

React

◆ JSX 문법

정수아

Contents

01 JSX란?

02 React 동작 원리

03 JSX 문법 작성 규칙



01

JSX란?

JSX란?

❖ JSX(JavaScript XML)

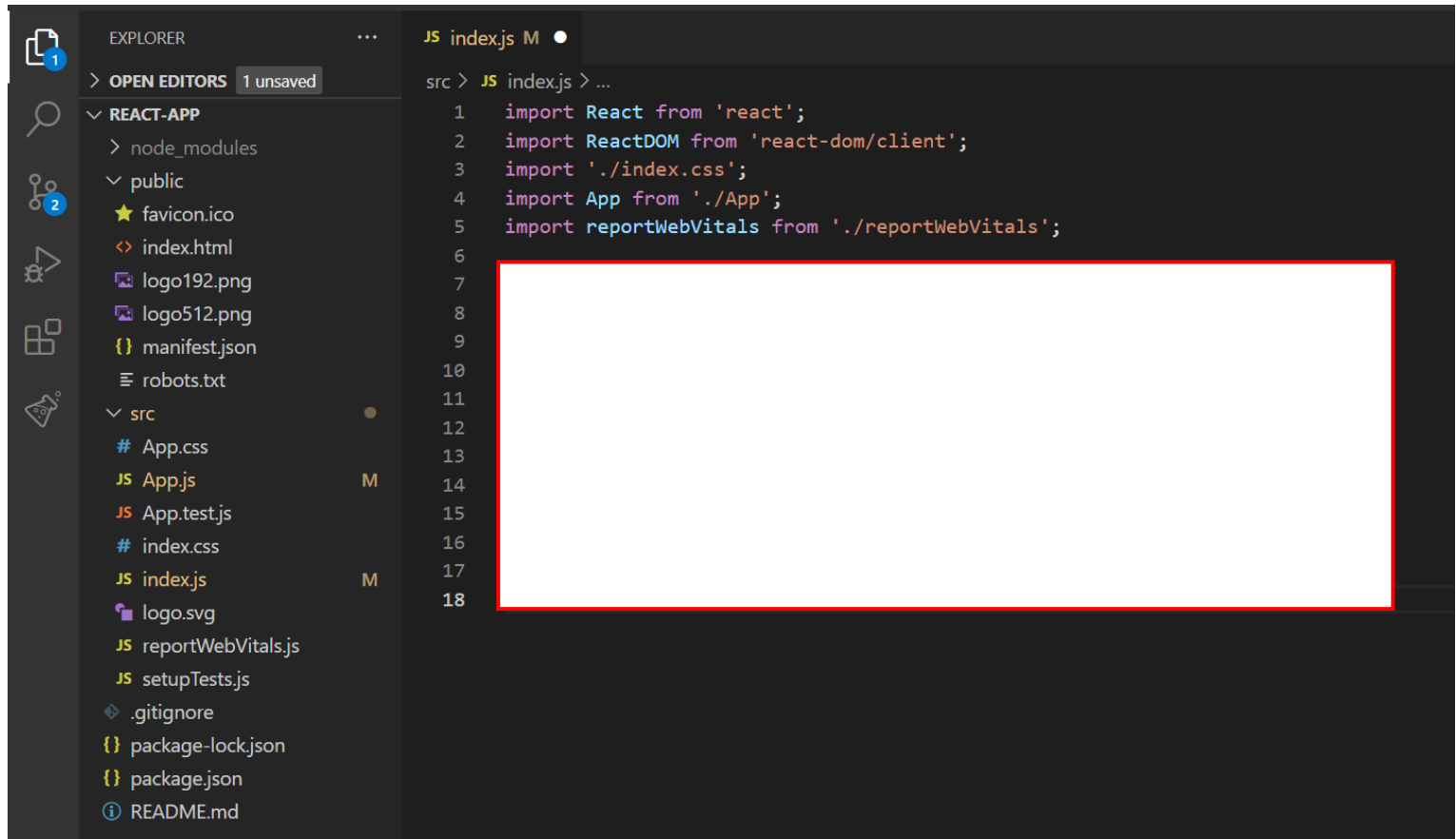
- 자바스크립트에 XML을 추가한 확장형 문법
- 특징
 - 하나의 파일에 자바스크립트와 HTML을 동시에 작성할 수 있어서 편리함
 - 개발자는 화면에만 집중해서 개발 가능
 - 리액트 엔진이 JSX를 내부적으로 자바스크립트 코드로 변환시켜서 작동시킴
- 예시

```
const element = <h1>Hello!</h1>;
```

