JSX 문법

안 화 수

JSX

JSX란 ?

JSX는 자바스크립트 확장 문법이며, XML과 매우 비슷하게 되어있다. JSX 형식으로 작성된 코드는 웹 브라우저에 실행되기 전에 코드가 번들링 되는 과정에서 바벨을 사용하여 일반 자바스크립트 형태로 변환되어서 실행된다.

react project 생성

❖ react project 생성

1. reactworkspace 폴더 생성

2. reactworkspace 폴더 이동후 react project 생성

npx create-react-app jsx // jsx 프로젝트 생성

reactworkspace 폴더 안에 jsx 프로젝트 폴더 생성됨

npm start // jsx 프로젝트 실행

3. Vs code 프로그램 실행후에
[File] – Open Folder 눌러서 jsx 프로젝트 폴더 선택

4. Vs code에서 Terminal 메뉴 눌러서 jsx 프로젝트 실행

npm start

JSX문법

1. root 태그가 반드시 1개 필요하다 : src/App.js

```
function App() {
  return (
    <div>
      <h1>Hello React</h1>
      <h1>React World</h1>
    </div>
  );
}

export default App;
```

JSX문법

2. JSX안에서 자바스크립트 표현식을 사용할때는 중괄호 { }로 감싸준다 : src/App2.js

```
function App2() {
    const name = '리액트';

    return (
      <div>
        <h1>{name} 안녕</h1>
        <h1>{name} World</h1>
      </div>
    );
  }

export default App2;
```

JSX문법

- **index.js 파일 수정 : src/index.js**

  App2.js 파일을 import해서 웹브라우저에 렌더링해서 출력한다.

  나머지 js파일들도 같은 방식으로 import해서 웹브라우저에 출력 하도록 해보자.

```
import App2 from './App2';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
        <React.StrictMode>
                <App2 />
        </React.StrictMode>
);
```

JSX문법

3. JSX안에서 조건부 연산자(삼항 연산자) 사용 : src/App3.js

```
function App3() {
    const name = '리액트';

    return (
      <div>
{name === '리액트' ? <h1>리액트 입니다.</h1> : <h1>리액트가 아닙니다.</h1> }
      </div>
    );
  }

export default App3;
```

JSX문법

3. JSX 외부에서는 기존 if문 사용 가능함 : src/App30.js

```
// JSX 외부에서는 기존 if문을 사용할 수 있다.
function App30(){
    let state = '';
    const login = 'y';
    if(login === 'y'){
        state = '로그인 성공';
    }else{
        state = '로그인 실패';
    }

    return(
        <div>
            {state}
        </div>
    );
}
export default App30;
```

JSX문법

4. and 연산자(&&)를 사용한 조건부 렌더링 : src/App4.js

```
function App4() {
    const name = '리액트';

    return (
      <div>
        {name === '리액트' &&  <h1>리액트 입니다.</h1>}
      </div>
    );
  }

export default App4;
```

JSX문법

5. CSS 설정할때 속성명을 카멜 표기법으로 작성 : src/App5.js

background-color는 backgroundColor와 같이 하이픈(-)이 없이 카멜 표기법으로 작성한다.

```
function App5() {
    const name = '리액트';
    const style = {
        backgroundColor : 'black',  // background-color -> backgroundColor
        color : 'aqua',
        fontSize : '48px',              // font-size -> fontSize
        fontWeight : 'bold',            // font-weight -> fontWeight
        padding : 16                    // px 단위 생략
    }
    return (
      <div style={style}>
            {name}
      </div>
    );
  }
export default App5;
```

JSX문법

6. 외부 css파일 : src/App.css

App.css 파일에 아래의 내용을 추가한다.

```
.react{
        background-color : aqua;
        color : black;
        font-size : 48px;
        font-weight: bold;
        padding : 16px;
}
```

JSX문법

6. 외부 css파일을 불러올때는 class대신에 className 속성 사용: src/App6.js

HTML에서 CSS 클래스를 사용할 때는 와 같이 class 라는 속성을 설정한다. 하지만 JSX에서는 class 대신에 className 으로 설정해 주어야 한다.

```
import './App.css'

function App6(){
    const name = '리액트';

    return (
        <div className='react'>{name}</div>
    )
}

export default App6;
```

JSX문법

7. 열린 태그 닫기 : src/App7.js

열린 태그는 반드시 닫아 주여야 한다.

<input>
 : 에러발생(잘못된 입력)

<input></input>
</br> : 올바른 입력

```
function App7(){
    return (
        <div>
            <form>
                이름 : <input><br>                          // 에러발생
                이름 : <input></input><br></br>    // 올바른 입력
            </form>
        </div>
    )
}
export default App7;
```

JSX문법

8. JSX의 주석문 : src/App8.js

JSX의 주석문 : {/* jsx주석문 입니다. */}

JSX 주석 단축키 : Ctrl + Shift + /

```
function App8(){
    // const name = '리액트';
    return (
        <div>
            {/* jsx주석문 입니다. */}
            {/* <div>{name}</div> */}
            // 이런 주석이나
            /* 이런 주석은 페이지에 그대로 나타나게 됩니다.*/
        </div>
    )
}
export default App8;
```