

리액트(React)

JSX와 리액트 컴포넌트 기초

김경민



index.html

MYAPP

> node_modules

> public

✓ src

> assets

App.css

App.jsx

index.css

main.jsx

.env

.gitignore

eslint.config.js

<> index.html

package-lock.json

package.json

README.md

vite.config.js

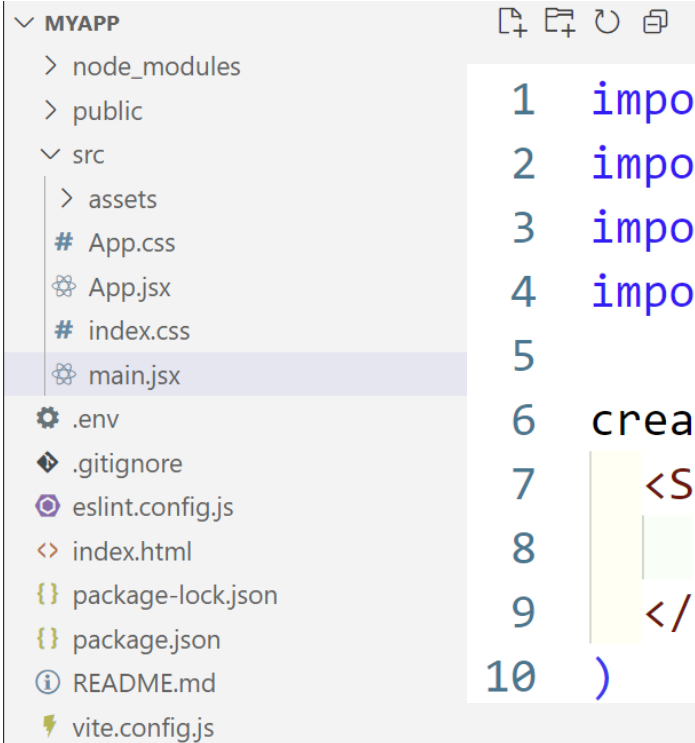
```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <link rel="icon" type="image/x-icon" href="/favicon.ico">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Vite + React</title>
8   </head>
9   <body>
10    <div id="root"></div>
11    <script type="module" src="/src/main.jsx"></script>
12  </body>
13 </html>
```

• 웹 애플리케이션의 진입점 (entry point)

- 마운트 포인트가 되는 DOM 요소인 `<div id="root"></div>`를 제공
- 리액트 애플리케이션의 모든 컴포넌트는 이 `<div>` 안에 렌더링
 - 렌더링된 결과물이 바로 SPA에서 말하는 단 1개의 페이지



src > main.jsx



```
1 import { StrictMode } from 'react'
2 import { createRoot } from 'react-dom/client'
3 import './index.css'
4 import App from './App.jsx'
5
6 createRoot(document.getElementById('root')).render(
7   <StrictMode>
8     <App />
9   </StrictMode>,
10 )
```

• React 애플리케이션의 진입점(entry point) 역할

- React 애플리케이션을 DOM에 렌더링
- 최상위 컴포넌트(App)를 root에 연결
- React.StrictMode 적용 (선택 사항)
- CSS 및 글로벌 스타일 임포트



src > App.jsx

MYAPP
node_modules
public
src
assets
App.css
App.jsx

- 애플리케이션의 최상위 컴포넌트(root component)
 - 전체 애플리케이션의 UI 구조를 정의하고, 다른 컴포넌트들을 포함하여 화면을 구성

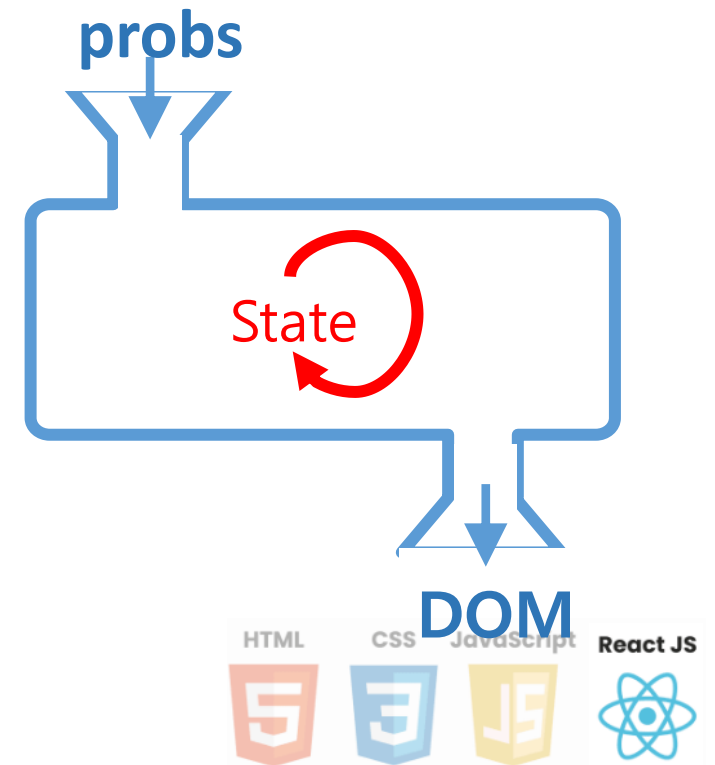
eslint.config.js
index.html
package-lock.json
package.json
README.md
vite.config.js

```
1 import reactLogo from './assets/react.svg'
2 import viteLogo from '/vite.svg'
3 import './App.css'
4 import { FaHome } from "react-icons/fa";
5
6 function App() {
7   return (
8     <div className="w-full h-full flex flex-col">
9       <div className="w-full flex justify-center items-center">
10        <a href="https://vite.dev" target="_blank">
11          <img src={viteLogo} className="logo" alt="Vite logo" />
12        </a>
13        <a href="https://react.dev" target="_blank">
14          <img src={reactLogo} className="logo react" alt="React logo" />
15        </a>
16      </div>
17      <h1 className="font-bold text-4xl">Vite + React</h1>
18      <p className="read-the-docs">
19        Click on the Vite and React logos to learn more
20      </p>
21      <p className="w-full flex justify-center text-4xl">
22        <FaHome />
23      </p>
24    </div>
25  )
26 }
27
28 export default App
```



