React

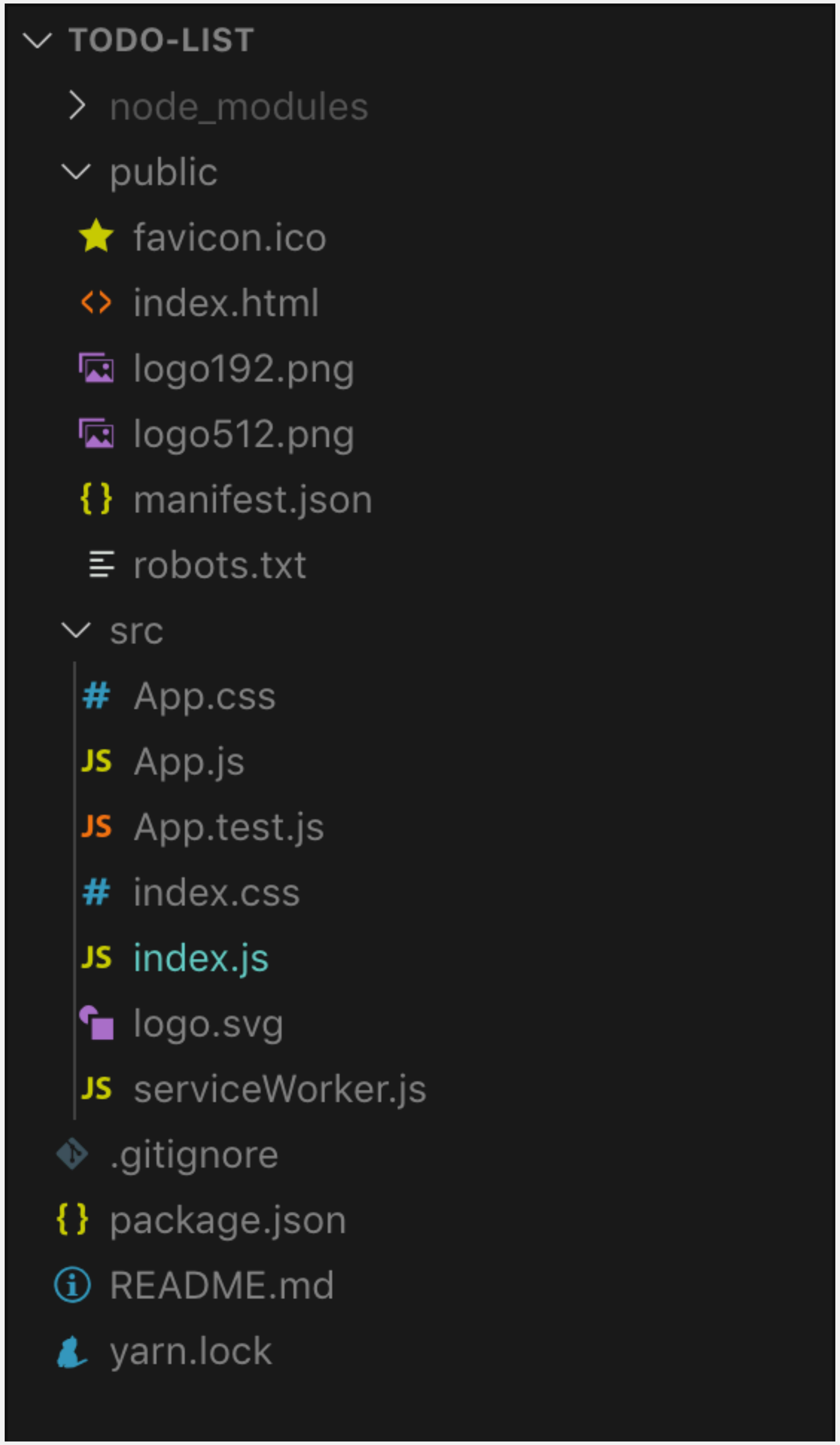
React

- A JavaScript library for building user interfaces
- Declarative
- Component-based
- Technology stack agnostic

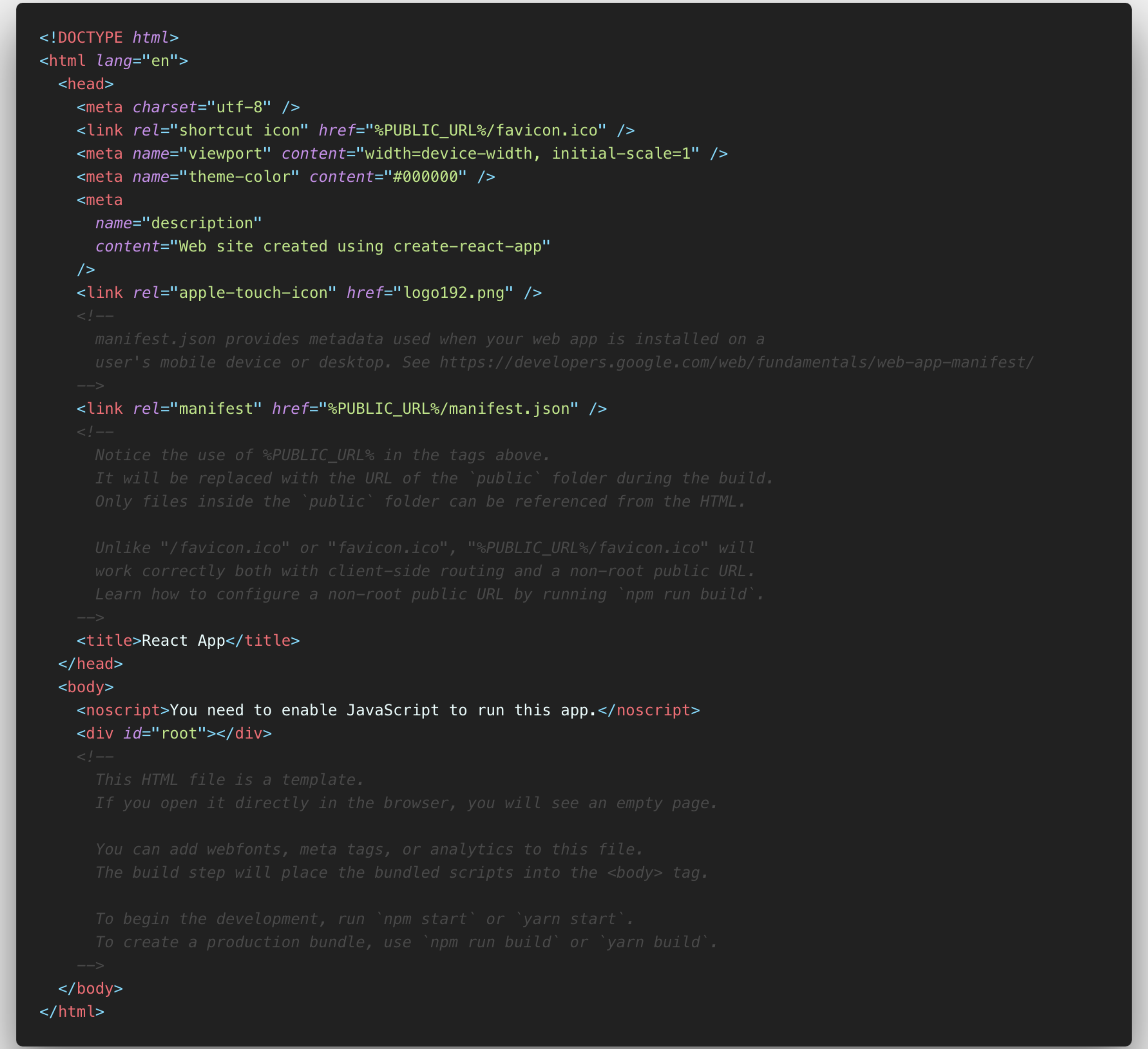
- Designed by Jordan Walke
- First deployed on Facebook’s newsfeed 2011
- Open sourced in May 2013
- Designed for speed, simplicity and scalability

