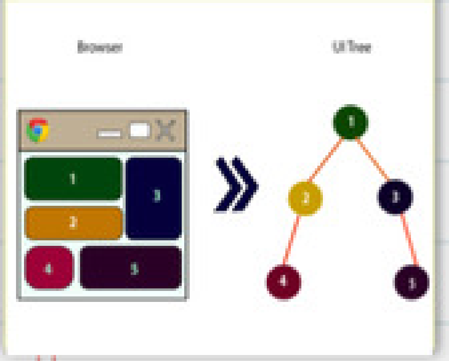
React Components

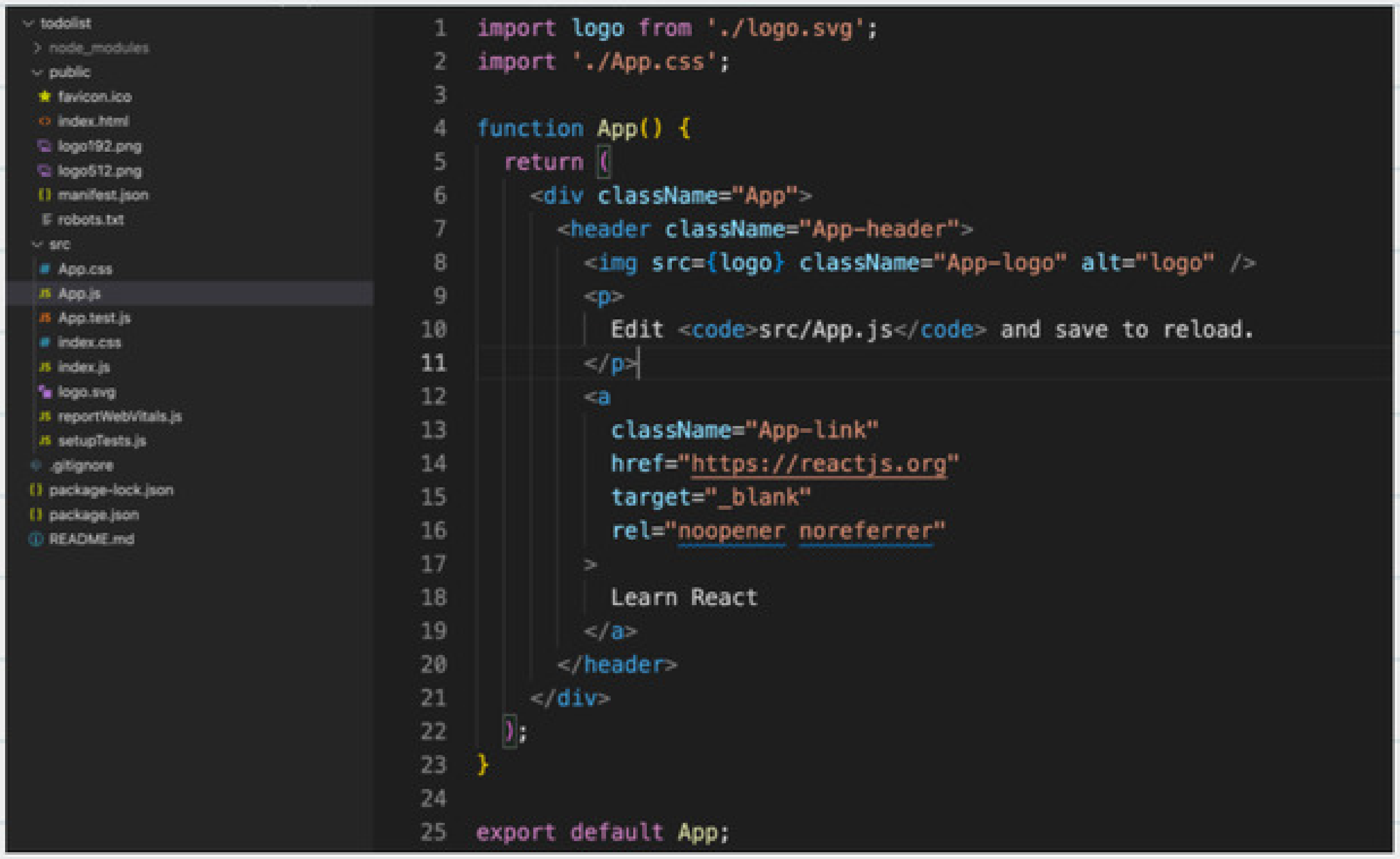


React Components

You already created your first React application.

This application comes with a series of files that do various things, mostly related to configuration, but there's one file that stands out: App.js.

