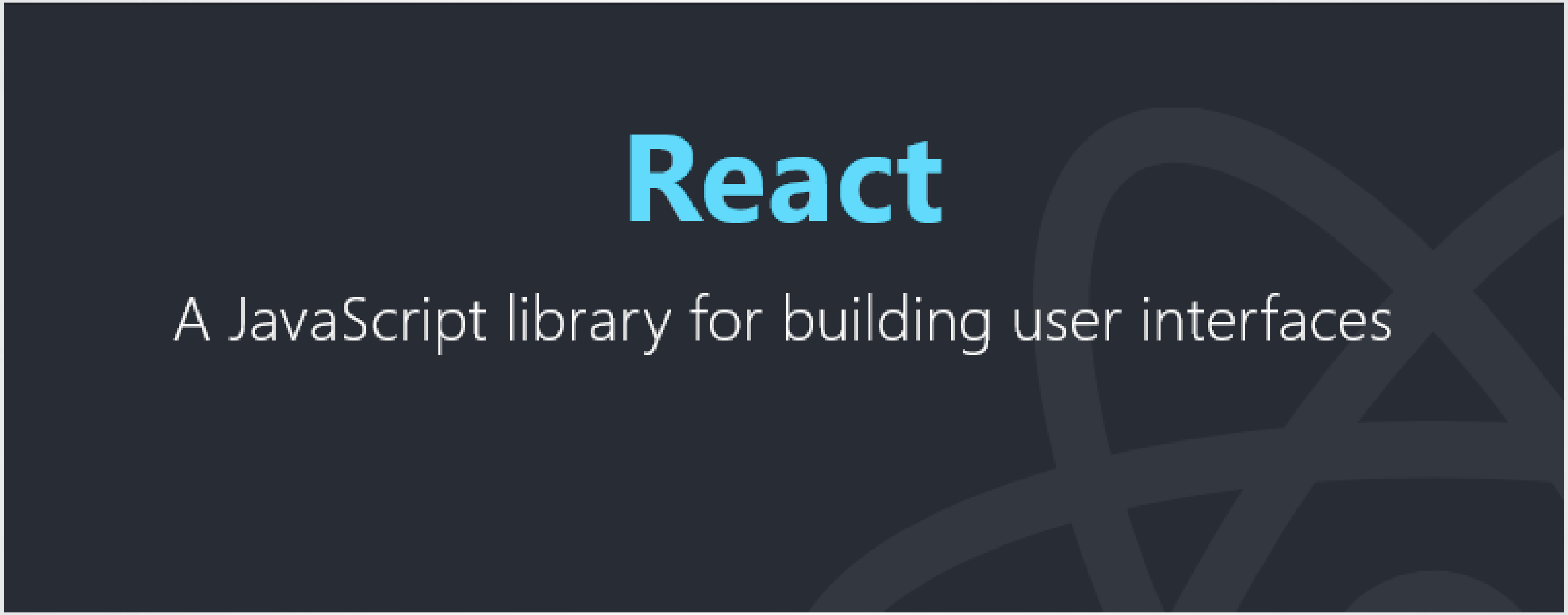


React

A JavaScript library for building user interfaces

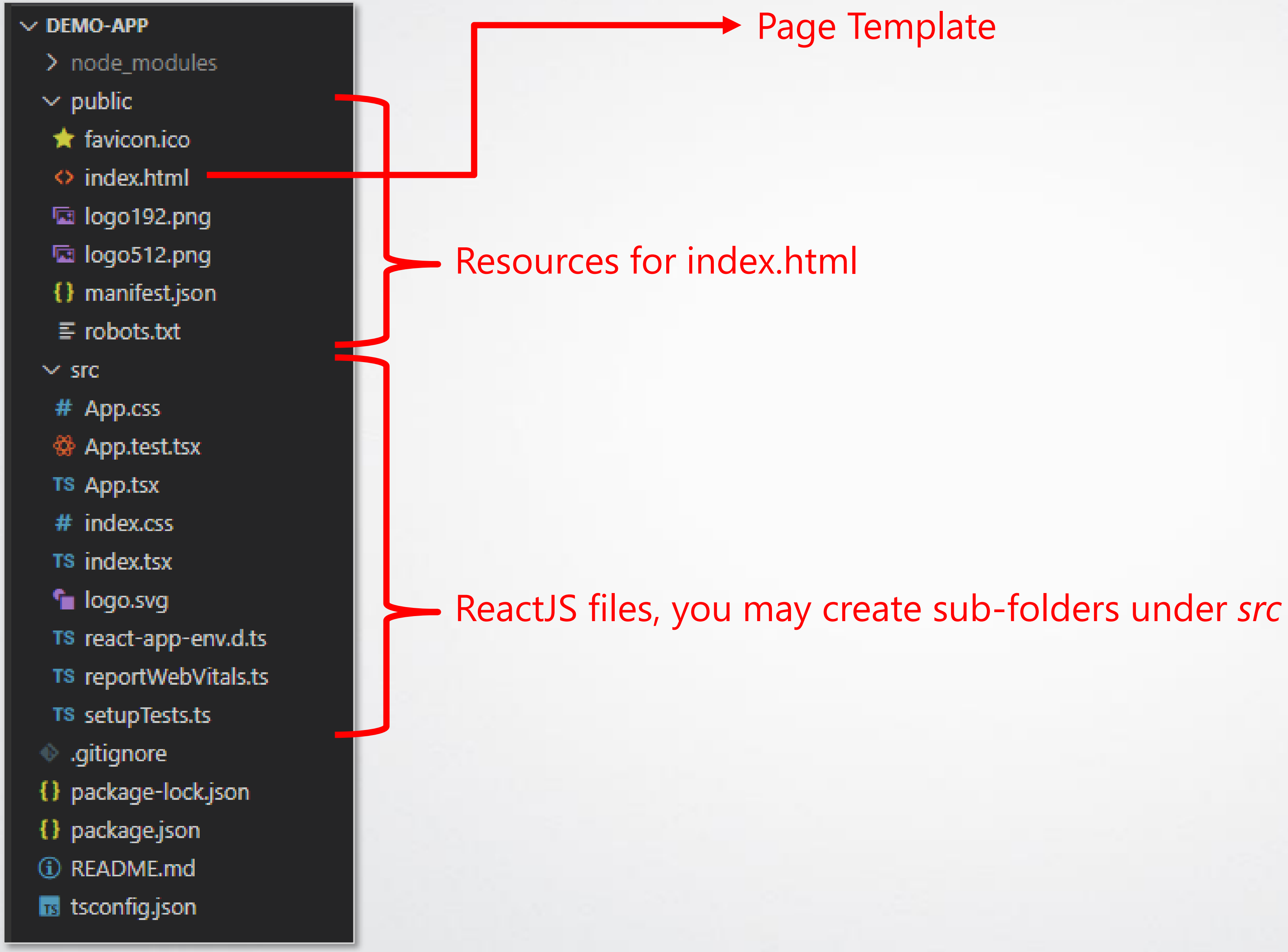
React Introduction

- Creating React app

```
npx create-react-app demo-app --template typescript
```

- Understanding the default files

Default Files



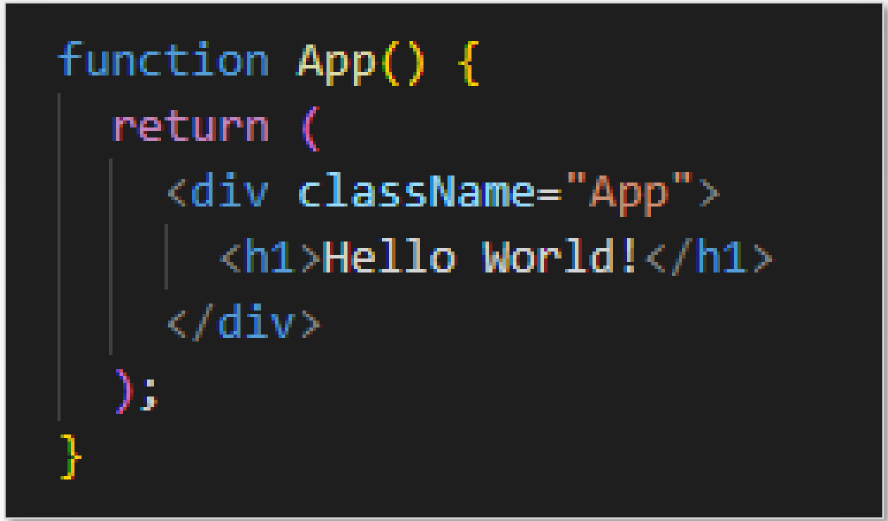
```
DEMO-APP
  node_modules
  public
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json
    robots.txt
  src
    App.css
    App.test.tsx
    App.tsx
    index.css
    index.tsx
    logo.svg
    react-app-env.d.ts
    reportWebVitals.ts
    setupTests.ts
  .gitignore
  package-lock.json
  package.json
  README.md
  tsconfig.json
```

