Otaku Tube

REACT JS NOTES (DAY 1)

Present by Sewak Dhakal

TOPICS FOR DAY 1

- What is React and JSx?
- Set Up local Environment
- Understanding our app
- Rewrite App
- Import - Export
- Our first component
- Markups in Jsx
- React Fragment
- Jsx with curly braces
- Structuring Components
- Style Components

INTRODUCTION

1. WHAT IS REACT?

- It's a library of JS for designing UI.

2. WHAT IS JSX?

- It's JavaScript Extension Syntax, that lets us write HTML directly inside JS.
- JSX syntax is not directly converted in JS, Babel transforms JSX into Js.

SETUP LOCAL ENVIRONMENT

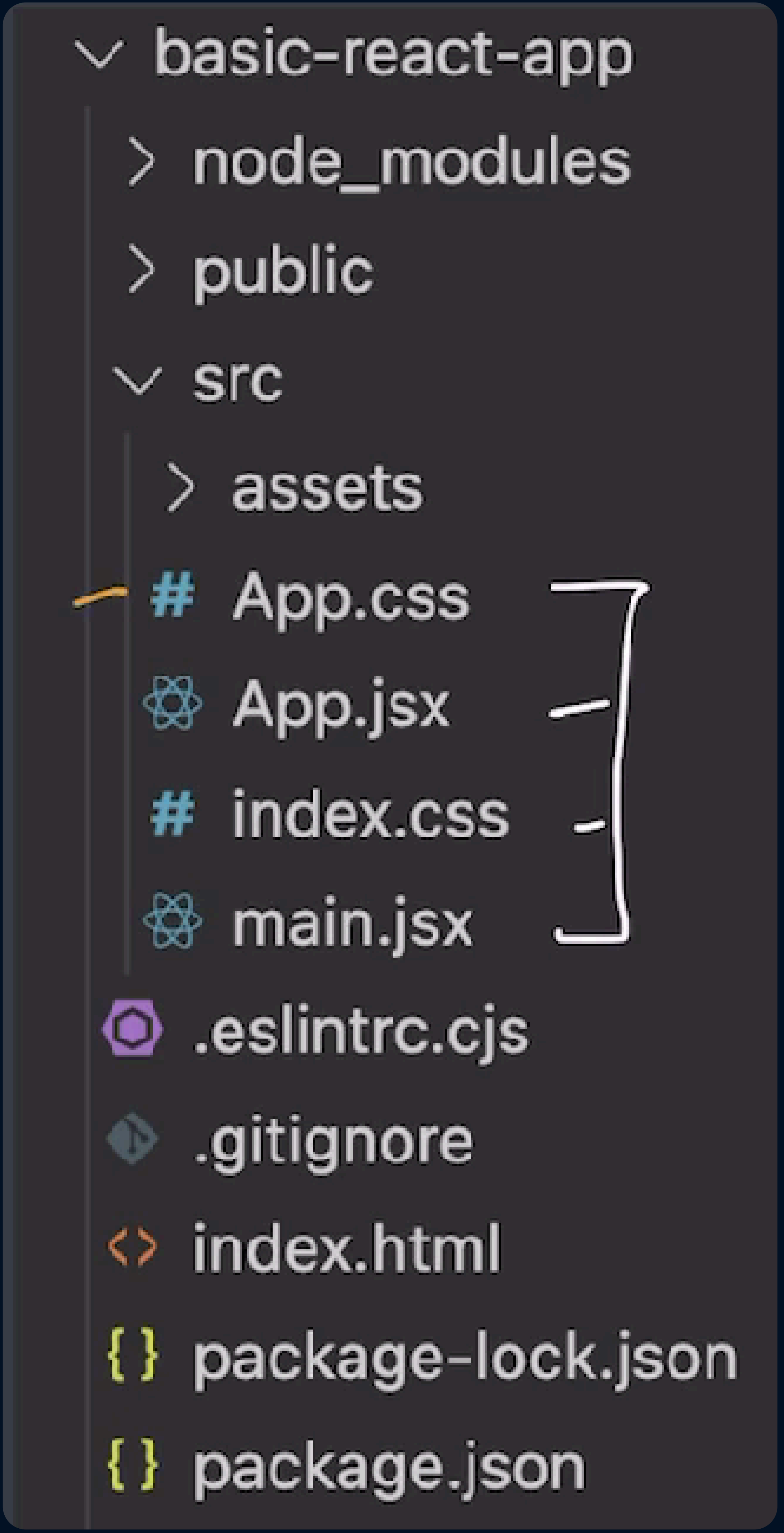
- Install Node.js
- Install Node.js
- Create a React project with `npm create vite@latest`
- Choose 'React' + JavaScript
- Run the app using `npm install` then `npm run dev`.

UNDERSTANDING OUR APP

🛠️ After setting up React + Vite in VS Code, you'll see a file structure like the one in the image.

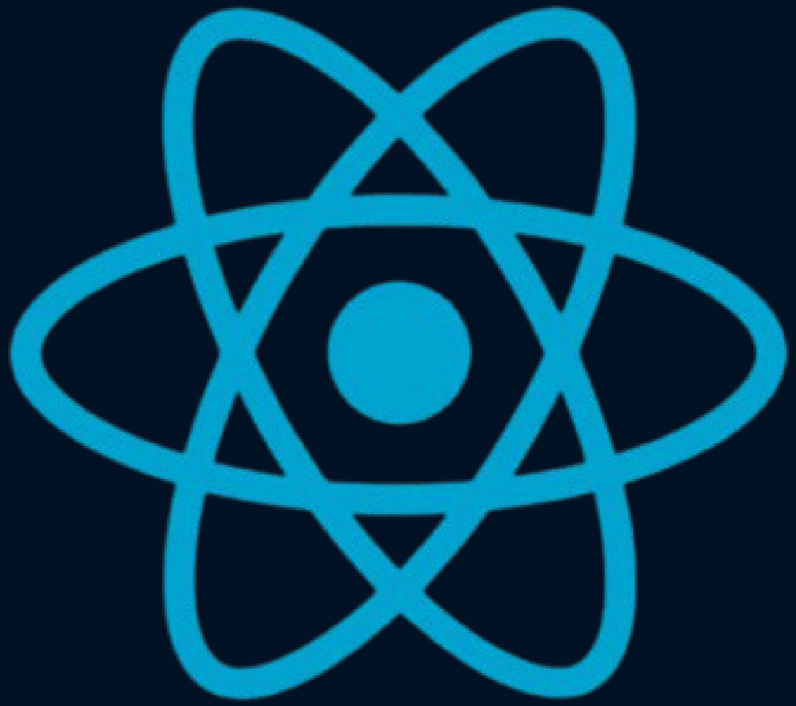
Your React app consists of components. The main component is App.jsx, which is rendered into the root div in index.html via main.jsx. Each component returns JSX that describes what appears on the screen.