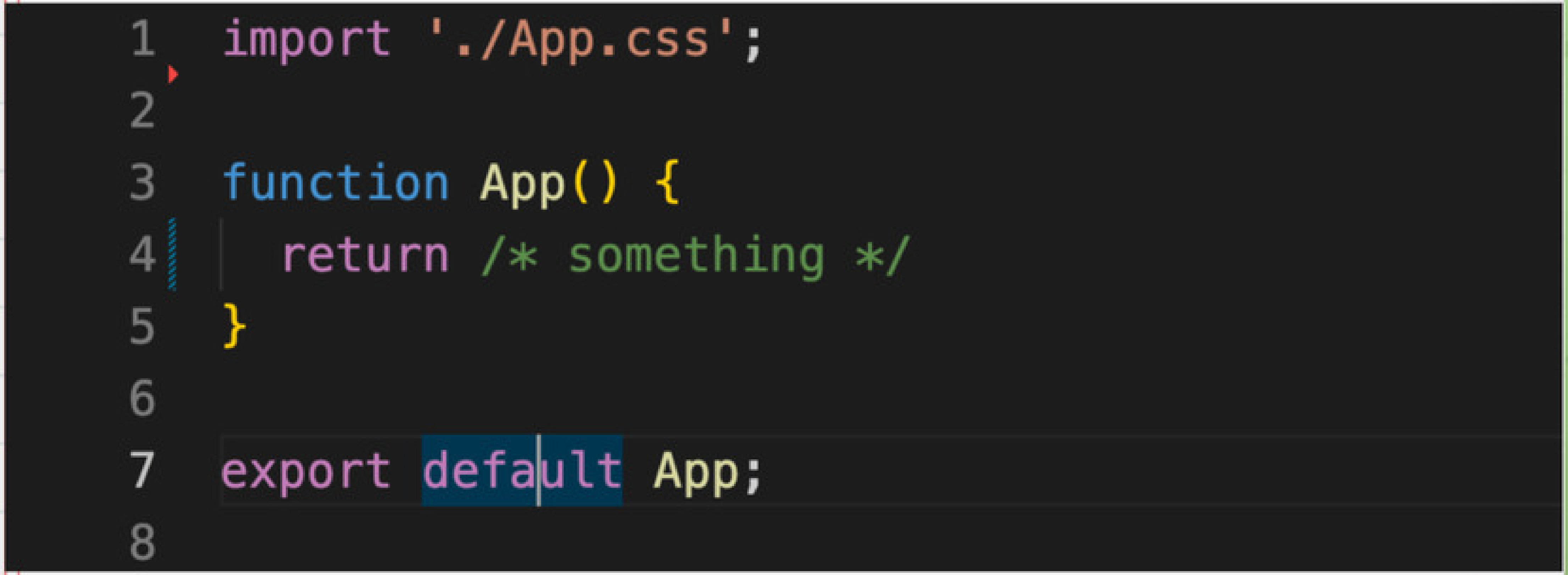
App.js is the first React Component you meet.



```
todolist
  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
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

```jsx
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;
```

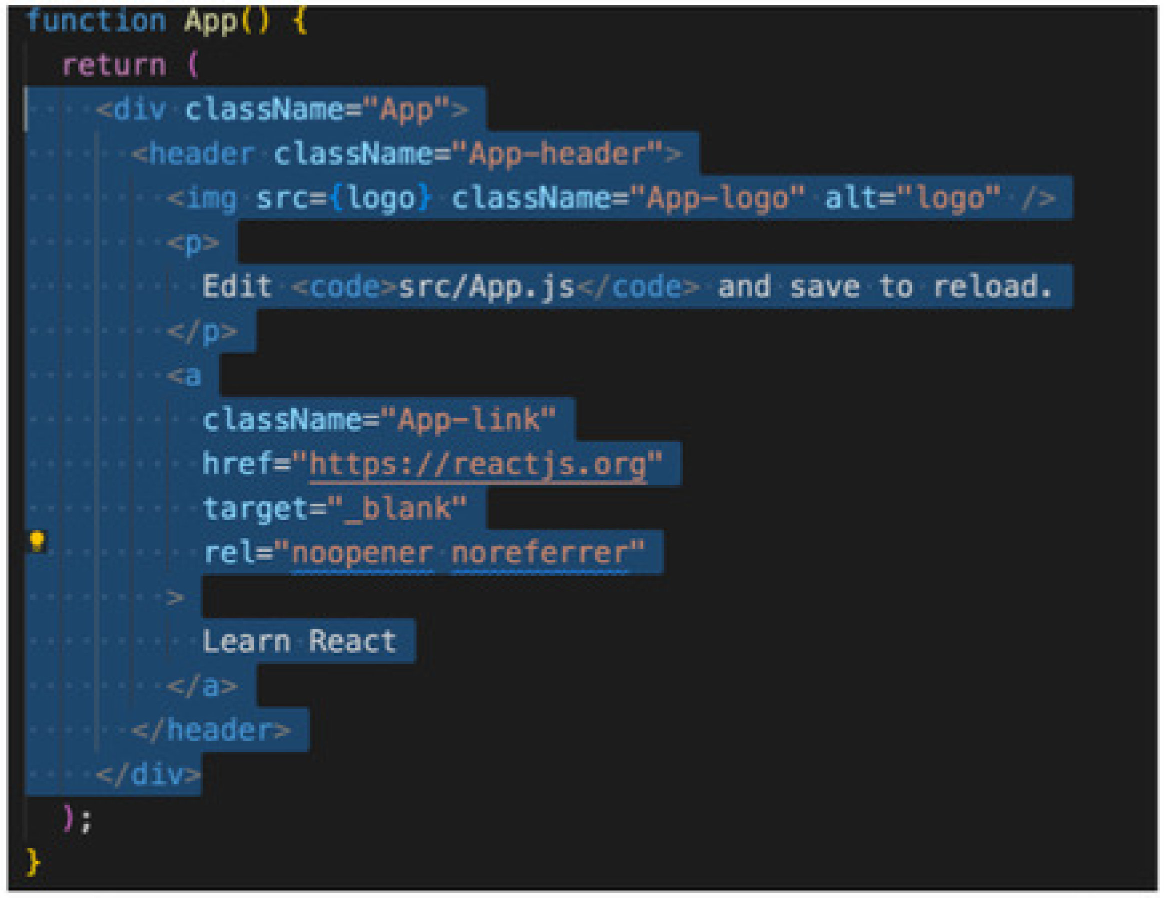
Let's start by analyzing this first component. I'm going to simplify this component code like this:



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

function App() {
  return /* something */
}

export default App;

```

You can see a few things here. We import some things, and we export a function called App.