Page Template

Resources for index.html

ReactJS files, you may create sub-folders under *src*

Understanding JSX

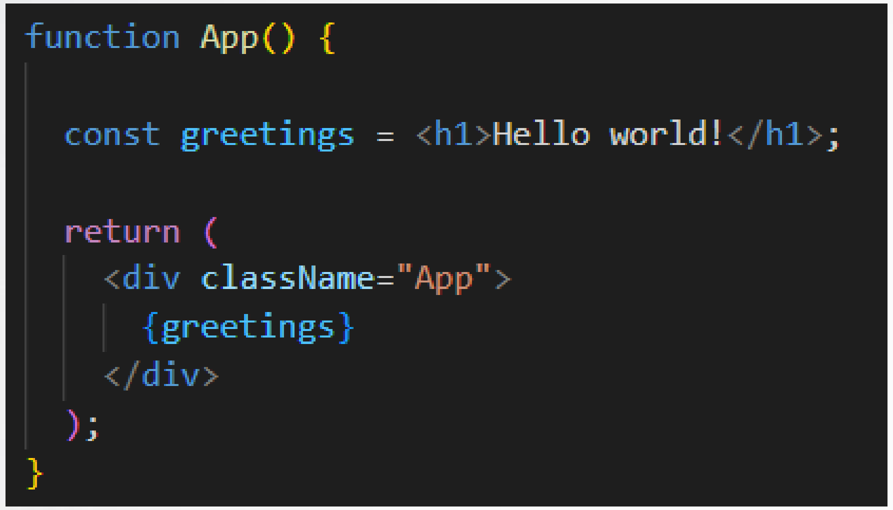


```
function App() {
  return (
    <div className="App">
      <h1>Hello World!</h1>
    </div>
  );
}
```

- Funny syntax – how does it work in JavaScript?
- This is known as JSX and it is an extension of JavaScript

