

● 리액트의 장점

- SAP(Single Page Application)

기존의 웹 페이지 방식은 MPA 방식으로 각 페이지 별로 HTML 파일이 따로 존재하며, 페이지를 이동하게 될 경우 브라우저에서는 해당 페이지의 HTML 파일을 받아와서 화면에 표시해 주었음.

그렇다면 수많은 페이지가 존재하는 복잡한 사이트의 경우 HTML 파일이 수십에서 수백 개가 될 텐데 어떻게 다 관리할까?

리액트는 하나의 HTML 파일의 <body> 태그의 내부가 텅 비어있다가 해당 페이지에 접속할 때, 그 페이지에 해당하는 콘텐츠를 가져와서 동적으로 <body> 내부를 채워 넣게 됨

- 리액트 홈페이지

A JavaScript library for building user interfaces.

사용자 인터페이스를 만들기 위한 자바스크립트 라이브러리

- 장점

1. 빠른 업데이트와 렌더링 속도

예를 들어 메뉴를 클릭시 새로운 내용을 보여주어야 하는데 리액트에서는 이런 빠른 업데이트를 위해 내부적으로 Virtual DOM이라는 것을 사용함.

DOM은 Document Object Model의 약자로 웹 페이지를 정의하는 하나의 객체
쉽게 표현하면 하나의 웹 사이트에 대한 정보를 모두 담고 있는 큰 그릇이라고 봄..

Virtual DOM은 웹페이지와 실제 DOM 사이에서 중간 매개체 역할을 함

2. 컴포넌트 기반

리액트에서는 모든 페이지가 컴포넌트로 구성되어 있음

3. 재사용성

4. 모바일 앱 개발 가능

리액트 네이티브(React Native)라는 모바일 환경 UI 프레임 워크를 사용하여
모바일 앱도 개발할 수 있음.

● 리액트의 단점

- 방대한 학습량

모든 라이브러리가 그렇지만 특히 리액트의 경우 기존과는 다른 방식의 UI 라이브러리이기 때문에 학습해야 할 것이 많음.

- 리액트 같은 경우에는 계속해서 버전 업데이트가 이루어지고 있어 새로운 내용들이 꾸준히 등장하므로 새로운 버전에 대한 내용을 학습해야 함.

- 높은 상태 관리 복잡도

리액트에서는 state라는 굉장히 중요한 개념이 있는데, 쉽게 말해 리액트 컴포넌트의 상태를 의미함.

성능 최적화를 위해 state를 잘 관리하는 것이 중요함.

(큰 규모의 프로젝트에서는 Redux, MobX, Recoil 등의 외부 상태 관리 라이브러리를 사용하는 경우가 많음)

● JSX란?

JavaScript와 XML/HTML을 합친 것

JSX의 X는 extension의 X를 의미함

예) `const element = <h1> Hello, World! </h1>`

● JSX의 역할

JSX는 내부적으로 XML/HTML 코드를 자바스크립트로 변환하는 과정을 거치게 됨.

그렇기 때문에 실제로 JSX로 코드를 작성해도 최종적으로는 자바스크립트 코드가 나오게 됨

JSX 코드를 자바스크립트 코드로 변환하는 역할을 하는 것이 바로 리액트의 `createElement()` 라는 함수 임.

● JSX의 장점

- 코드가 간결하고 가독성이 향상됨

● JSX의 장점

- 코드가 간결하고 가독성이 향상됨

JSX를 사용

```
<div> Hello, {name} </div>
```

JSX를 사용 안 함

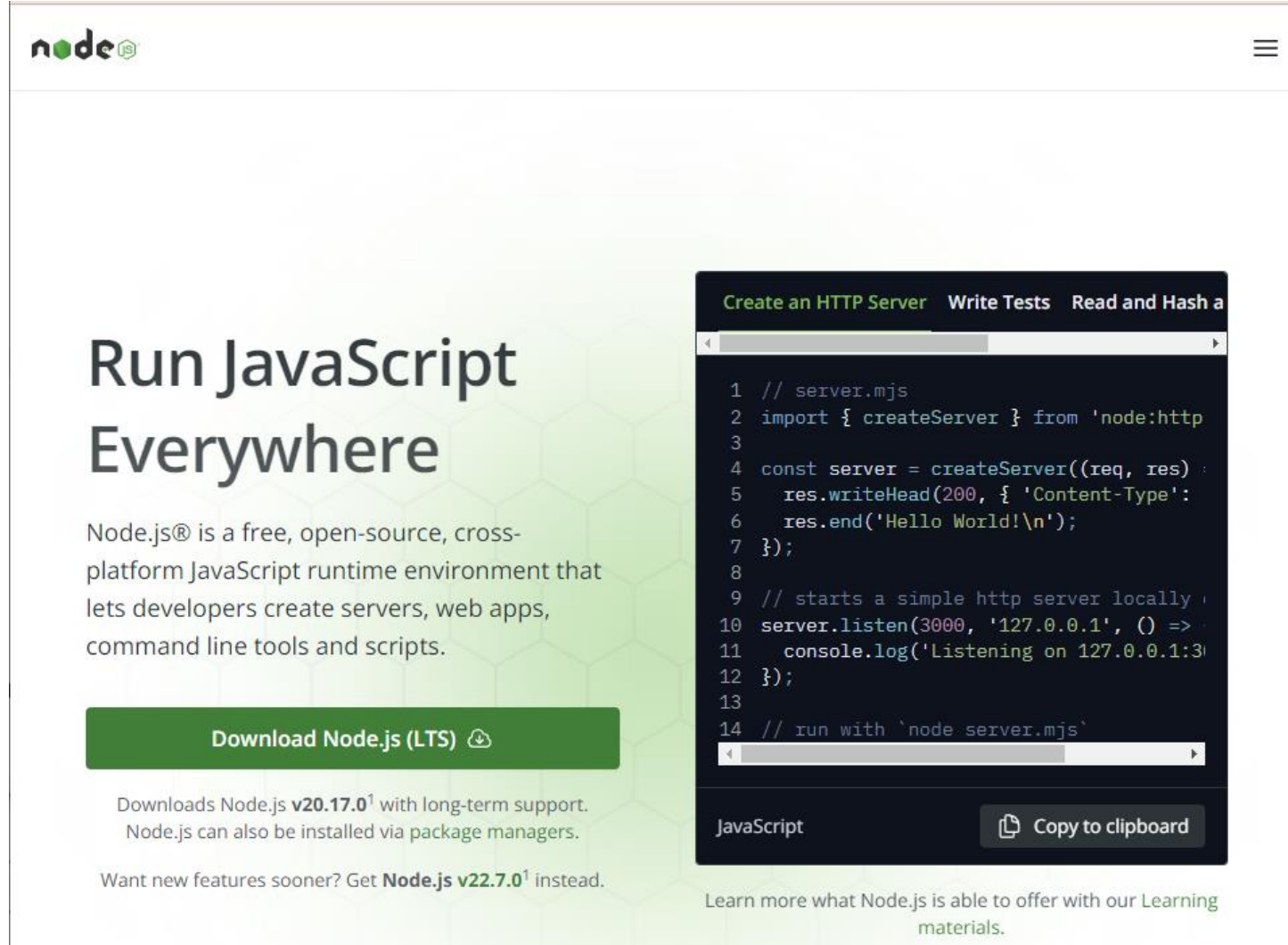
```
React.createElement('div', null, 'Hello, ${name}');
```

작업 환경 설정

- 리액트 프로젝트를 만들 때는 Node.js를 반드시 먼저 설치해야 함
- Node.js는 크롬 V8 자바 스크립트 엔진으로 빌드한 자바스크립트 런타임
- 이것은 웹 브라우저 환경이 아닌 곳에서도 자바스크립트를 사용하여 연산할 수 있음
- 리액트 애플리케이션은 웹 브라우저에서 실행되는 코드이므로 Node.js와 직접적인 연관은 없지만 프로젝트를 개발하는 데 필요한 주요한 도구들이 Node.js를 사용하기 때문에 설치하는 것임
- Node.js를 설치하면 Node.js 패키지 매니저 도구인 npm이 설치됨
- npm으로 수많은 개발자가 만든 패키지(재사용 가능한 코드)를 설치하고 설치한 패키지의 버전을 관리할 수 있음
- 리액트 역시 하나의 패키지임

- node.js 설치

1. <https://nodejs.org/en> 공식 사이트에서 내려 받아 설치



The screenshot shows the Node.js website homepage. The main heading is "Run JavaScript Everywhere". Below it, a paragraph describes Node.js as a free, open-source, cross-platform JavaScript runtime environment. A green button labeled "Download Node.js (LTS)" is prominent. Below the button, text indicates that the download includes Node.js v20.17.0 with long-term support. A code editor window is open on the right, displaying a JavaScript snippet for creating an HTTP server. The code includes comments and uses the `http` module. A "Copy to clipboard" button is visible at the bottom of the code editor.

Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

Download Node.js (LTS)

Downloads Node.js **v20.17.0**¹ with long-term support.
Node.js can also be installed via [package managers](#).

Want new features sooner? Get **Node.js v22.7.0**¹ instead.

```
1 // server.mjs
2 import { createServer } from 'node:http'
3
4 const server = createServer((req, res) :
5   res.writeHead(200, { 'Content-Type':
6     res.end('Hello World!\n');
7 });
8
9 // starts a simple http server locally
10 server.listen(3000, '127.0.0.1', () =>
11   console.log('Listening on 127.0.0.1:3000')
12 });
13
14 // run with `node server.mjs`
```

JavaScript [Copy to clipboard](#)

Learn more what Node.js is able to offer with our [Learning materials](#).

- 설치 확인

node -v

C:\> 선택 명령 프롬프트

```
Microsoft Windows [Version 10.0.19045.4780]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\WUser>node -v  
v20.12.2
```

```
C:\Users\WUser>
```

- yarn 설치

Node.js를 설치할 때, 패키지를 관리해 주는 npm이라는 도구가 설치됨

yarn은 npm을 대체할 수 있는 도구로 npm 보다 빠르며 효율적인 캐시 시스템과 기타 부가 기능을 제공함

```
$ npm install -global yarn
```

```
C:\Users\User>npm install -global yarn
```

```
added 1 package in 6s
```

```
npm notice
```

```
npm notice New minor version of npm available! 10.5.0 -> 10.8.2
```

```
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.2
```

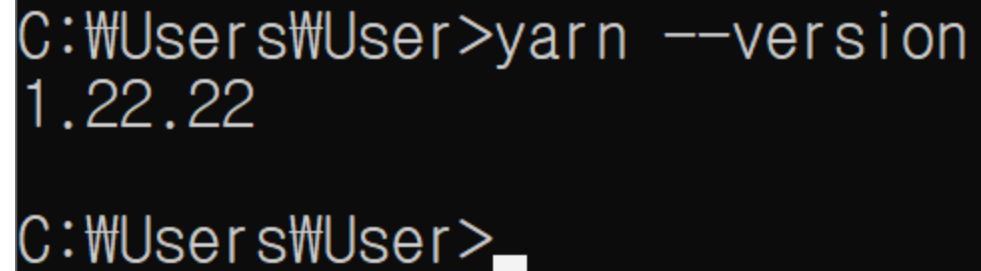
```
npm notice Run npm install -g npm@10.8.2 to update!
```

```
npm notice
```

```
C:\Users\User>
```

설치 확인

\$ yarn --version

A screenshot of a Windows command prompt window with a black background. The text is white. The first line shows the command 'C:\Users\WUser>yarn --version' and the second line shows the output '1.22.22'. The third line shows the prompt 'C:\Users\WUser>' with a white cursor.

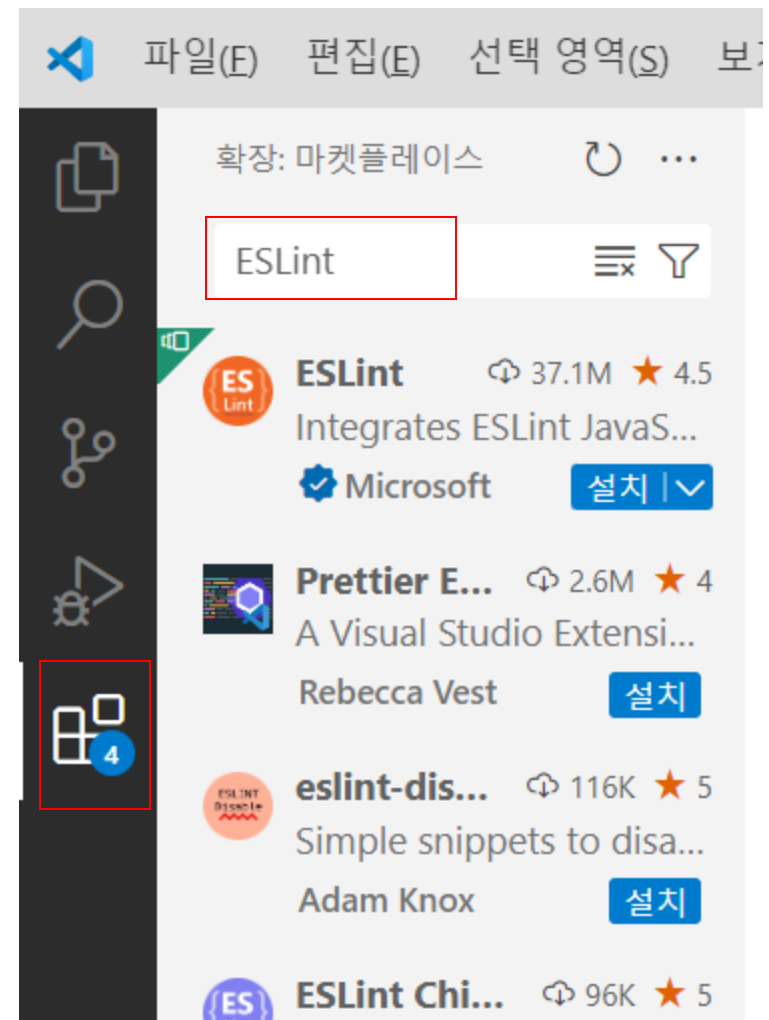
```
C:\Users\WUser>yarn --version
1.22.22
C:\Users\WUser>
```

에디터 설치 (Visual Studio Code)