Hello world

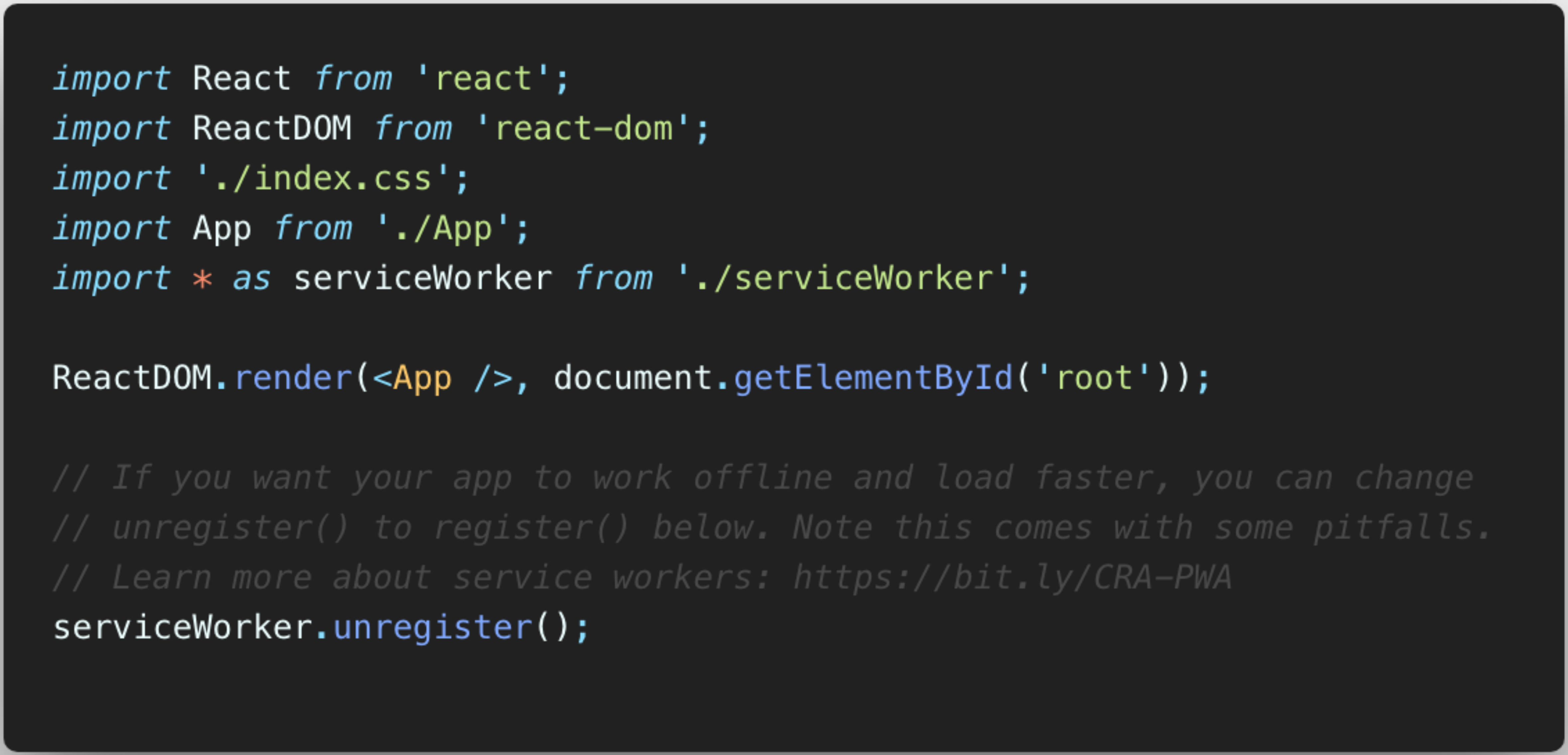
```
npx create-react-app my-app
cd my-app
npm run start
```



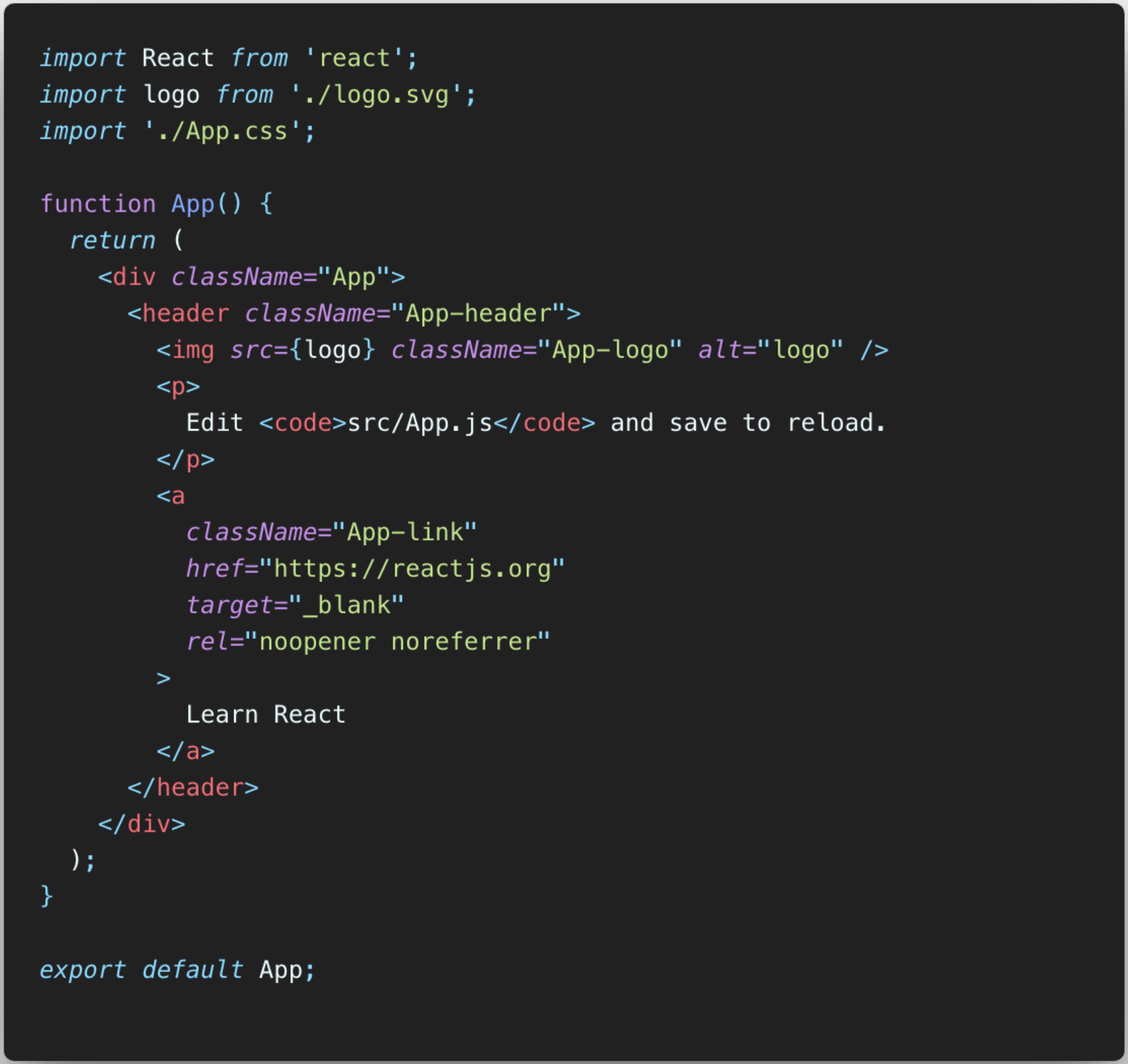
```
TODO-LIST
  node_modules
  public
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json
    robots.txt
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    serviceWorker.js
  .gitignore
  package.json
  README.md
  yarn.lock
```