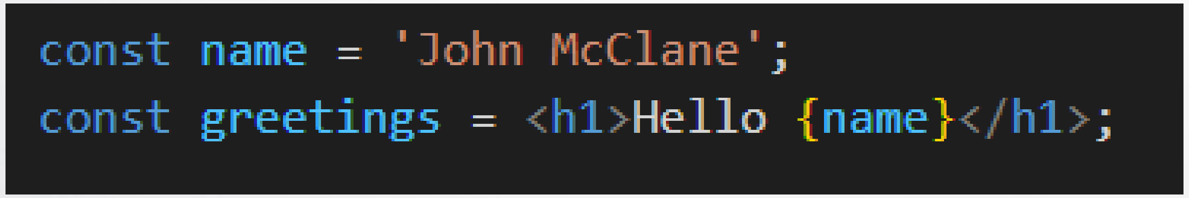
Understanding JSX

- JSX can also be assigned to a variable



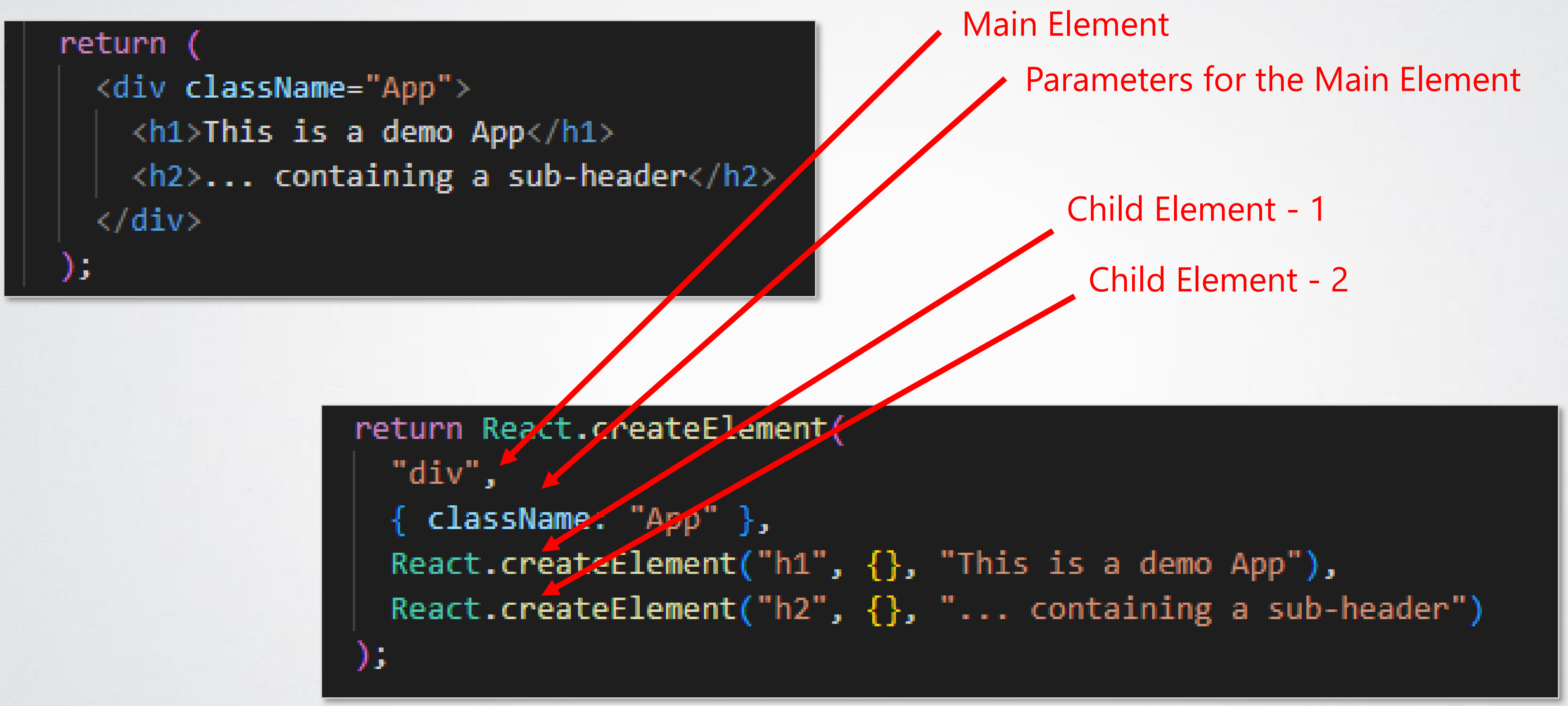
```
function App() {

  const greetings = <h1>Hello world!</h1>;

  return (
    <div className="App">
      {greetings}
    </div>
  );
}
```

- We can also use variables in JSX by wrapping it in curly braces



```
const name = 'John McClane';
const greetings = <h1>Hello {name}</h1>;
```

A closer look at JSX

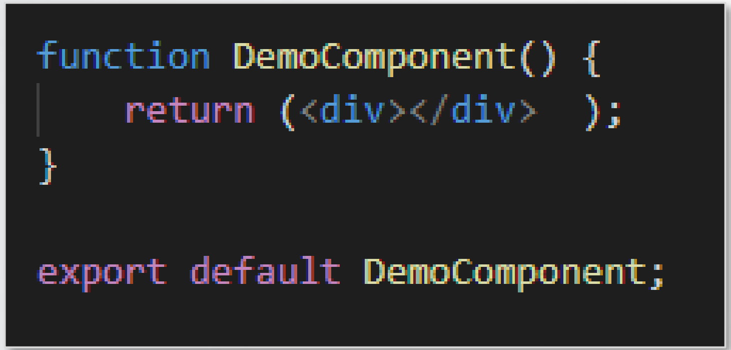


```
return (
  <div className="App">
    <h1>This is a demo App</h1>
    <h2>... containing a sub-header</h2>
  </div>
);
```

```
return React.createElement(
  "div",
  { className: "App" },
  React.createElement("h1", {}, "This is a demo App"),
  React.createElement("h2", {}, "... containing a sub-header")
);
```