컴포넌트(Component)

- UI를 재사용 가능한 개별적인 조각으로 사용자 정의 태그 생성
 - props라고 하는 임의의 입력을 받은 후, 화면에 어떻게 표시되는지를 기술하는 React 엘리먼트를 반환
 - 컴포넌트는 반드시 하나의 요소를 반환
 - 여러 요소가 있다면 **프래그먼트로 감싸서 반환**(`<>...</>`)
 - 컴포넌트명은 반드시 **대문자로 시작**해야 함
 - JSX 문법으로 작성



src > 01 > Hello.js

src > 01 > Hello.jsx > ...

```
1 export default function Hello() {  
2   return (  
3     <div>Hello React!!</div>  
4   )  
5 }
```

Hello.jsx

```
1 import reactLogo from './assets/react.svg'  
2 import viteLogo from '/vite.svg'  
3 import './App.css'  
4 import Hello from './01/Hello'  
5  
6 function App() {  
7   return (  
8     <div className="w-full h-full flex flex-col">  
9 > <div className="w-full flex justify-center items-center">...  
16 </div>  
17 <Hello />  
18 </div>  
19 )  
20 }  
21  
22 export default App
```

App.jsx

• 함수형 컴포넌트

- 자바스크립트 함수로 작성
- return문에 JSX 코드를 작성하여 반환



JSX(JavaScript XML)

- JavaScript의 확장 문법

- React에서 UI를 작성할 때 사용하는 JavaScript 확장 문법
- HTML과 유사한 문법으로 React 컴포넌트를 쉽게 작성
- JSX 주요 특징
 - JavaScript 안에서 HTML을 작성할 수 있음
 - Babel을 통해 JavaScript로 변환됨
 - XML과 유사한 문법을 사용하지만, JavaScript 표현식을 포함할 수 있음
 - Virtual DOM을 활용하여 렌더링 최적화 가능



JSX(JavaScript XML) 기본 문법

- 단일 루트 요소 반환

- 여러 개의 요소를 반환하고 싶다면 fragments라 불리는 `<></>` 를 사용

- 자바스크립트 표현식

- 중괄호(`{}`) 내에 위치

- 자바스크립트 예약어와 같은 속성명을 사용할 수 없음

- class 속성은 `className`, for 속성은 `htmlFor`

- 스타일 적용

- 스타일 이름을 **카멜표기법**으로 사용
 - `background-color => backgroundColor`

- 스타일은 오브젝트로 선언하여 표현식으로 작성하거나 인라인 스타일은 `{{}}`안에 작성



JSX(JavaScript XML) 기본 문법

- 반드시 종료 태그 작성

- 예) <Hello />

- 조건부 렌더링

- 삼항연산자를 이용하여 조건부 렌더링

- &&(AND)연산자를 이용한 조건부 렌더링

- 특정조건을 만족할때만 내용을 보여주고, 만족하지 않을 때는 렌더링 하지 않는 경우

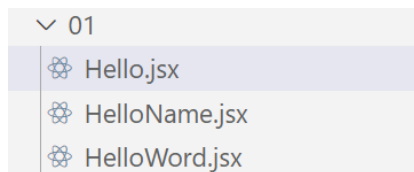
- ||(OR)연산자를 이용한 조건부 렌더링

- 컴포넌트내부에서 undefined만 반환하면 오류가 발생하므로 OR연산자를 사용하여 해당값이 undefind인 경우일때 사용할 값을 지정하여 오류를 방지