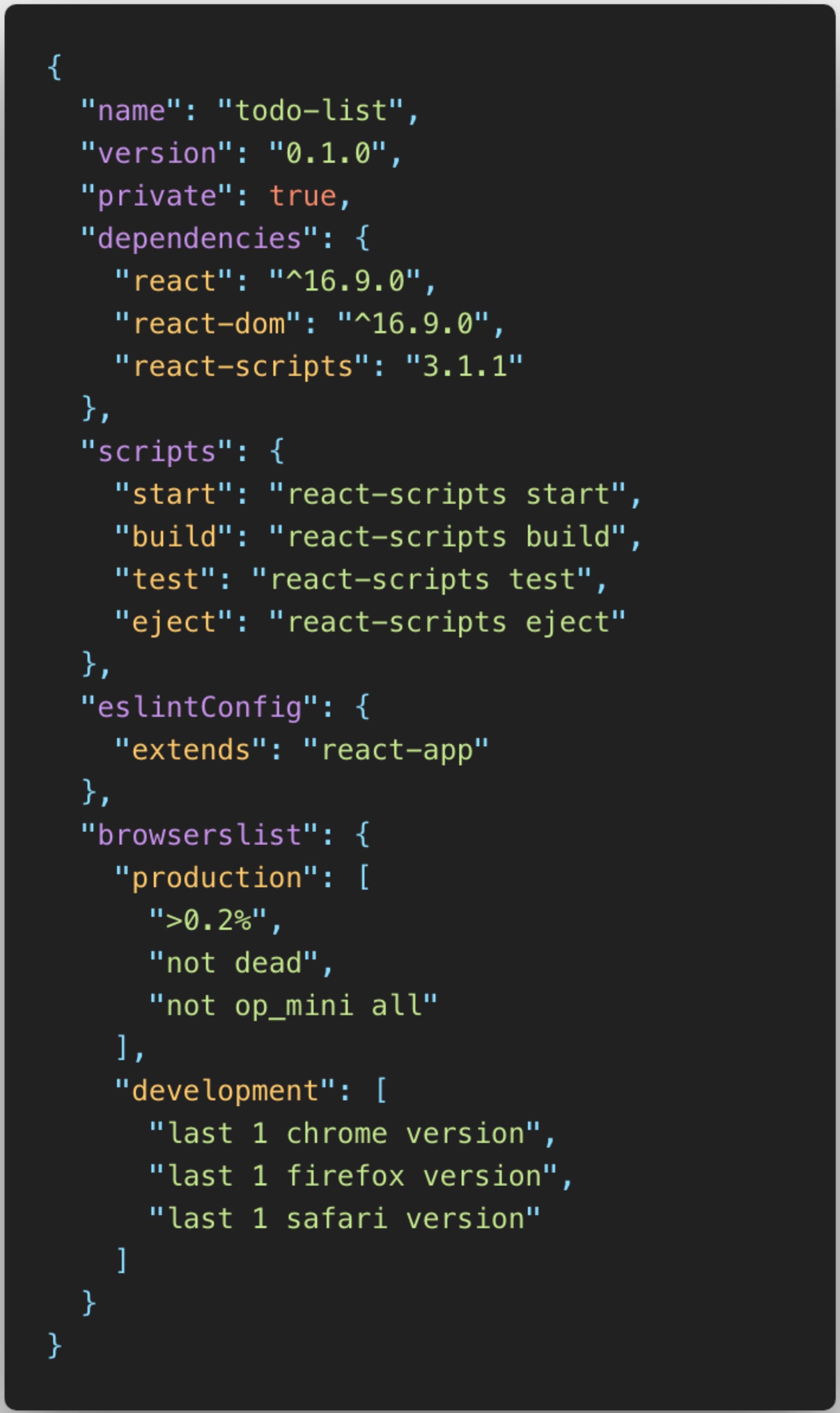
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```



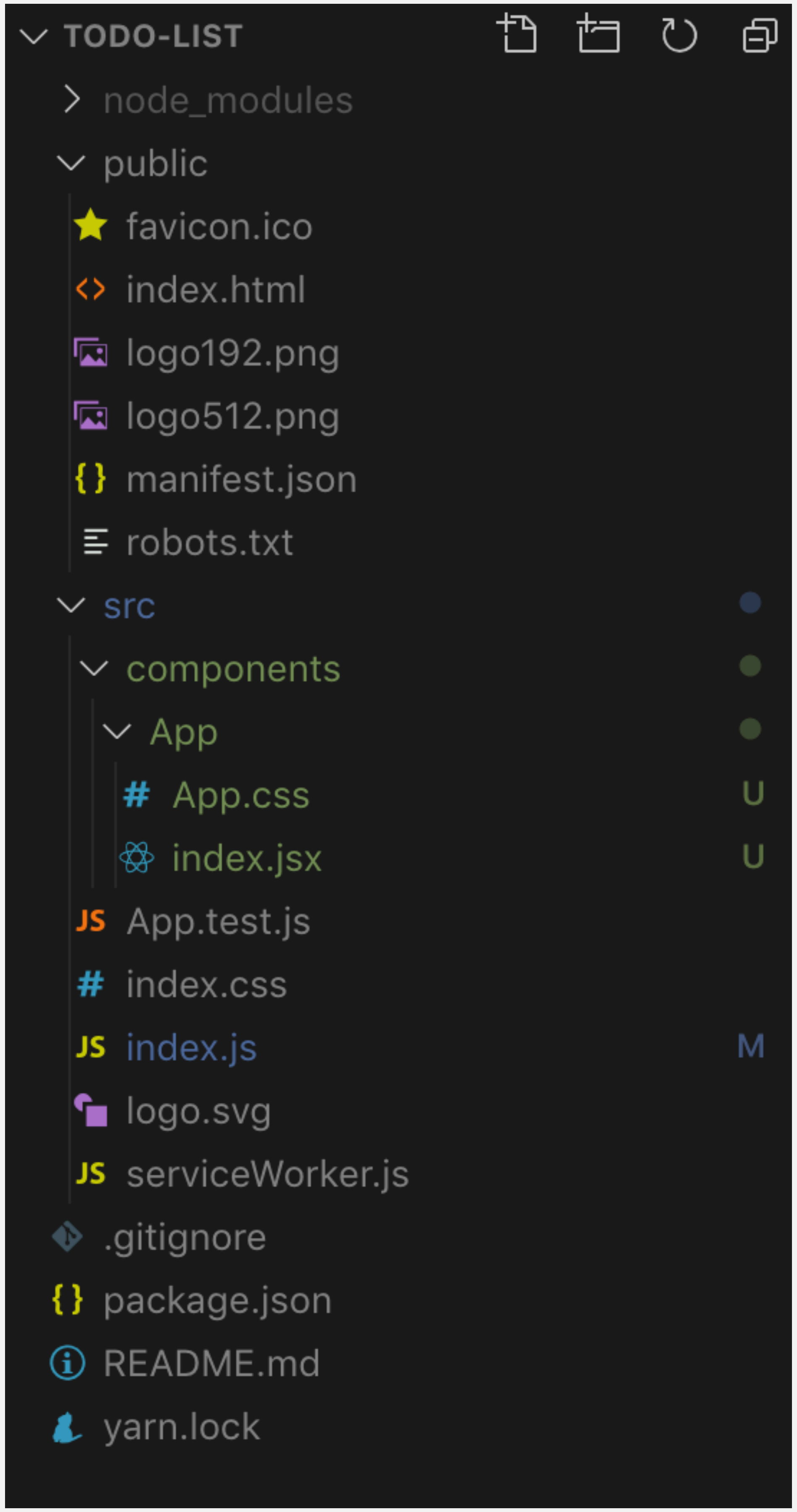
```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```



