04_JSX

JSX

JSX란 JavaScript 확장 문법으로, React에서 UI를 작성할 때 사용됩니다.

React에서 JSX를 사용하면, 더욱 직관적이고 가독성이 좋은 코드를 작성할 수 있습니다.

또한, React 컴포넌트 내에서 HTML과 JavaScript 코드를 함께 작성할 수 있어서, 코드의 유지보수가 용이해집니다.

React_JSX는 React에서 UI를 작성하는 데 필수적인 기술 중 하나이며, React를 학습하고 개발하는 데 있어서 중요한 개념입니다.

JSX의 기본 작성규칙

JSX에서는 HTML 태그와 함께 JavaScript 표현식을 함께 사용할 수 있습니다.

이를 위해서는 JSX에서는 JavaScript 표현식을 중괄호({ })로 감싸주어야 합니다.

```
const name = "React";
const element = <h1>Hello, {name}!</h1>;
```

위 코드에서는, `name`이라는 변수를 선언하고, `element`라는 변수에 HTML 태그와 함께 중괄호를 사용하여 `name` 변수를 출력하도록 했습니다. 이렇게 하면, `element` 변수가 HTML 태그와 함께 출력되는 동시에, `{name}` 부분은 `React`라는 문자열로 대체되어 출력됩니다.

JSX에서는 HTML 태그의 속성값에도 JavaScript 표현식을 사용할 수 있습니다. 이때, 속성값에 JavaScript 표현식을 사용할 경우, 속성값을 중괄호로 감싸주어야 합니다.

```
const element = <img src={user.avatarUrl} />;
```

위 코드에서는, `img` 태그의 `src` 속성값에 `user.avatarUrl` 변수를 사용하도록 했습니다. 이때, `src` 속성값을 중괄호로 감싸주어야만 한다는 것에 주의해야 합니다.

```
const element = <div className="container">Hello, World!</div>;
```

위 코드에서는, `div` 태그를 사용하여 `Hello, World!`라는 텍스트를 출력하도록 했습니다. 이때, `div` 태그의 속성으로 `className`을 사용하여 CSS 클래스를 지정하였습니다.

JSX 작성 방법 응용편

JSX에서는 다양한 방법으로 JavaScript 표현식을 사용할 수 있습니다.

이번에는 JSX 작성 방법을 응용하여 다양한 UI 요소를 작성해보겠습니다.

1. 조건문과 반복문 사용하기

JSX에서는 JavaScript의 조건문과 반복문을 사용하여 동적인 UI를 만들 수 있습니다.

src > exam > Exam6.js

- 조건문

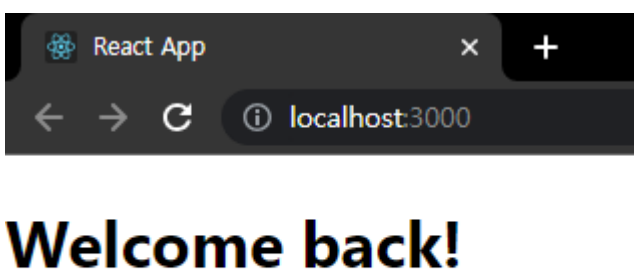
```
function Exam6_1(props) {
    const isLogin = props.isLogin;
```

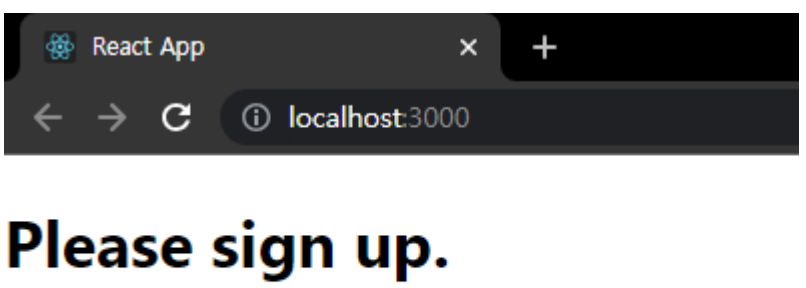
```
    if (isLogin) {
        return <h1>Welcome back!</h1>;
    } else {
        return <h1>Please sign up.</h1>;
    }
}

export default Exam6_1;
```

```
// App.js
import Exam6_1 from './exam/Exam6';

function App() {
  return (
    <div className="App">
      <Exam6_1 isLogin={true}/>
    </div>
  );
}

export default App;
```

위 코드에서는, Exam6이라는 컴포넌트를 정의하고, isLogin 이라는 props를 전달하도록 했습니다.

