JSX

▶ JSX(Javascript XML)

```
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

자바스크립트 + XML 을 추가한 확장형 문법

브라우저에서 실행되기 전 바벨(transpilers)을 통해 자바스크립트 형태로 변환됨

옆의 코드는 App 컴포넌트를 만드는 코드

HTML도 아니고 문자열 탬플릿도 아님

이런 코드를 JSX라고 부른다

JSX는 자바스트립트의 확장 문법이며 XML과 매우 비슷하다

장점

- 보기 쉽고 익숙하다
- 활용도가 높다

▶ JSX (JavaScript XML) 문법

1. 반드시 부모 요소 하나로 감싸야 한다
2. return되는 JSX는 하나의 요소로 감싸져야 함
3. JSX 안에서 자바스크립트 표현식을 쓸 수 있다
4. return문 안에서 IF문 대신 조건부 연산자 사용

   JSX 내부의 자바스크립트에서 if문을 사용할 수 없다(즉, retur문 안에서)

   대신 { }안에 조건부 연산자를 사용(삼항 연산자)
5. 속성 name class 대신 className으로 사용
6. 항상 닫는태그가 있어야한다.
7. 주석 {/* 주석 */}

▶ JSX (JavaScript XML) 문법

1. 반드시 부모 요소 하나로 감싸야 한다

```
function App() {
  return (
    <h1>REACT</h1>
    <h3>리액트 문법</h3>
  );
}

export default App;
```

[오류]

```
function App() {
  return (
    <div>
      <h1>REACT</h1>
      <h3>리액트 문법</h3>
    </div>
  );
}

export default App;
```

[react 모든 버전 사용]

```
import { Fragment } from 'react/jsx-runtime';

function App() {
  return (
    <Fragment>
      <h1>REACT</h1>
      <h3>리액트 문법</h3>
    </ Fragment >
  );
}

export default App;
```

[react v16이상부터 도입]

```
function App() {
  return (
    <>
      <h1>REACT</h1>
      <h3>리액트 문법</h3>
    </>
  );
}

export default App;
```

[Fragment 생략 가능]

▶ JSX (JavaScript XML) 문법

2. return되는 JSX는 하나의 요소로 감싸져야 함(한줄 일 때는 상관없음)

```
import React from 'react';

function App() {
  const name = ' 리액트';
  return <h1> { name } </h1>
}
export default App;
```

```
import React from 'react';

function App() {
  const name = ' 리액트';
  return (
     <div>
         <h1> { name } </h1>
         <h2>기초 실습</h2>
    </div>
  );
}
export default App;
```

▶ JSX (JavaScript XML) 문법

3. JSX 안에서 자바스크립트 표현식을 쓸 수 있다

```
import React from 'react';

function App() {
  const name = “REACT”;
  return (
    <>
      <h1>{ name }</h1>
      <h3>리액트 문법</h3>
    </>
  );
}

export default App;
```

[react 모든 버전 사용]

▶ JSX (JavaScript XML) 문법

4. return안에서는 IF문 대신 조건부 연산자 사용(삼항 연산자)

```
import React from 'react';

function App() {
  const name = 'REACT';
  return (
    <div>
        { name === 'REACT' ? <h1>REACT이다</h1>  : <h1>REACT가 아니다.</h1> }
   </div>
  );
}

export default App;
```

```
{ name === 'REACT' ? <h1>REACT이다</h1>  : <h1>REACT가 아니다.</h1> }
```

아래처럼 사용 가능

```
{ name === 'REACT' && <h1>REACT이다</h1> }
{ name === 'REACT' || <h1>REACT가 아니다</h1> }
```

react에서
어떤 조건에 참(true)일 때 출력하고 싶다면 &&(AND) 연산자를 사용
어떤 조건에 거짓(false)일 때 출력하고 싶다면 ||(OR) 연산자를 사용

▶ JSX (JavaScript XML) 문법

5. 속성 name class 대신 className으로 사용

```
import React from 'react';

function App() {
  const name = ' 리액트';
  return <div className = “cname”> { name } </div>
}

export default App;
```

▶ JSX (JavaScript XML) 문법

6. 항상 닫는태그가 있어야한다

기존 html 태그	JSX 를 이용한 태그
<img src="이미지파일경로">	<img src="이미지파일경로"></img> 또는
 	 </br> 또는
<input>	<input></input>

▶ JSX (JavaScript XML) 문법

7. 리턴 안에서 주석 {/* 주석 */}

```
import React from 'react';

function App() {
  const name = ' 리액트';
  return (
    <>
        {/* 주석처리 */}
        <div className = “react”  // 시작 태그를 여러줄로 작성하면 여기에 주석 처리 가능
        > { name }
        </div>
        // 주석처리 안됨
        /* 이 주석도 처리 안됨 그대로 출력 */
    </>
  );
}

export default App;
```