```jsx
import React from 'react';
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;
```

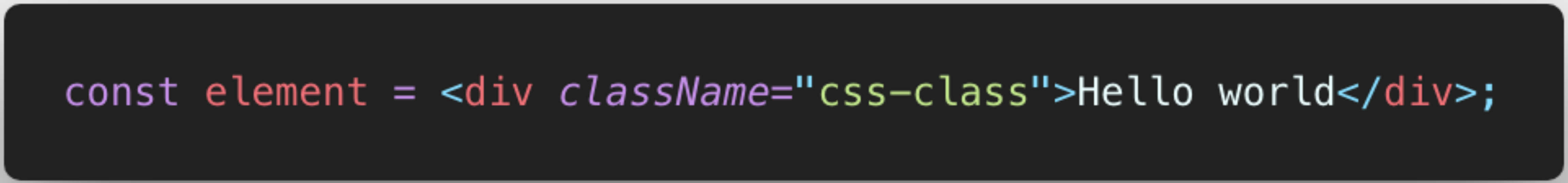


```json
{
  "name": "todo-list",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.9.0",
    "react-dom": "^16.9.0",
    "react-scripts": "3.1.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

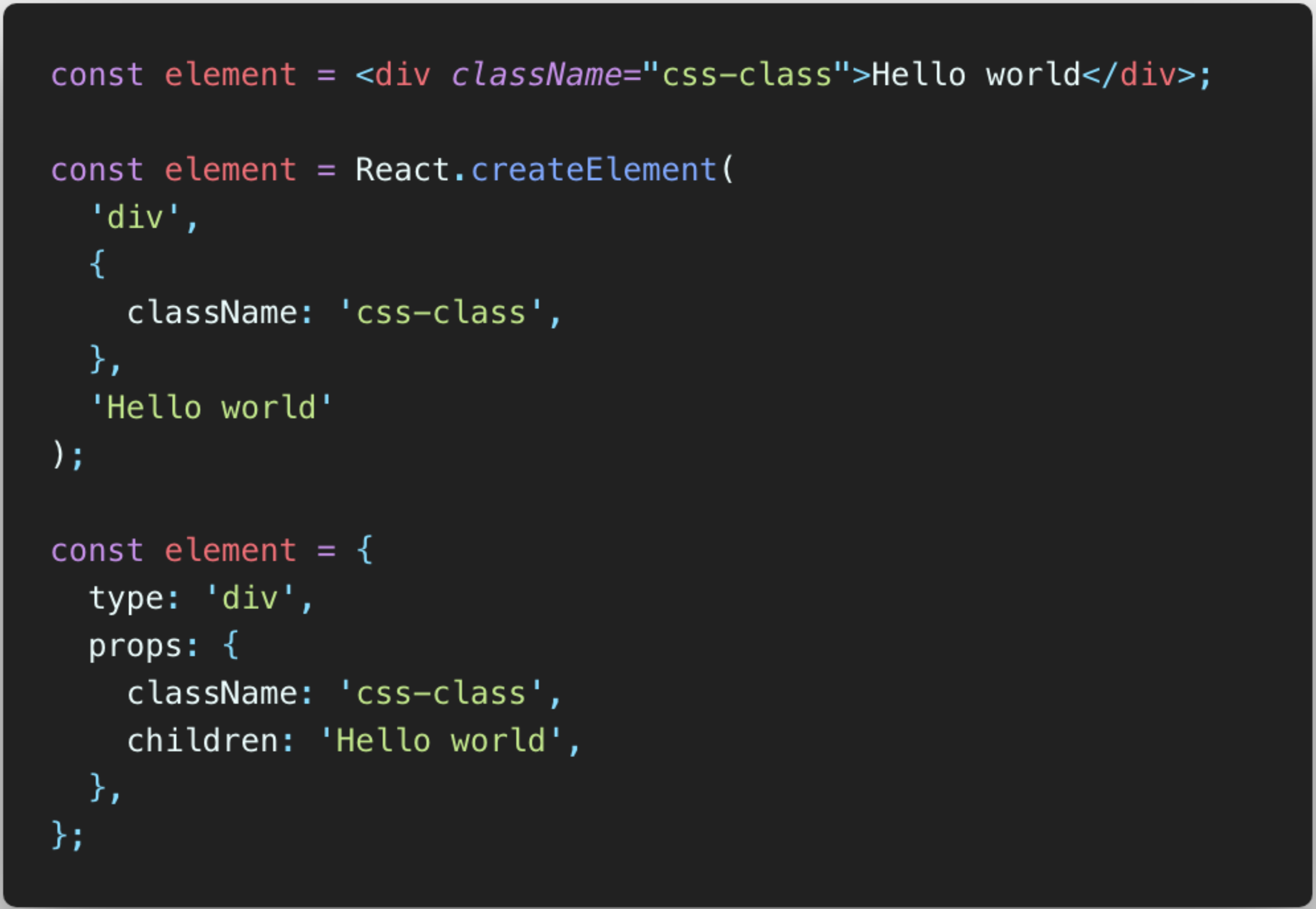


```
TODO-LIST
  node_modules
  public
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json
    robots.txt
  src
    components
      App
        App.css          U
        index.jsx        U
    App.test.js
    index.css
    index.js             M
    logo.svg
    serviceWorker.js
  .gitignore
  package.json
  README.md
  yarn.lock
```

Elements



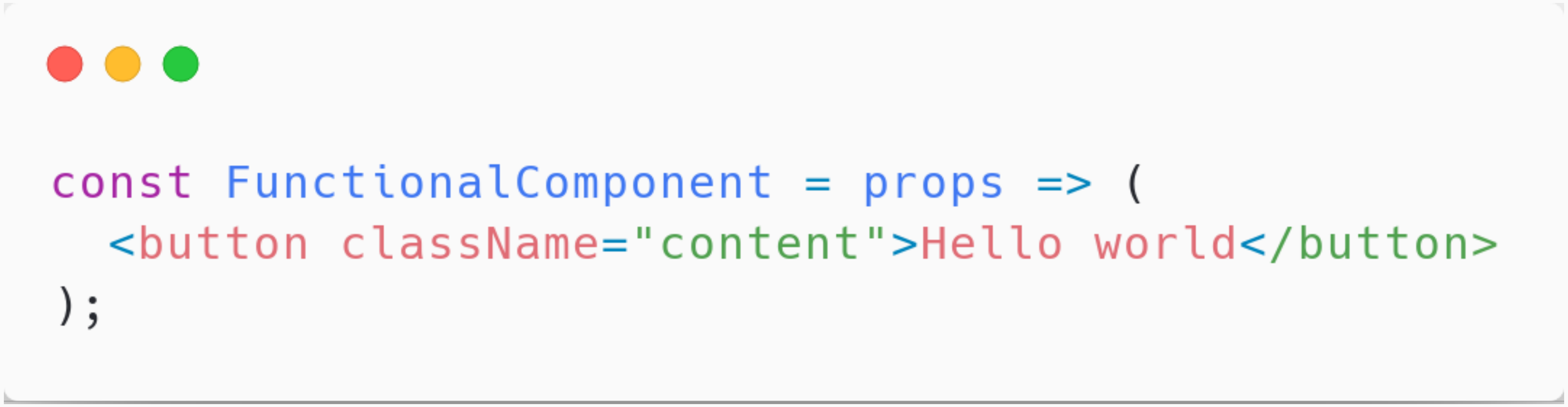
```
const element = <div className="css-class">Hello world</div>;
```



```
const element = <div className="css-class">Hello world</div>;

const element = React.createElement(
  'div',
  {
    className: 'css-class',
  },
  'Hello world'
);

const element = {
  type: 'div',
  props: {
    className: 'css-class',
    children: 'Hello world',
  },
};
```