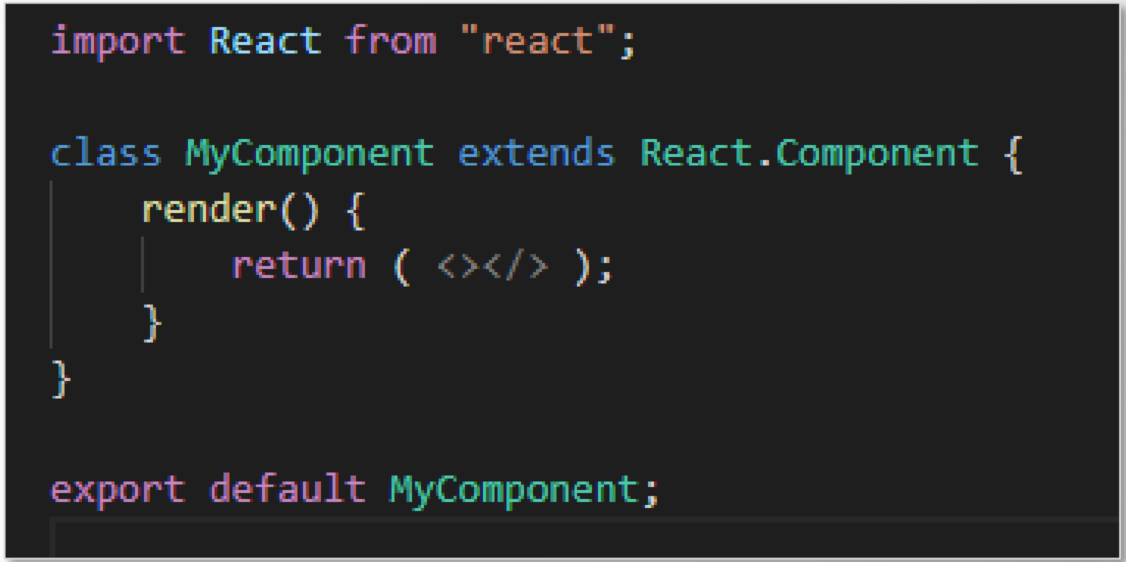
Class vs Functional Components

- Functional Component



```
function DemoComponent() {
    return (<div></div>  );
}

export default DemoComponent;
```

- Class Component



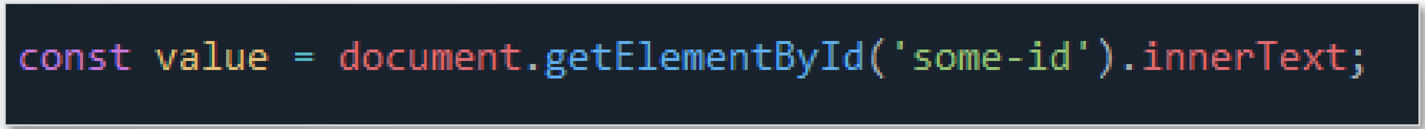
```
import React from "react";

class MyComponent extends React.Component {
    render() {
        return ( <></> );
    }
}

export default MyComponent;
```

How Component Functions are Executed?

- Components are functions that are called
- Once the component is executed, any change in the variable in the component are not automatically updated on UI

React Virtual DOM

- DOM stands for Document Object Model
  - it is a structured representation of the HTML elements that are present in a webpage or web-app



```
const value = document.getElementById('some-id').innerText;
```

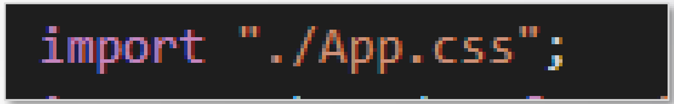
- React Virtual DOM
  - React uses Virtual DOM exists which is like a lightweight copy of the actual DOM(a virtual representation of the DOM
  - React maintains the virtual DOM as a tree and uses the tree to find the best possible ways to make these changes to the real DOM
  - Hence the update through Virtual DOM is faster

Styling Components

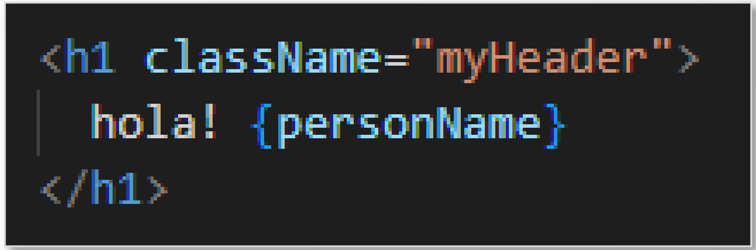
- CSS files
- Styled-components
- Module CSS

