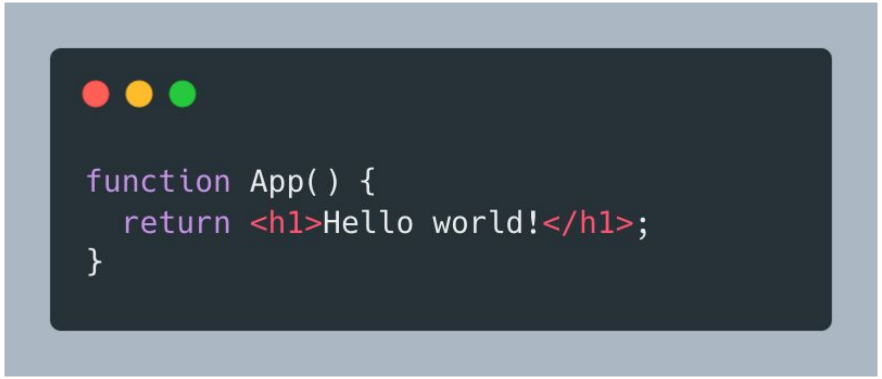
Functional Component

- Sometimes referred to as “dumb” or “stateless” components as they simply accept data and display them in some form; that is they are mainly responsible for rendering UI.
- React lifecycle methods (for example, componentDidMount) cannot be used in functional components.
- There is no render method used in functional components.
- These are mainly responsible for UI and are typically presentational only (For example, a Button component).
- Functional components can accept and use props.

Class Based Component VS Function Component



```
class App extends React.Component {
  render() {
    return <h1>Hello world !</h1>;
  }
}
```



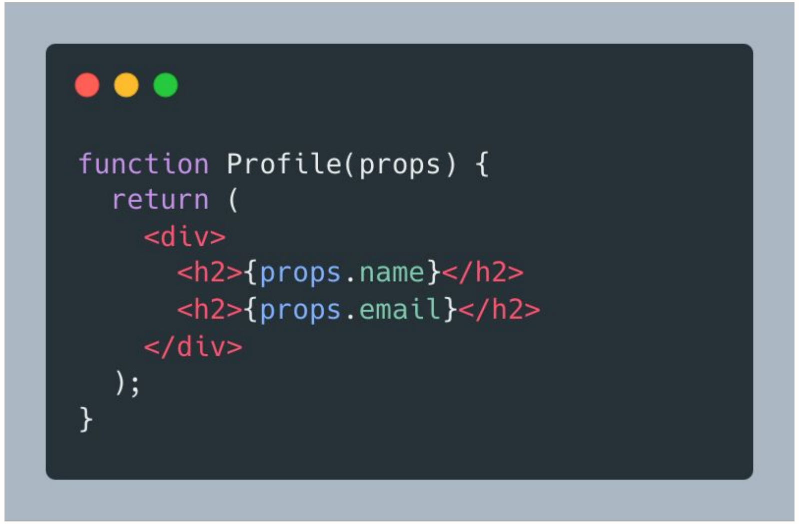
```
function App() {
  return <h1>Hello world!</h1>;
}
```

Props

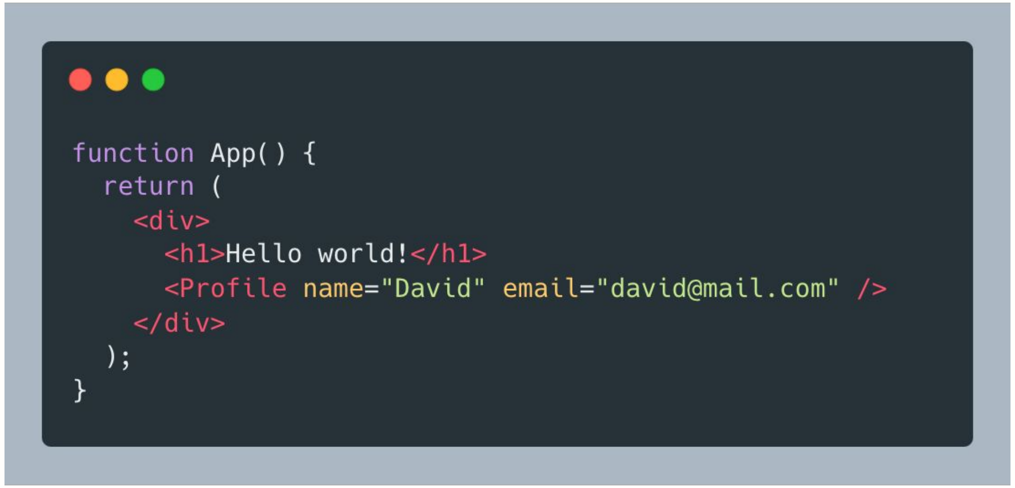
React allows us to pass information to a Component using something called props (stands for properties).