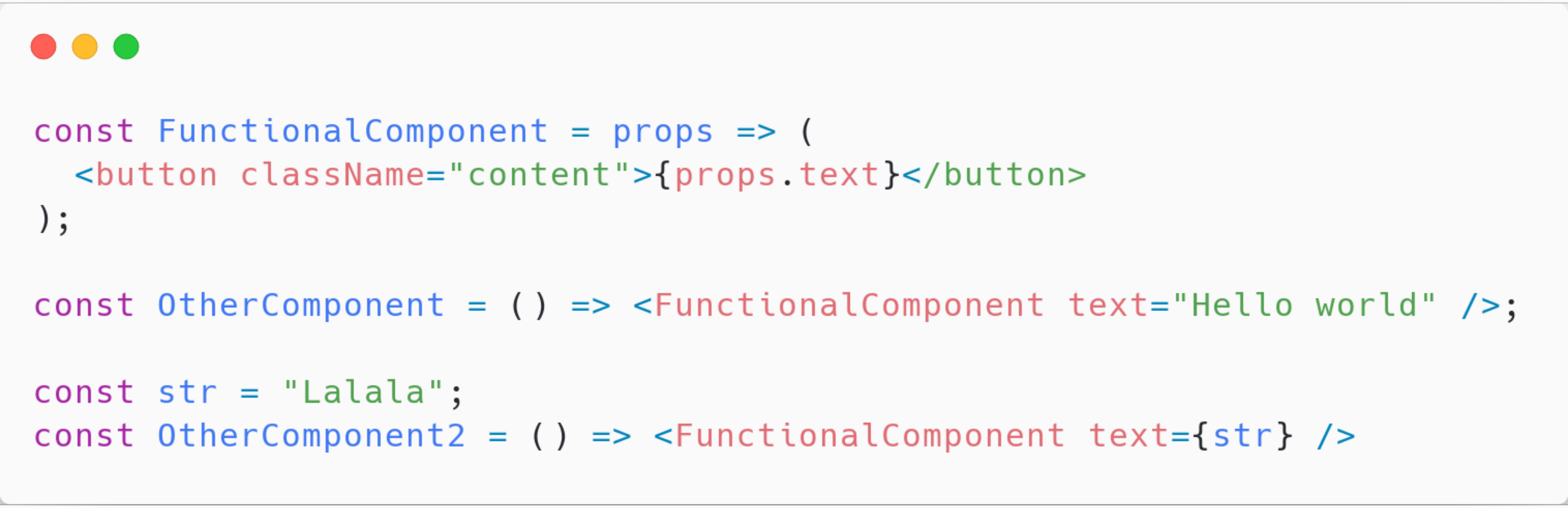
Components

Functional component



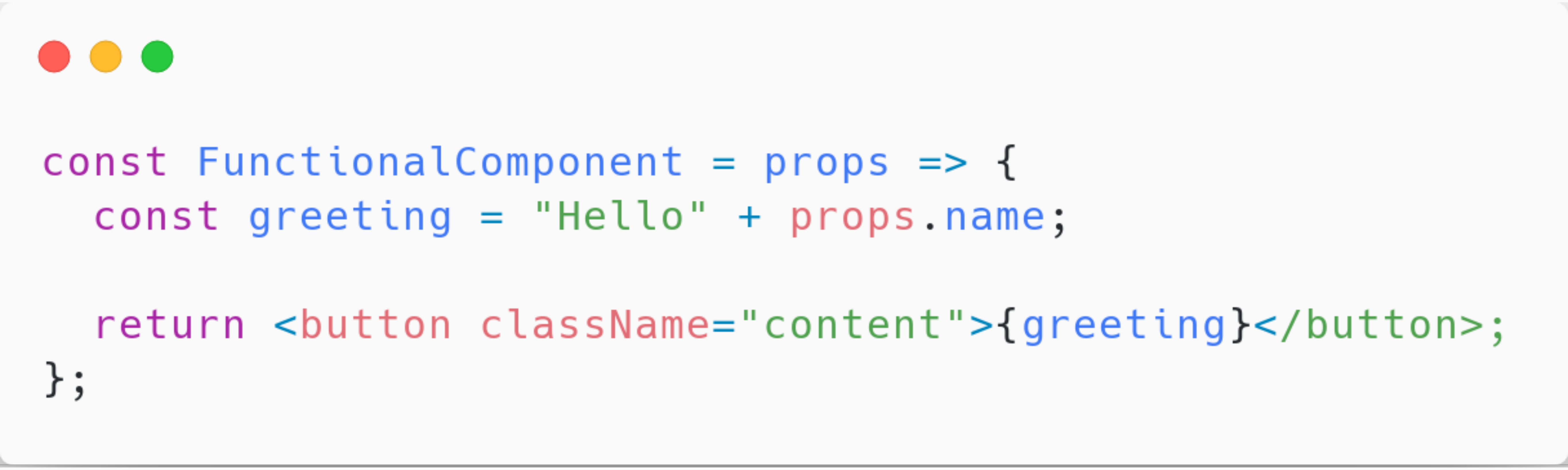
```
const FunctionalComponent = props => (
  <button className="content">Hello world</button>
);
```

Functional component



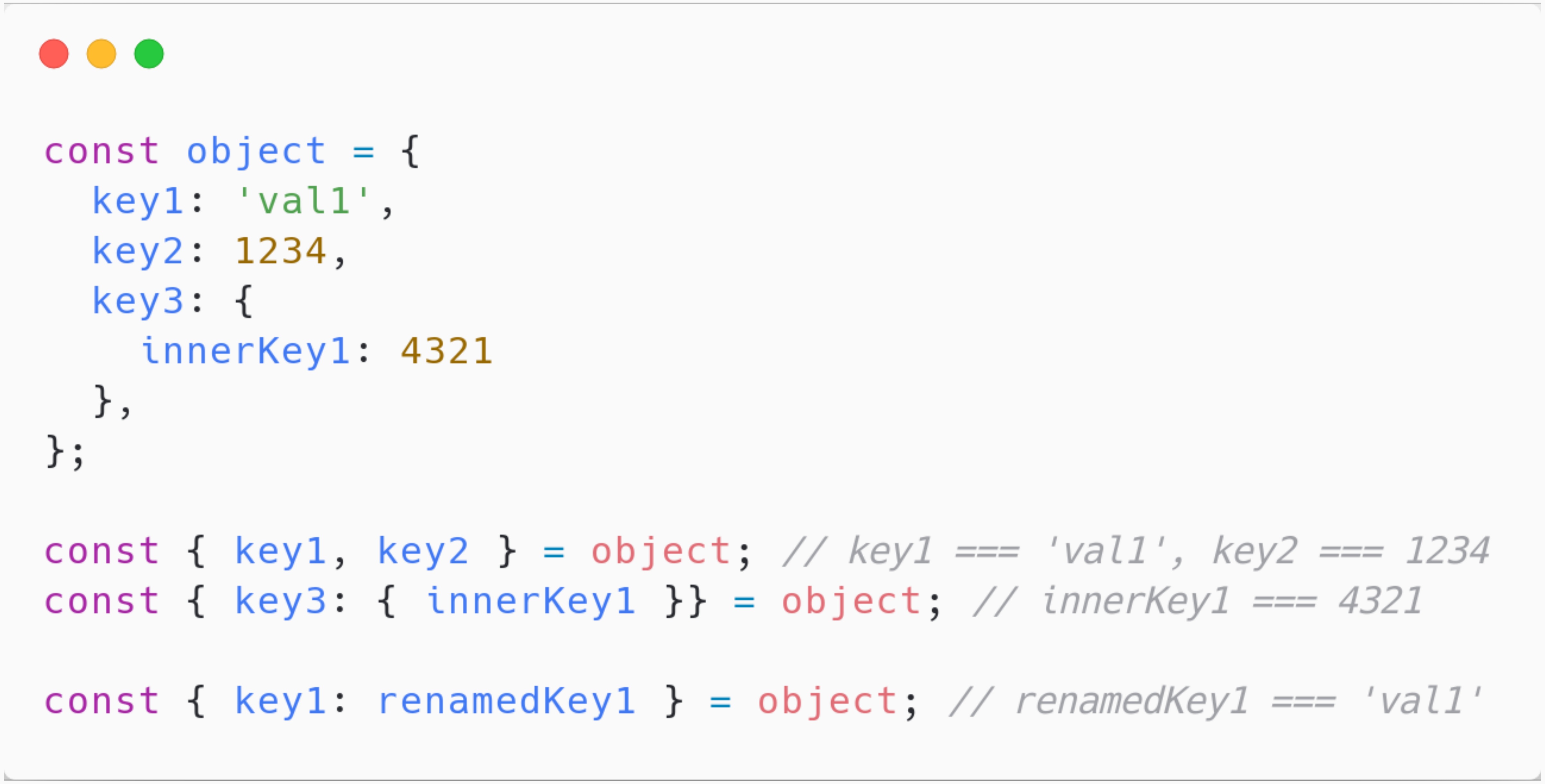
```
const FunctionalComponent = props => (
  <button className="content">{props.text}</button>
);

const OtherComponent = () => <FunctionalComponent text="Hello world" />;

const str = "Lalala";
const OtherComponent2 = () => <FunctionalComponent text={str} />
```