[실습] JSX를 사용하는 이유 - 자바스크립트

❖ src/index.js 파일 수정

- import 아래 내용 삭제하기



[실습] JSX를 사용하는 이유 - 자바스크립트

❖ 자바스크립트 코드로 내용 작성

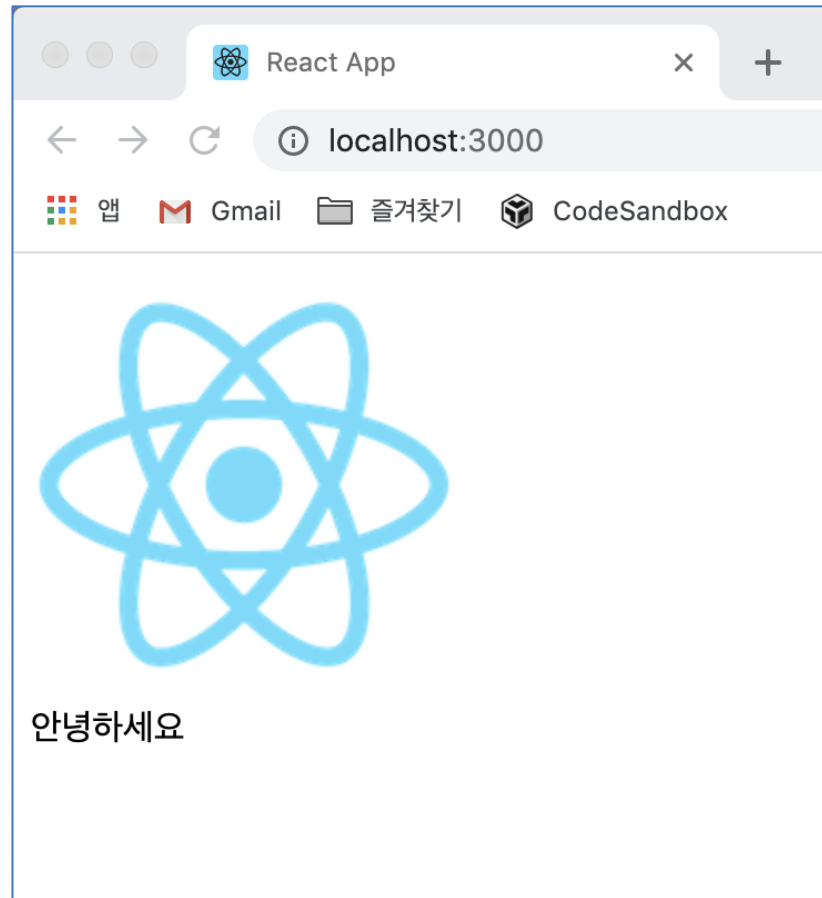
```
var img = document.createElement("img");  
img.setAttribute("src", "logo512.png");  
  
var divChild = document.createElement("div");  
divChild.innerText = "안녕하세요";  
  
var divParent = document.createElement("div");  
divParent.append(img);  
divParent.append(divChild);  
  
var root = document.getElementById("root");  
root.append(divParent);
```