<https://code.visualstudio.com/>

VS 확장 프로그램 설치

- ESLint : 자바스크립트 문법 및 코드 스타일을 검색해 주는 도구
- Reactjs Code Snippets : 리액트 컴포넌트 및 라이프사이클 함수를 작성할 때 단축 단어를 사용하여 간편하게 코드를 자동으로 생성해 낼 수 있는 코드 스니펫 모음
검색했을 때 유사한 결과가 여러 개 나올 수 있는데 제작자가 charalampos karypidis인 것을 설치
- Prettier-Code formatter : 코드 스타일을 자동으로 정리해 주는 도구



VS Code 언어 한국어로 설정하기

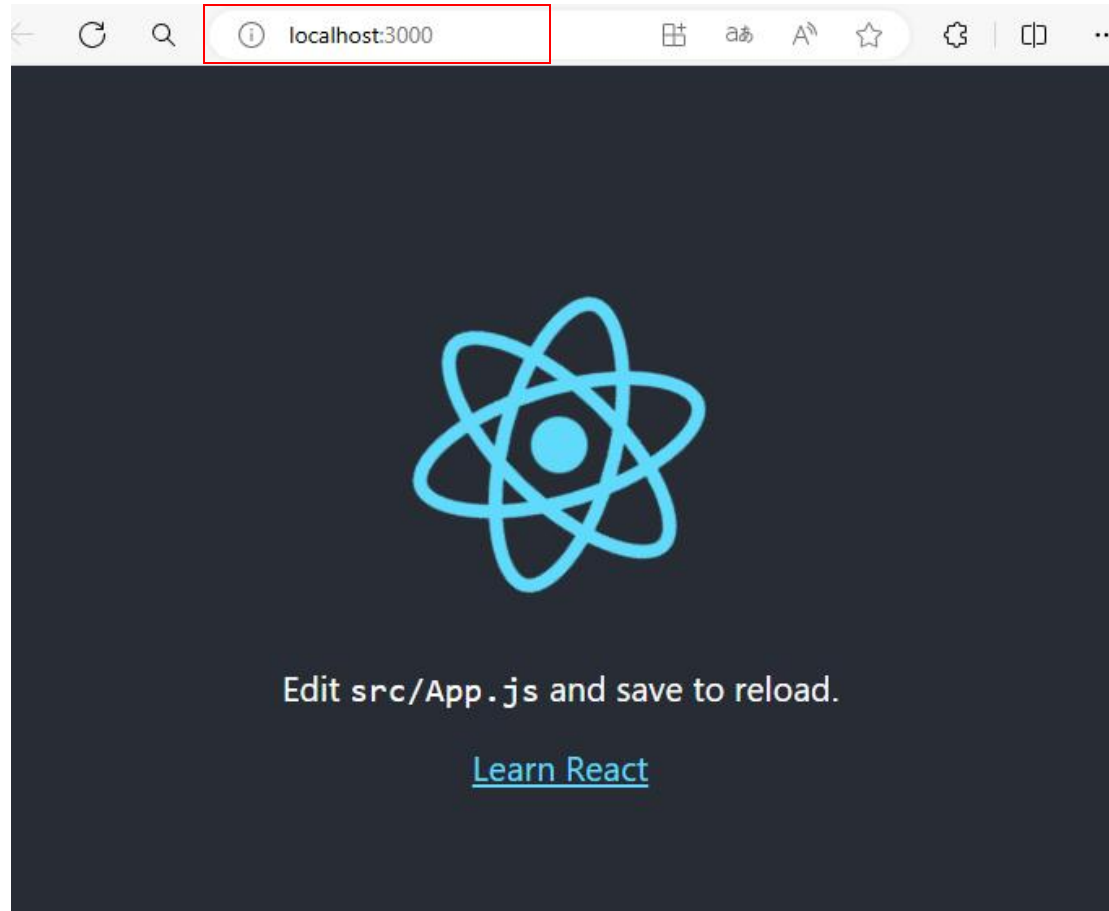
- 확장 마켓플레이스에서 Korean Language Pack for Visual Studio Code 설치
- 설치 후 VS Code에서 F1을 누른 후 "Configure Display Language"를 입력하고 나서 Enter를 누름
- 이렇게 하면 local.json 이라는 파일이 열리는데, 여기서 ko로 설정한 뒤 재시작 함

- 먼저 react 폴더를 생성한 후 폴더 안에 들어가

\$ yarn create react-app hello-app(프로젝트 이름) 생성

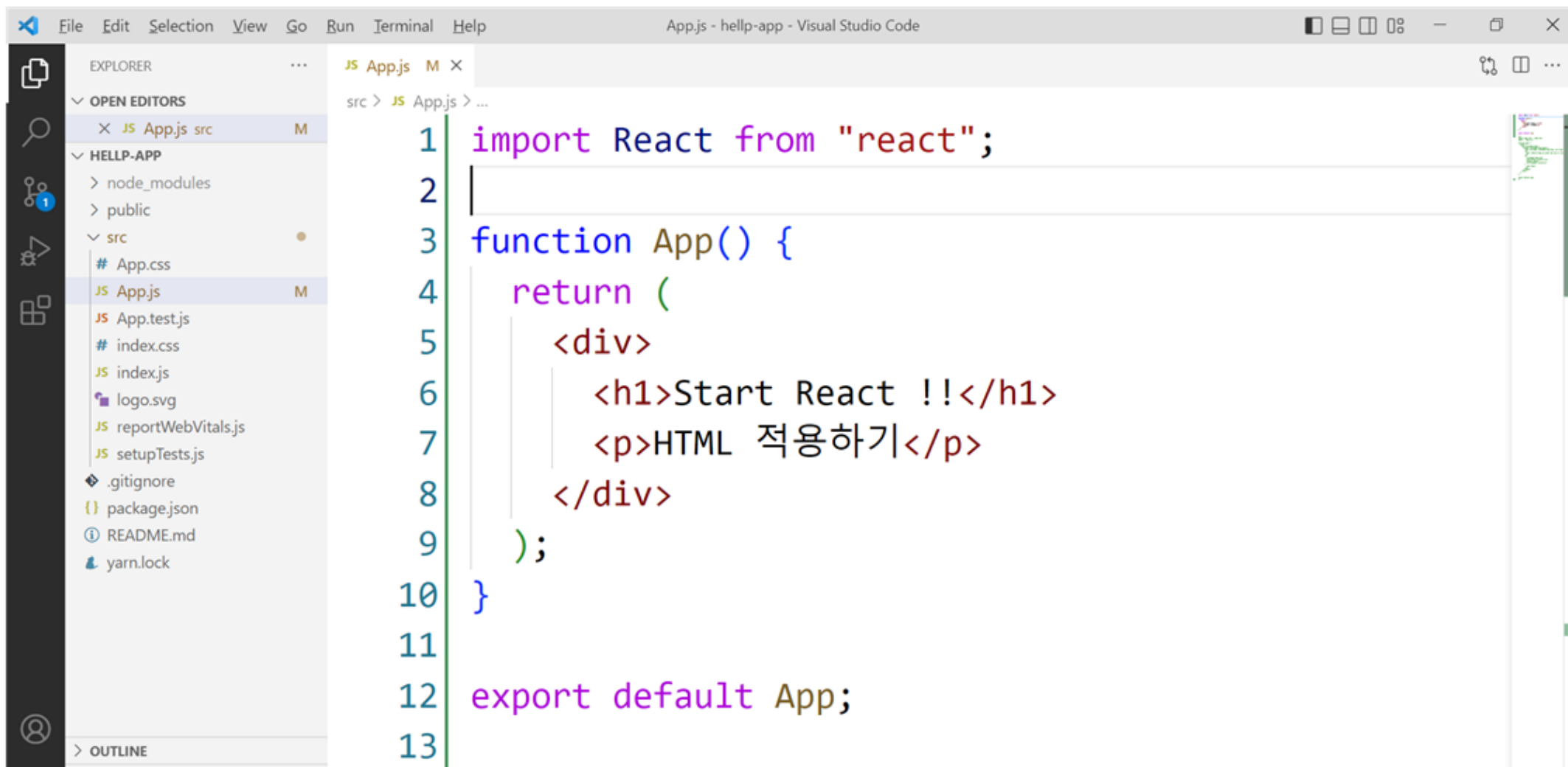
\$ cd hello-app

\$ yarn start

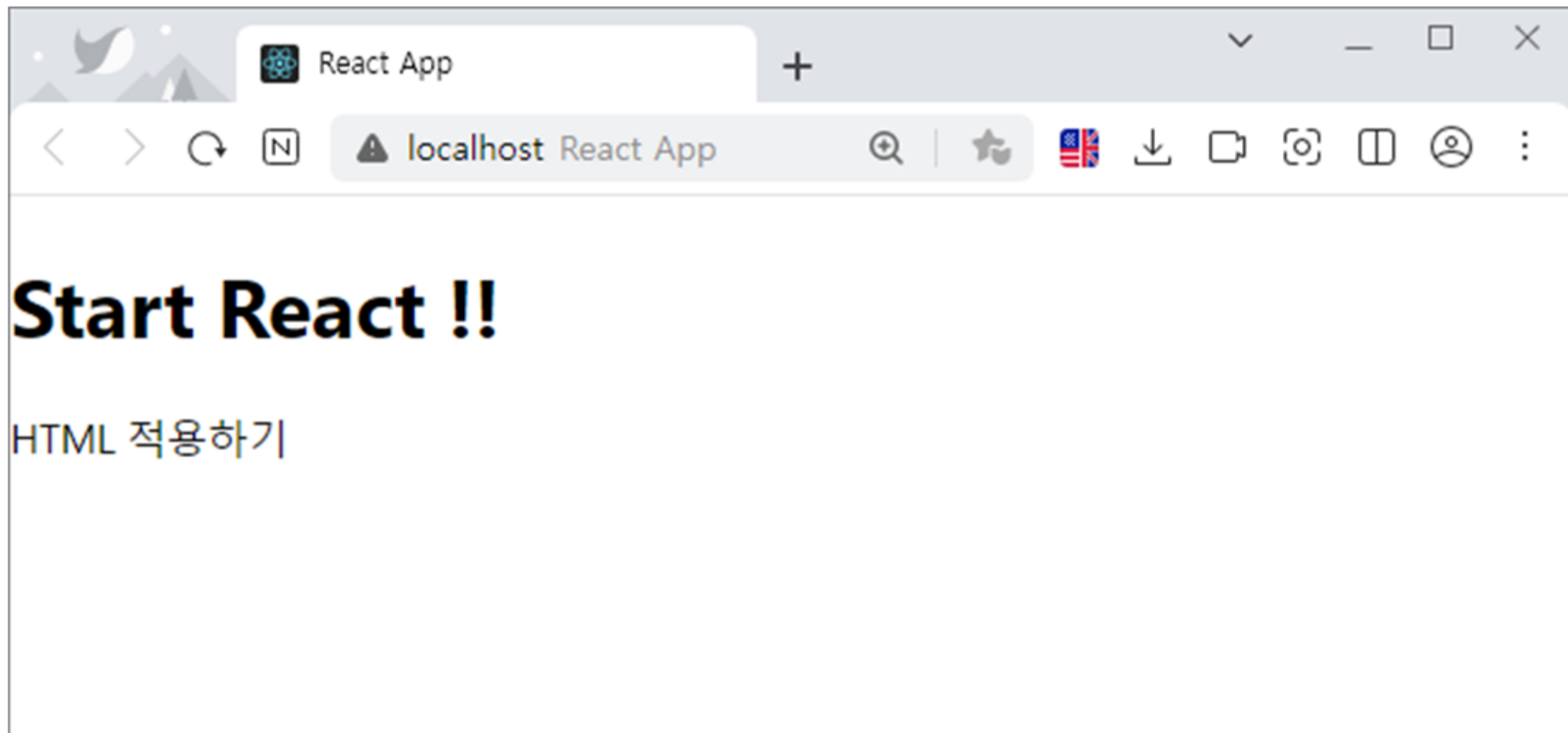


● JSX 코드 작성 예

- 홈 화면인 App.js 파일을 다음과 같이 수정



```
1 import React from "react";
2
3 function App() {
4   return (
5     <div>
6       <h1>Start React !!</h1>
7       <p>HTML 적용하기</p>
8     </div>
9   );
10 }
11
12 export default App;
13
```



● .jsx에 css 적용

```
1  div{
2    background-color: rgb(162,127,243);
3    color:rgb(255,2555,255);
4    padding:40px;
5    font-family: 고딕;
6    text-align: center;
7  }
8
9  h1{
10   color:white;
11   background-color: #2EFE2E;
12   padding: 10px;
13   font-family: 궁서;
14 }
```

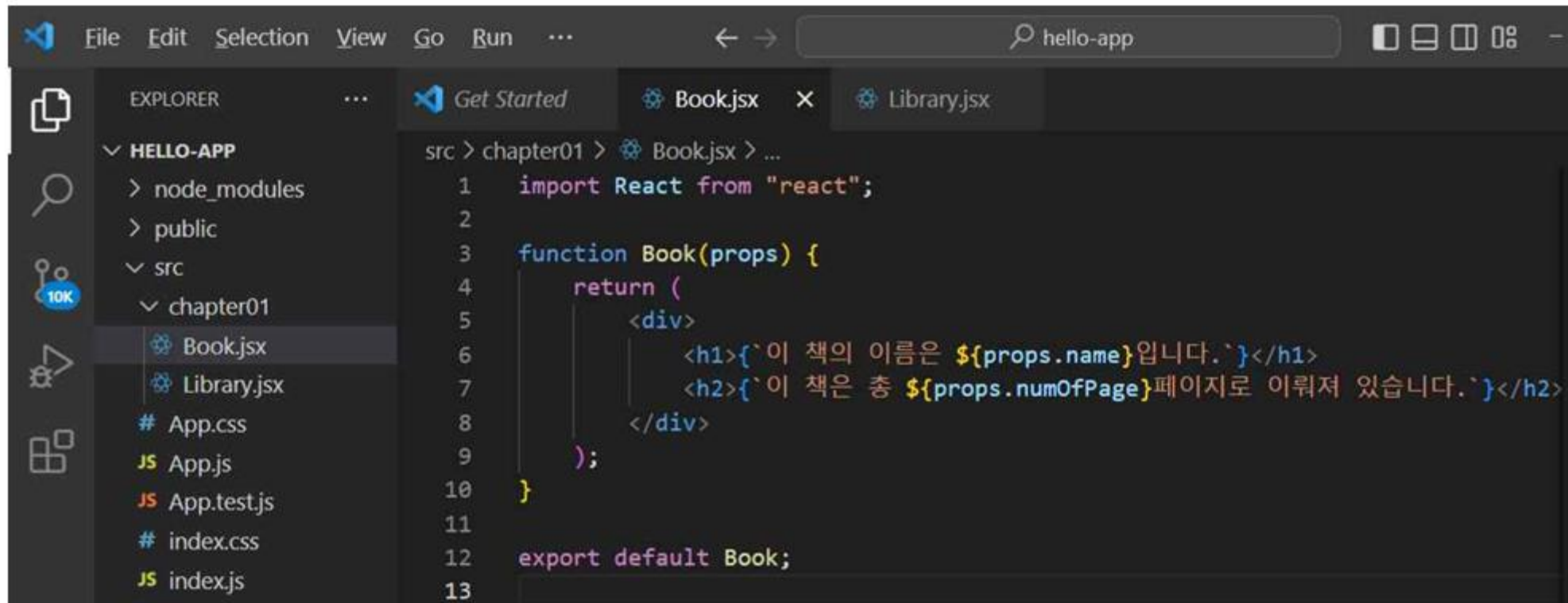
* App.js에 다음 코드 추가



```
1 import React from "react";  
2 import "./App.css";
```