Functional component



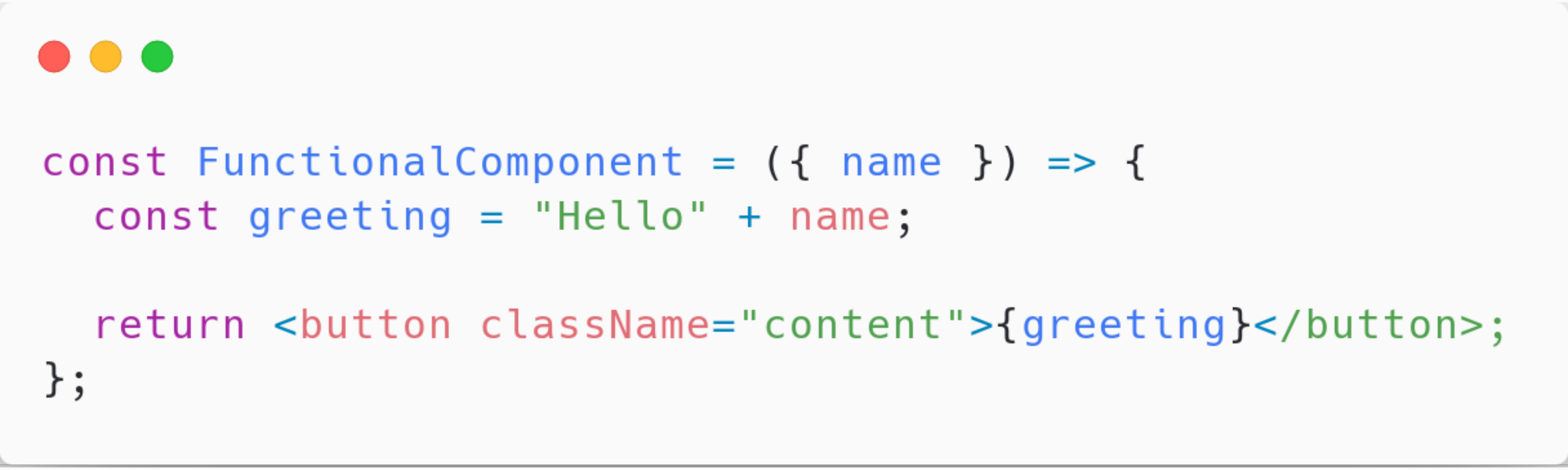
```jsx
const FunctionalComponent = props => {
  const greeting = "Hello" + props.name;

  return <button className="content">{greeting}</button>;
};
```

Object destructuring



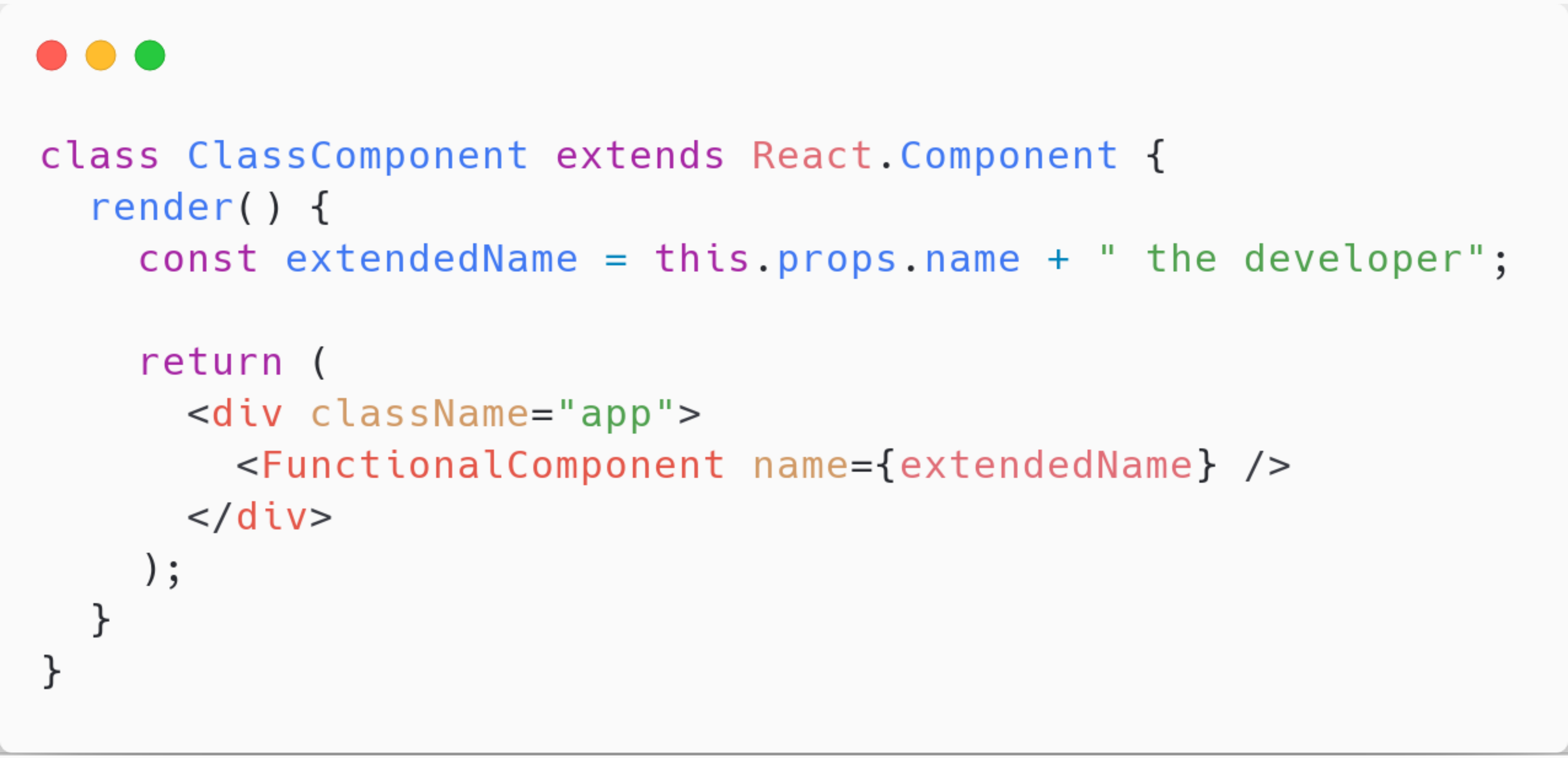
```
const object = {
  key1: 'val1',
  key2: 1234,
  key3: {
    innerKey1: 4321
  },
};

const { key1, key2 } = object; // key1 === 'val1', key2 === 1234
const { key3: { innerKey1 }} = object; // innerKey1 === 4321

const { key1: renamedKey1 } = object; // renamedKey1 === 'val1'
```