- 실행

```
$ npm start
```

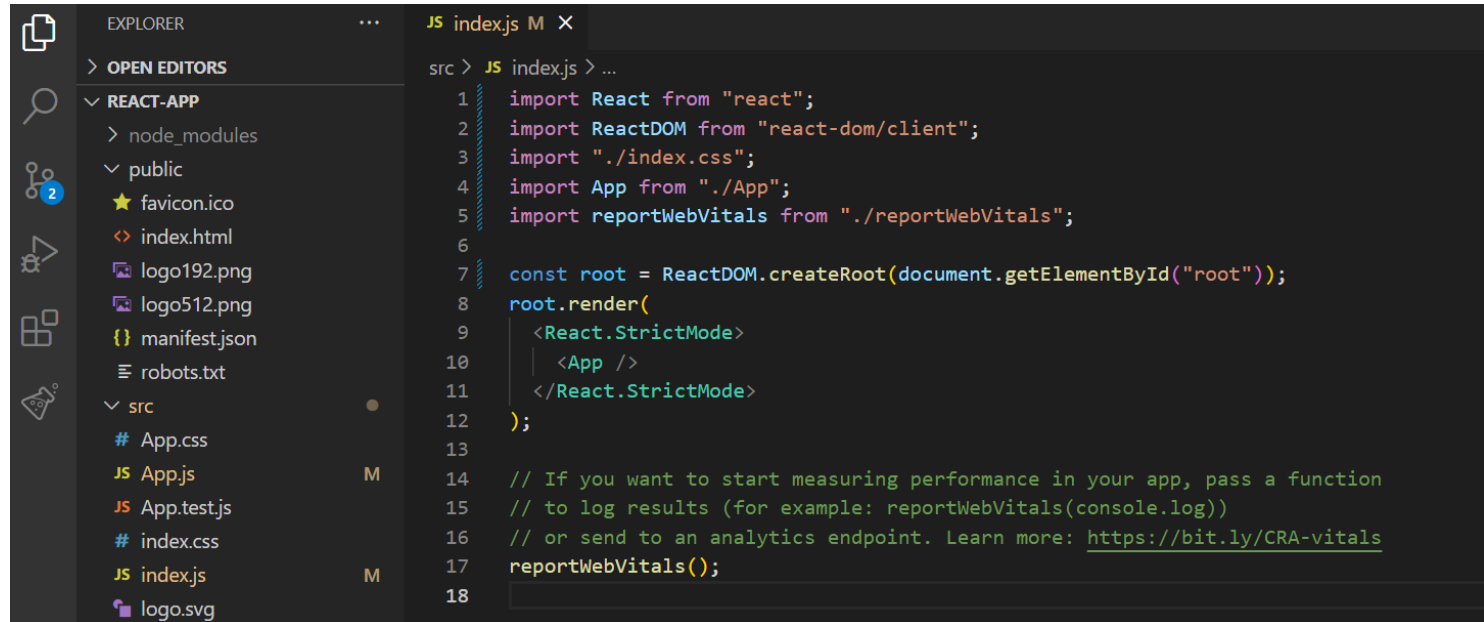
[실습] JSX를 사용하는 이유 - 자바스크립트

❖ 실행 결과



[실습] JSX를 사용하는 이유 - JSX

❖ src/index.js 파일 내용 되돌리기



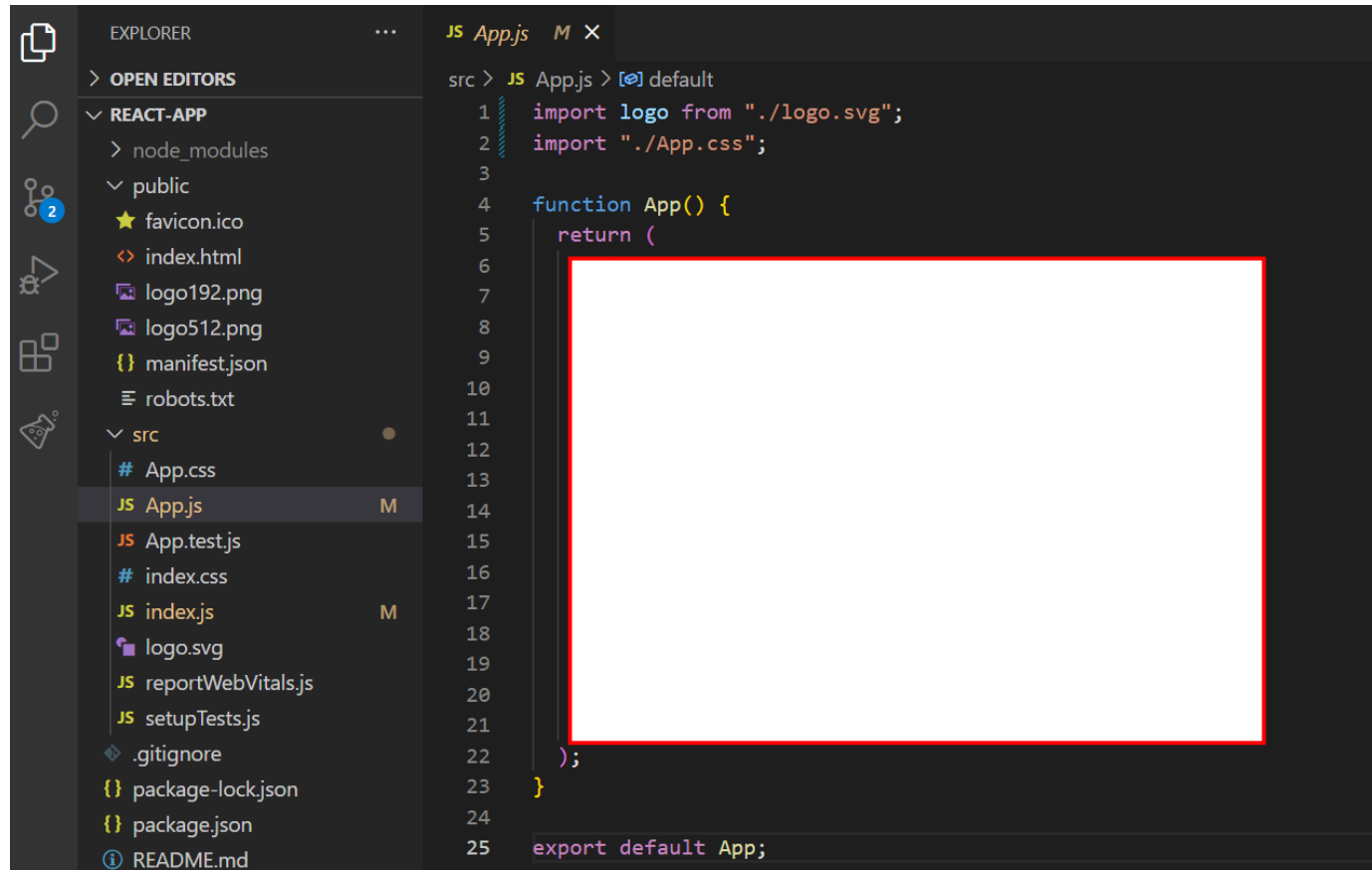
The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure for 'REACT-APP'. The 'src' directory is expanded, showing files like 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', and 'logo.svg'. The 'index.js' file is selected. The main editor area shows the code for 'src > JS index.js'. The code imports React, ReactDOM, and App, and uses ReactDOM.createRoot to render the App component into the root element of the document.

```
src > JS index.js > ...
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 );
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 reportWebVitals();
18
```


[실습] JSX를 사용하는 이유 - JSX

❖ src/App.js 파일 내용 수정

- return() 함수 내용 삭제하기



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left and the Editor pane on the right. The Explorer sidebar shows the project structure for 'REACT-APP', with the 'src' folder expanded. The 'App.js' file is selected. The Editor pane shows the content of 'App.js' with line numbers 1 through 25. A red rectangular box highlights the content of the 'return' statement in the 'App' function, which is currently empty. The code in the editor is as follows:

```
src > JS App.js > default
1  import logo from './logo.svg';
2  import './App.css';
3
4  function App() {
5    return (
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22  );
23 }
24
25 export default App;
```