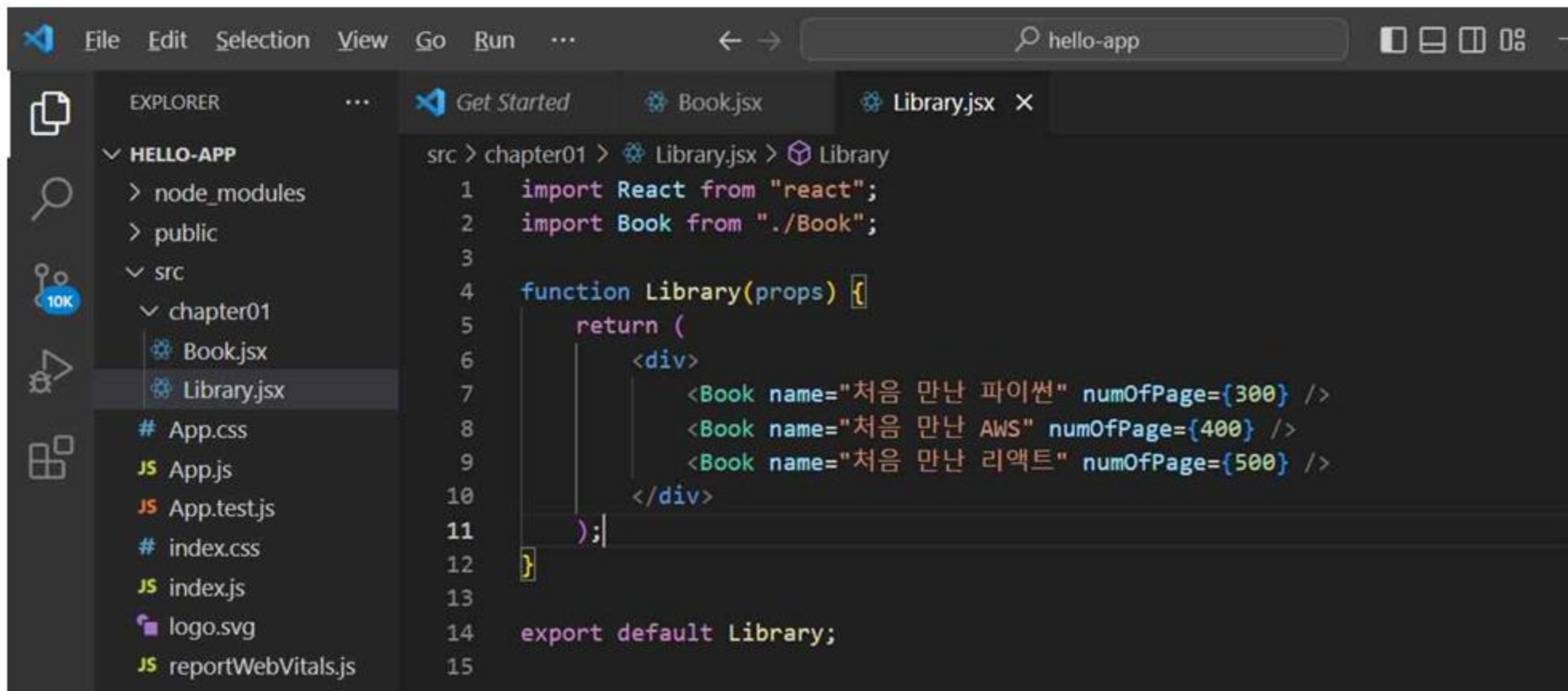
● - chapter01 폴더 생성후 Book.jsx 파일 생성



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays the project structure for 'HELLO-APP', including 'node_modules', 'public', 'src', and 'chapter01'. The 'chapter01' folder is expanded, showing 'Book.jsx' and 'Library.jsx'. The main editor area displays the code for 'Book.jsx', which imports React and defines a 'Book' function component. The component returns a JSX element with two paragraphs. The first paragraph uses the 'name' prop, and the second uses the 'numOfPage' prop. The code is as follows:

```
src > chapter01 > Book.jsx > ...
1  import React from "react";
2
3  function Book(props) {
4    return (
5      <div>
6        <h1>`이 책의 이름은 ${props.name}입니다.`</h1>
7        <h2>`이 책은 총 ${props.numOfPage}페이지로 이뤄져 있습니다.`</h2>
8      </div>
9    );
10 }
11
12 export default Book;
13
```

- Libray.jsx파일 생성



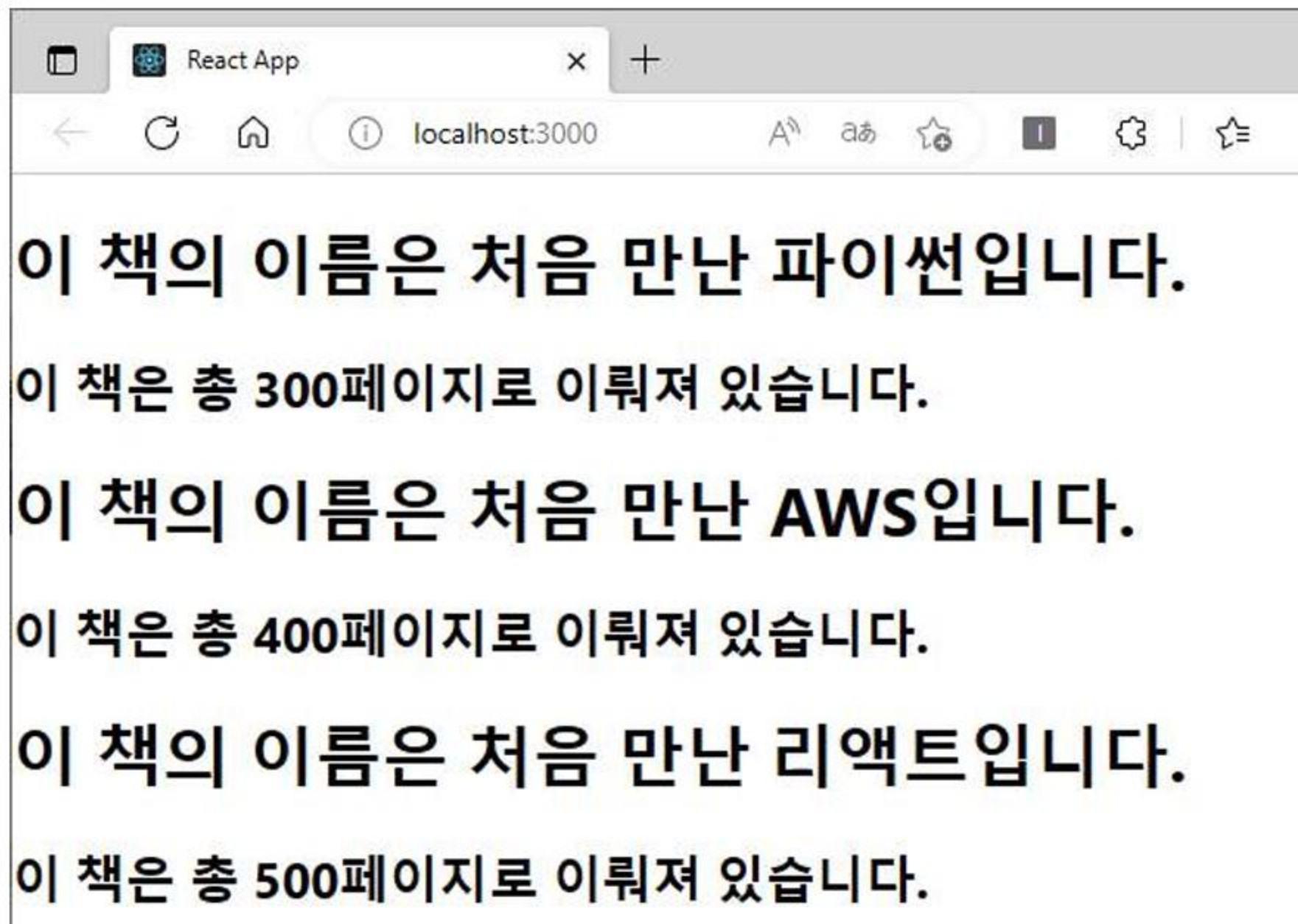
The screenshot shows the Visual Studio Code interface with the Explorer panel on the left and the Editor panel on the right. The Explorer panel shows the project structure for 'HELLO-APP', with 'src > chapter01 > Library.jsx' selected. The Editor panel shows the content of 'Library.jsx'.

```
1  import React from "react";
2  import Book from "../Book";
3
4  function Library(props) {
5    return (
6      <div>
7        <Book name="처음 만난 파이썬" numOfPage={300} />
8        <Book name="처음 만난 AWS" numOfPage={400} />
9        <Book name="처음 만난 리액트" numOfPage={500} />
10      </div>
11    );
12  }
13
14  export default Library;
```

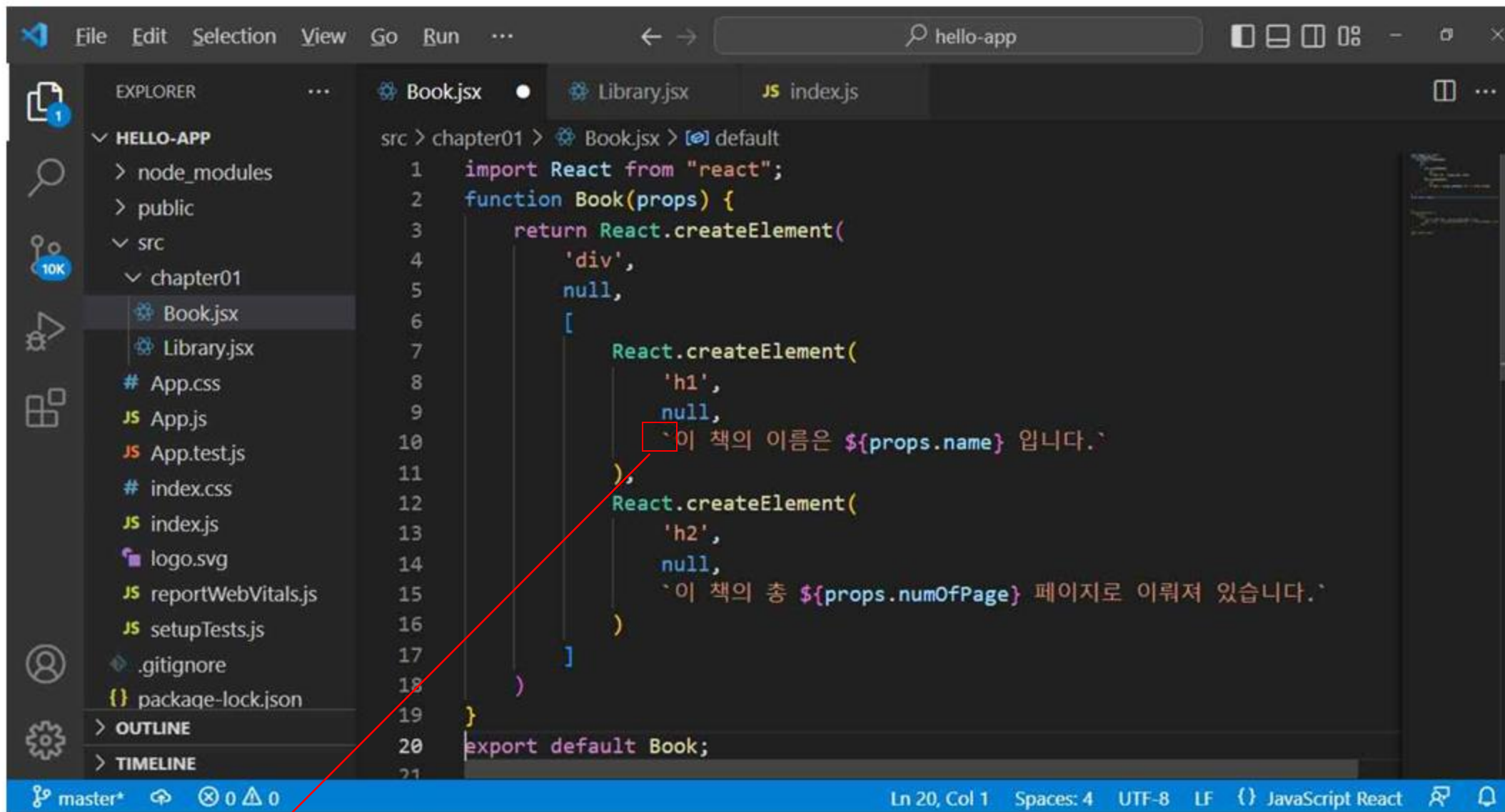
- 위에서 만든 컴포넌트를 화면에 랜더링 하기 위해 index.js 파일을 수정해야 함.