Using CSS

- css files can be imported in Component
- Class names can be provided as className



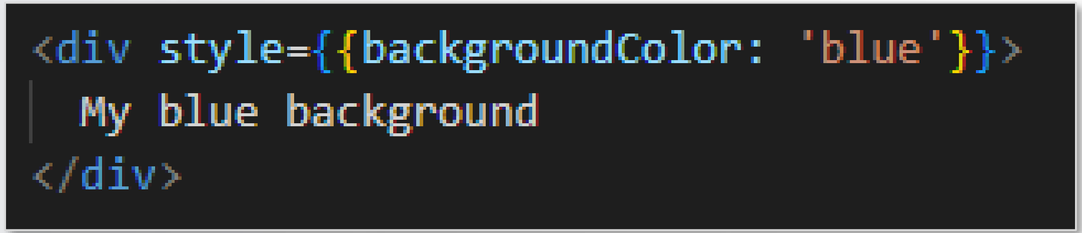
```
import "./App.css";
```



```
<h1 className="myHeader">
  hola! {personName}
</h1>
```

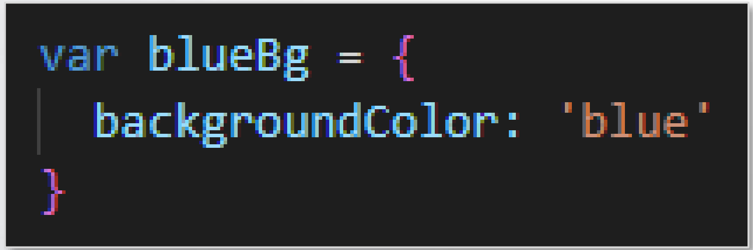
Using CSS

- In-line styles can be provided

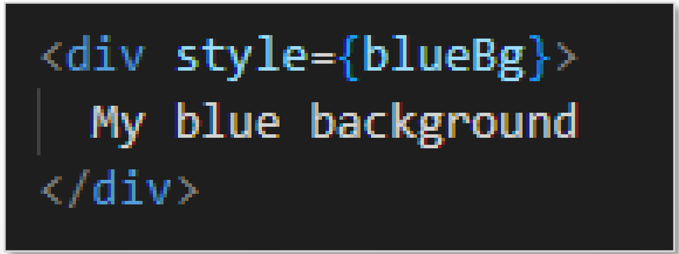


```
<div style={{backgroundColor: 'blue'}}>
  My blue background
</div>
```

- Objects can also be used to provide in-line style (also called CSS-in-JS)



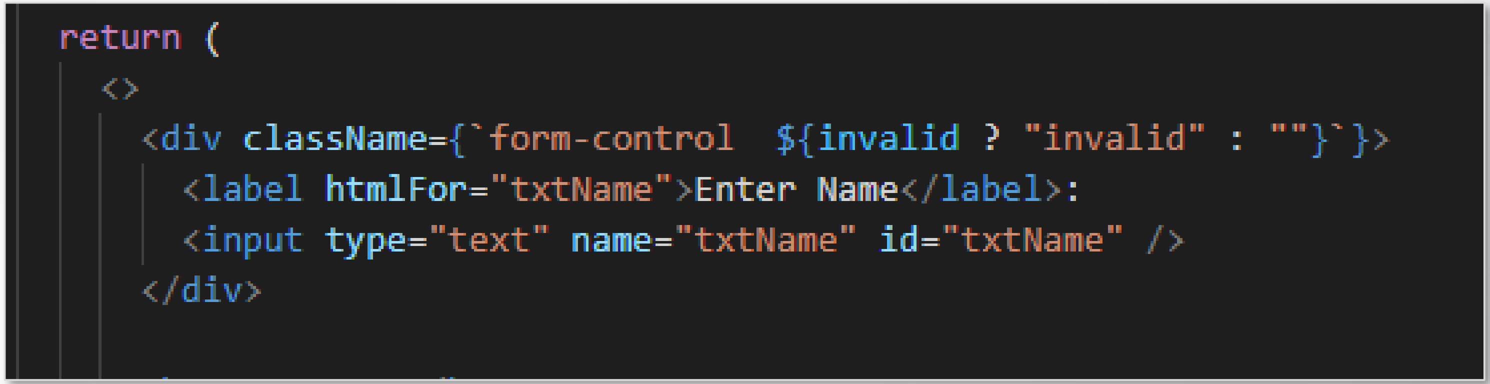
```
var blueBg = {
  backgroundColor: 'blue'
}
```



```
<div style={blueBg}>
  My blue background
</div>
```