Functional component



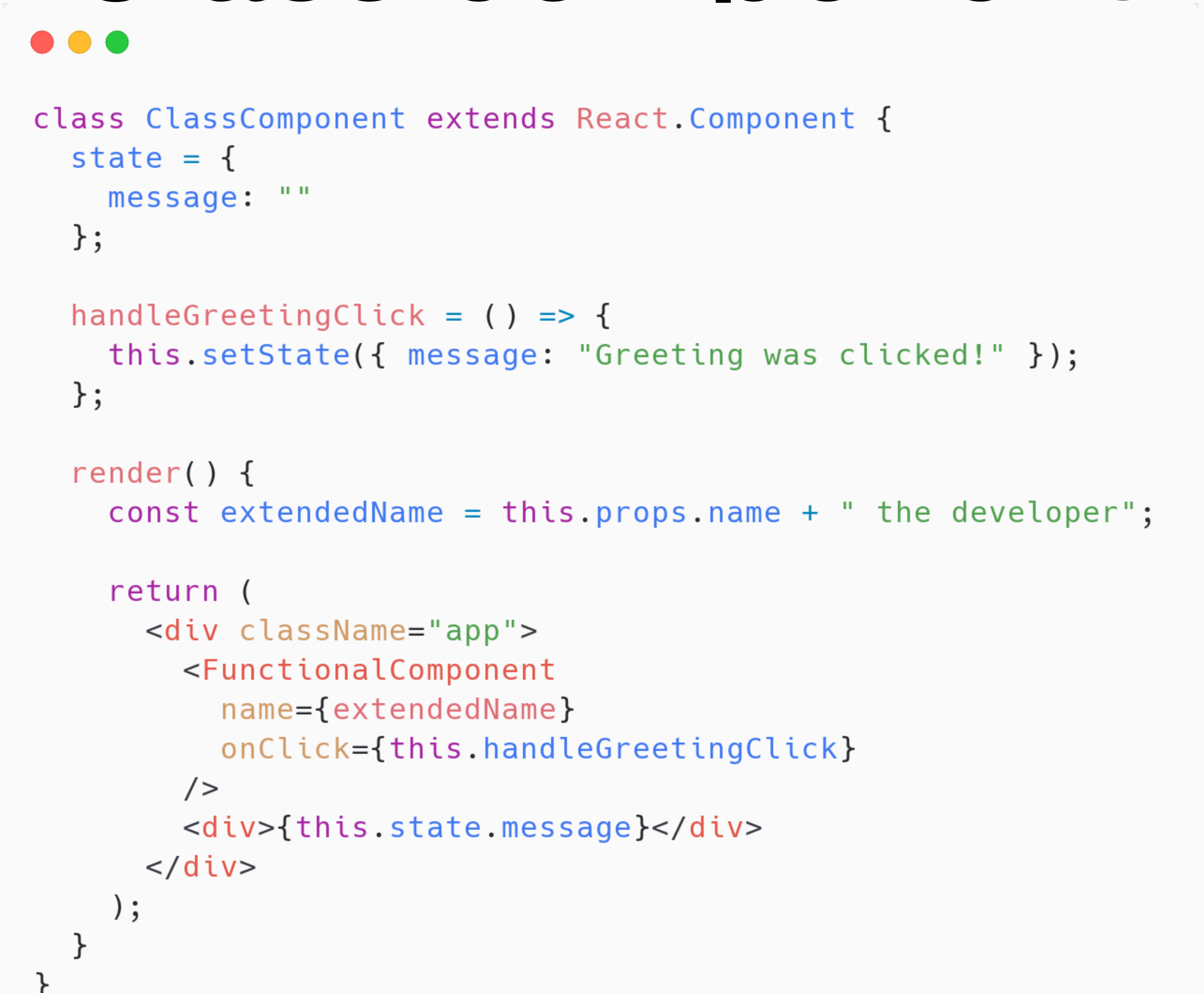
```
const FunctionalComponent = ({ name }) => {
  const greeting = "Hello" + name;

  return <button className="content">{greeting}</button>;
};
```

Class component



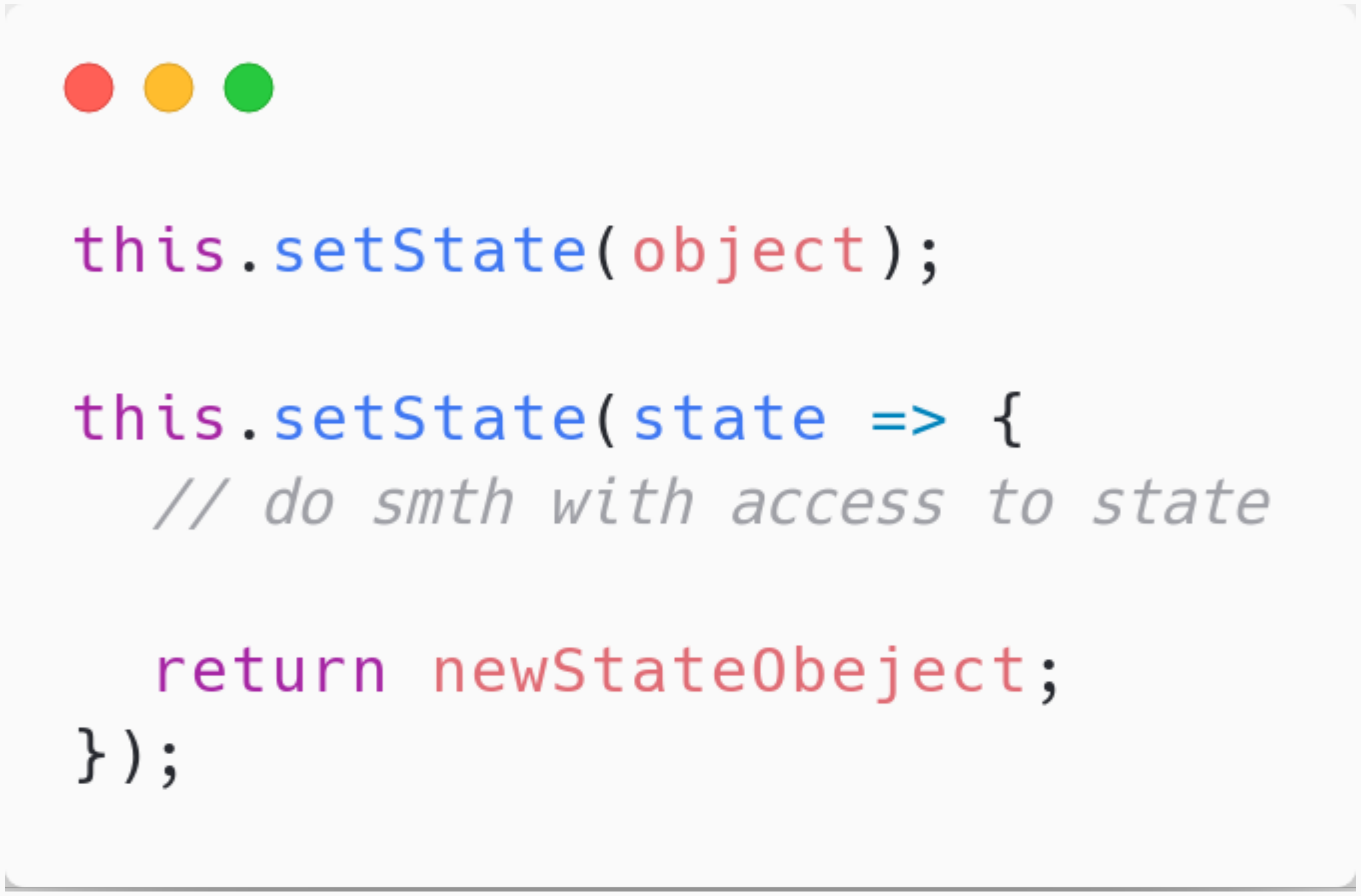
```
class ClassComponent extends React.Component {
  render() {
    const extendedName = this.props.name + " the developer";

    return (
      <div className="app">
        <FunctionalComponent name={extendedName} />
      </div>
    );
  }
}
```