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import Library from './chapter01/Library'
4 const root = ReactDOM.createRoot(document.getElementById('root'));
5 root.render(
6   <React.StrictMode>
7     <Library />
8   </React.StrictMode>
9 );
```

다음과 같이 랜더링 됨



* JSX를 사용하지 않고 작성한 Book.jsx



The screenshot shows the Visual Studio Code editor with the 'Book.jsx' file open. The Explorer sidebar on the left shows the project structure: 'HELLO-APP' with subfolders 'node_modules', 'public', and 'src'. Under 'src', there is a 'chapter01' folder containing 'Book.jsx' and 'Library.jsx'. The main editor area displays the code for 'Book.jsx'. The code is as follows:

```
1 import React from "react";
2 function Book(props) {
3   return React.createElement(
4     'div',
5     null,
6     [
7       React.createElement(
8         'h1',
9         null,
10        `이 책의 이름은 ${props.name} 입니다.`
11      ),
12      React.createElement(
13        'h2',
14        null,
15        `이 책의 총 ${props.numOfPage} 페이지로 이뤄져 있습니다.`
16      )
17    ]
18  )
19 }
20 export default Book;
```

A red box highlights the backtick character (``) at the start of the string template on line 10. A red arrow points from this box to the text '백틱 기호임' (Backtick symbol) located below the editor. The status bar at the bottom indicates 'Ln 20, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and 'JavaScript React'.

백틱 기호임

JSX 문법

- 컴포넌트에 여러 요소가 있다면 반드시 부모 요소 하나로 감싸야 함
- App.js 파일의 App 컴포넌트 함수 내부를 모두 지우고 다음과 같이 작성 해 봄

```
2 function App(){
3   return (
4     <h1>Start React !!</h1>
5     <p>HTML 적용하기</p>
6   )
7 }
8
9 export default App;
10
11
```


index.js 도 다음과 같이 수정

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import App from './App'
4 const root = ReactDOM.createRoot(document.getElementById('root'));
5 root.render(
6   <React.StrictMode>
7     <App />
8   </React.StrictMode>
9 );
10
```

문제로 컴파일 됨 :

./src/App.js에 오류가 있습니다.

모듈 빌드 실패 (./node_modules/babel-loader/lib/index.js에서) :

SyntaxError: D:\react\hello-app\src\App.js: 인접한 JSX 요소는 바깥쪽 태그로 래핑해야 합니다. JSX 프래그먼트를 원하셨나요<>... </>?

(6:11) 4 | 5 | <h1> 반응 시작 !!</H1> > 6 | <p>HTML 적용하기</p> | ^ 7 | 8 |) 9 | } 생성자 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:349:19)에서

FlowParserMixin.raise에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:3247:19)

FlowParserMixin.jsxParseElementAt (D : \ react \ hello-app node_modules \ @babel \ parser \ lib \ index.js : 6780 : 18)

FlowParserMixin.jsxParseElement에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:6787:17)

FlowParserMixin.parseExprAtom에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:6797:19)

FlowParserMixin.parseExprSubscripts에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:10562:23)

FlowParserMixin.parseUpdate에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:10547:21)

FlowParserMixin.parseMaybeUnary에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:10527:23)

FlowParserMixin.parseMaybeUnaryOrPrivate에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:10381:61)

FlowParserMixin.parseExprOps에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:10386:23)

FlowParserMixin.parseMaybeConditional에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:10363:23)

FlowParserMixin.parseMaybeAssign (D : \ react \ hello-app node_modules \ @babel \ parser \ lib \ index.js : 10326 : 21)

D:\react\hello-app\node_modules\@babel\parser\lib\index.js:5618:39에서

FlowParserMixin.tryParse에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:3585:20)

FlowParserMixin.parseMaybeAssign (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:5618:18)에서

D:\react\hello-app\node_modules\@babel\parser\lib\index.js:10296:39에서

FlowParserMixin.allowInAnd에서 (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:11911:12)

FlowParserMixin.parseMaybeAssignAllowIn (D:\react\hello-app\node_modules\@babel\parser\lib\index.js:10296:17)에서

FlowParserMixin.parseParenAndDistinguishExpression (D : \ react \ hello-app \ node_modules \ @babel \ parser \ lib \ index.js :

11172 : 28)

다음과 같이 App.js를 부모 요소로 감싸면 오류가 해결 됨

```
1
2 function App(){
3   return (
4     <div>
5       <h1>Start React !!</h1>
6       <p>HTML 적용하기</p>
7     </div>
8   )
9 }
10 export default App;
```

리액트 컴포넌트에서 요소 여러 개를 왜 하나의 요소로 꼭 감싸 주어야 할까요?

그것은 Virtual DOM에서 컴포넌트 변화를 감지해 낼 때 효율적으로 비교할 수 있도록 컴포넌트 내부는 하나의 DOM 트리 구조로 이루어져야 한다는 규칙이 있기 때문

리액트 v16 이상 부터 도입된 Fragment라는 기능을 사용하면 div 요소를 사용하지 않아도 됨

```
1 import { Fragment } from "react";
2
3 function App(){
4   return (
5     <Fragment>
6       <h1>Start React !!</h1>
7       <p>HTML 적용하기</p>
8     </Fragment>
9   )
10 }
11 export default App;
12
```

Fragment는 다음과 같은 형태로도 표현할 수 있음

```
1 function App(){
2   return (
3     <>
4       <h1>Start React !!</h1>
5       <p>HTML 적용하기</p>
6     </>
7   )
8 }
9 export default App;
10
```

● 엘리먼트(Element)란?

- 요소, 성분이란 뜻으로 리액트 앱을 구성하는 요소를 의미하며, 리액트 앱을 구성하는 가장 작은 블록을 지칭 함.

```
const element = <h1> Hello, world </h1>
```

이 코드는 JSX를 사용하여 작성된 코드로써, 이 코드가 실행될 때, 대입 연산자의 오른쪽 부분은 리액트의 createElement() 함수를 사용하여 엘리먼트를 생성하게 됨.

결국 이렇게 생성된 것이 바로 리액트 엘리먼트가 되며, 리액트는 이 엘리먼트를 이용해서 실제 화면에서 보게 될 DOM 엘리먼트를 생성 함.

● 엘리먼트의 생김새

리액트 엘리먼트는 자바스크립트 객체 형태로 존재.

엘리먼트는 컴포넌트 유형(예:Button)과 속성 (예:color) 및 내부의 모든 자식(children)에 대한 정보를 포함하고 있는 일반적인 자바스크립트 객체임.

이 객체는 마음대로 변경할 수 없는 불변성을 갖고 있으므로 한 번 생성되면 바꿀 수 없음

엘리먼트의 실제 모습

```
{
  type:'button',
  props: {
    className: 'bg-green',
    children: {
      type: 'b',
      props: {
        children: 'Hello, element!'
      }
    }
  }
}
```

위 코드는 버튼을 나타내기 위한 엘리먼트이며 실제로 랜더링 된다면 다음과 같은 DOM 엘리먼트가 됨

```
<button class='bg-green'>  
  <b>  
    'Hello, element!'  
  </b>  
</button>
```


● 엘리먼트 렌더링 하기

```
<div id="root"> </div>
```

단순하지만 이 코드는 모든 리액트 앱에 필수적으로 들어가는 아주 중요한 코드임.

이것을 **Root DOM node**라고 함.

이 <div> 태그 안에 있는 모든 것이 리액트 DOM 에 의해 관리되기 때문에 중요 함.

오직 리액트만으로 만들어진 모든 웹사이트들은 단 하나의 Root DOM node를 가지게 됨.

root <div> 에 리액트를 렌더링하기 위해서는 다음과 같은 코드를 사용

```
const element = <h1> 안녕, 리액트 </h1>
```

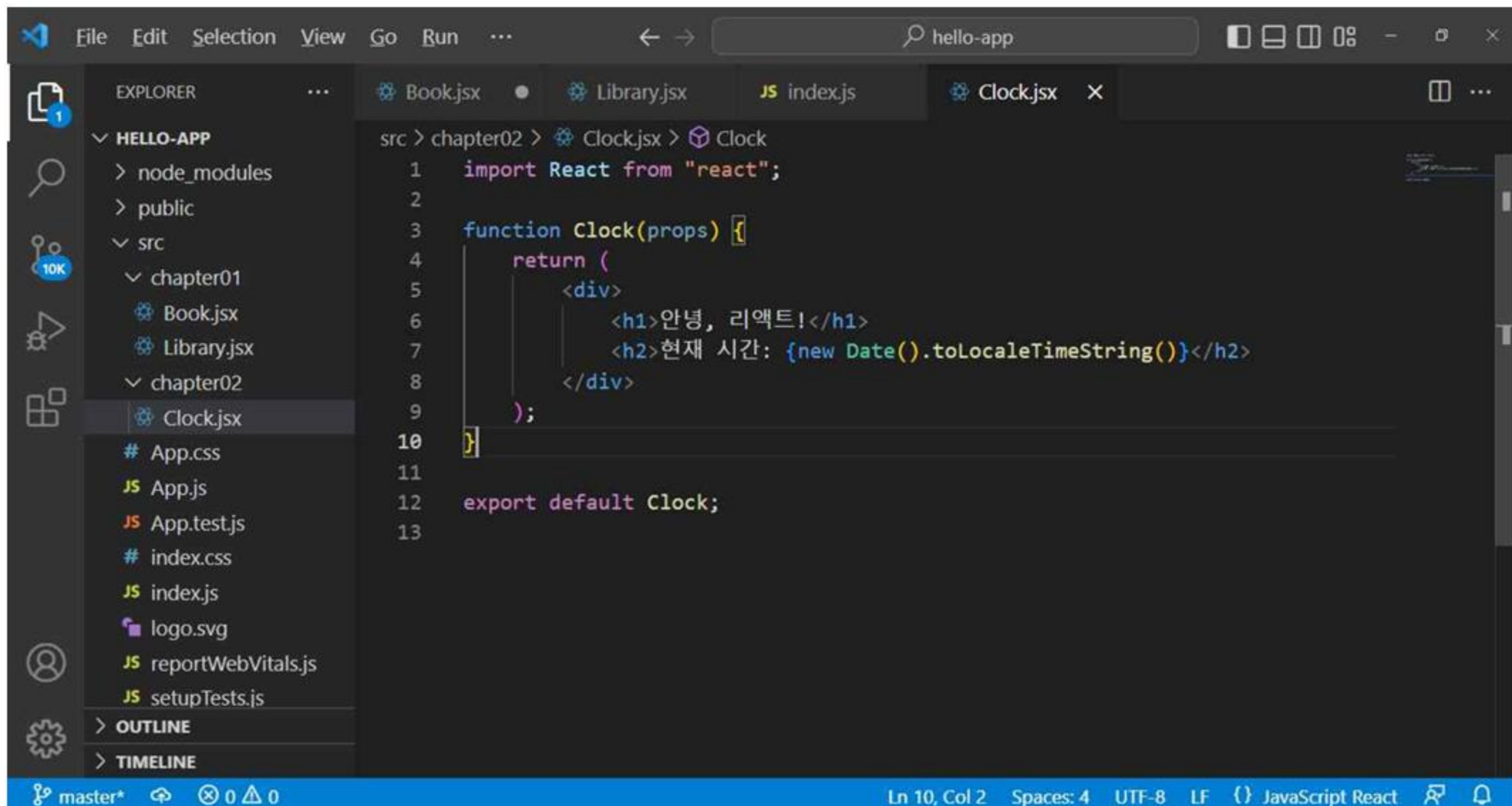
```
ReactDOM.render(element, document.getElementById('root'));
```

- 렌더링된 엘리먼트 업데이트 하기

리액트 엘리먼트의 중요한 특징 중 하나가 불변성으로 한 번 생성된 엘리먼트는 바꿀수가 없기 때문에 엘리먼트를 업데이트 하기 위해서는 다시 생성해야 함.

* 시계 만들기

-chapter02 폴더 생성 후 Clock.jsx 파일 생성



- index.js 수정



```
File Edit Selection View Go Run Terminal Help
• index.js - hellp-app - Visual Studio Code

JS App.js M JS index.js 1, M • Clock.jsx U # App.css M
src > JS index.js > ...
6 import Clock from "../chapter02/Clock";
7
8 setInterval(() => {
9   const root = ReactDOM.createRoot(document.getElementById("root"));
10  root.render(
11    <React.StrictMode>
12    <Clock />
13    </React.StrictMode>
14  );
15 }, 1000);
```

← → ↻ ⓘ localhost:3000



안녕, 리액트!

현재 시간: 오전 8:22:11

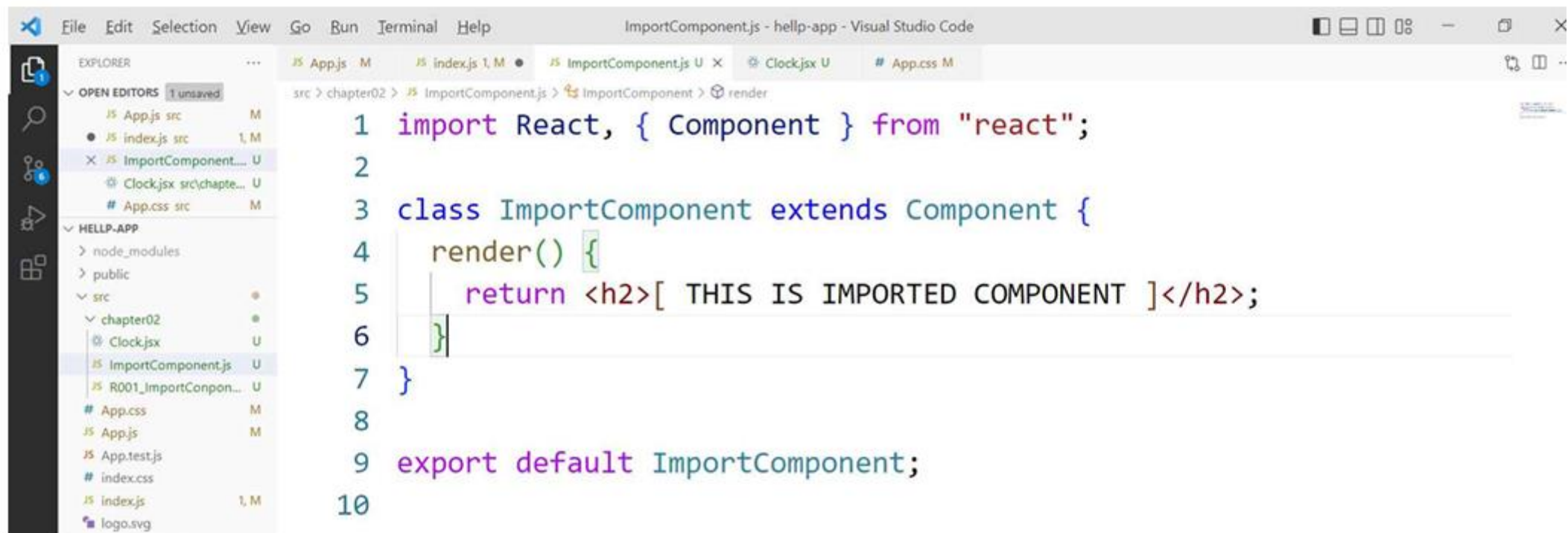
● 컴포넌트

리액트 컴포넌트는 개념적으로 자바스크립트의 함수와 같이 입력을 받아 출력을 한다는 것에서 비슷하지만, 리액트 컴포넌트에서의 입력은 바로 props 라는 것이고, 출력은 엘리먼트가 됨.