App is a function that, in the original example, returns something that at first sight looks quite strange.
It looks like HTML but it has some JavaScript embedded into it.
That is JSX, a special language we use to build a component's output. We'll talk more about JSX in the next section.



```
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>
  );
}
```

Take 12 minutes to watch the next video

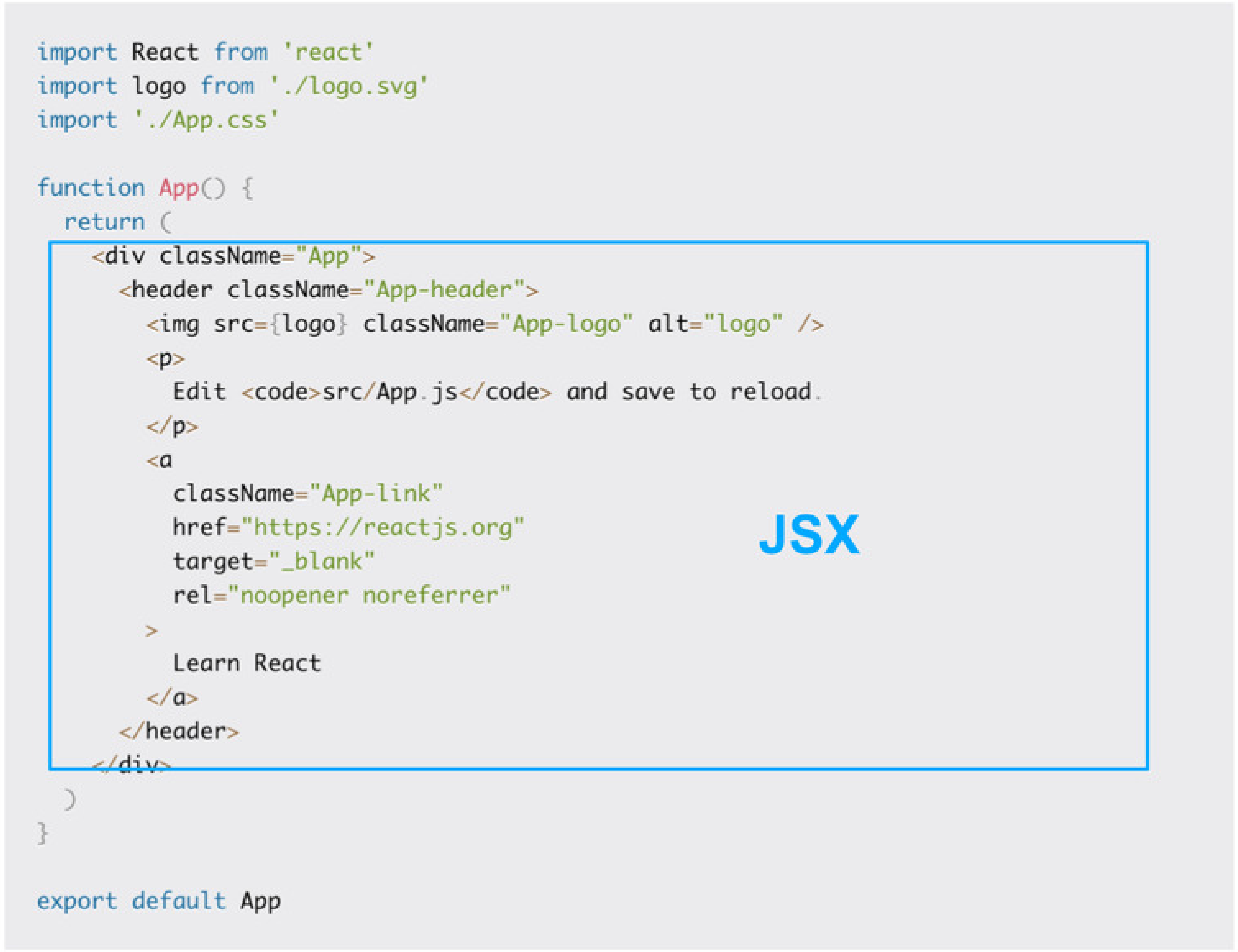
Introduction to JSX: React Crash Course For Beginners Tutorial

Introduction to JSX

We can't talk about React without first explaining JSX.

In the last section you met your first React component, the `App` component defined in the default application built by `create-react-app`.

We call JSX everything wrapped inside the parentheses returned by the component:



```
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
```

This *looks* like HTML, but it's not really HTML. It's a little different.

And it's a bit strange to have this code inside a JavaScript file. This does not look like JavaScript at all!

Using JSX to compose a UI

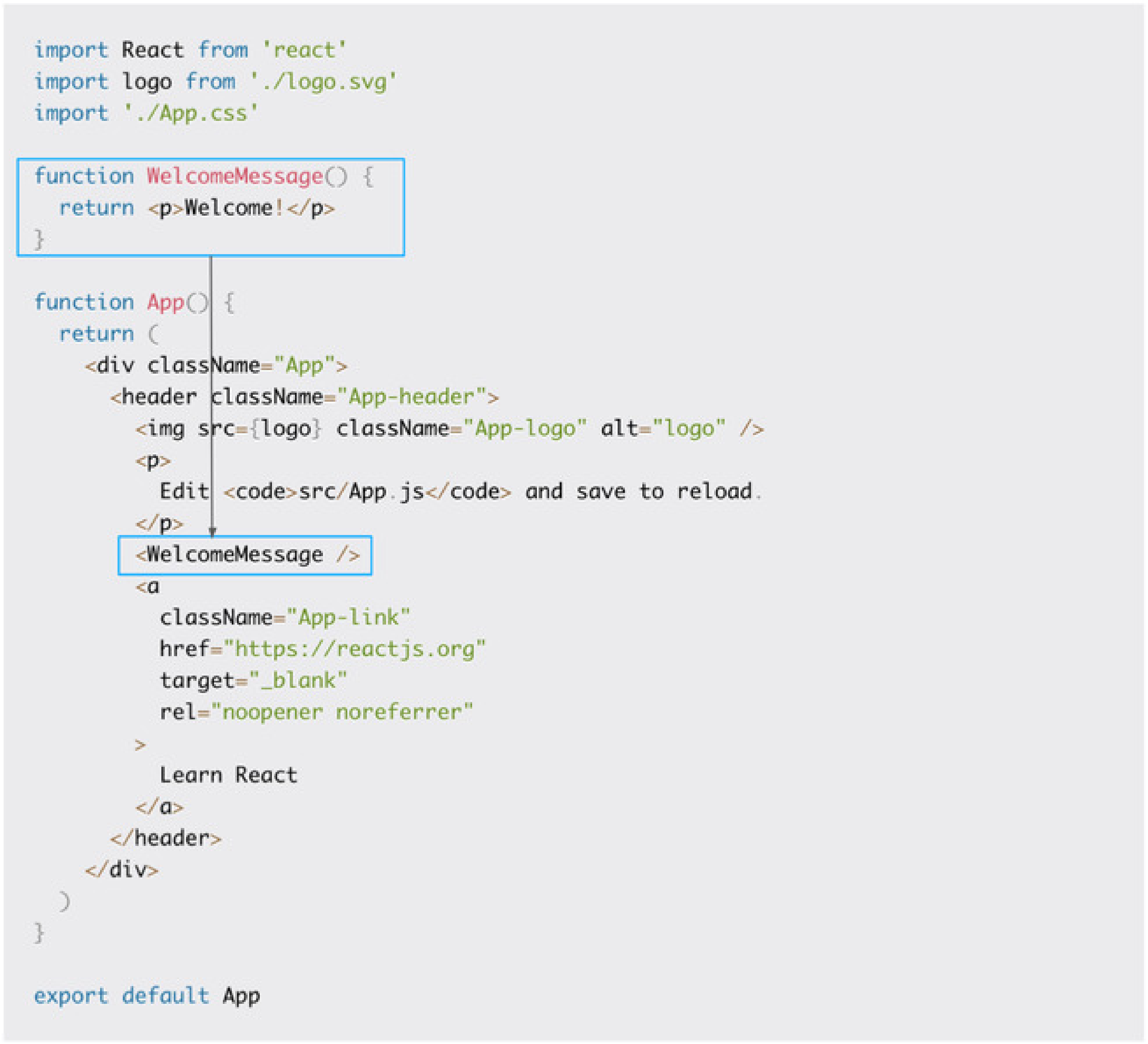
- In particular, in a React component you can import other React components, and you can embed them and display them.
- A React component is usually created in its own file, because that's how we can easily reuse it (by importing it) in other components.
- But a React component can also be created in the same file of another component, if you plan to only use it in that component. There's no "rule" here, you can do what feels best to you.