Class component



```
class ClassComponent extends React.Component {
  state = {
    message: ""
  };

  handleGreetingClick = () => {
    this.setState({ message: "Greeting was clicked!" });
  };

  render() {
    const extendedName = this.props.name + " the developer";

    return (
      <div className="app">
        <FunctionalComponent
          name={extendedName}
          onClick={this.handleGreetingClick}
        />
        <div>{this.state.message}</div>
      </div>
    );
  }
}
```

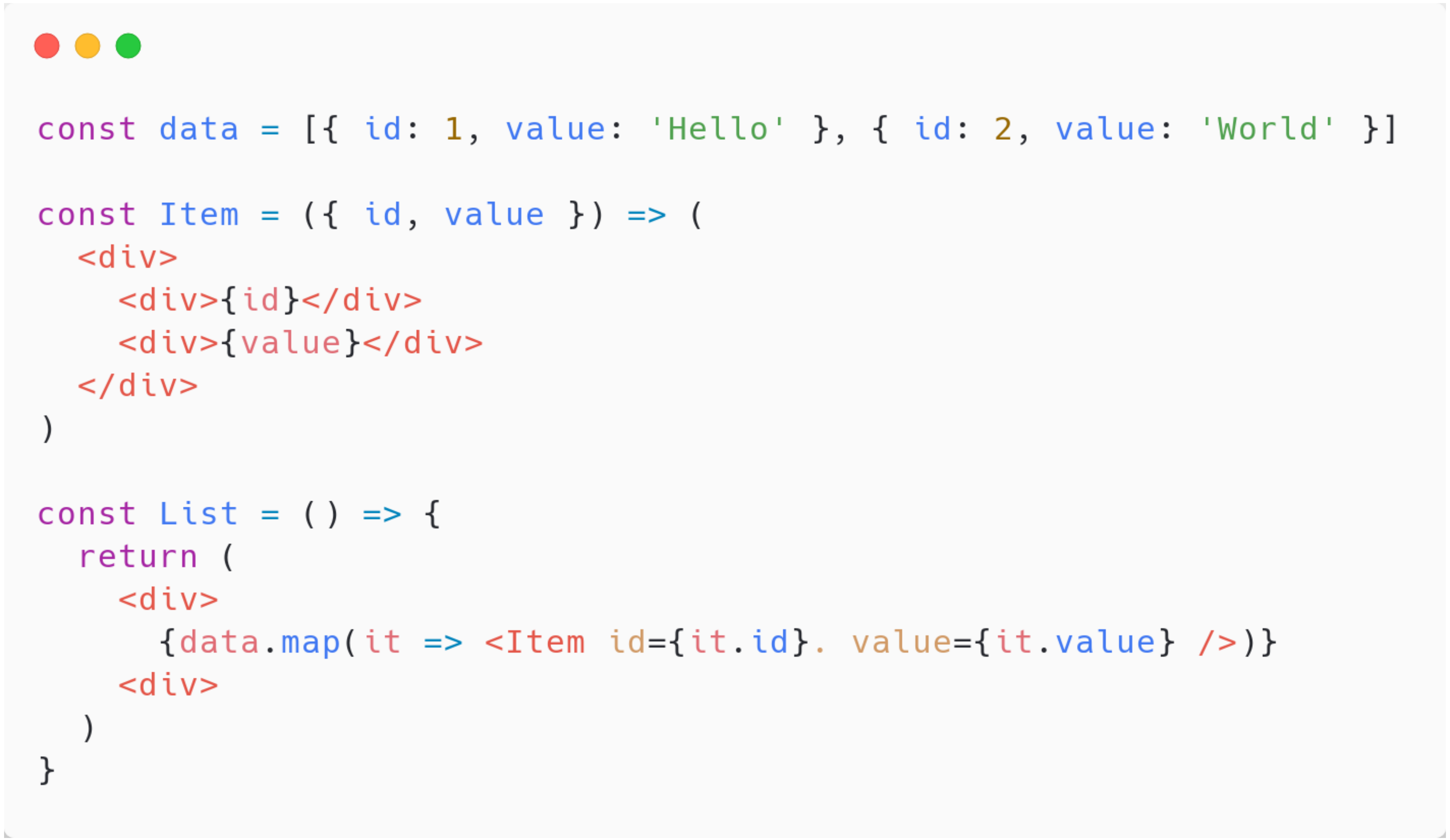
Class component



```
this.setState(object);

this.setState(state => {
  // do smth with access to state

  return newStateObeject;
});
```

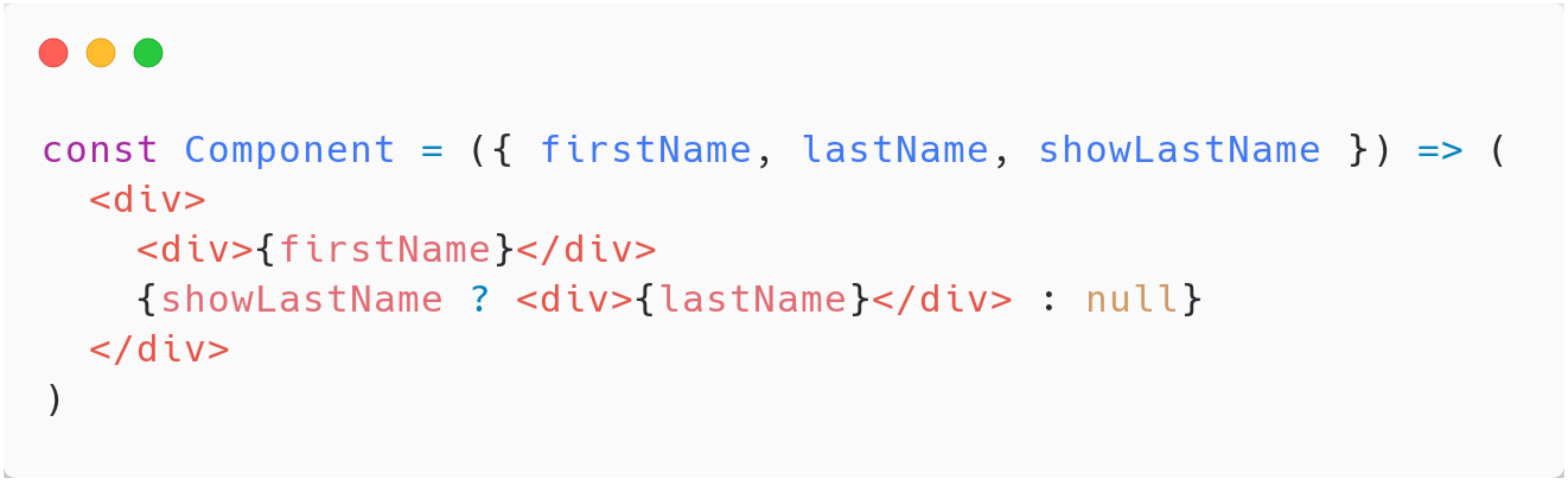
Реализация counter’a

Map data to components



```
const data = [{ id: 1, value: 'Hello' }, { id: 2, value: 'World' }]

const Item = ({ id, value }) => (
  <div>
    <div>{id}</div>
    <div>{value}</div>
  </div>
)

const List = () => {
  return (
    <div>
      {data.map(it => <Item id={it.id}. value={it.value} />)}
    <div>
  )
}
```

Conditional rendering



```
const Component = ({ firstName, lastName, showLastName }) => (
  <div>
    <div>{firstName}</div>
    {showLastName ? <div>{lastName}</div> : null}
  </div>
)
```