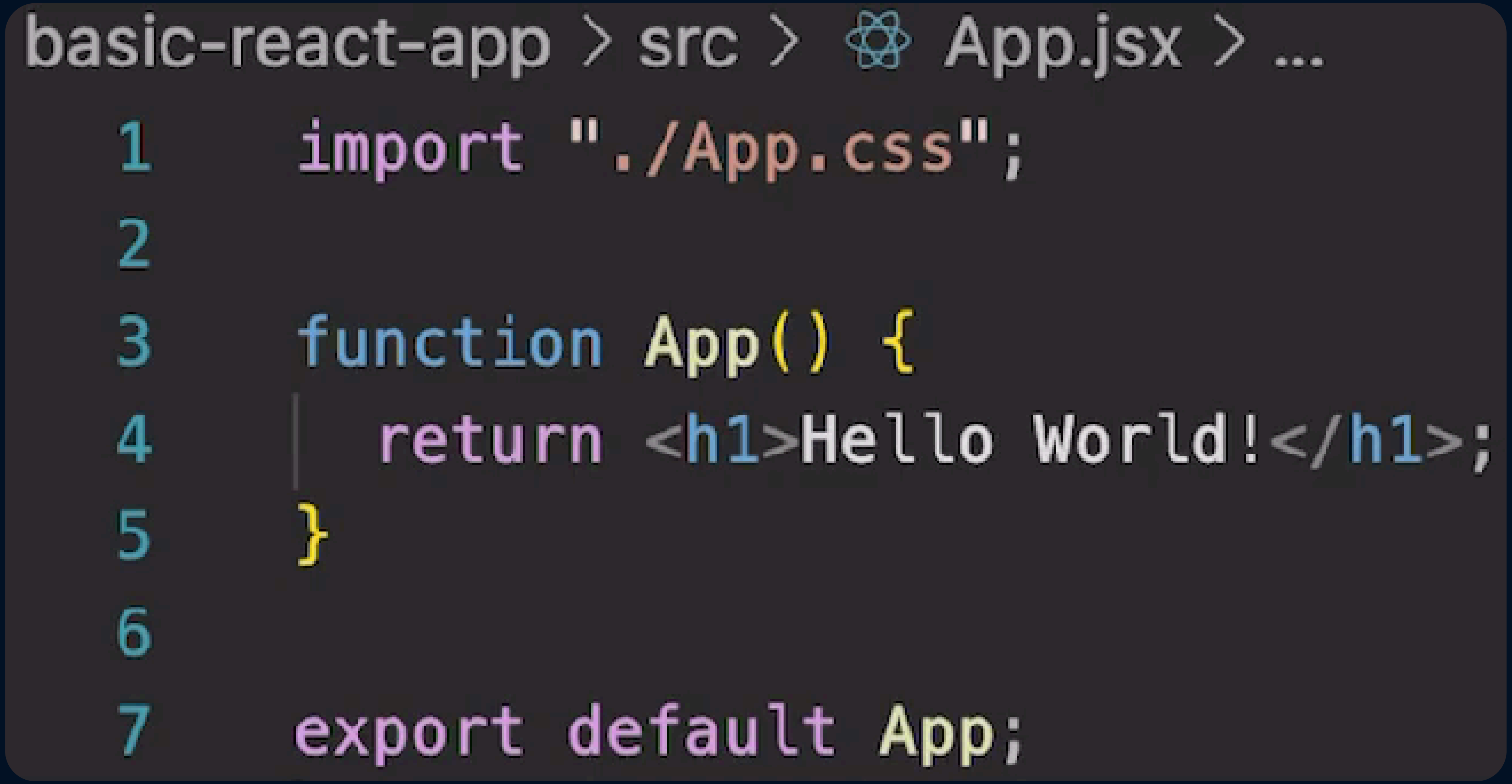
App.css – Contains the styling (CSS) for the App component.
App.jsx – Main component that defines the UI structure of your app.
index.css – Global styles that apply to the entire React app.
main.jsx – Entry point that renders the App component into the DOM.



RE-WRITE APP

You can re-write the default App component to show your own content. Replace the JSX inside App.jsx's return statement to create your custom UI.





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

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

export default App;
```

IMPORT - EXPORT

- Use `export default ComponentName` to export a component. Use `import ComponentName from './ComponentName'` to bring it into another file.

Import

```
import Title from "./Title.jsx";
```

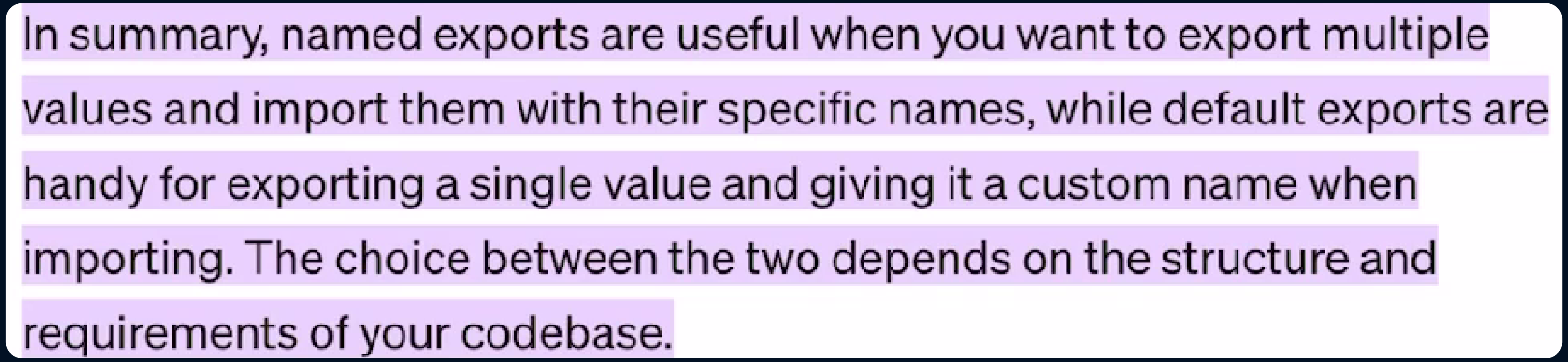
Default Export