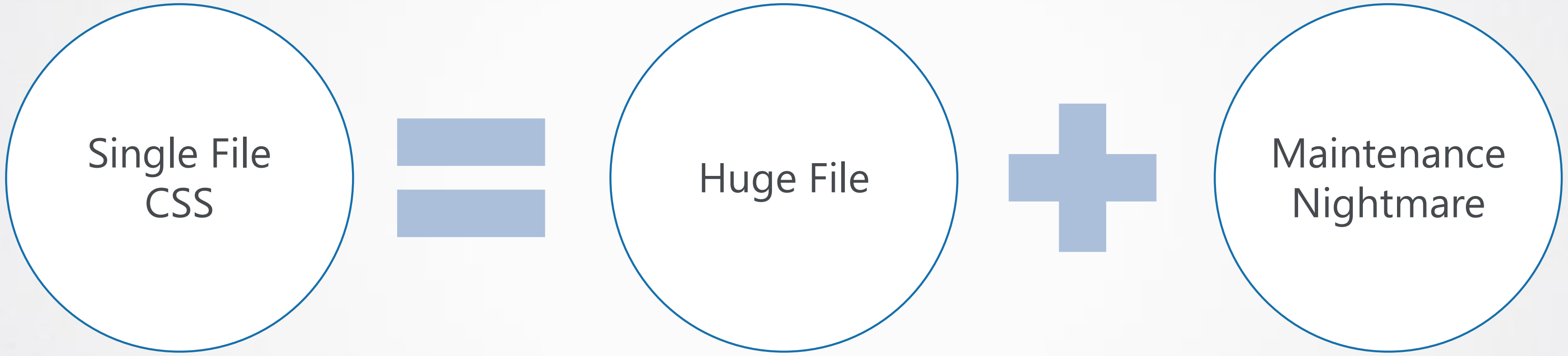
CSS: Conditional Styling

- Conditional styling can be applied



```
return (
  <>
    <div className={`form-control  ${invalid ? "invalid" : ""}`}>
      <label htmlFor="txtName">Enter Name</label>:
      <input type="text" name="txtName" id="txtName" />
    </div>
```

Module CSS

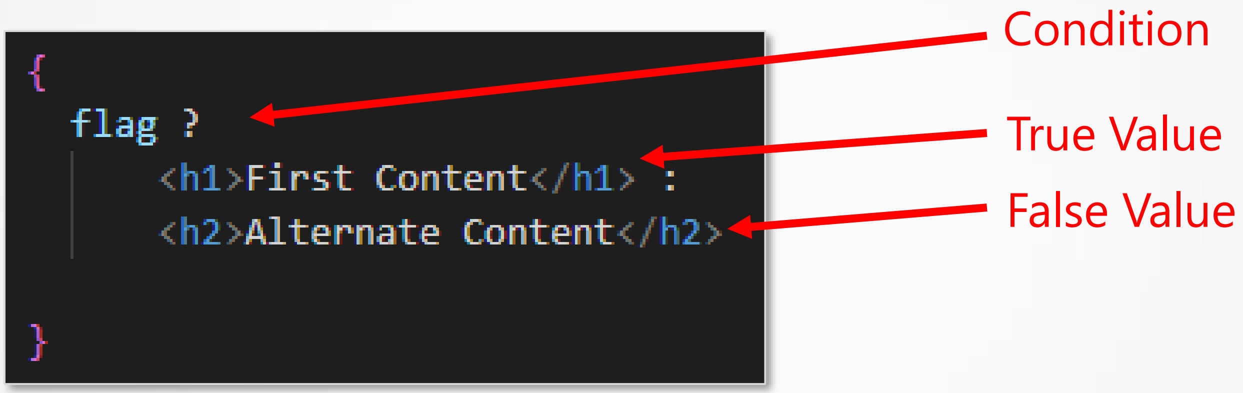


Module CSS

- react-scripts@2.0.0 and higher supports CSS Modules
- CSS Modules are regular stylesheets using naming conventions like – [name].module.css
  - E.g. Button.module.css or Button.module.scss or Button.module.sass