[실습] JSX를 사용하는 이유 - JSX

❖ return() 함수 내에 JSX 코드로 내용 작성

```
import logo from "./logo.svg";
import "./App.css";

function App() {
  return (
    <div>
      
      <div>안녕하세요</div>
    </div>
  );
}

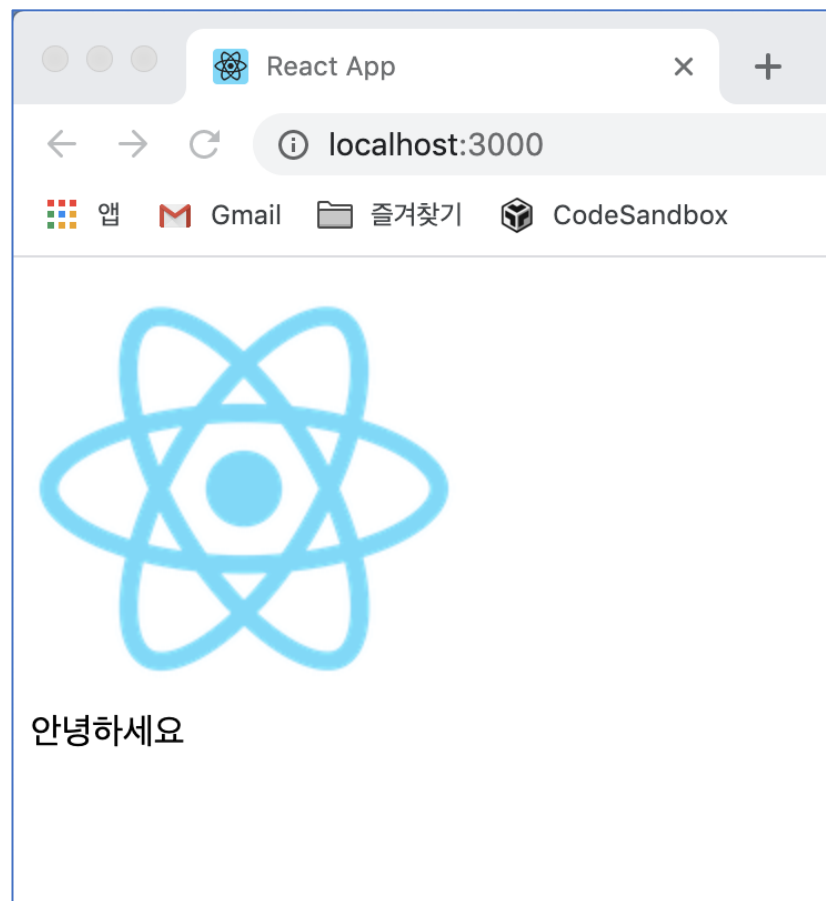
export default App;
```

- 실행

```
$ npm start
```

[실습] JSX를 사용하는 이유 - JSX

❖ 실행 결과



[실습] JSX를 사용하는 이유 - JSX

❖ 개발자 도구로 확인

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

❖ public 폴더 안 index.html 파일 확인

```
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
</body>
```

JSX를 사용하는 이유

❖ 코드량 감소

```
var img = document.createElement("img");  
img.setAttribute("src", "logo512.png");  
  
var divChild = document.createElement("div");  
divChild.innerText = "안녕하세요";  
  
var divParent = document.createElement("div");  
divParent.append(img);  
divParent.append(divChild);  
  
var root = document.getElementById("root");  
root.append(divParent);
```



```
<div>  
    
  <div>안녕하세요</div>  
</div>
```

- 리액트 엔진이 JSX를 내부적으로 자바스크립트 코드로 변환

```
function App() {  
  return (  
    React.createElement("div", null,  
      React.createElement("img", { src: "logo512.png" }),  
      React.createElement("div", null, "안녕하세요")  
    )  
  );  
}
```