```
export default Title;
```

Named Export

```
export { Title };
```

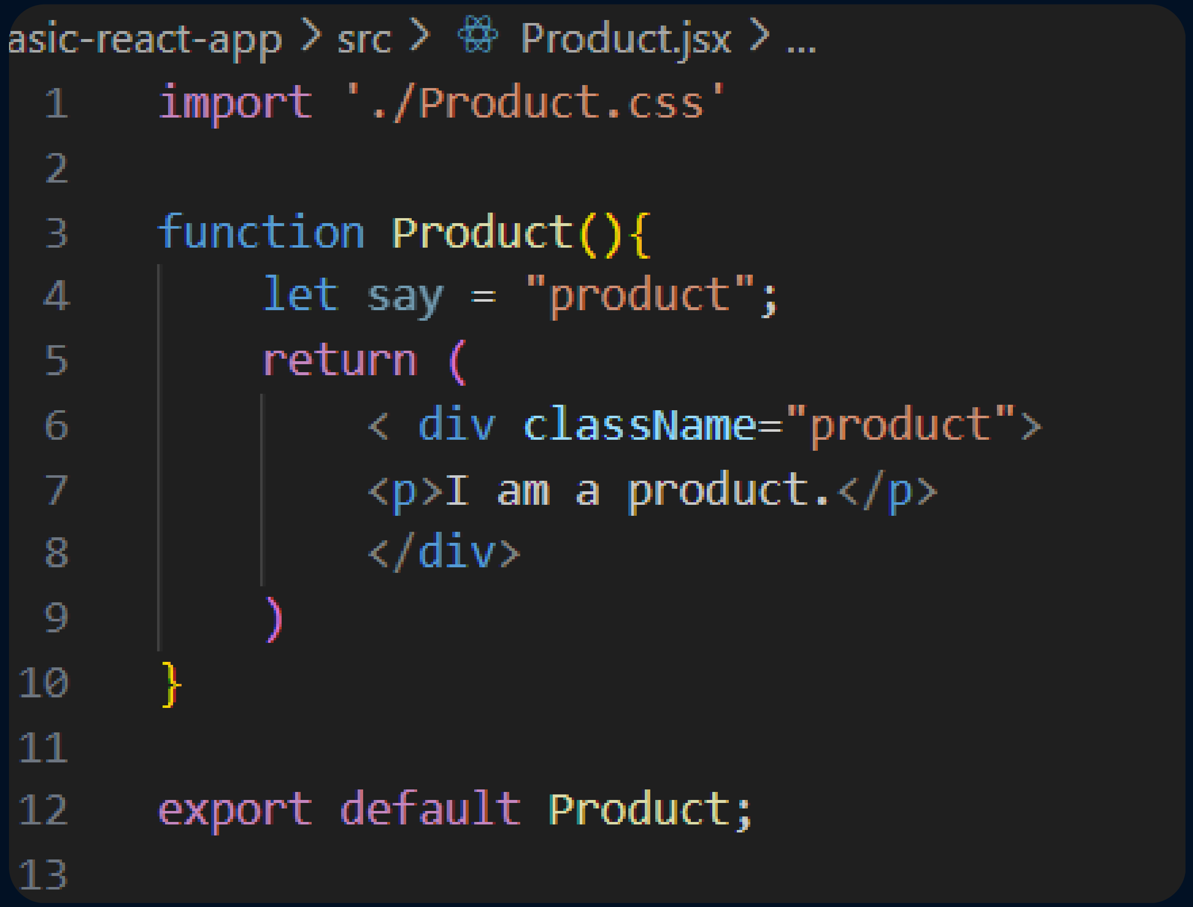
```
import Title from "./Title.jsx";
```



In summary, named exports are useful when you want to export multiple values and import them with their specific names, while default exports are handy for exporting a single value and giving it a custom name when importing. The choice between the two depends on the structure and requirements of your codebase.

OUR FIRST COMPONENT

- Component is a reusable and independent piece of code.
- Create a new file (e.g., Title.jsx) and define a function that returns JSX. Export it and import it in App.jsx to display it. Components must start with a capital letter.
- Creating a component:

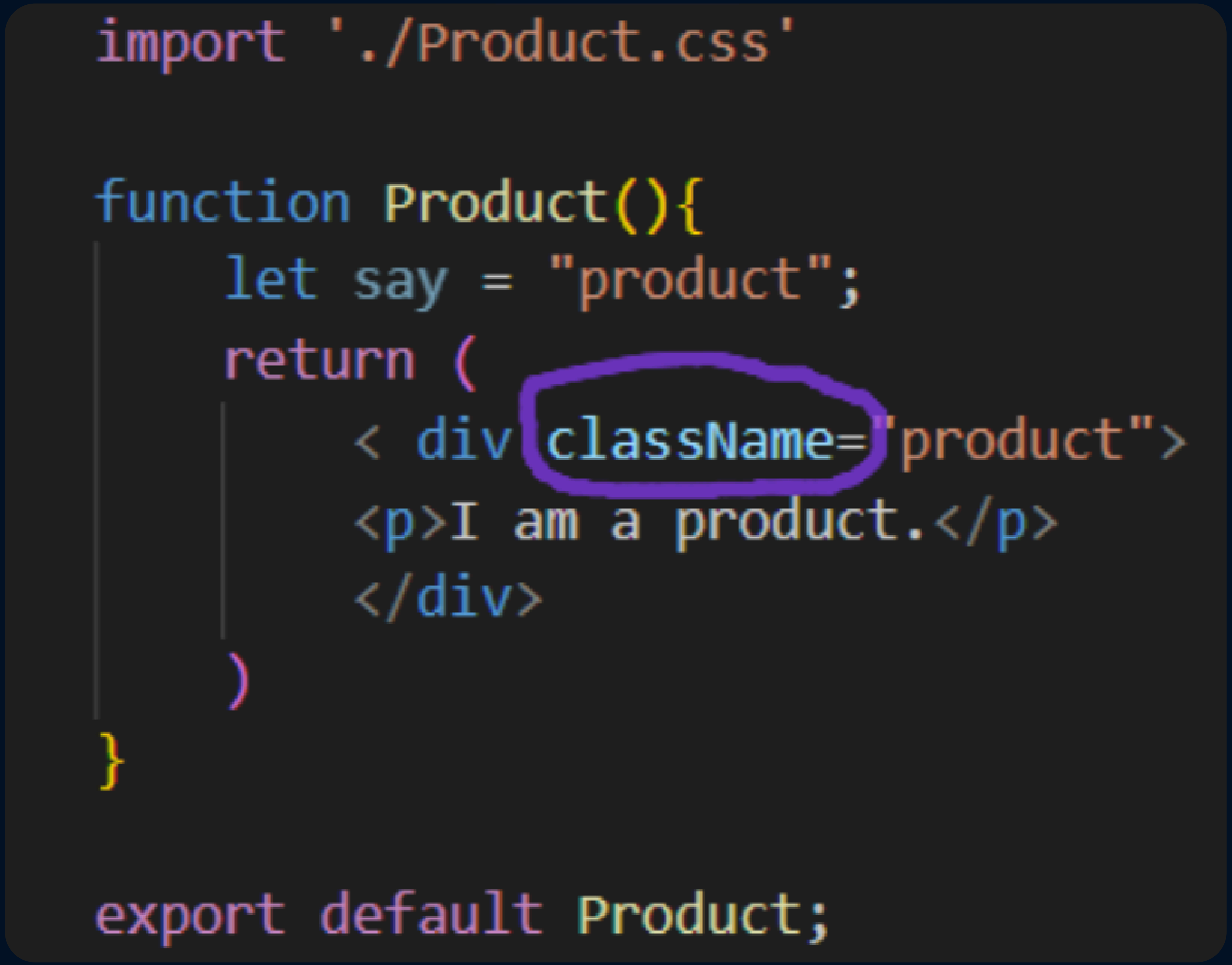

asic-react-app > src > Product.jsx > ...