결국 리액트 컴포넌트가 해주는 역할은 어떠한 속성들을 입력으로 받아서 그에 맞는 리액트 엘리먼트를 생성하여 리턴해 주는 것.

* Component 사용하기

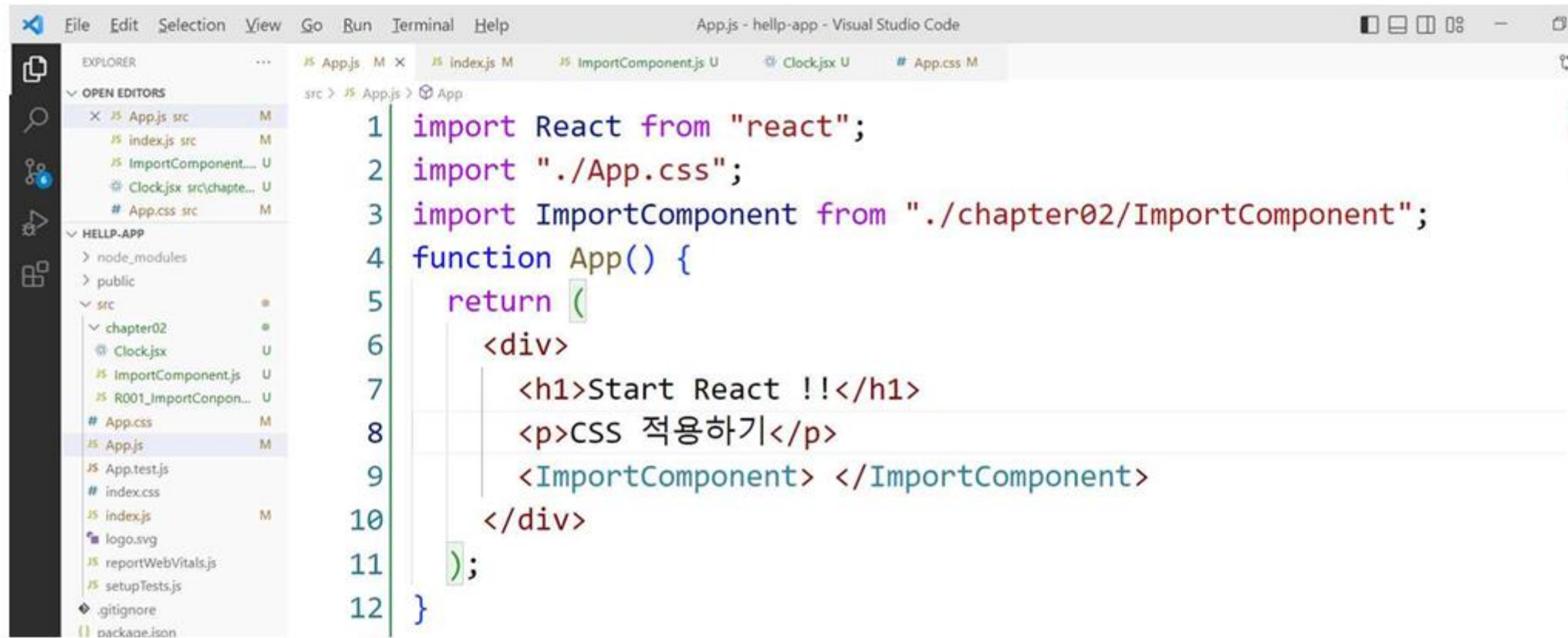
- chapter02에 ImportComponent.js 파일 생성



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left and the Editor window on the right. The Explorer sidebar shows the project structure with 'chapter02' expanded, and 'ImportComponent.js' is highlighted. The Editor window shows the code for 'ImportComponent.js'.

```
1 import React, { Component } from "react";
2
3 class ImportComponent extends Component {
4   render() {
5     return <h2>[ THIS IS IMPORTED COMPONENT ]</h2>;
6   }
7 }
8
9 export default ImportComponent;
10
```

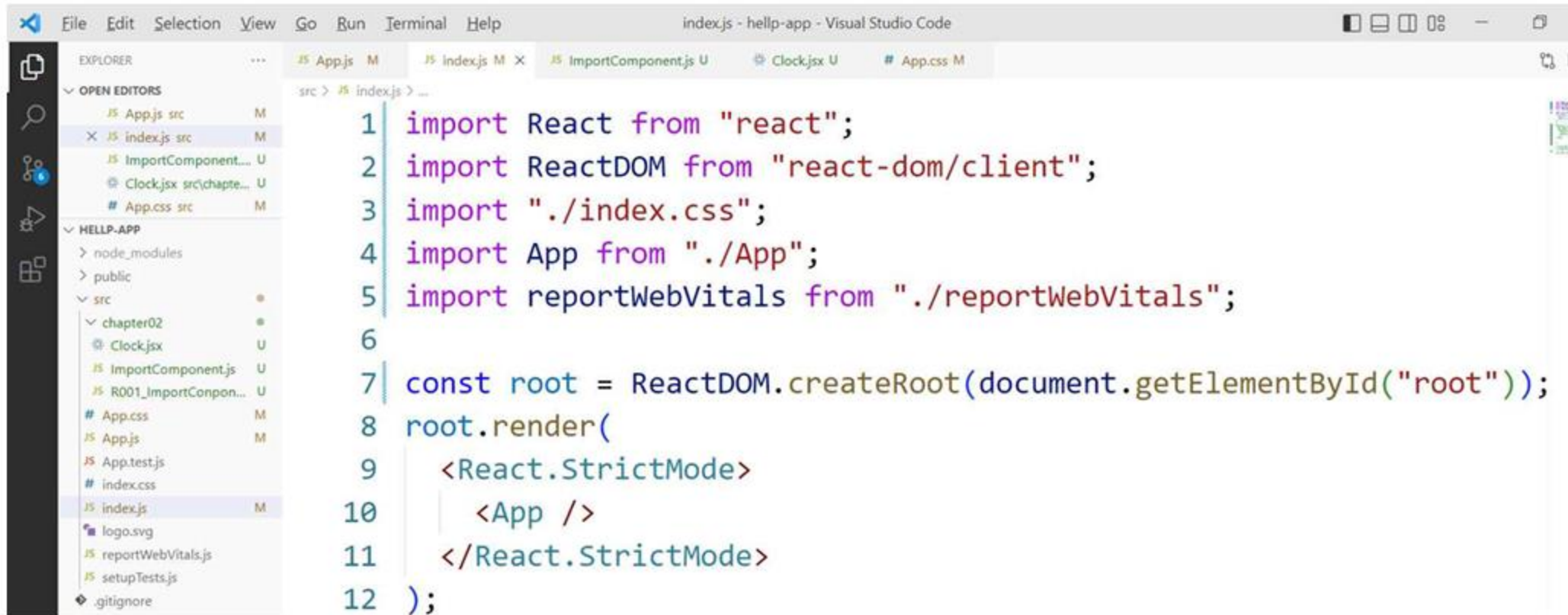
- App.js를 수정



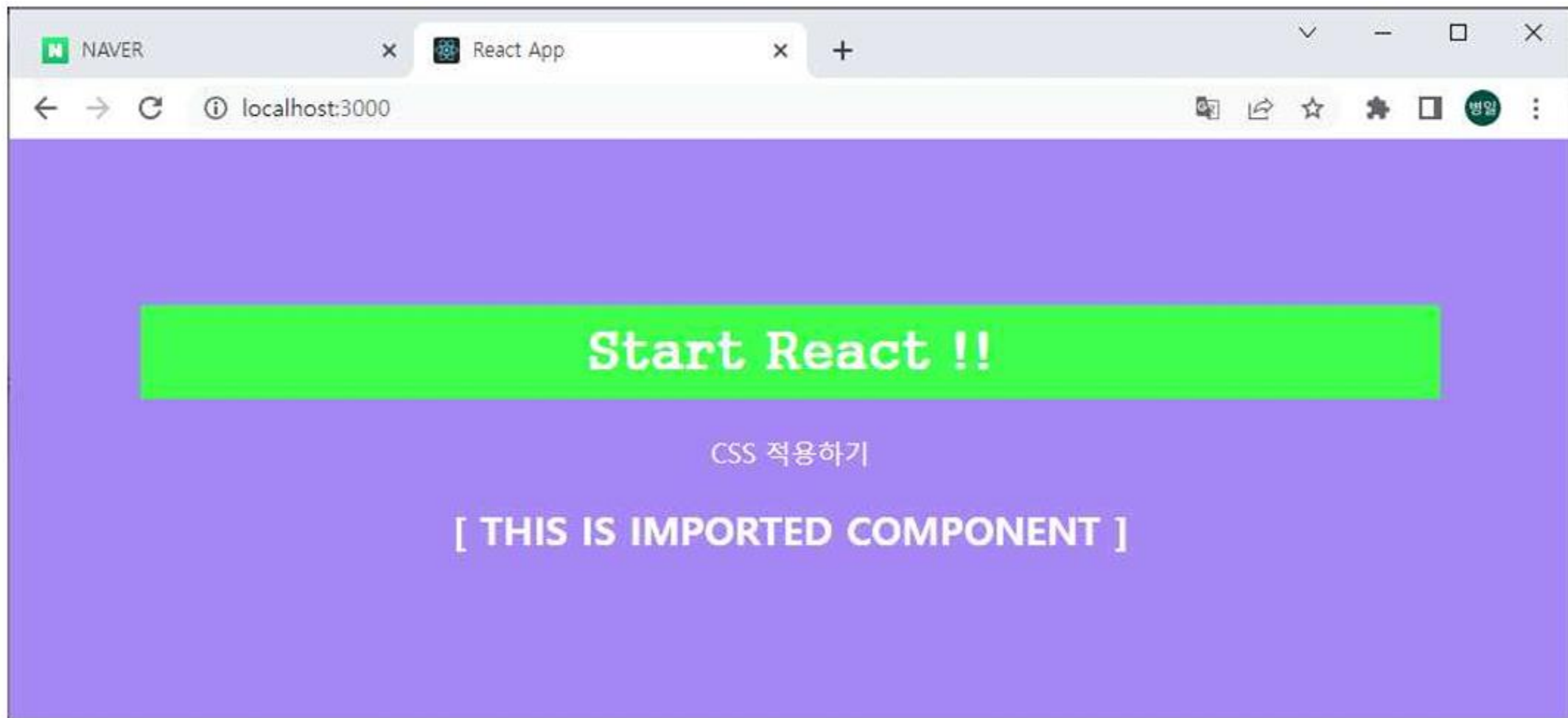
```
File Edit Selection View Go Run Terminal Help App.js - hellp-app - Visual Studio Code

src > JS App.js > App
1 import React from "react";
2 import "./App.css";
3 import ImportComponent from "../chapter02/ImportComponent";
4 function App() {
5   return (
6     <div>
7       <h1>Start React !!</h1>
8       <p>CSS 적용하기</p>
9       <ImportComponent> </ImportComponent>
10    </div>
11  );
12 }
```


- index.js 수정



```
1 import React from "react";
2 import ReactDOM from "react-dom/client";
3 import "./index.css";
4 import App from "./App";
5 import reportWebVitals from "./reportWebVitals";
6
7 const root = ReactDOM.createRoot(document.getElementById("root"));
8 root.render(
9   <React.StrictMode>
10     <App />
11   </React.StrictMode>
12 );
```



● props

property의 약어로 속성을 뜻함, 즉 리액트 컴포넌트의 속성을 말함.

● props의 특징

- 읽기 전용 : 값을 변경할 수 없음.

그러므로 리액트 컴포넌트의 props는 바꿀 수 없고, 같은 props가 들어오면 항상 같은 엘리먼트를 리턴해야 함.

● props 사용법

먼저 JSX를 사용하는 경우에는 다음과 같이 키와 값으로 이루어진 키-값 쌍의 형태로 컴포넌트에 props를 넣을 수 있음.

```
function App(props) {  
  return (  
    <Profile  
      name="프론트엔드"  
      introduction="안녕하세요, 리액트입니다"  
      viewCount={1500}  
    />  
  );  
}
```

- * props에 값을 넣을 때에 문자열 이외에 정수, 변수, 그리고 다른 컴포넌트등이 들어갈 경우에는 중괄호를 사용해서 감싸주어야 함.
물론 문자열을 중괄호로 감싸도 상관 없음.