Conditional Rendering

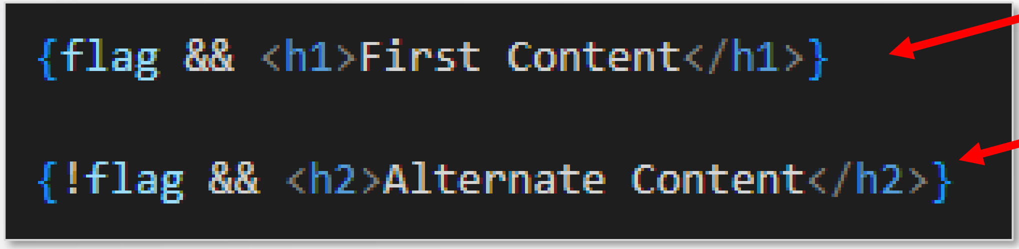
- Sometimes we need to show content conditionally
- Though long statements like *if blocks* are not allowed in JSX, the ternary operator can be used



```
{
  flag ?
      <h1>First Content</h1> :
      <h2>Alternate Content</h2>

}
```

Conditional Rendering

- Alternatively *ampersand hack* can be used

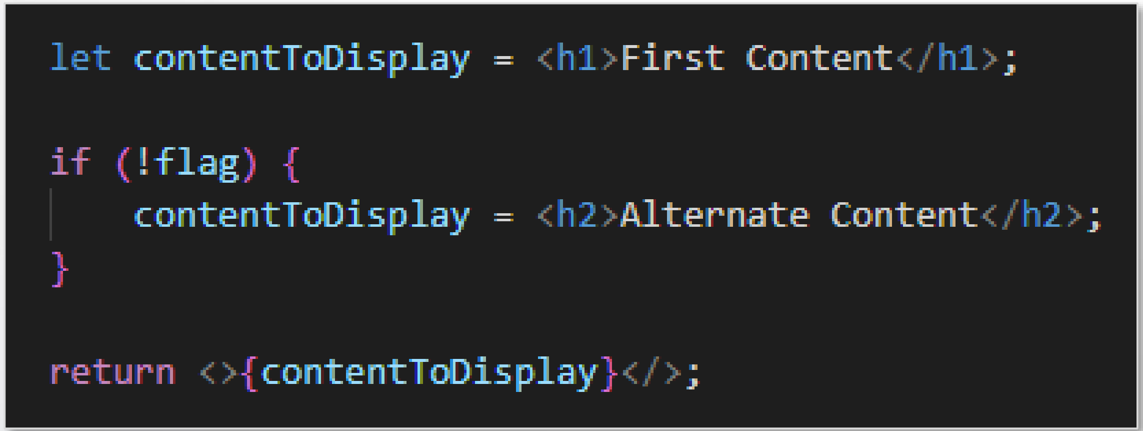


```
{flag && <h1>First Content</h1>}

{!flag && <h2>Alternate Content</h2>}
```

Display if flag is true

Display if flag is false

- JSX can also be assigned to variable

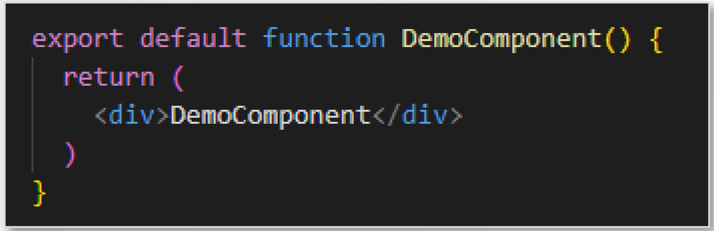


```
let contentToDisplay = <h1>First Content</h1>;

if (!flag) {
    contentToDisplay = <h2>Alternate Content</h2>;
}

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

Splitting Components

- Creating new components
- Passing props
- Handling events

Creating new components



```
export default function DemoComponent() {
  return (
    <div>DemoComponent</div>
  )
}
```