Exam6 컴포넌트 내에서는 isLogin값이 `true`인 경우, Welcome back!이라는 텍스트가 출력되고, isLogin 값이 `false`인 경우, Please sign up.이라는 텍스트가 출력됩니다.



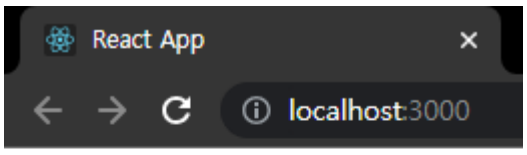


- 반복문

```
...
export function Exam6_2() {
    const numbers = [1, 2, 3, 4, 5];
    const listItems = numbers.map((number) => <li>{number}</li>);

    return <ul>{listItems}</ul>
}
```

```
// App.js
import Exam6_2 from './exam/Exam6';

function App() {
  return (
    <div className="App">
      {/* <Exam6_1 isLogin={true}/> */}
```

```
      <Exam6_2 />
    </div>
  );
}

export default App;
```



- 1
- 2
- 3
- 4
- 5

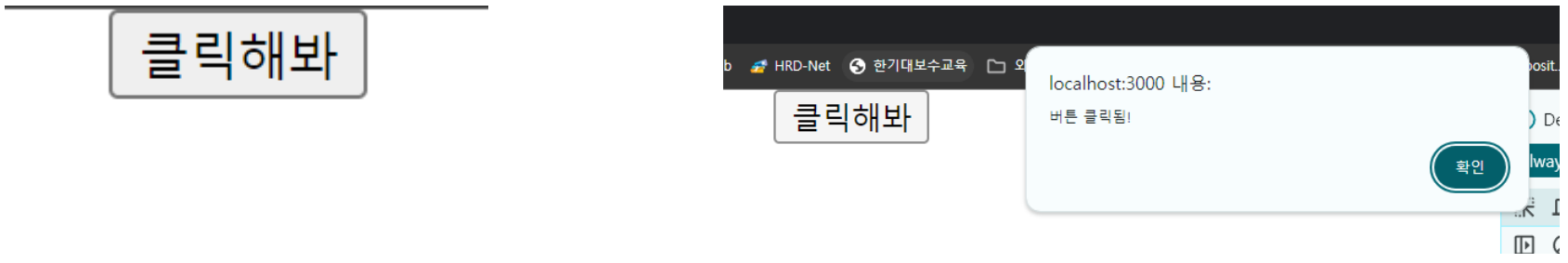
위 코드에서는, `numbers`라는 배열을 정의하고, `map` 함수를 사용하여 `listItems`라는 배열을 생성하였습니다. `listItems` 배열을 `ul` 태그 내부에 출력하면, `1`, `2`, `3`, `4`, `5`라는 숫자 리스트가 출력됩니다.

2. 이벤트 처리하기

JSX에서는 HTML 태그와 마찬가지로 이벤트를 처리할 수 있습니다.

```
...
export function Exam6_3(props) {
    const handleClick = () => {
      alert('버튼 클릭됨!');
    }

    return (
      <button onClick={handleClick}>
        {props.label}
      </button>
    );
}
```

```
// App.js
import Exam6_1, { Exam6_2, Exam6_3 } from './exam/Exam6';

function App() {
  return (
    <div className="App">
       {/* <Exam6_1 isLogin={true}/> */}
      {/* <Exam6_2 /> */}
      <Exam6_3 label="클릭해봐"/>
    </div>
  );
}

export default App;
```



주의사항

React에서 함수형 컴포넌트를 작성할 때, **컴포넌트가 반환하는 JSX 요소는 단 하나의 최상위 요소로 감싸져 있어야 합니다.** 이는 JSX에서 여러 요소를 반환할 경우 발생할 수 있는 문제를 방지하기 위함입니다.

이유는 React가 컴포넌트를 렌더링할 때, 컴포넌트가 반환하는 JSX 요소가 하나의 최상위 요소로 감싸져 있지 않으면, 렌더링이 실패할 수 있기 때문입니다. 따라서, **여러 요소를 반환해야 하는 경우에는, 하나의 최상위 요소로 감싸주어야 합니다.**

```
function App() {
  return (
    <div>
      <h1>Hello, World!</h1>
      <p>Welcome to my app.</p>
    </div>
  );
}
```

위 코드에서는, `div` 태그를 사용하여 `h1` 태그와 `p` 태그를 감싸주었습니다. 이렇게 하면, 함수형 컴포넌트가 반환하는 JSX 요소가 하나의 최상위 요소로 감싸져 있으므로, 렌더링이 정상적으로 수행됩니다.