* 또한 중괄호를 사용하면 다음과 같이 props 의 값으로 컴포넌트를 넣을수 있음

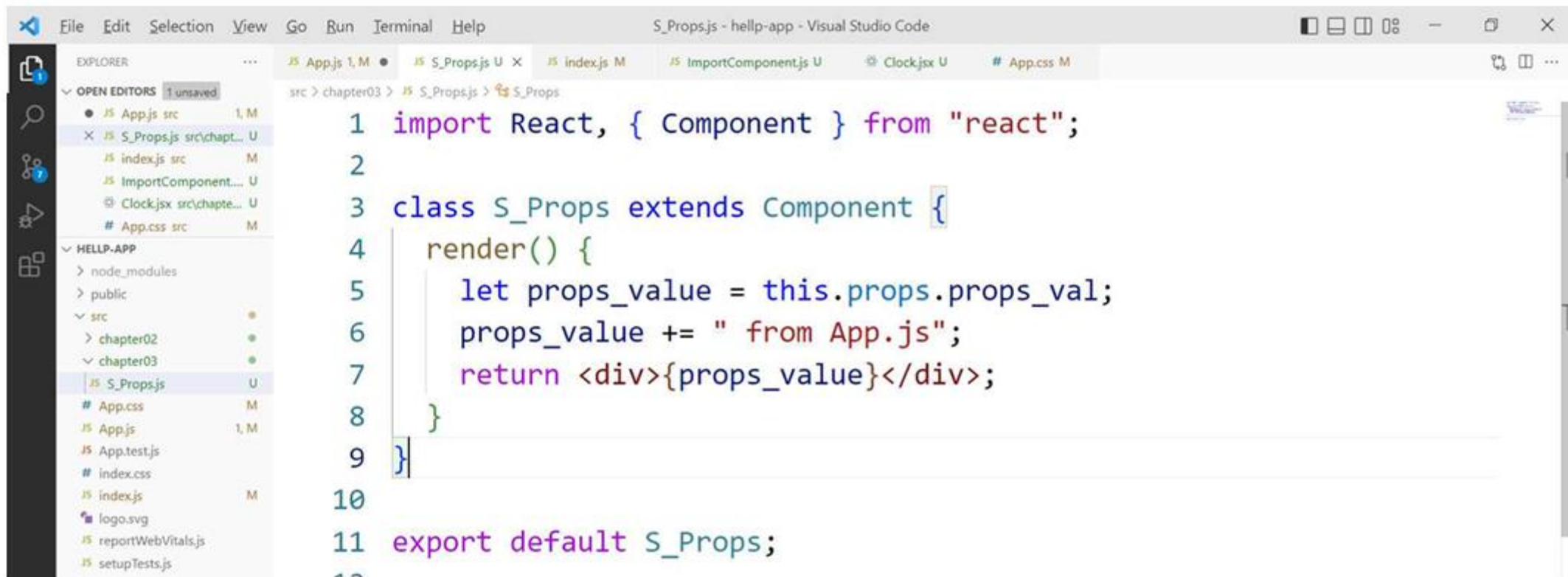
```
function App(props) {  
  return (  
    <Layout  
      width={2560}  
      height={1400}  
      header = {  
        <Header title=“산대특과정” />  
      }  
      footer = {  
        <Footer />  
      }  
    />  
  );  
}
```

Layout 컴포넌트의 props로는 정숫값을 가진 width, height 와 리엑트 엘리먼트인 footer가 들어오게 됨.

이처럼 JSX를 사용하는 경우에는 간단하게 컴포넌트에 props를 넣을 수 있음

● props 사용하기

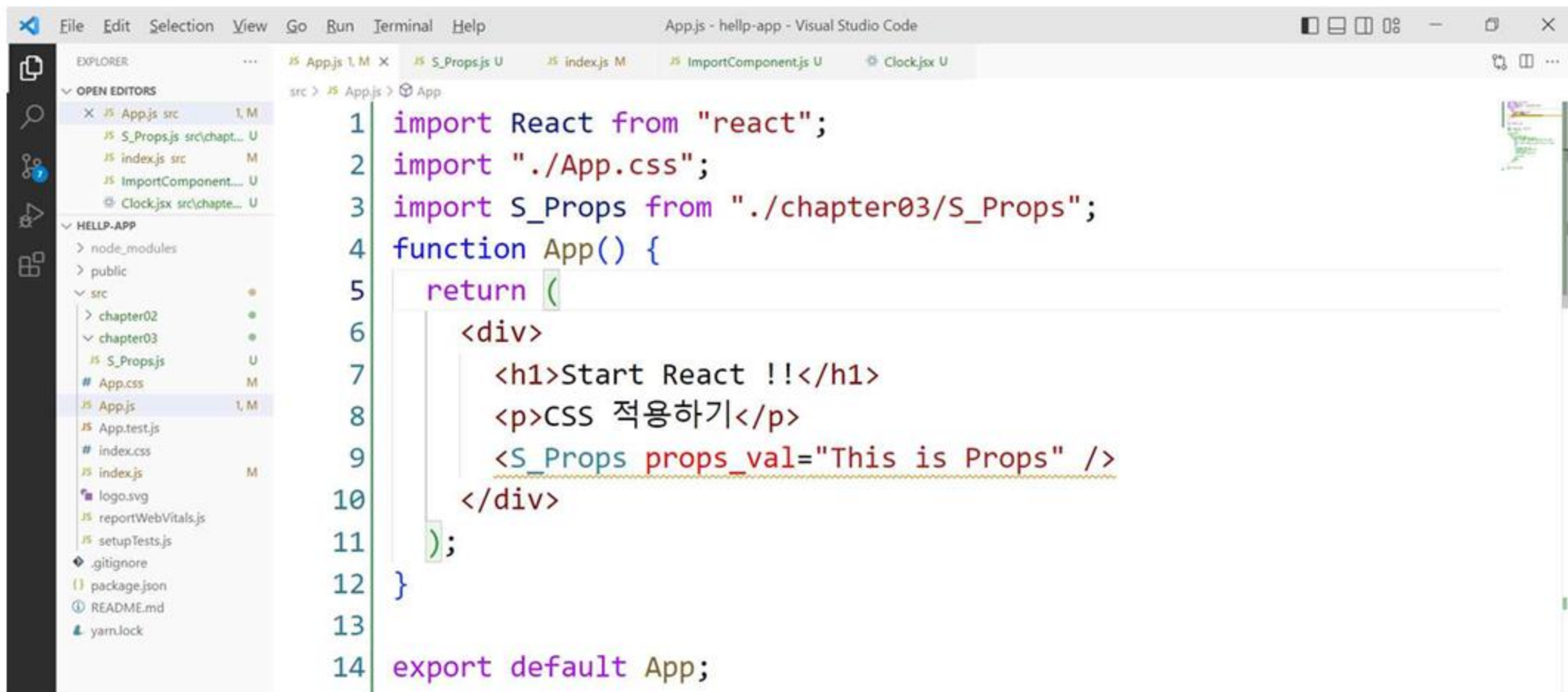
- chapter03 폴더 생성하고 S_Props.js 파일 생성



The screenshot shows the Visual Studio Code interface with the S_Props.js file open in the editor. The Explorer sidebar on the left shows the project structure, including the src directory and the chapter03 folder. The editor displays the following code:

```
1 import React, { Component } from "react";
2
3 class S_Props extends Component {
4   render() {
5     let props_value = this.props.props_val;
6     props_value += " from App.js";
7     return <div>{props_value}</div>;
8   }
9 }
10
11 export default S_Props;
```

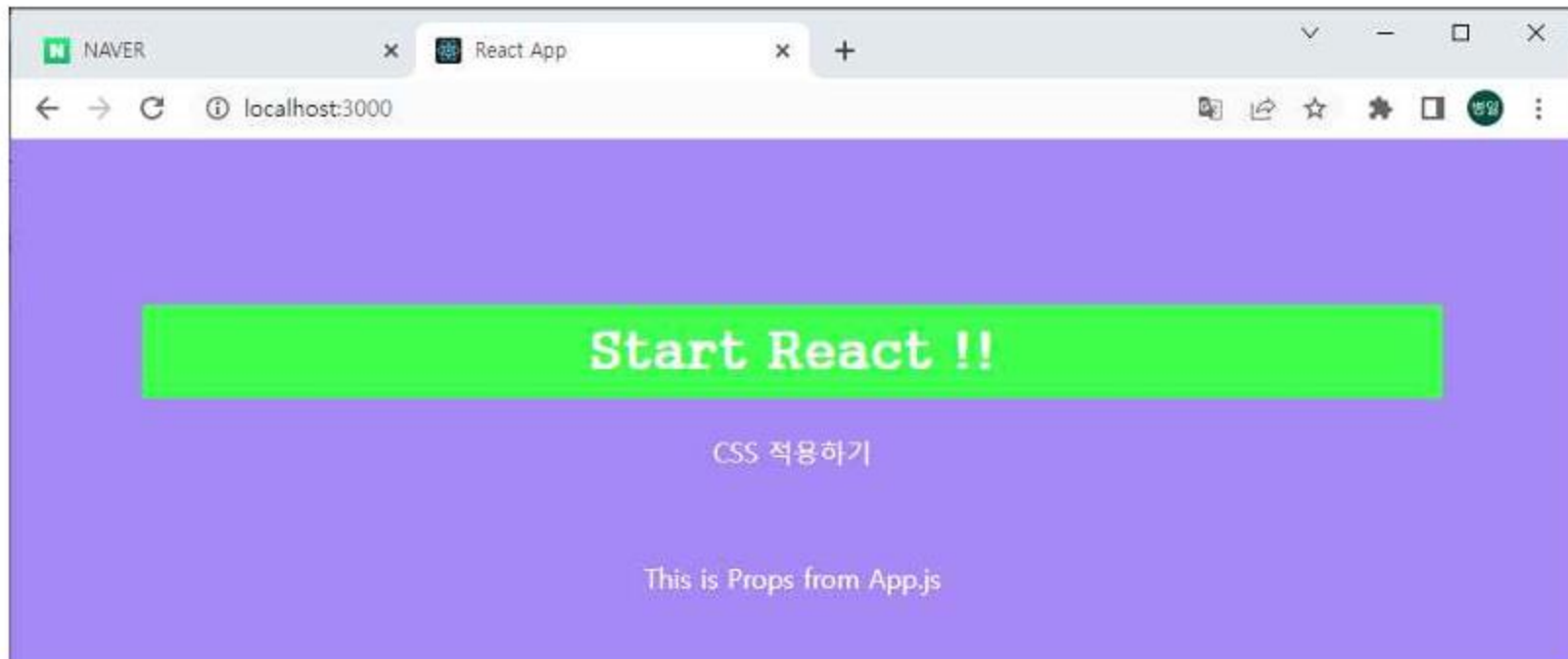
- App.js 파일 수정



```
File Edit Selection View Go Run Terminal Help
App.js - help-app - Visual Studio Code

EXPLORER
OPEN EDITORS
  X App.js src 1, M
  App.js src\chapt... U
  index.js src M
  ImportComponent... U
  Clock.jsx src\chapte... U
HELPP-APP
  > node_modules
  > public
  > src
    > chapter02
    > chapter03
      App.js U
      App.css M
      App.js 1, M
      App.test.js
      index.css
      index.js M
      logo.svg
      reportWebVitals.js
      setupTests.js
      .gitignore
      package.json
      README.md
      yarn.lock

src > App.js > App
1 import React from "react";
2 import "./App.css";
3 import S_Props from "../chapter03/S_Props";
4 function App() {
5   return (
6     <div>
7       <h1>Start React !!</h1>
8       <p>CSS 적용하기</p>
9       <S_Props props_val="This is Props" />
10    </div>
11  );
12 }
13
14 export default App;
```



- props는 부모 컴포넌트가 자식 컴포넌트에 데이터를 전달할 때 사용
- props를 전달 받는 자식 컴포넌트에서는 데이터를 수정할 수 없음
- 데이터를 변경하기 위해서는 컴포넌트 내부에서만 사용하는 변수에 값을 넣어 사용해야 함.
- `this.props.` 뒤에 상위 컴포넌트(App.js)에서 전달 받은 props 변수명을 붙이면, 해당 데이터를 사용할 수 있음
- 데이터를 수정할 경우, props 자체가 아닌 컴포넌트 내부 변수(`props_value`)에 옮겨 가공.

- 컴포넌트의 종류

크게 클래스 컴포넌트와 함수 컴포넌트로 나눌수 있음

- 함수 컴포넌트

```
function Welcome(props) {  
  return <h1> 안녕, {props.name} </h1>
```

- 클래스 컴포넌트

```
class Welcome extends React.Component {  
  render() {  
    return <h1> 안녕, {this.props.name} </h1>  
  }  
}
```

함수 컴포넌트와의 가장 큰 차이점은 React.Component를 상속받아서 만든다는 것.

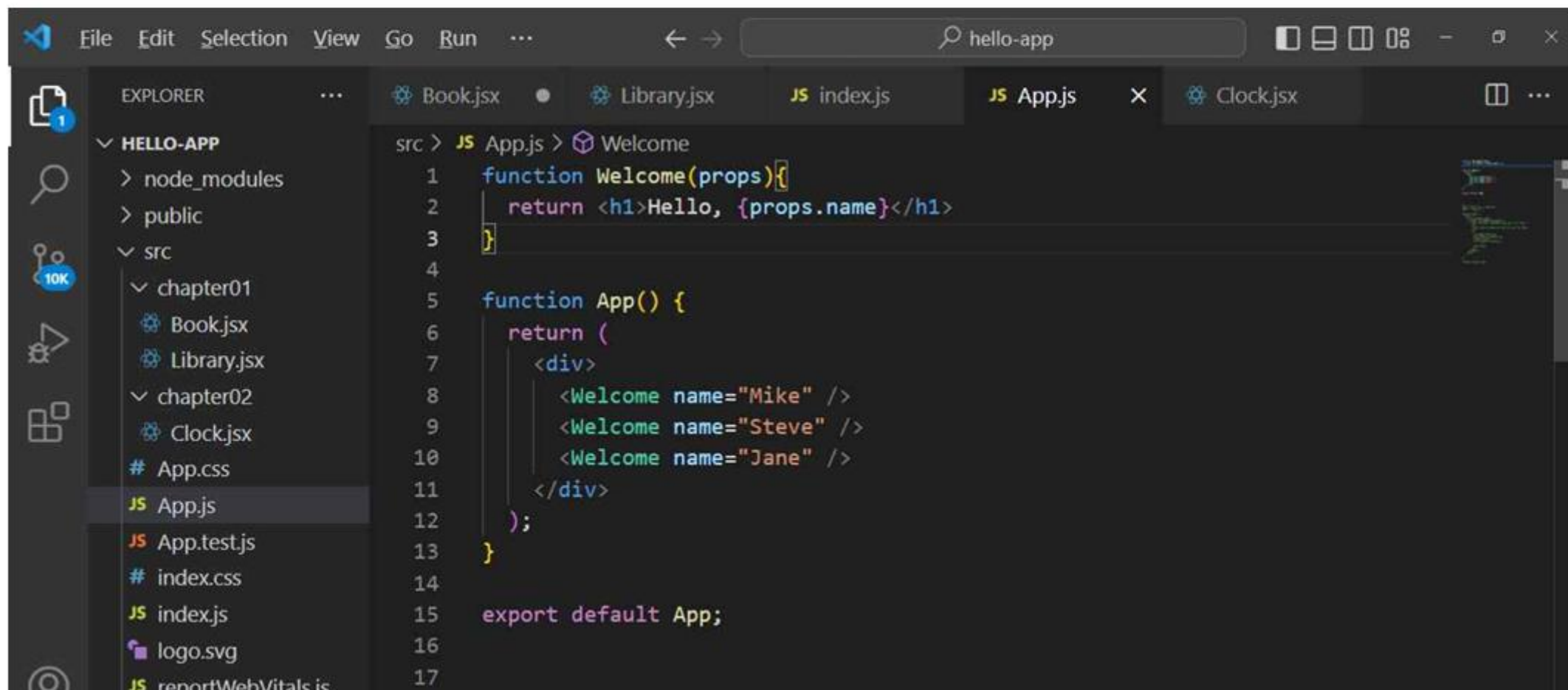
● 컴포넌트 이름 짓기

- 컴포넌트들은 항상 대문자로 시작해야 함.
- 왜냐하면 리엑트는 소문자로 시작하는 컴포넌트를 DOM 태그로 인식하기 때문
- 예를 들어 `<div>`나 ``과 같이 사용하는 것은 내장 컴포넌트라는 것을 뜻하며, 'div'나 'span'과 같은 문자열 형태로 `React.createElement()`에 전달됩니다.