We're going to create a `WelcomeMessage` component:

```
function WelcomeMessage() {
  return <p>Welcome!</p>
}
```

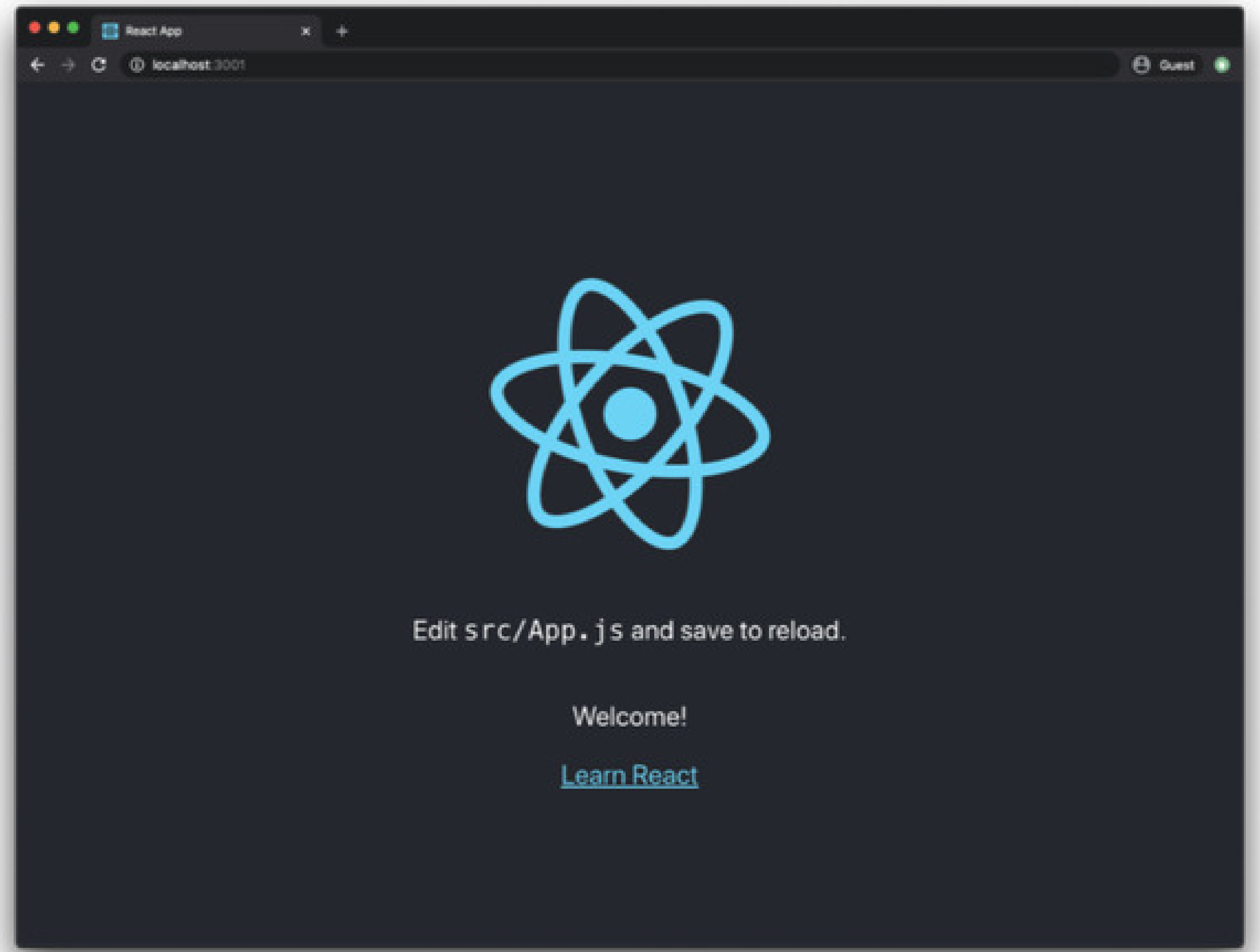
See? It's a simple function that returns a line of JSX that represents a `p` HTML element.