```jsx
import './Product.css'

function Product(){
    let say = "product";
    return (
        < div className="product">
        <p>I am a product.</p>
        </div>
    )
}

export default Product;

```

WRITING MARKUPS IN JSX

Rules to be followed:

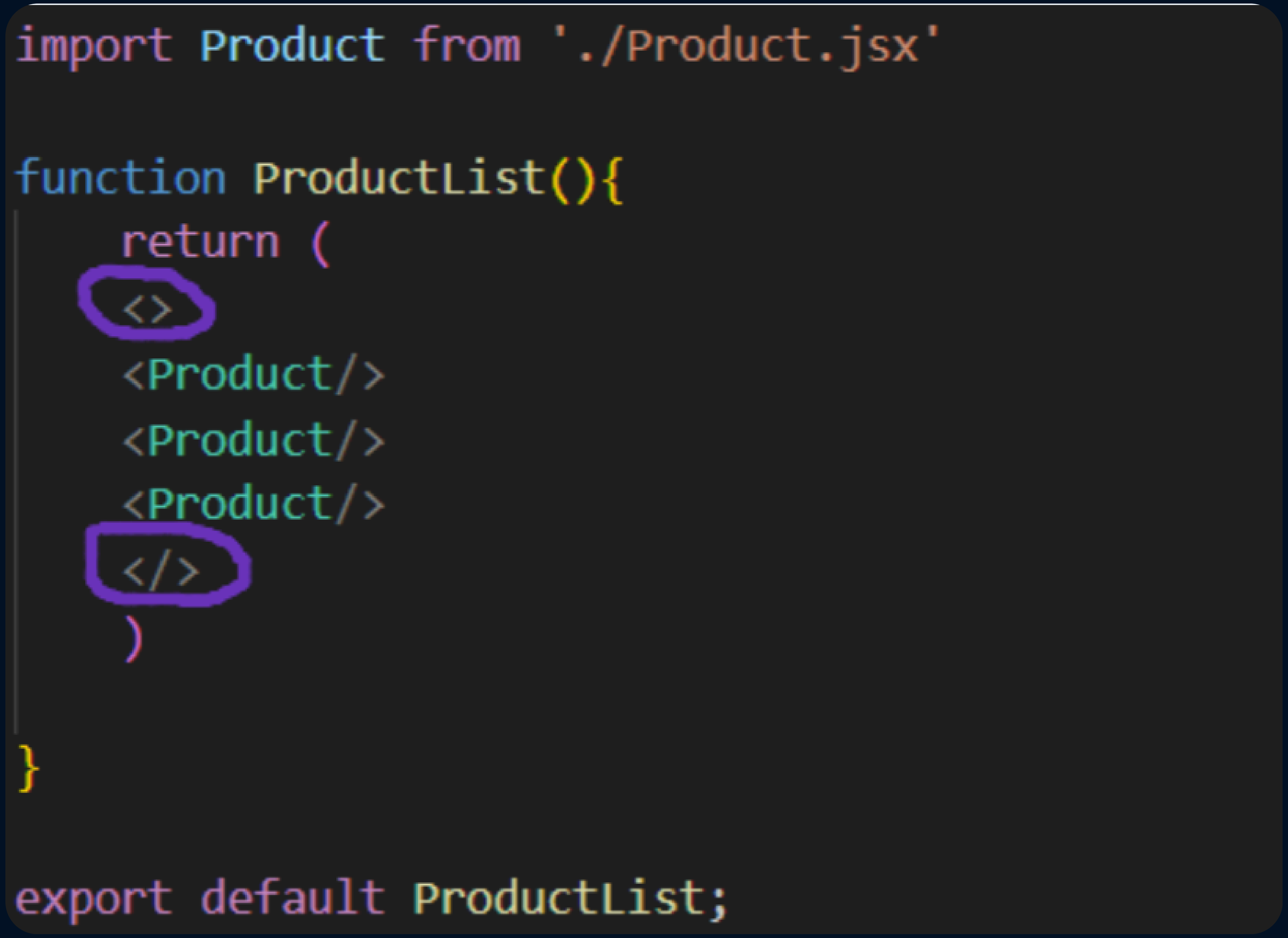
- **Return a single root element.**
- **Close all tags.**
- **camelCase most of the things.**