하지만 `<Foo />`와 같이 대문자로 시작하는 경우에는 `React.createElement(Foo)`의 형태로 컴파일되며 자바스크립트 파일 내에서 사용자가 정의했거나 임포트한 컴포넌트를 가리키기 때문에 컴포넌트 이름은 항상 대문자로 시작해야 함.

* 컴포넌트 합성

여러 개의 컴포넌트를 합쳐서 하나의 컴포넌트로 만드는 것
App.js 파일을 다음과 같이 수정한 후 index.js 파일도 수정 후 결과 확인





Hello, Mike

Hello, Steve

Hello, Jane

* 컴포넌트 추출

- 컴포넌트 합성과 반대로 복잡한 컴포넌트를 쪼개서 여러 개의 컴포넌트로 나누는 것.

즉, 큰 컴포넌트에서 일부를 추출해서 새로운 컴포넌트를 만드는 것

컴포넌트가 작아질수록 해당 컴포넌트의 기능과 목적이 명확해지고, props도 단순해지기 때문에 다른곳에서 사용할 수 있는 확률이 높아지므로 컴포넌트 추출을 잘 활용하게 되면 컴포넌트의 재사용성이 올라가게 됨.

● 컴포넌트 추출 과정

```
function Comment(props) {  
  return (  
    <div className="comment">  
      <div className="user-info">  
        <img className="avata"  
          src={props.author.avatarUrl}  
          alt={props.author.name}  
        />  
        <div className="user-info-name">  
          {props.author.name}  
        </div>  
      </div>  
  
      <div className="comment-text">  
        {props.text}  
      </div>  
  
      <div className="comment-date">  
        {formatDate(props.date)}  
      </div>  
    );  
  }  
}
```

- 먼저 Avatar 컴포넌트를 추출

```
function Avator(props) {  
  return(  
    <img className="avatar"  
      src={props.user.avatorUrl}  
      alt={props.user.name}  
    />  
  );  
}
```

이렇게 추출된 Avatar 컴포넌트를 실제로 Comment 컴포넌트에 반영하면

```
function Comment(props) {  
  return (  
    <div className="comment">  
      <div className="user-info">  
        <Avatar user={props.author} />  
        <div className="user-info-name">  
          {props.author.name}  
        </div>  
      </div>  
  
      <div className="comment-text">  
        {props.text}  
      </div>  
  
      <div className="comment-date">  
        {formatDate(props.date)}  
      </div>  
    );  
  }  
}
```


다음은 사용자 정보를 담고 있는 부분을 UserInfo라는 컴포넌트로 추출

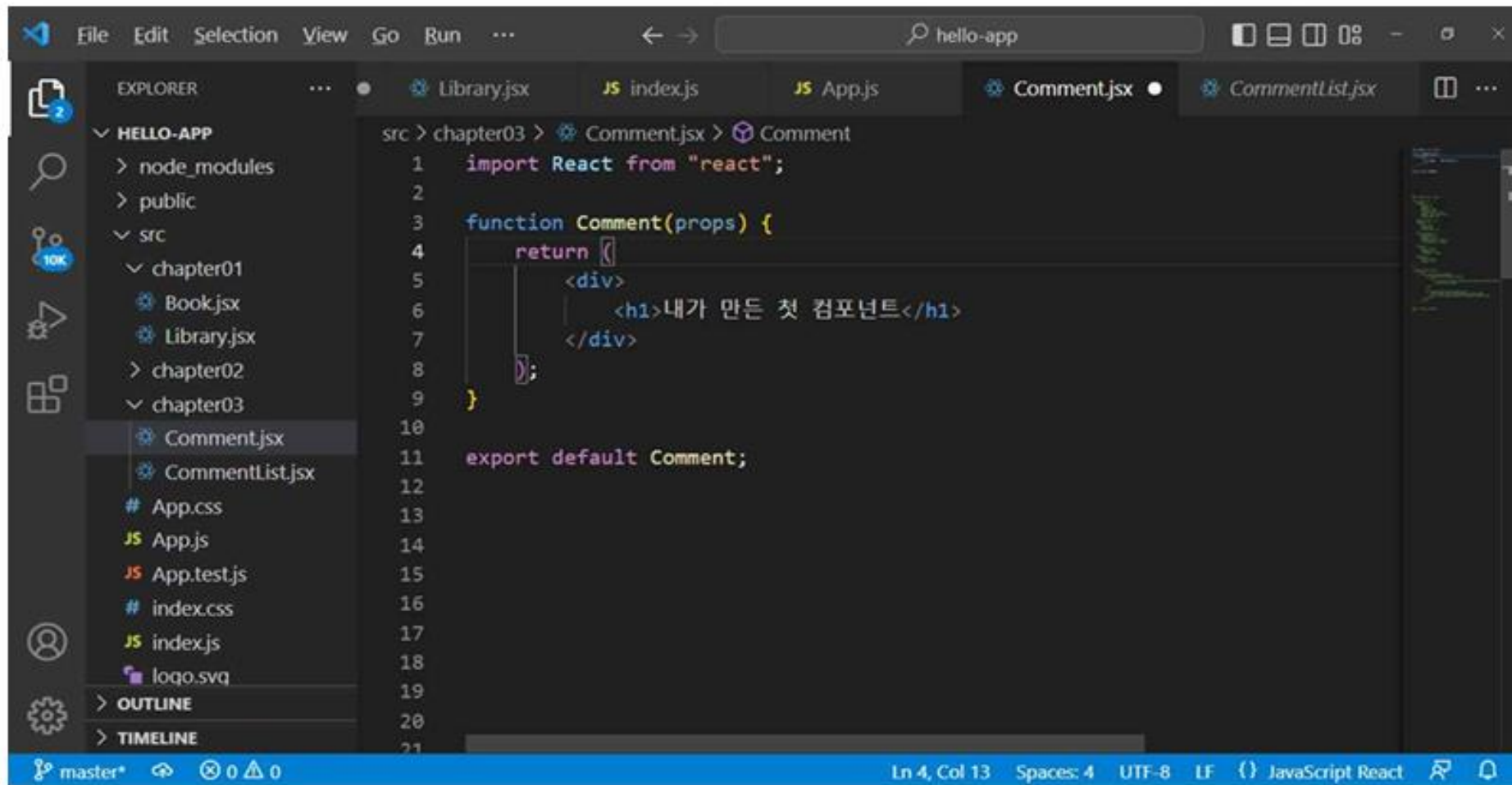
```
function UserInfo(props) {  
  return(  
    <div className="user-info">  
      <Avatar user={props.user} />  
      <div className="user-info-name">  
        {props.user.name}  
      </div>  
    </div>  
  );  
}
```

추출된 UserInfo 컴포넌트를 Comment 컴포넌트에 반영

```
function Comment(props) {  
  return (  
    <div className="comment">  
      <UserInfo user={props.author} />  
      <div className="comment-text">  
        {props.text}  
      </div>  
  
      <div className="comment-date">  
        {formatDate(props.date)}  
      </div>  
    </div>  
  );  
}
```

● 댓글 컴포넌트 만들기

Comment.jsx 파일 생성

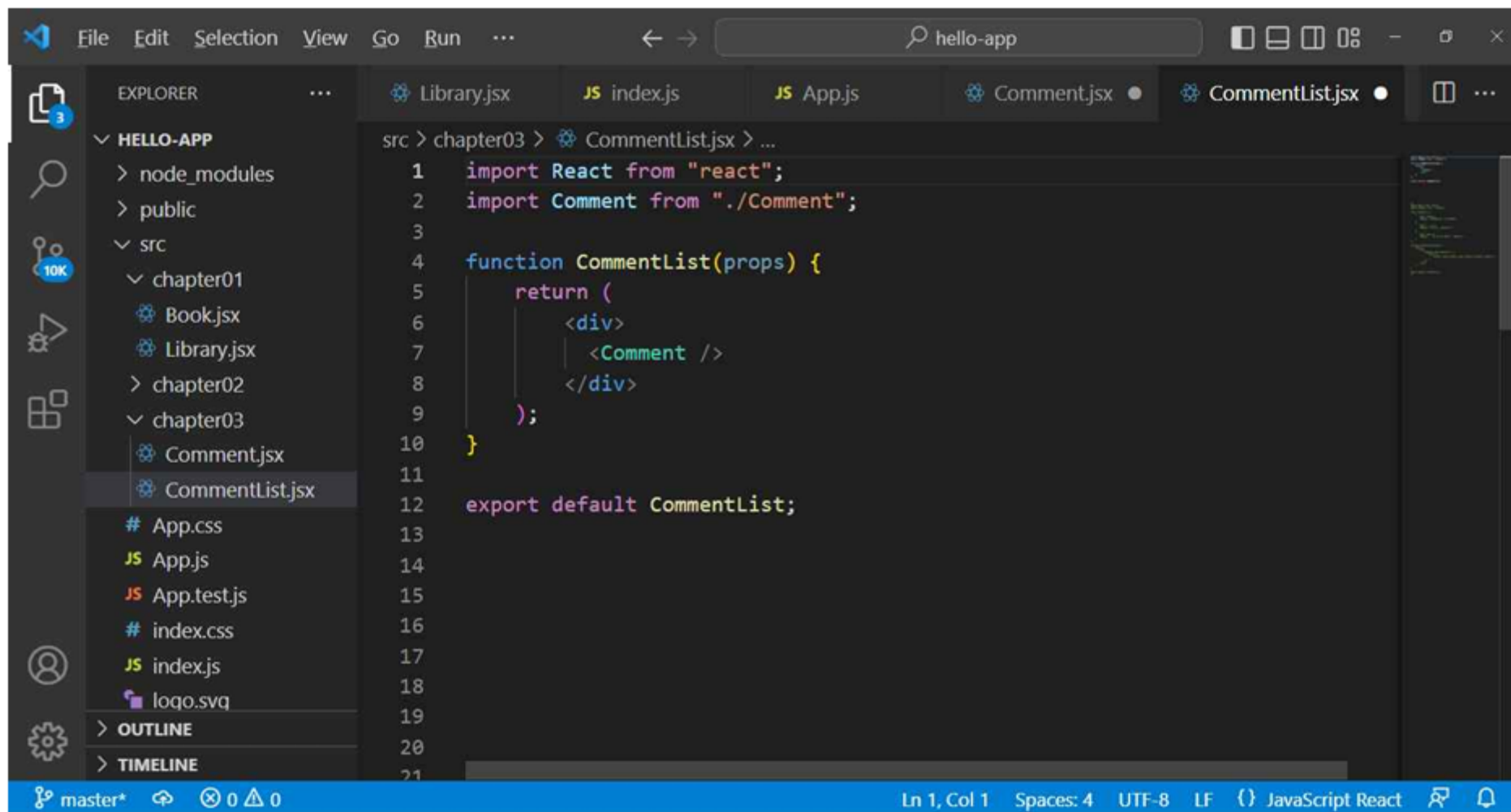


The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left displays the project structure for 'HELLO-APP', with 'src > chapter03 > Comment.jsx' selected. The main editor window shows the code for 'Comment.jsx' with the following content:

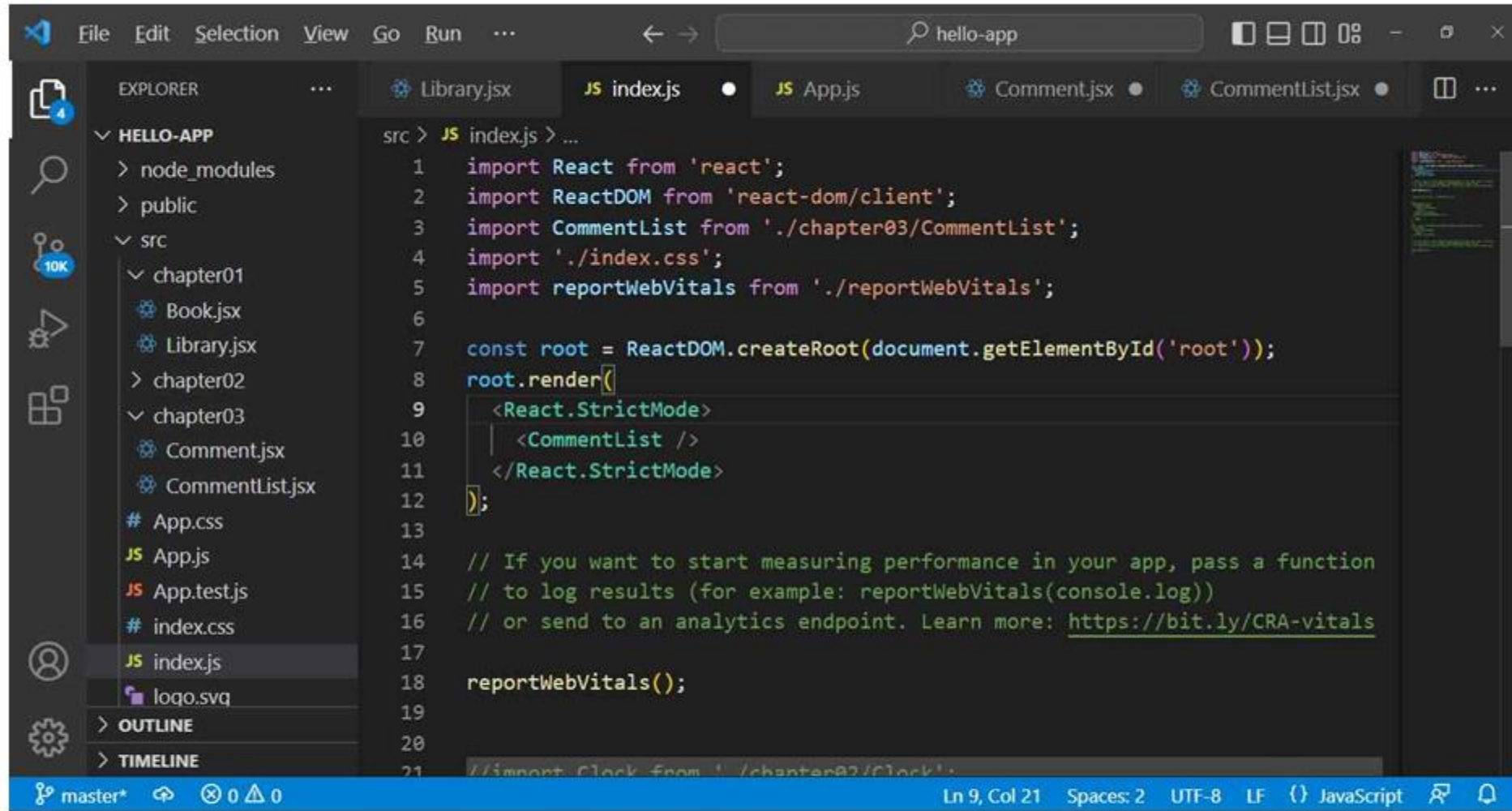
```
1 import React from "react";
2
3 function Comment(props) {
4   return (
5     <div>
6       <h1>내가 만든 첫 컴포넌트</h1>
7     </div>
8   );
9 }
10
11 export default Comment;
```

The status bar at the bottom indicates the current position is 'Ln 4, Col 13' with 'Spaces: 4', 'UTF-8' encoding, and 'LF' line endings. The language mode is set to 'JavaScript React'.

- CommentList.jsx 파일 생성



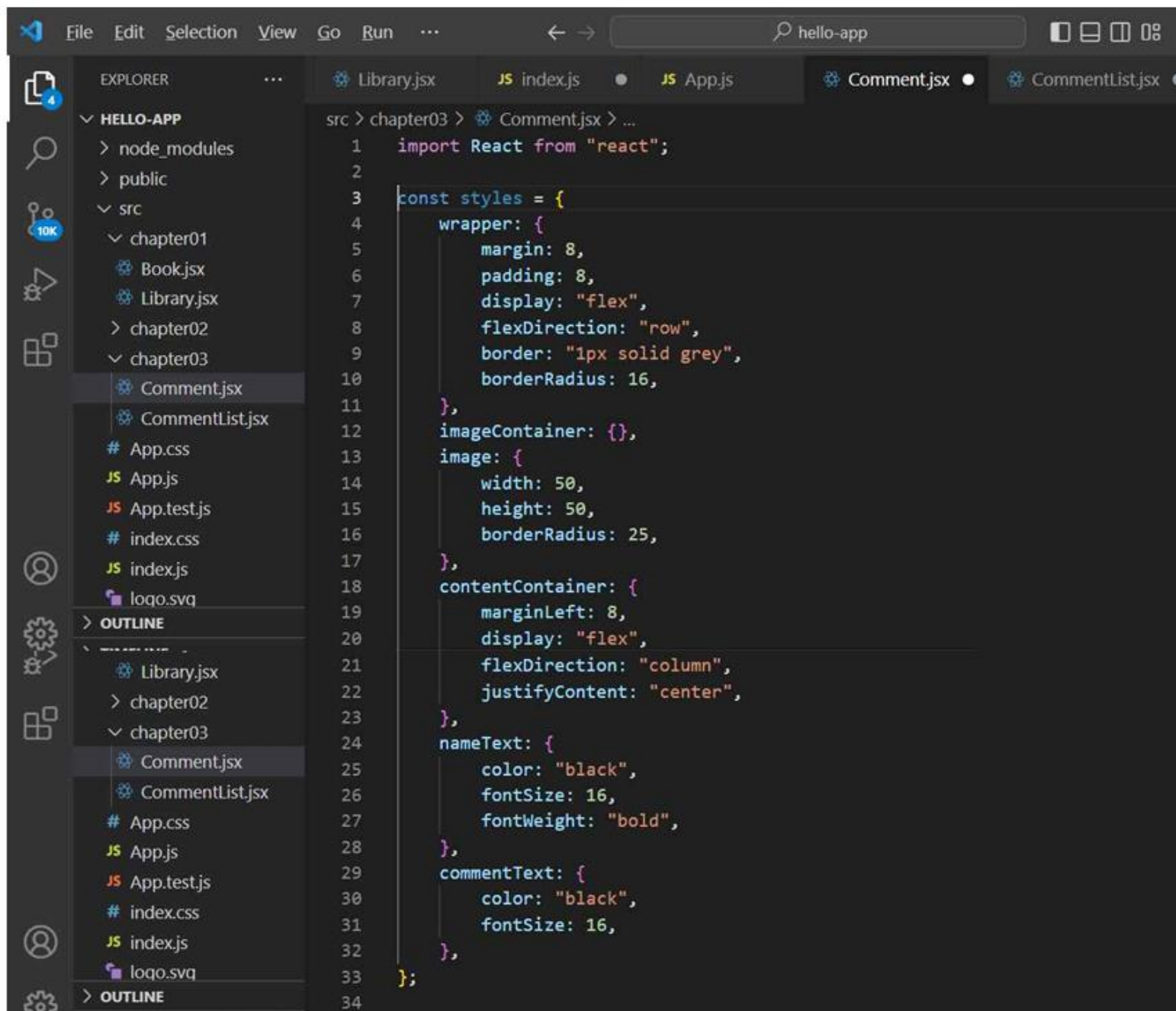
- index.js 수정



```
src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import CommentList from './chapter03/CommentList';
4  import './index.css';
5  import reportWebVitals from './reportWebVitals';
6
7  const root = ReactDOM.createRoot(document.getElementById('root'));
8  root.render(
9    <React.StrictMode>
10     <CommentList />
11   </React.StrictMode>
12 );
13
14 // If you want to start measuring performance in your app, pass a function
15 // to log results (for example: reportWebVitals(console.log))
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17
18 reportWebVitals();
19
20
21 //import Clock from './chapter03/Clock';
```

내가 만든 첫 컴포넌트

- Comment 에 CSS 적용



The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left displays the project structure for 'HELLO-APP', including 'node_modules', 'public', and 'src'. The 'src' directory is expanded, showing 'chapter01', 'chapter02', and 'chapter03'. Under 'chapter03', the files 'Comment.jsx' and 'CommentList.jsx' are listed. The 'Comment.jsx' file is selected and its content is displayed in the main editor area. The code defines a 'styles' object for a comment container, including properties for 'wrapper', 'imageContainer', 'image', 'contentContainer', 'nameText', and 'commentText'. The 'wrapper' style includes 'margin', 'padding', 'display', 'flexDirection', 'border', and 'borderRadius'. The 'imageContainer' style includes 'width', 'height', and 'borderRadius'. The 'image' style includes 'width', 'height', and 'borderRadius'. The 'contentContainer' style includes 'marginLeft', 'display', 'flexDirection', and 'justifyContent'. The 'nameText' style includes 'color', 'fontSize', and 'fontWeight'. The 'commentText' style includes 'color' and 'fontSize'.

```
src > chapter03 > Comment.jsx > ...
1  import React from "react";
2
3  const styles = {
4    wrapper: {
5      margin: 8,
6      padding: 8,
7      display: "flex",
8      flexDirection: "row",
9      border: "1px solid grey",
10     borderRadius: 16,
11   },
12   imageContainer: {},
13   image: {
14     width: 50,
15     height: 50,
16     borderRadius: 25,
17   },
18   contentContainer: {
19     marginLeft: 8,
20     display: "flex",
21     flexDirection: "column",
22     justifyContent: "center",
23   },
24   nameText: {
25     color: "black",
26     fontSize: 16,
27     fontWeight: "bold",
28   },
29   commentText: {
30     color: "black",
31     fontSize: 16,
32   },
33 };
34
```

```
35 function Comment(props) {
36   return (
37     <div style={styles.wrapper}>
38       <div style={styles.imageContainer}>
39         
43       </div>
44
45       <div style={styles.contentContainer}>
46         <span style={styles.nameText}>이병일</span>
47         <span style={styles.commentText}>
48           제가 만든 첫 컴포넌트 입니다.
49         </span>
50       </div>
51     </div>
52   );
53 }
54
55 export default Comment;
56
```



이병일

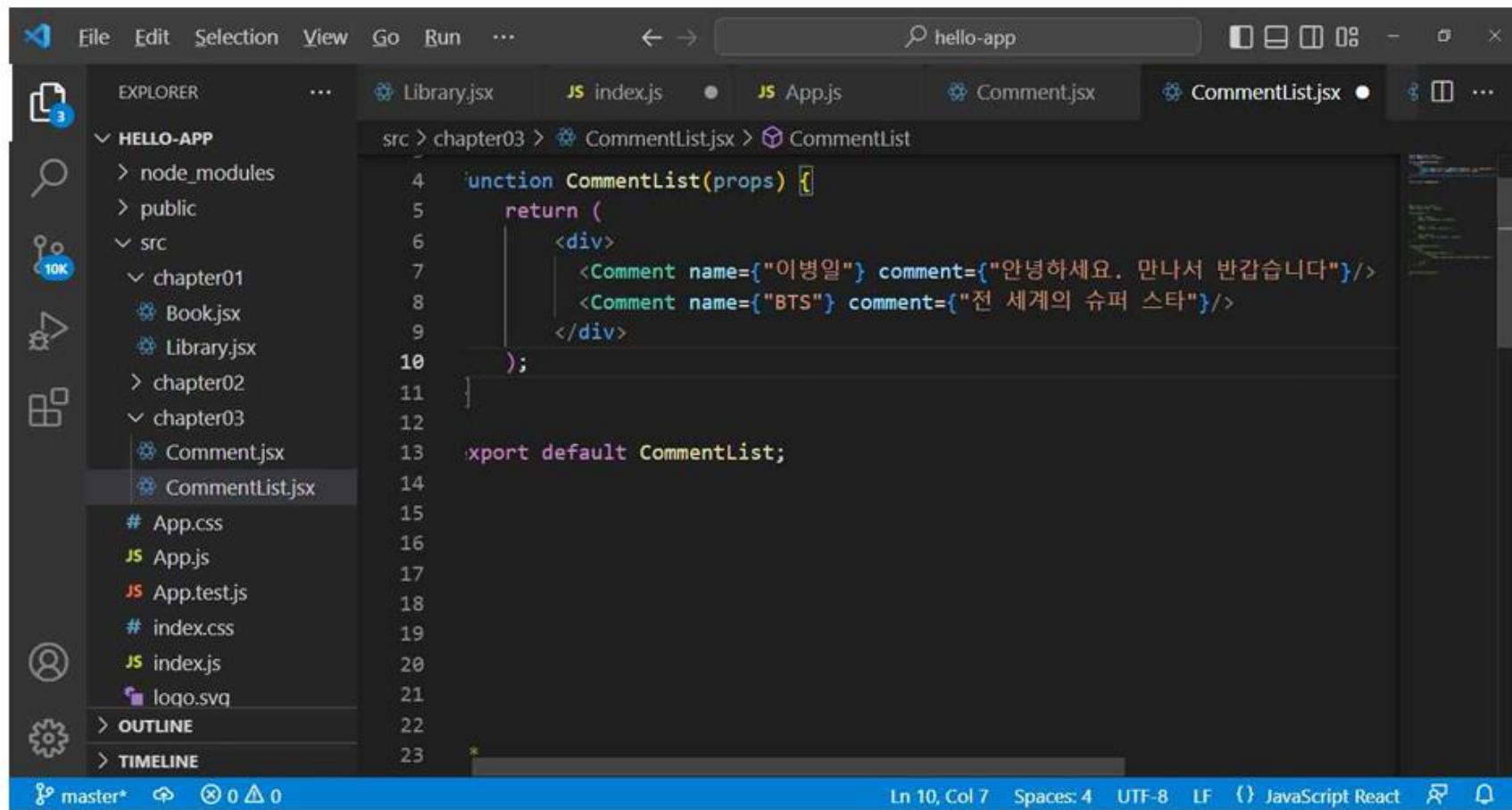
제가 만든 첫 컴포넌트입니다.

- Comment 컴포넌트에 표시되는 작성자 이름과 댓글 내용을 동적으로 변경할 수 있도록 하기 위해 props를 추가.

```
37 function Comment(props) {
38   return (
39     <div style={styles.wrapper}>
40       <div style={styles.imageContainer}>
41         
7        <Comment name={"이병일"} comment={"안녕하세요. 만나서 반갑습니다"}/>
8      </div>
9    );
10 }
11
12 export default CommentList;
```



- 댓글을 하나 더 추가해 보자



```
4 function CommentList(props) {  
5   return (  
6     <div>  
7       <Comment name={"이병일"} comment={"안녕하세요. 만나서 반갑습니다"}/>  
8       <Comment name={"BTS"} comment={"전 세계의 슈퍼 스타"}/>  
9     </div>  
10  );  
11 }  
12  
13 export default CommentList;  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23
```