Props are objects which can be used inside a component.

Props



```
function Profile(props) {
  return (
    <div>
      <h2>{props.name}</h2>
      <h2>{props.email}</h2>
    </div>
  );
}
```



```
function App() {
  return (
    <div>
      <h1>Hello world!</h1>
      <Profile name="David" email="david@mail.com" />
    </div>
  );
}
```

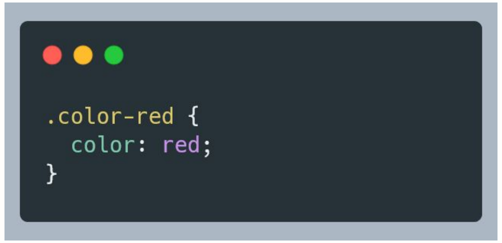
Add Styling

There is several ways to styling component in React JS:

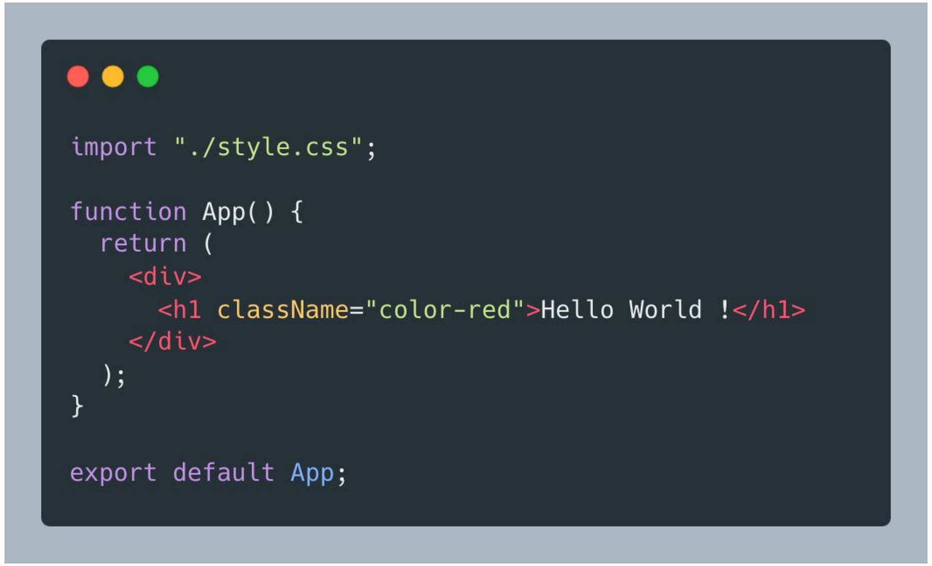
1. CSS Classes (Using CSS stylesheet outside component)
2. Inline Styles (Direct styling in an element)
3. Styled components

CSS Classes

Create a css stylesheet as usual, import that CSS file to the component we will use, and then give the class name to the element to be styled into the "className" props. Take a look at the following example.



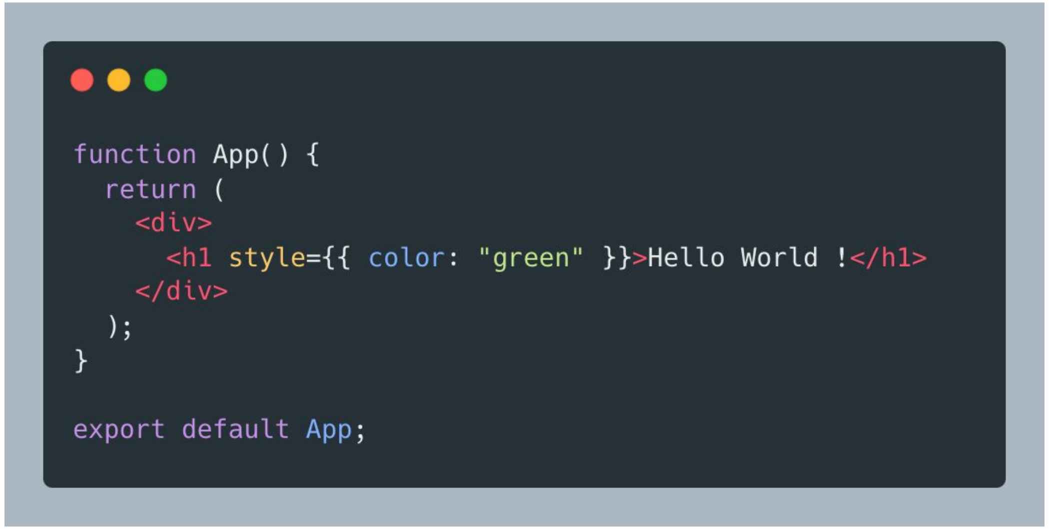
```css
.color-red {
  color: red;
}
```



```jsx
import "./style.css";

function App() {
  return (
    <div>
      <h1 className="color-red">Hello World !</h1>
    </div>
  );
}

export default App;
```