```
import './Product.css'

function Product(){
    let say = "product";
    return (
        < div className="product">
        <p>I am a product.</p>
        </div>
    )
}

export default Product;
```

REACT FRAGMENT

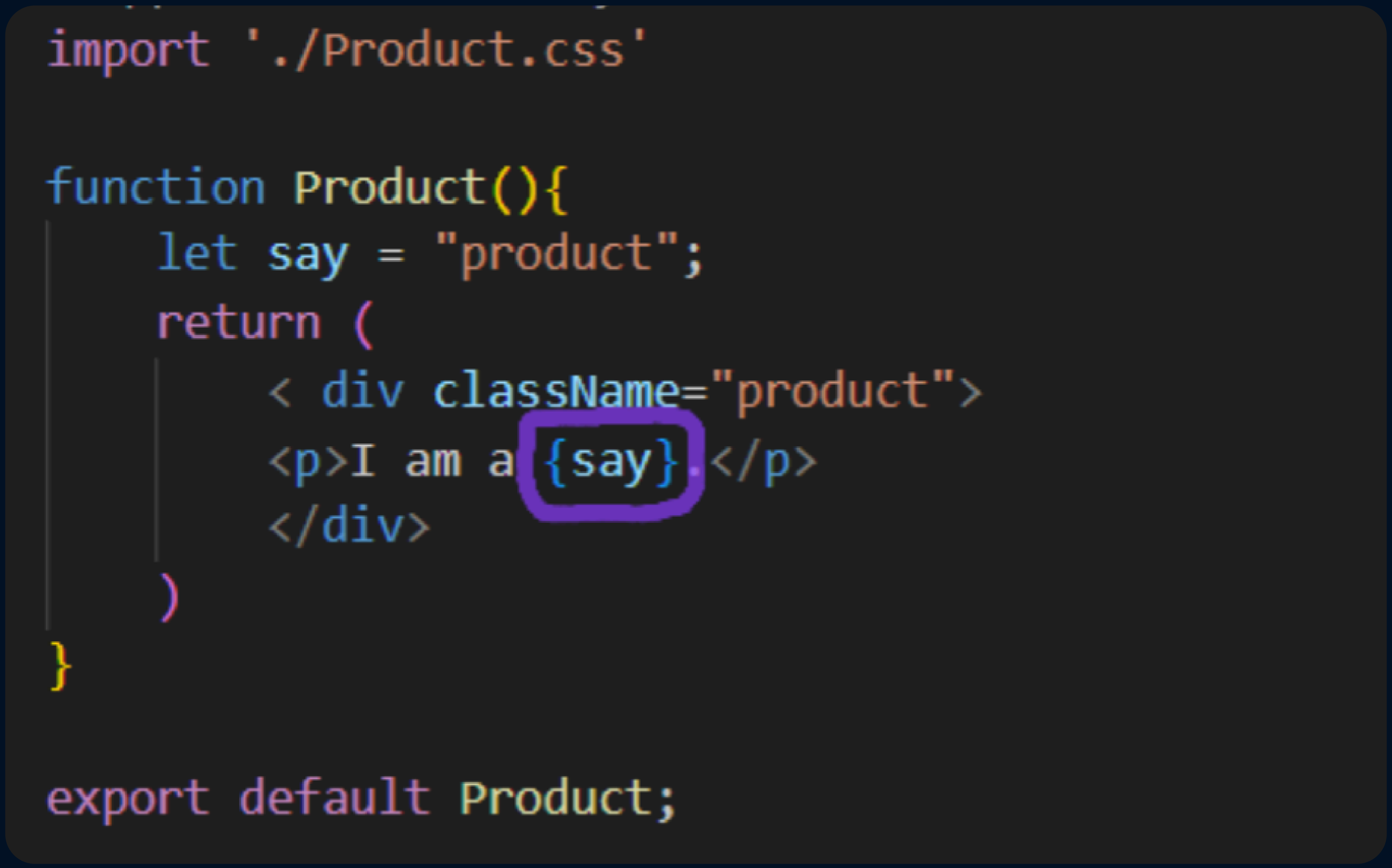
Fragments let you group elements without adding extra nodes to the DOM. Use `<>` or `` to wrap elements inside a component



```
import Product from './Product.jsx'

function ProductList(){
    return (
  <>
    <Product/>
    <Product/>
    <Product/>
  </>
    )