Now inside the `App` component JSX we can add `<WelcomeMessage />` to show this component in the user interface:



```
import React from 'react'
import logo from './logo.svg'
import './App.css'

function WelcomeMessage() {
  return <p>Welcome!</p>
}

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>
        <WelcomeMessage />
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  )
}

export default App
```

And here's the result. Can you see the "Welcome!" message in the screen?



Edit `src/App.js` and save to reload.

Welcome!

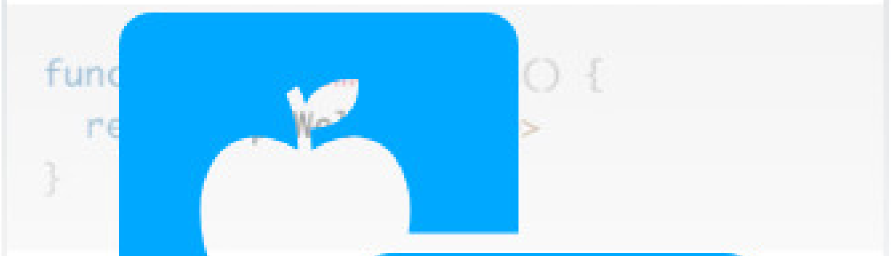
Learn React

We say that `WelcomeMessage` is a **child component** of App, and `App` is its parent componnet.

Match the word/sentences with the image.

Instructions

HTML



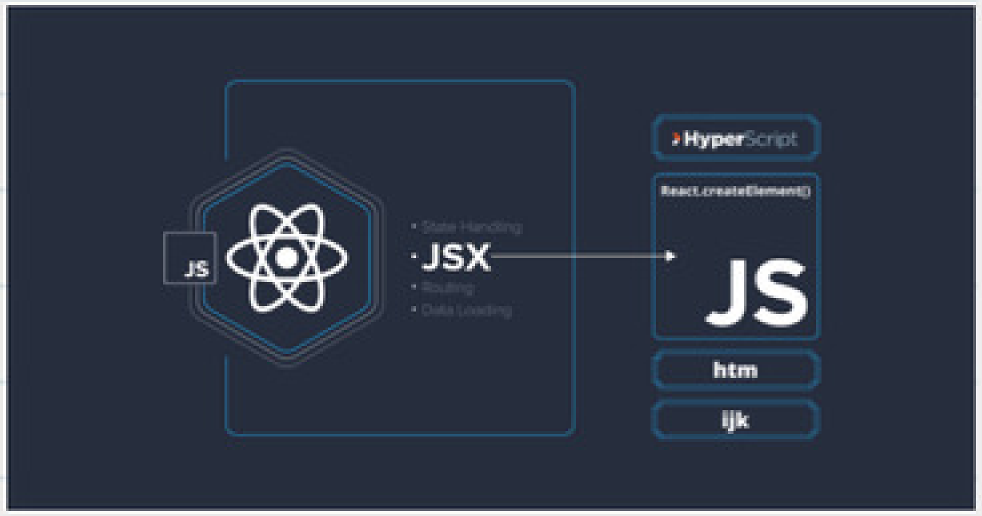
JSX

Matching Pairs

The difference between JSX and HTML

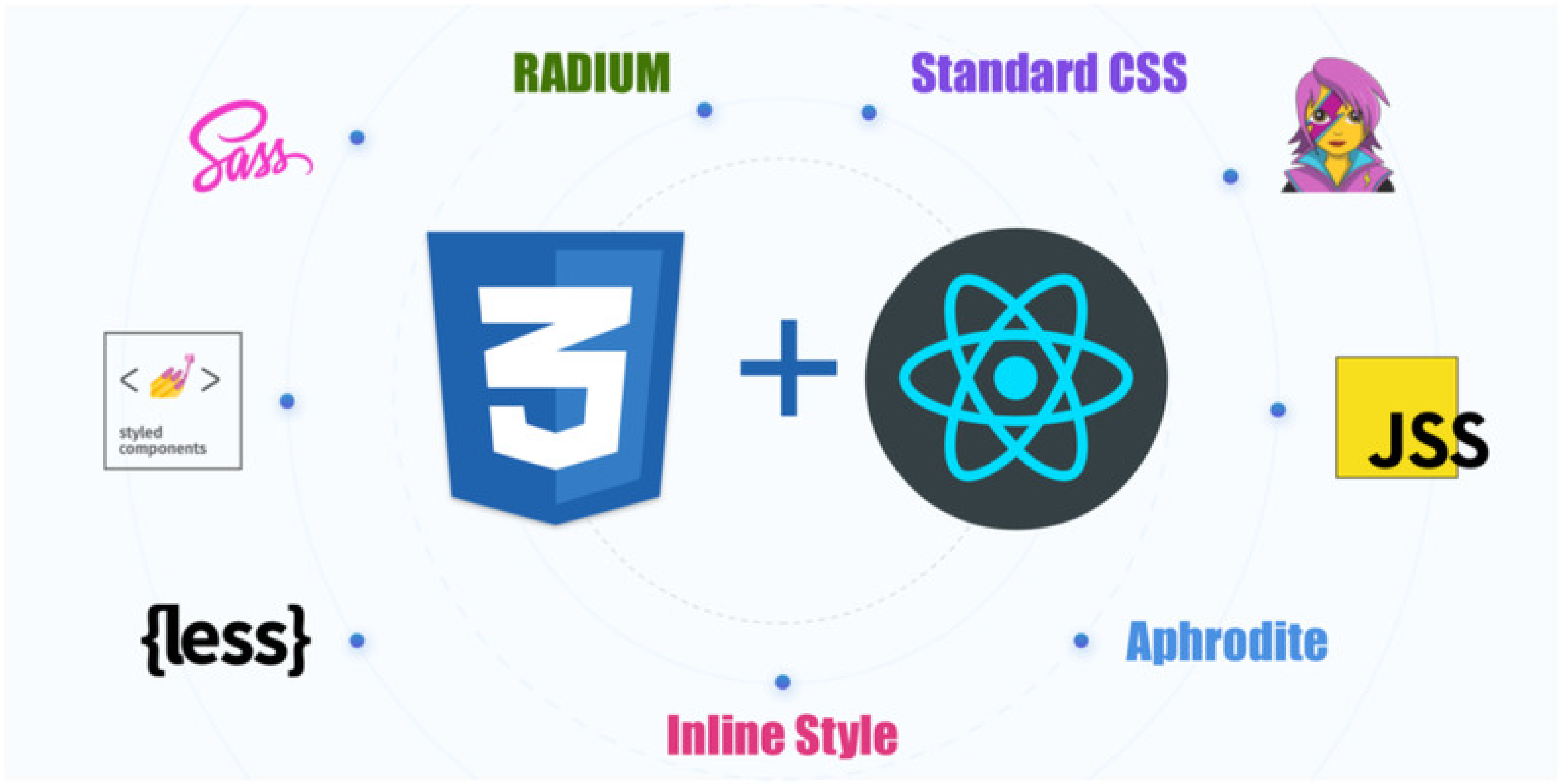
JSX kind of looks like HTML, but it's not.