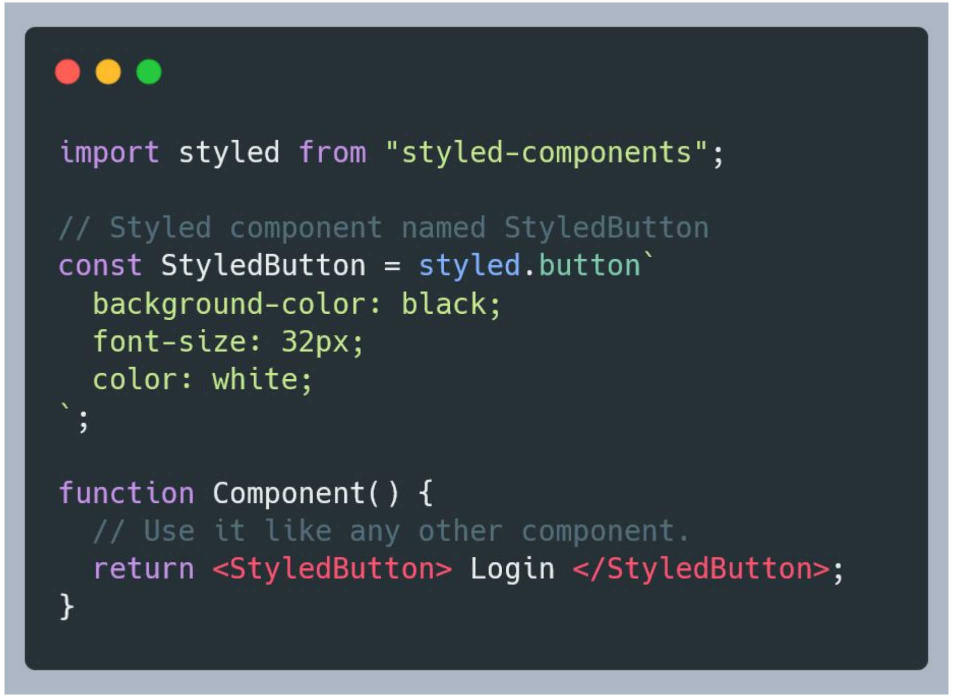
Inline Styles

Styling directly inside the element with props “style={}” and then put object in which there is css command. Take a look at the following example.



```
function App() {
  return (
    <div>
      <h1 style={{ color: "green" }}>Hello World !</h1>
    </div>
  );
}

export default App;
```

Styled Components



```
import styled from "styled-components";

// Styled component named StyledButton
const StyledButton = styled.button`
  background-color: black;
  font-size: 32px;
  color: white;
`;

function Component() {
  // Use it like any other component.
  return <StyledButton> Login </StyledButton>;
}
```

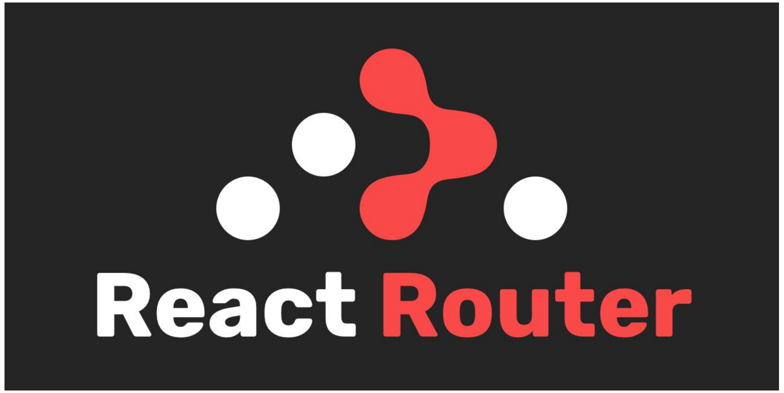
Install:

npm i styled-components

Routing

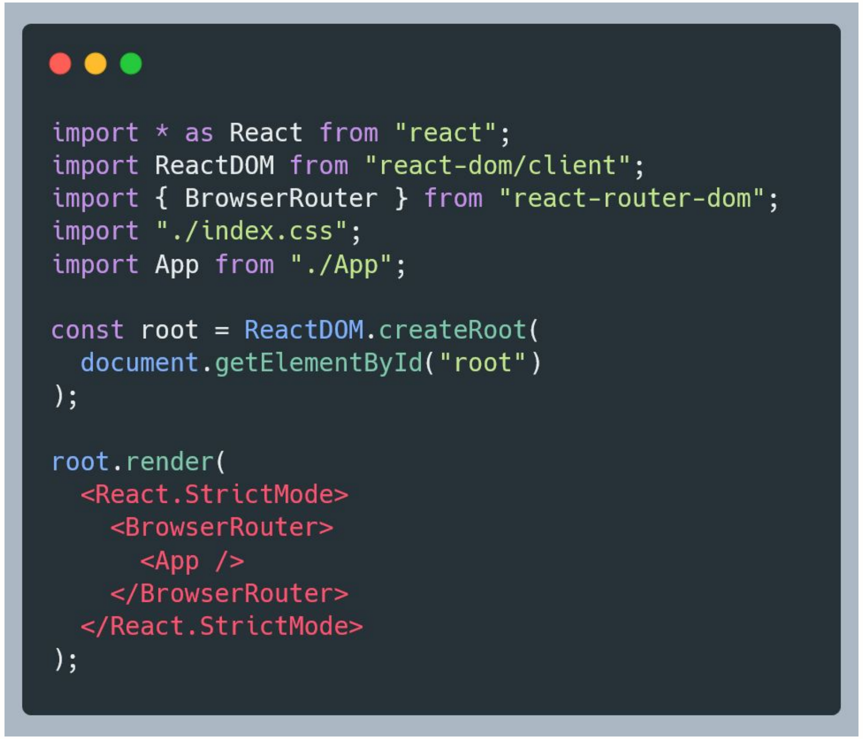
Routing is a process in which a user is directed to different pages based on their action or request. **React Router** is used to define multiple routes in the application.

First install React Router dependencies on your React JS project: ***npm install react-router-dom@6***