}

export default ProductList;
```

JSX WITH CURLY BRACES

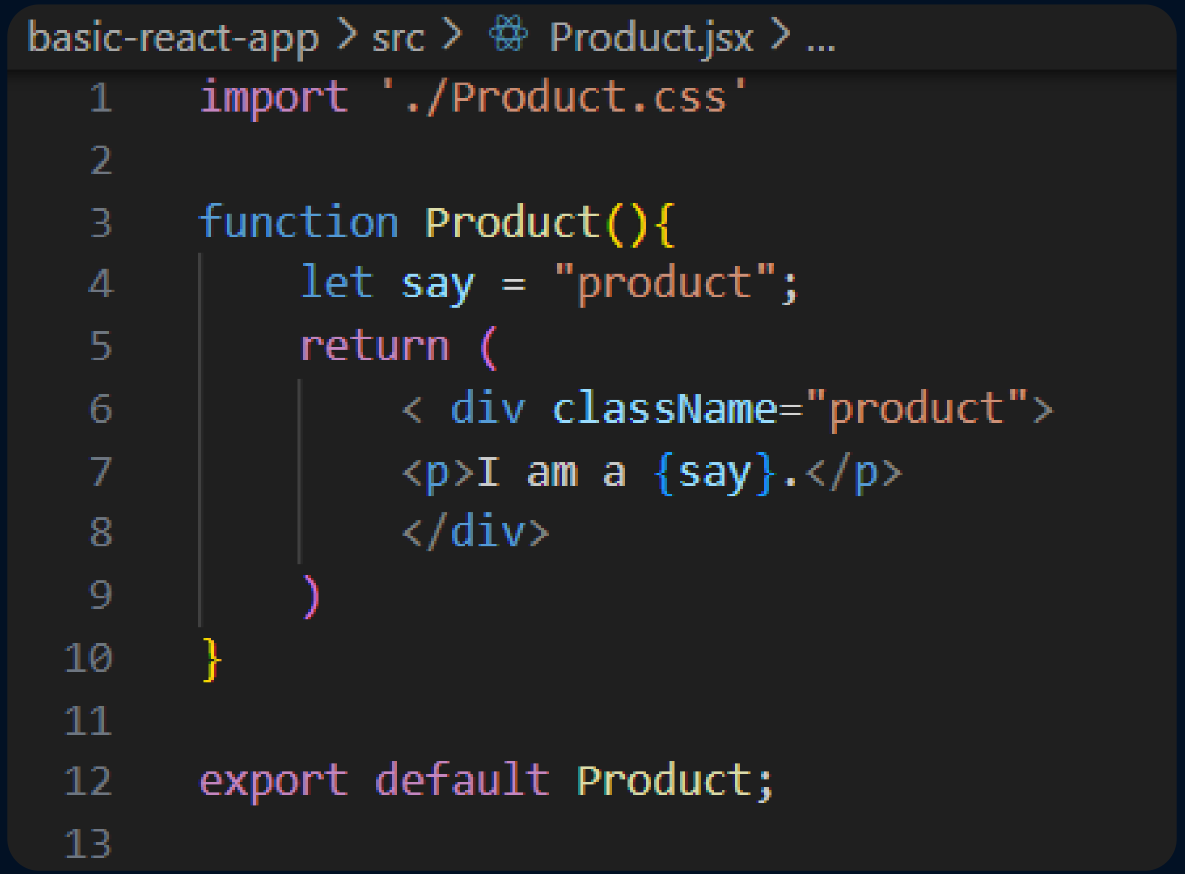
In JSX, you use `{}` to embed JavaScript expressions inside your markup, like `{5+5}` or `{props.title}`.



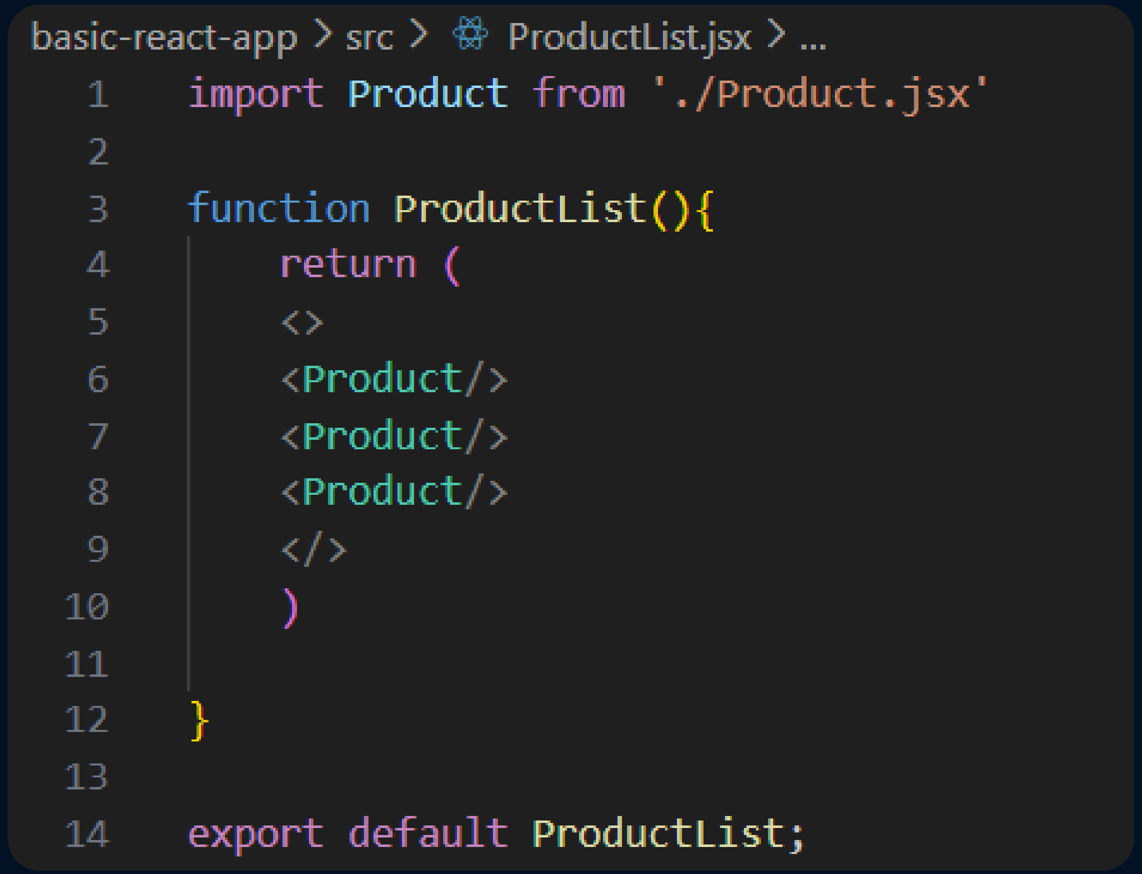
```
import './Product.css'

function Product(){
    let say = "product";
    return (
        < div className="product">
        <p>I am a {say}.</p>
        </div>
    )
}