JSX를 사용하는 이유

❖ 가독성 향상

- 중첩된 선언형 구조를 잘 나타냄

❖ 팀의 생산성 향상

- HTML과 비슷하여 전문 개발자가 아니더라도 직접 코드 수정 가능

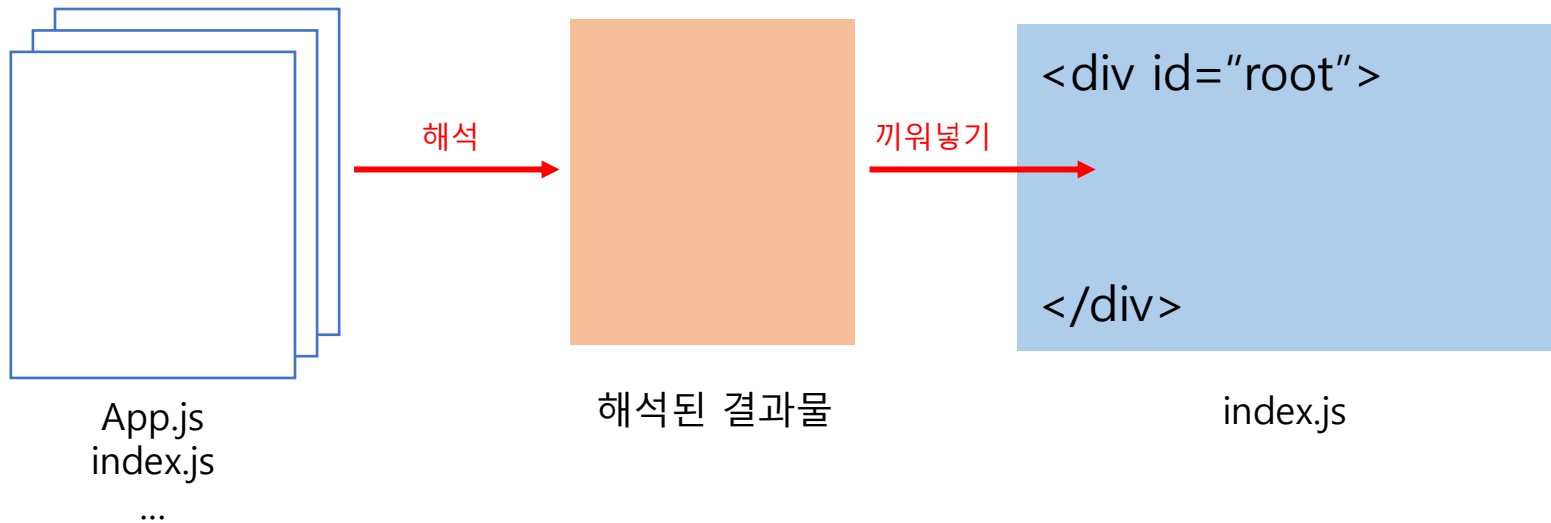


02

React 동작 원리

React 동작 원리

❖ 작성된 코드를 리액트가 자바스크립트를 이용하여 해석



React 동작 원리

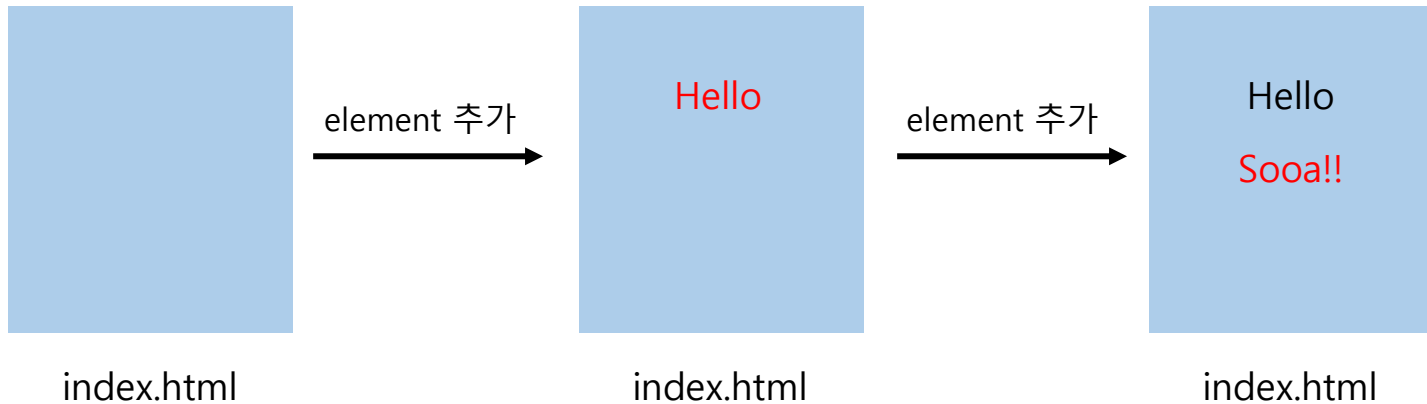
❖ src/index.js 파일

```
const root = ReactDOM.createRoot(document.getElementById("root"));  
  
root.render(그려라  
  <React.StrictMode>  
    <App /> App 컴포넌트를  
  </React.StrictMode>  
);
```