JSX 특징

- 컴포넌트는 반드시 부모 요소 하나만 반환
 - 리액트 Virtual DOM에서 변화를 효율적으로 감지
 - `<Fragment></Fragment>`, `<></>`



```
1 import HelloWorld from './HelloWord'
2 import HelloName from './HelloName'
3 export default function Hello() {
4   return (
5     <>
6       <HelloWord />
7       <HelloName />
8     </>
9   )
10 }
```

```
src > 01 > @ HelloWorld.jsx > ...
1 export default function HelloWorld() {
2   return (
3     <div>Hello </div>
4   )
5 }
```

```
src > 01 > @ HelloName.jsx > ...
1 export default function HelloName() {
2   return (
3     <div>React!!</div>
4   )
5 }
```

```
<div id="root">
  <div class="w-full h-full flex flex-co
    <div class="w-full flex justify-cente
      <div>Hello </div>
      <div>React!!</div>
    </div>
  </div>
```



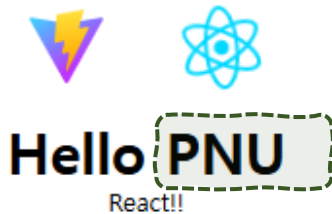
JSX 특징

• 자바스크립 표현식

- {}에 작성
- undefined를 반환하지 않도록 처리

src > 01 > HelloWord.jsx > ...

```
1 export default function HelloWord() {  
2   const name = 'PNU';  
3   return (  
4     <div className="text-4xl font-bold">  
5       Hello {name}  
6     </div>  
7   )  
8 }
```



src > 01 > HelloWord.jsx > ...

```
1 export default function HelloWord() {  
2   // const name = 'PNU';  
3   const name = undefined;  
4   return (  
5     <div className="text-4xl font-bold">  
6       Hello {name || '이름이 정의되지 않았습니다.'}  
7     </div>  
8   )  
9 }
```

Hello 이름이 정의되지 않았습니다.



src > 01 > HelloWord.jsx > ...

```
1 export default function HelloWord() {  
2   // const name = 'PNU';  
3   const name = undefined;  
4   return (  
5     <div className="text-4xl font-bold">  
6       Hello {name && '이름이 정의되지 않았습니다.'}  
7     </div>  
8   )  
9 }
```

falsy 값으로 ||, && 연산
- false 값으로 간주되는 값
false, 0, -0, "",
null, undefined, NaN

HTML



Hello
React!!



JS




JSX 특징



• 조건부 랜더링

- JSX 내부에는 조건문 사용할 수 없어 삼항 연산자 사용

```
1 export default function HelloWord() {  
2   const name = 'PNU' ;  
3   // const name = undefined;  
4   return (  
5     <div className="text-4xl font-bold">  
6       {  
7         name ? `${name}님 반갑습니다.` : 'Hello'  
8       }  
9     </div>  
10  )  
11 }
```

 
PNU님 반갑습니다.
React!!

```
1 export default function HelloWord() {  
2   // const name = 'PNU' ;  
3   const name = undefined;  
4   return (  
5     <div className="text-4xl font-bold">  
6       {  
7         name ? `${name}님 반갑습니다.` : 'Hello'  
8       }  
9     </div>  
10  )  
11 }
```

 
Hello
React!!

ipt React JS




JSX 특징



• 조건부 랜더링

- JSX 내부에는 조건문 사용할 수 없어 삼항 연산자 사용

```
1 export default function HelloWord() {  
2   const name = 'PNU' ;  
3   // const name = undefined;  
4   return (  
5     <div className="text-4xl font-bold">  
6       {  
7         name ? `${name}님 반갑습니다.` : 'Hello'  
8       }  
9     </div>  
10  )  
11 }
```

 
PNU님 반갑습니다.
React!!

```
1 export default function HelloWord() {  
2   // const name = 'PNU' ;  
3   const name = undefined;  
4   return (  
5     <div className="text-4xl font-bold">  
6       {  
7         name ? `${name}님 반갑습니다.` : 'Hello'  
8       }  
9     </div>  
10  )  
11 }
```

 
Hello
React!!

ipt React JS



JSX 특징

• 스타일 적용

- 스타일 이름을 **카멜표기법**으로 사용
- 오브젝트로 선언

src > 01 > HelloDate.jsx > ...

```
1 export default function HelloDate() {  
2   const divStyle = {  
3     backgroundColor: "black",  
4     color : "white",  
5     padding : "10px"  
6   }  
7   return (  
8     <div style={divStyle}>  
9       현재시간 :  
10      <span style={{color : "yellow", fontWeight: "bold"}}>  
11        {new Date().toLocaleTimeString()}  
12      </span>  
13    </div>  
14  )  
15 }
```

스타일을 오브젝트로 정의

스타일을 표현식으로 오브젝트로 정의



Hello

React!!

현재시간 : 오전 1:31:01



해결문제

- 다음 그림과 같이 컴포넌트를 분리하여 작성

