📌 JSX(JavaScript XML)

🚀 1. JSX란?

📖 개념

- **JSX(JavaScript XML)** 는 JavaScript에서 XML 문법을 사용할 수 있도록 만든 확장 문법입니다.
- React에서 UI를 정의할 때 사용되며, HTML과 유사한 문법으로 컴포넌트를 쉽게 작성할 수 있습니다.
- Babel을 통해 JavaScript 코드로 변환되어 실행됩니다.

✅ JSX의 주요 특징

1. **HTML과 유사한 문법** → JavaScript 코드 안에서 직관적으로 UI를 정의 가능
2. **컴포넌트 기반 구조와 결합 가능** → React의 컴포넌트 개념을 효율적으로 활용
3. **보안성이 뛰어남** → XSS(크로스 사이트 스크립팅) 공격 방지
4. **Virtual DOM과 연계하여 성능 최적화** → 최소한의 변경 사항만 실제 DOM에 반영됨

💡 2. 왜 JSX를 사용하는가?

✅ JSX의 장점

1. **가독성 향상**: HTML과 유사한 문법을 사용하여 코드의 가독성이 뛰어남
2. **컴포넌트 기반 개발 지원**: UI를 작은 컴포넌트로 나누어 재사용 가능
3. **JavaScript와 강력한 결합**: JavaScript 변수, 함수 등을 JSX 내부에서 쉽게 사용 가능
4. **Virtual DOM 최적화**: JSX는 React의 Virtual DOM을 최적화하여 빠른 렌더링 가능

❌ JSX의 단점

1. **추가적인 컴파일 필요**: Babel을 통해 변환이 필요하여 실행 속도가 약간 느려질 수 있음
2. **JavaScript와 완전히 동일하지 않음**: 특정 문법(JSX 규칙)에 따라 작성해야 함
3. **브라우저에서 직접 실행 불가**: JSX는 브라우저에서 바로 실행되지 않고 JavaScript 코드로 변환 후 실행됨

🛠 3. JSX 문법 기본

📌 1) JSX 기본 문법

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

- JSX 문법을 사용하여 UI를 정의할 수 있으며, **HTML과 유사한 문법**을 가짐

📌 2) JavaScript 표현식 사용

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

- `{}`를 사용하여 JavaScript 표현식을 JSX 내부에서 사용할 수 있음

📌 3) 속성(Attribute) 사용

```
const image = <img src="logo.png" alt="React Logo" />;
```

- JSX에서 속성을 설정할 때 **HTML과 유사하지만 camelCase 문법을 사용** (`class → className`)

📌 4) 스타일 적용

```
const style = {
  color: "blue",
  fontSize: "20px"
};

const element = <h1 style={style}>Styled Text</h1>;
```

- JSX에서 스타일을 적용할 때는 **객체 형태**로 정의해야 함

⚡ 4. JSX의 주요 특징

📌 1) 단일 부모 요소가 필요함

```
// 오류 발생 (두 개의 요소가 병렬로 존재)
return (
  <h1>Hello</h1>
  <p>JSX</p>
);
```

```
// 올바른 방식 (하나의 부모 요소로 감싸기)
return (
  <div>
    <h1>Hello</h1>
    <p>JSX</p>
  </div>
);
```

- JSX에서는 하나의 부모 요소로 감싸야 오류가 발생하지 않음

📌 2) 프래그먼트(Fragments) 사용 가능

```
return (
  <>
    <h1>Hello</h1>
    <p>JSX</p>
  </>
);
```

- `<>...</>`을 사용하면 불필요한 `<div>` 태그 없이 여러 요소를 반환 가능

📌 3) if 문 대신 삼항 연산자 사용

```
const isLoggedIn = true;
return (
  <div>
    {isLoggedIn ? <h1>Welcome back!</h1> : <h1>Please sign in</h1>}
  </div>
);
```

- JSX 내부에서는 `if` 문을 직접 사용할 수 없고, **삼항 연산자**를 사용해야 함

🔥 5. JSX와 React 컴포넌트

📌 1) 함수형 컴포넌트에서 JSX 사용

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

- JSX는 **React 컴포넌트의 반환값**으로 활용될 수 있음

📌 2) 클래스형 컴포넌트에서 JSX 사용

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

- 클래스형 컴포넌트에서도 JSX를 사용하여 UI를 정의 가능

🎯 6. 적용

✅ **JSX의 기본 문법을 이해하고 HTML과의 차이점을 숙지**

✅ **JavaScript 표현식을 JSX 내부에서 올바르게 사용하는 방법 익히기**

✅ **스타일 적용 시 객체 형태 사용 및 camelCase 속성 적용하기**

✅ **React 컴포넌트에서 JSX를 활용하여 동적 UI 생성**

✅ **JSX에서 조건부 렌더링 시 삼항 연산자 활용하기**

🏁 7. 마무리

- JSX는 React에서 UI를 쉽게 정의할 수 있도록 도와주는 **JavaScript 확장 문법**
- HTML과 유사하지만, **JavaScript 표현식을 포함할 수 있고 camelCase 속성을 사용**하는 점이 다름
- 실무에서는 JSX를 활용하여 **컴포넌트 기반 개발을 효과적으로 수행**하며, React의 성능을 극대화 가능