export default Product;
```

STRUCTURING COMPONENTS

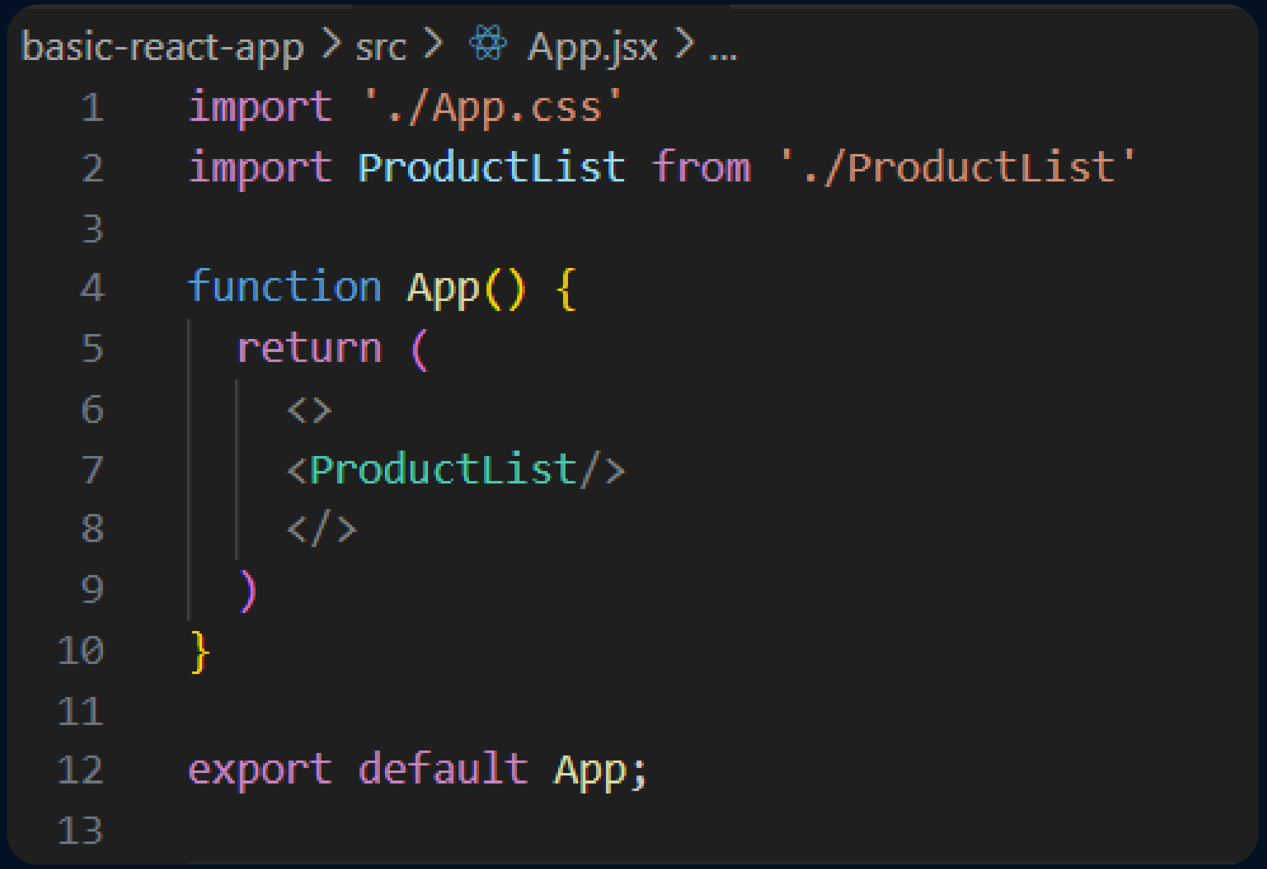
Organize your project by placing components in a `/components` folder. Keep each component in its own file, and use folders to group related components.


basic-react-app > src > Product.jsx > ...

```jsx
import './Product.css'

function Product(){
    let say = "product";
    return (
        < div className="product">
        <p>I am a {say}.</p>
        </div>
    )
}

export default Product;
```


basic-react-app > src > ProductList.jsx > ...

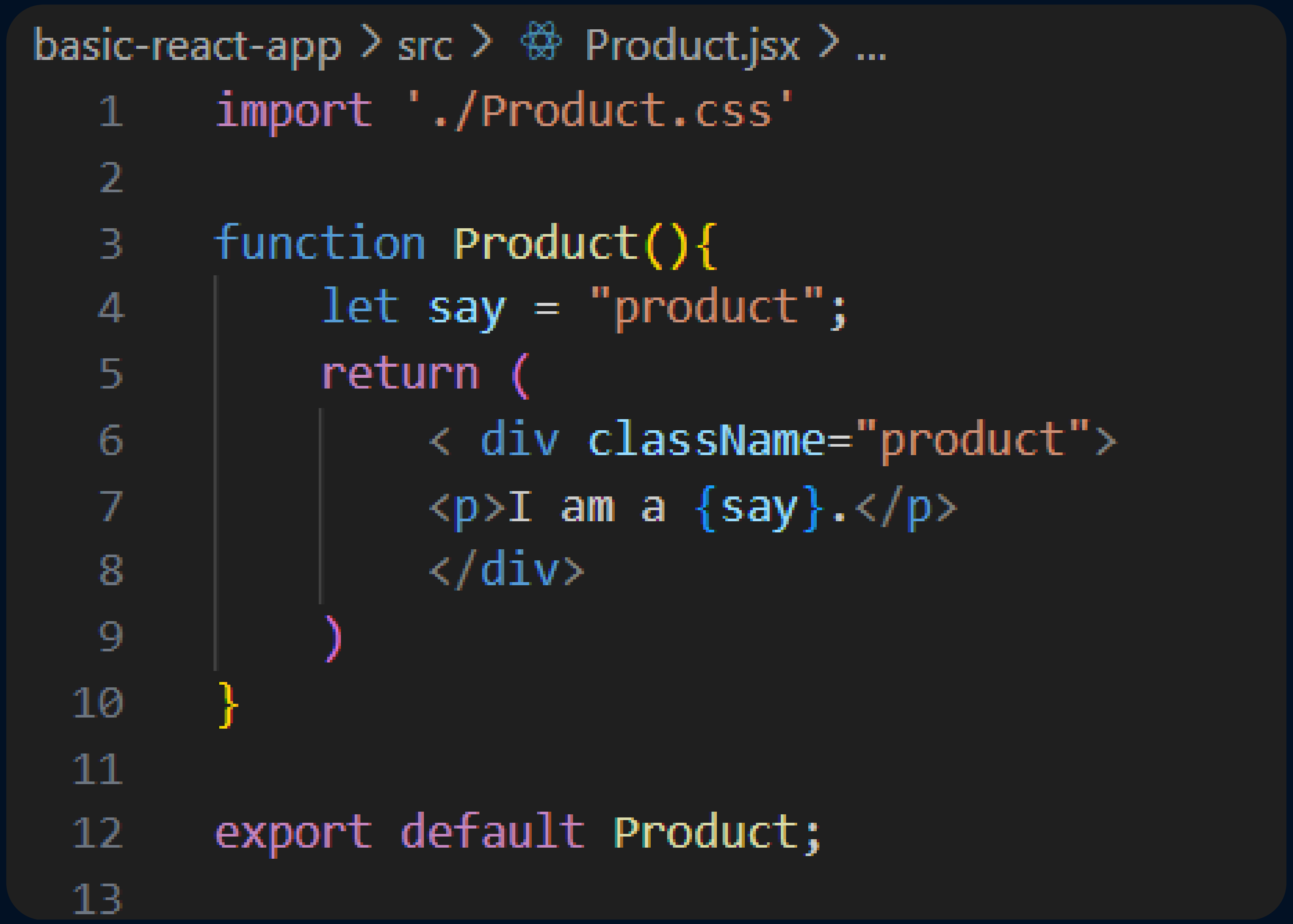
```jsx
import Product from './Product.jsx'