React Router

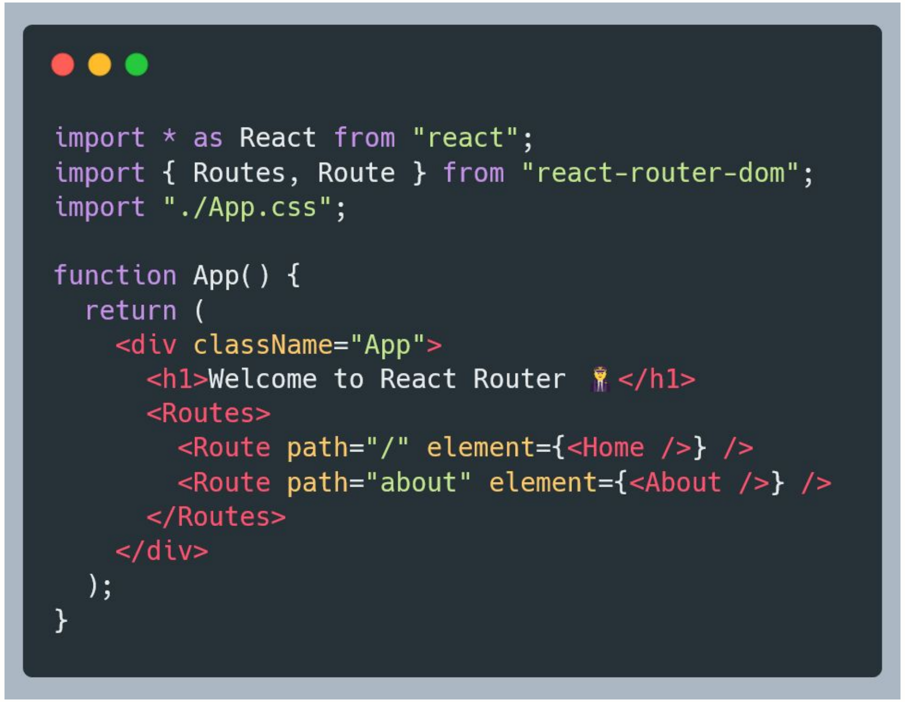
Once your project is set up and React Router is installed as a dependency, open the src/index.js in your text editor. Import BrowserRouter from react-router-dom near the top of your file and wrap your app in a <BrowserRouter>:



```
import * as React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
import "./index.css";
import App from "./App";

const root = ReactDOM.createRoot(
  document.getElementById("root")
);

root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

React Router

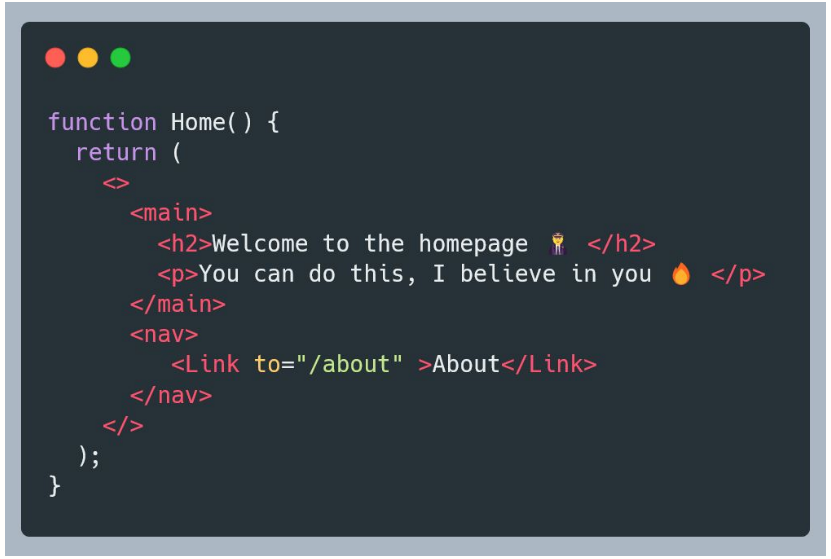
Now you can use React Router anywhere in your app! For a simple example, open src/App.js and replace the default markup with some routes:



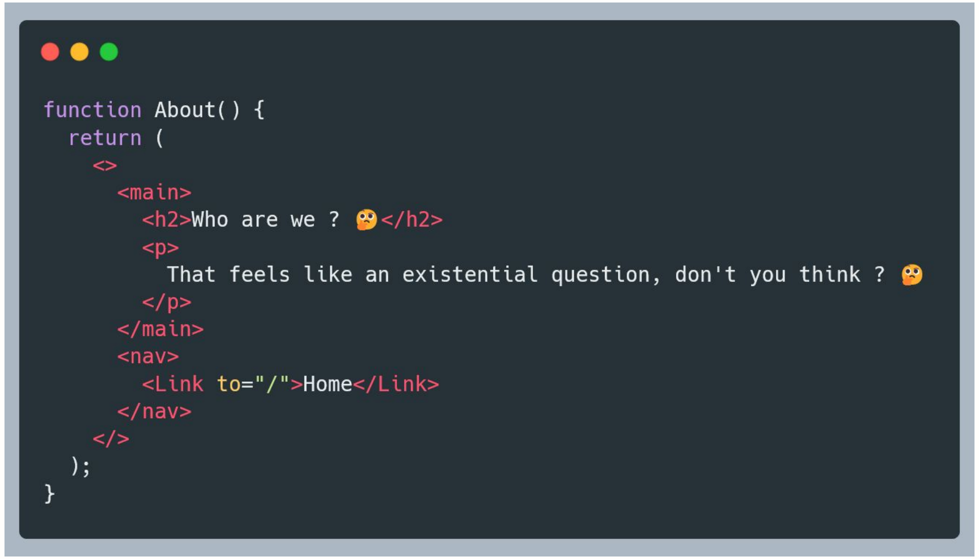
```
import * as React from "react";
import { Routes, Route } from "react-router-dom";
import "./App.css";

function App() {
  return (
    <div className="App">
      <h1>Welcome to React Router 🧑‍✈️</h1>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="about" element={<About />} />
      </Routes>
    </div>
  );
}
```

React Router

Now, still in src/App.js, create your route components:



```
function Home() {
  return (
    <>
      <main>
        <h2>Welcome to the homepage 🕴 </h2>
        <p>You can do this, I believe in you 🔥 </p>
      </main>
      <nav>
        <Link to="/about" >About</Link>
      </nav>
    </>
  );
}
```



```
function About() {
  return (
    <>
      <main>
        <h2>Who are we ? 🥺</h2>
        <p>
          That feels like an existential question, don't you think ? 🥺
        </p>
      </main>
      <nav>
        <Link to="/">Home</Link>
      </nav>
    </>
  );
}
```

Exercise

- Create your Portfolio Website using ReactJS

Thank You!

Purwadhika
Digital Technology School