HTML



One of the differences might be quite obvious if you looked at the `App` component JSX: there's a strange attribute called `className`.

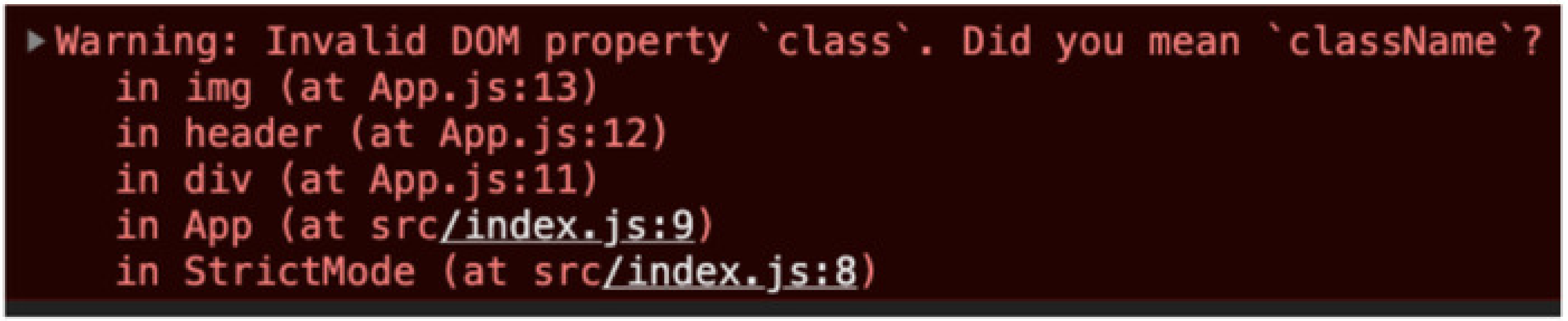
In HTML we use the `class` attribute. It's probably the most widely used attribute, for various reasons. One of those reasons is CSS. The `class` attribute allows us to style HTML elements easily, and CSS frameworks like Tailwind put this attribute to the center of the CSS user interface design process.



That's how we ended up with `className` instead of `class`.

You need to remember this especially when you're copy/pasting some existing HTML.

React will try its best to make sure things don't break, but it will raise a lot of warnings in the Developer Tools:



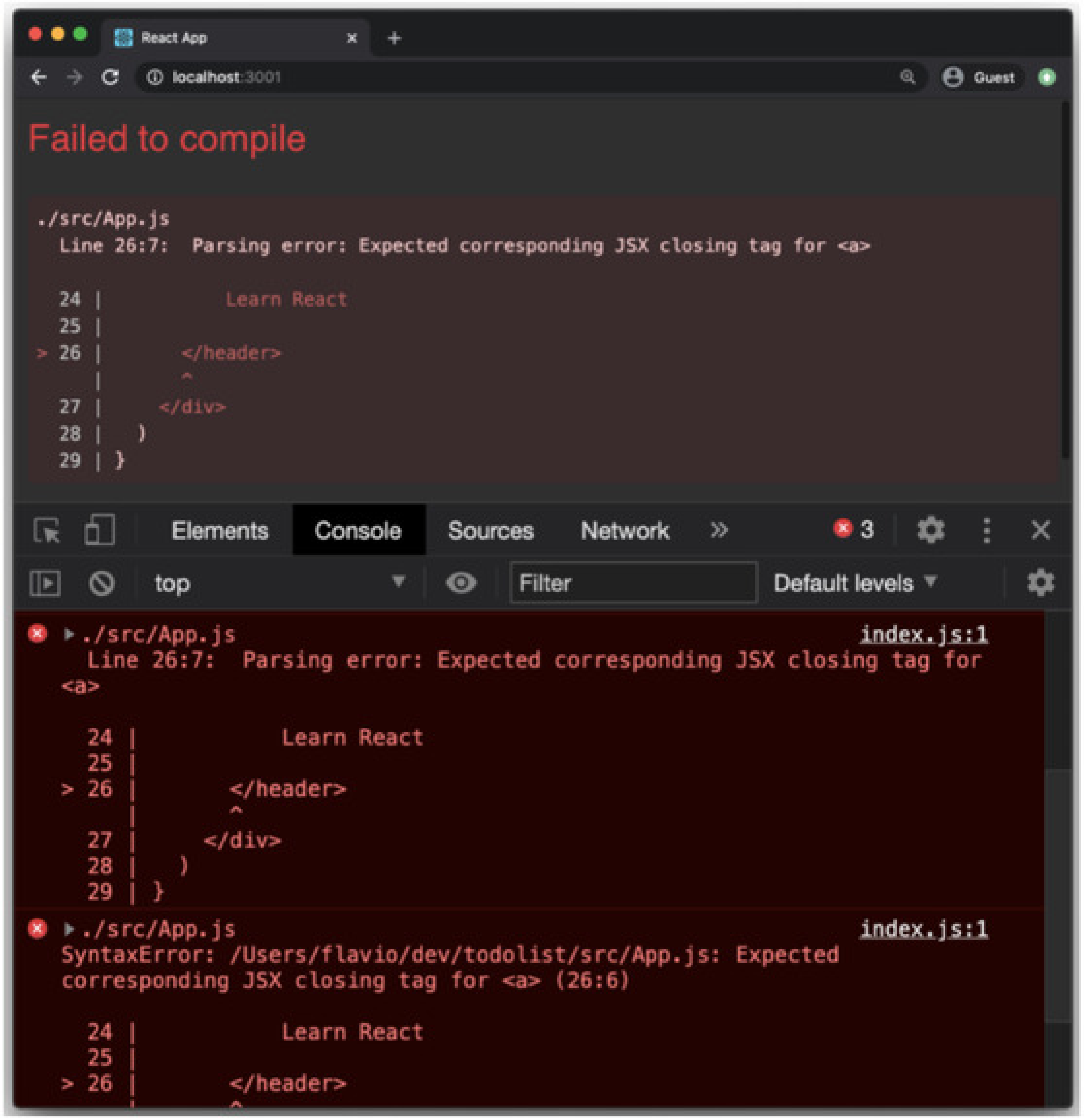
```
▶Warning: Invalid DOM property `class`. Did you mean `className`?
    in img (at App.js:13)
    in header (at App.js:12)
    in div (at App.js:11)
    in App (at src/index.js:9)
    in StrictMode (at src/index.js:8)
```