아이디가 'root'인 엘리먼트에

React 동작 원리

❖ 리액트가 화면을 그리는 방식



- 추가 및 제거되는 element만 업데이트



02

JSX 문법 작성 규칙

JSX 문법 작성 규칙

❖ 반드시 하나의 부모 태그로 감싸는 형태여야 함

- 에러 발생 코드

```
function App() {  
  return (  
    <div>안녕하세요.</div>  
    <div>정수아입니다.</div>  
  );  
}
```

- 정상 작동 코드

```
function App() {  
  return (  
    <div>  
      <div>안녕하세요.</div>  
      <div>정수아입니다.</div>  
    </div>  
  );  
}
```

JSX 문법 작성 규칙

❖ 반드시 하나의 부모 태그로 감싸는 형태여야 함

- 의미 없는 태그인 경우, `<></>` 이렇게 작성 가능

```
function App() {  
  return (  
    <>  
      <div>안녕하세요.</div>  
      <div>정수아입니다.</div>  
    </>  
  );  
}
```

JSX 문법 작성 규칙

❖ 자바스크립트 표현식은 {} 괄호로 감싸는 형태여야 함

```
function App() {  
  const name = 'Sooa'  
  return (  
    <div>  
      <div>안녕하세요.</div>  
      <div>{name}</div>  
    </div>  
  );  
}
```

JSX 문법 작성 규칙

❖ if, for와 같은 구문은 사용할 수 없음

- 자바스크립트 표현식이 아니기 때문
- 방법 1) 조건부 연산자(삼항 연산자)를 사용

```
function App() {  
  const name = 'Sooa'  
  return (  
    <div>  
      { name === 'Sooa' ? (<h1>선생님</h1>) : (<h1>학생</h1>) }  
    </div>  
  );  
}
```

JSX 문법 작성 규칙

- 방법 2) AND 연산자(&&)를 사용
 - 특정 조건을 만족할 때 내용을 보여주고, 만족하지 않을 때 아무것도 렌더링하지 않아야 하는 경우에 사용

```
function App() {  
  const name = 'Sooa'  
  return (  
    <div>  
      { name === 'Sooa' ? (<h1>선생님</h1>) : null }  
    </div>  
  );  
}
```

```
function App() {  
  const name = 'Sooa'  
  return (  
    <div>  
      { name === 'Sooa' && <h1>선생님</h1> }  
    </div>  
  );  
}
```


JSX 문법 작성 규칙

❖ HTML 태그는 반드시 닫아야 함

- XML 마크업 규칙을 따르기 때문
- HTML의 경우 다음 태그들은 닫지 않아도 됨
 - ,
, <input> 등
 - JSX의 경우 태그를 닫지 않으면 에러 발생

```
function App() {  
  return (  
    <div>  
      <div>안녕하세요.</div>  
        
    </div>  
  );  
}
```

JSX 문법 작성 규칙

❖ class 대신 className을 사용

- class는 자바스크립트의 예약어이기 때문에 사용할 수 없음
- JSX에서는 className을 사용해야 함

```
function App() {  
  const name = 'Sooa'  
  return (  
    <div className="myclass">  
      <div>안녕하세요.</div>  
      <div>{name}</div>  
    </div>  
  );  
}
```

JSX 문법 작성 규칙

❖ CSS Style

- JSX에서 태그에 style을 적용할 때는 객체 형태로 작성
- 속성 이름은 카멜 표기법으로 작성

```
function App() {  
  const name = 'Sooa'  
  const style = {  
    backgroundColor : 'red',  
    fontSize : '12px'  
  }  
  return (  
    <div style={style}>  
      <div>안녕하세요.</div>  
      <div>{name}</div>  
    </div>  
  );  
}
```

JSX 문법 작성 규칙

❖ 주석

- JSX 내에서는 `{/*...*/}`와 같은 형식으로 사용

```
function App() {  
  const name = 'Sooa'  
  return (  
    <div>  
      {/* 주석 사용 방법 */}  
      <div>안녕하세요.</div>  
      <div>{name}</div>  
    </div>  
  );  
}
```

THANK 😊 YOU