function ProductList(){
    return (
    <>
    <Product/>
    <Product/>
    <Product/>
    </>
    )

}

export default ProductList;
```


basic-react-app > src > App.jsx > ...

```jsx
import './App.css'
import ProductList from './ProductList'

function App() {
  return (
    <>
    <ProductList/>
    </>
  )
}

export default App;
```

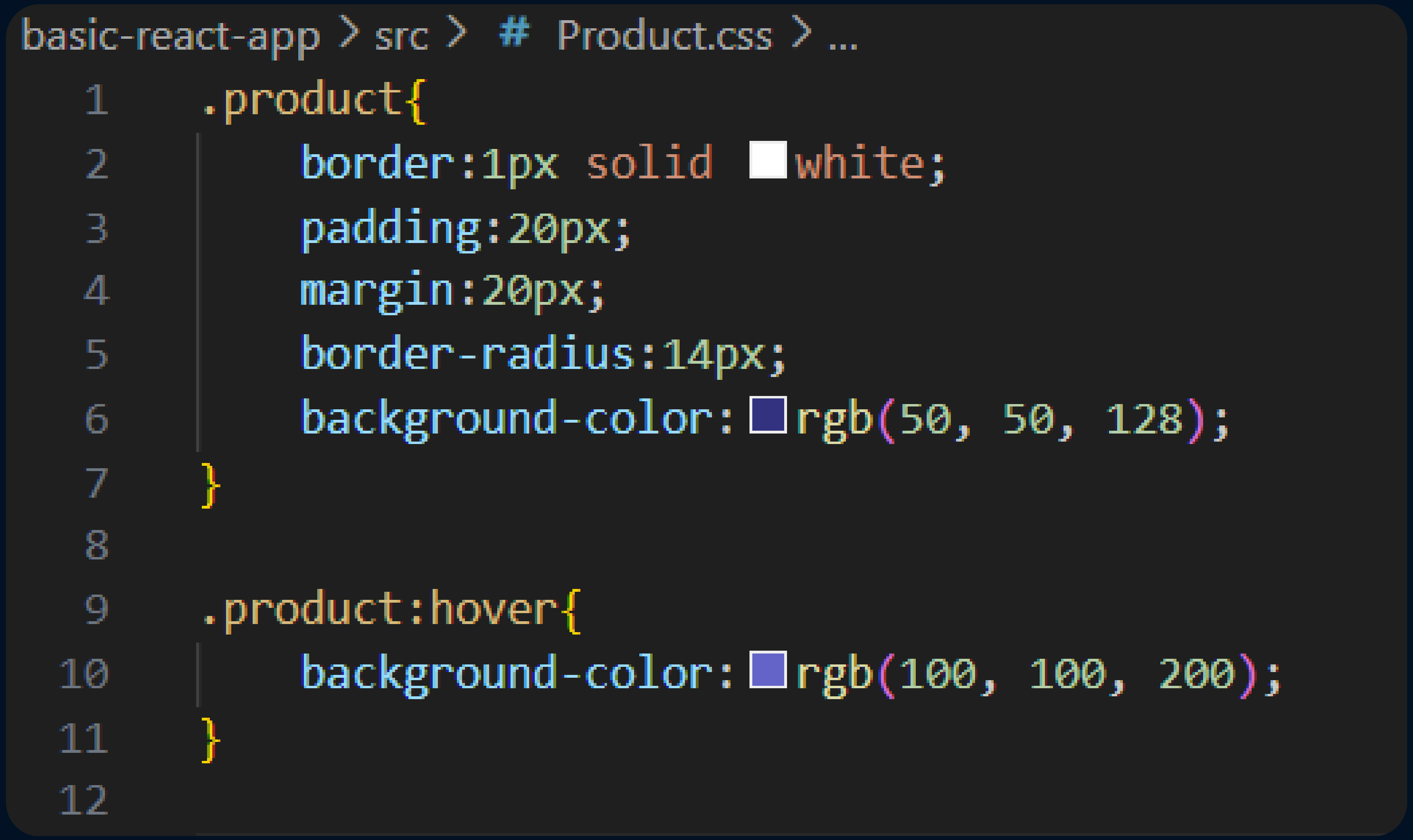
STYLE COMPONENTS

You can style components using CSS files (imported into the component), inline styles, or CSS-in-JS libraries like styled-components.
Example: `div style={{ color: 'red' }}>Hello</div>`.



basic-react-app > src > Product.jsx > ...

```jsx
import './Product.css'

function Product(){
    let say = "product";
    return (
        < div className="product">
        <p>I am a {say}.</p>
        </div>
    )
}

export default Product;

```



basic-react-app > src > # Product.css > ...

```css
.product{
    border:1px solid ⬜white;
    padding:20px;
    margin:20px;
    border-radius:14px;
    background-color:▨rgb(50, 50, 128);
}

.product:hover{
    background-color:▨rgb(100, 100, 200);
}

```