This is not the only HTML feature that suffers from this problem, but it's the most common one.

Another big difference between JSX and HTML is that HTML is very *relaxed*, we can say. Even if you have an error in the syntax, or you close the wrong tag, or you have a mismatch, the browser will try its best to interpret the HTML without breaking.

It's one of the core features of the Web. It is very forgiving.

JSX is not forgiving. If you forget to close a tag, you will have a clear error message:



Failed to compile

```
./src/App.js
  Line 26:7:  Parsing error: Expected corresponding JSX closing tag for <a>

  24 |           Learn React
  25 |
> 26 |         </header>
     |         ^
  27 |       </div>
  28 |     )
  29 | }
```

```
▸./src/App.js                                              index.js:1
  Line 26:7:  Parsing error: Expected corresponding JSX closing tag for
<a>

  24 |           Learn React
  25 |
> 26 |         </header>
     |         ^
  27 |       </div>
  28 |     )
  29 | }
▸./src/App.js                                              index.js:1
SyntaxError: /Users/flavio/dev/todolist/src/App.js: Expected
corresponding JSX closing tag for <a> (26:6)

  24 |           Learn React
  25 |
> 26 |         </header>
```

Embedding JavaScript in JSX

One of the best features of React is that we can easily embed JavaScript into JSX.

Other frontend frameworks, for example Angular and Vue, have their own specific ways to print JavaScript values in the template, or perform things like loops.

React doesn't add new things. Instead, it lets us use JavaScript in the JSX, by using curly brackets.

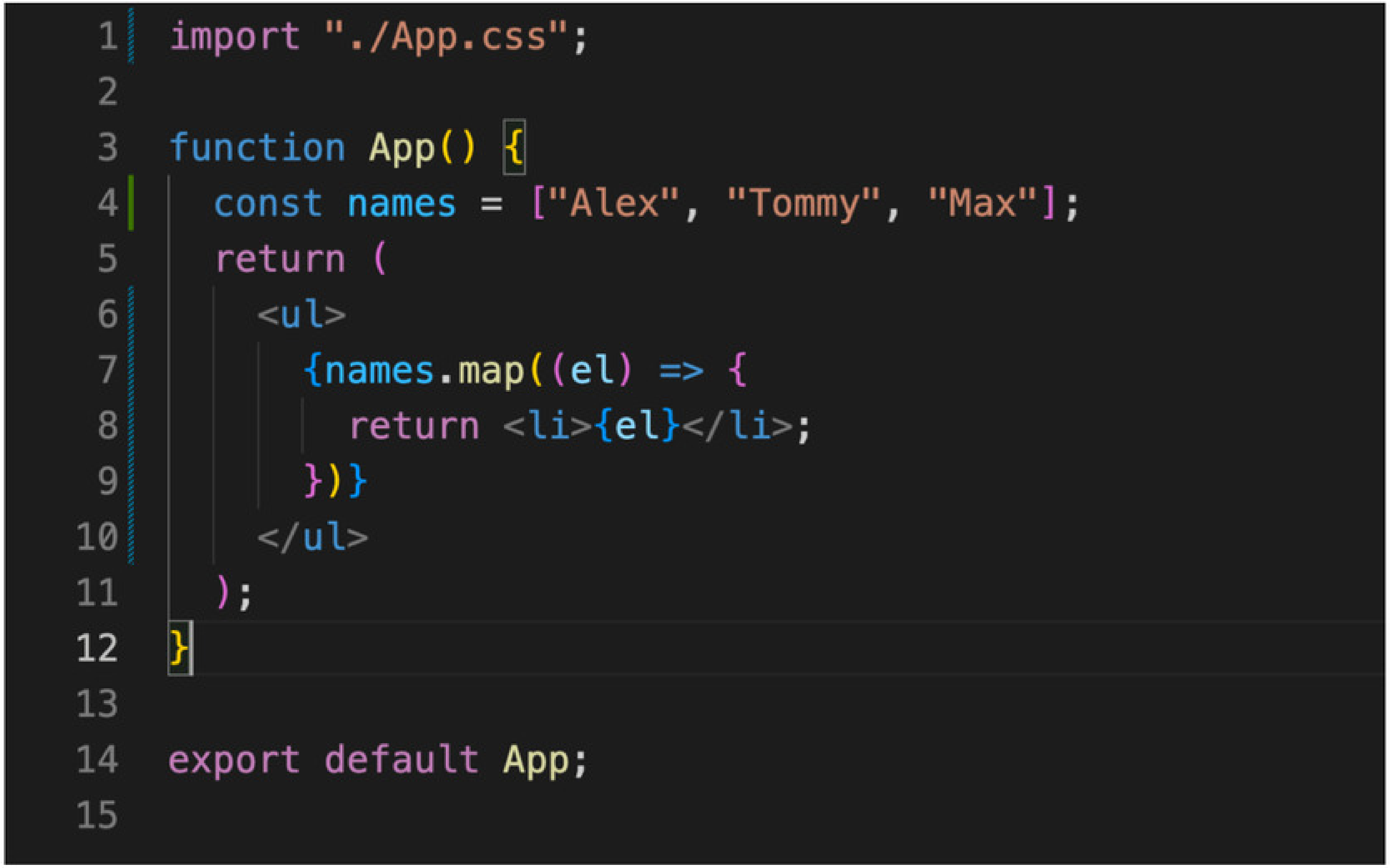
We import the `logo` SVG file using

```
import logo from './logo.svg'
```

and then in the JSX we assign this SVG file to the `src` attribute of an `img` tag:

```
<img src={logo} className="App-logo" alt="logo" />
```

Generating lists with Array map, the `key` property

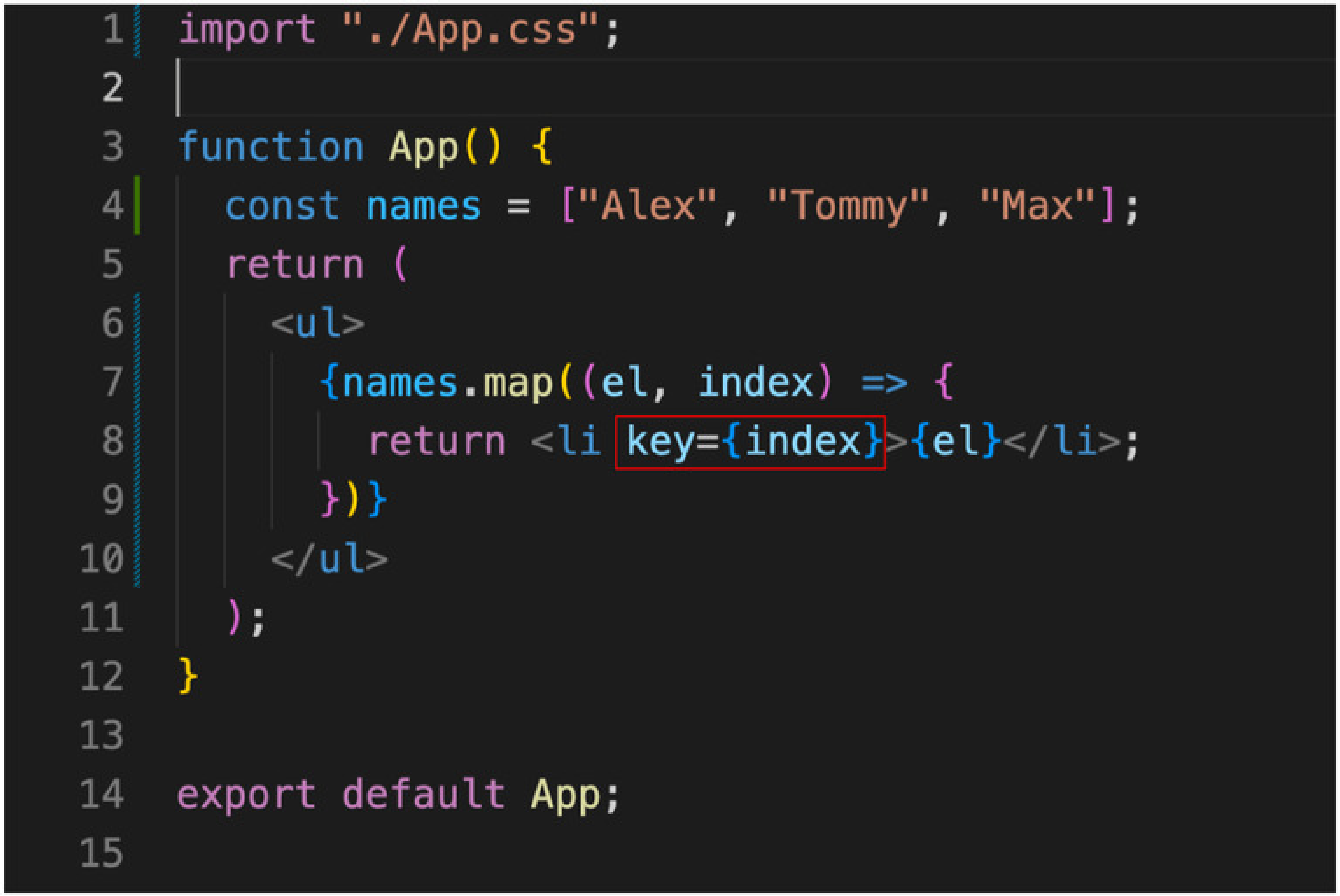


```
import "./App.css";

function App() {
  const names = ["Alex", "Tommy", "Max"];
  return (
    <ul>
      {names.map((el) => {
        return <li>{el}</li>;
      })}
    </ul>
  );
}

export default App;

```

When you run this code, you'll be given a warning that a key should be provided for list items. A "key" is a special string attribute you need to include when creating lists of elements. We'll discuss why it's important in the next section.

Keys

Keys help React identify which items have changed, are added, or are removed. Keys should be given to the elements inside the array to give the elements a stable identity:



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

function App() {
  const names = ["Alex", "Tommy", "Max"];
  return (
    <ul>
      {names.map((el, index) => {
        return <li key={index}>{el}</li>;
      })}
    </ul>
  );
}

export default App;
```

Time To Climb

Time To Climb

Time to practice