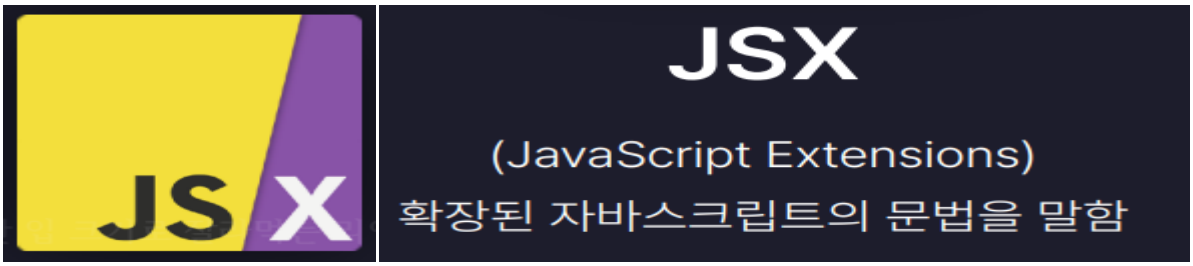
8253jang@daum.net
장희정

JSX란?



- A syntax extension to JavaScript : 자바스크립트의 확장 문법

- JavaScript와 XML/HTML을 합치 것.

예시)

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

위 문장은 왼쪽의 javaScript코드과 오른쪽의 html코드가 결합되어 있는 JSX코드 이다.

- JSX는 내부적으로 XML/HTML코드를 자바스크립트로 변환하는 과정을 거쳐 최종적으로 자바스크립트 코드가 나온다.

- JSX코드를 자바스크립트 코드로 변환하는 역할을 하는 것이 React의 createEelement() 라는 함수이다.

- JSX문법을 사용하면 리엑트에서 내부적으로 모두 createElement라는 함수를 사용하도록 변환된다. 그리고 최종적으로 이 createElement()함수를 호출한 결과로 자바스크립트 객체가 나오게 된다.

- **React에서 JSX를 사용하는 것은 필수는 아니지만 사용했을때 코드가 더욱 간결해지고 생산성과 가독성이 올라 가기때문에 사용을 권장한다.**

JSX의 장점

1) 코드가 간결해진다.

ex) JSX를 사용했을때

```
<div> Hello, {name} </div>
```

ex) JSX사용 안 했을때

```
React.createElement('div', null, 'Hello, ${name}' );
```

2) 가독성이 향상된다.

3) Injection Attack이라 불리는 해킹방법을 방어함으로써 보안성이 높다.

예시)

```
const  title = "사용자입력한값-악의적인 코드입력(보안위험의 가능성이 있는 코드)"
const element =  <h1> {title} </h1>
```

기본적으로 ReactDOM은 렌더링하기 전에 삽입된 값을 모두 문자열로 변환한다.
그렇기 때문에 명시적으로 선언되지 않은 값은 괄호사이에 들어 갈 수 없다.

(XSS - cross site scripting attacks 방어)

JSX 문법

1) 컴포넌트의 부모 요소는 반드시 단 하나 여야 한다.

단순히 감싸기 위해서 새로운 div를 사용 하고 싶지 않다면 아래 그림 오른쪽과 같이 Fragment 라는 것을 사용하면 된다. (이 기능은 v16.2 에 도입 되었다)

8253jang@daum.net
장희정

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

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

프래그먼트의 특징

 태그를 사용 한 것과 무엇이 다른지?

<React.Fragment></ React.Fragment> ==➔ <></>

<>: Fragment라고 하며, 브라우저 상의 HTML 트리 구조에서 흔적을 남기지 않고 그룹화를 해준다. 그룹화를 하는 이유는 실행될 때 JSX에 작성한 내용은 하나의 JavaScript 객체로 변환되는데 하나의 태그로 감싸지지 않으면 변환이 되지 않는다.

src/App.js

```
import React, { Component } from 'react';
class App extends Component {
  render() {
    return (
      <div>
        <div>Hello</div>
        <div>Bye</div>
      </div>
    );
  }
}
export default App;
```

src/App.js

```
import React, { Component } from 'react';
class App extends Component {
  render() {
    return (
      <React.Fragment>
        <div>Hello</div>
        <div>Bye</div>
      </React.Fragment>
    );
  }
}
export default App;
```

2)JSX 안에서, JavaScript 표현을 사용하는 방법은 매우 간단하다. { } 로 wrapping 하면 된다.

src/App.js

```
    sayHey(){
       console.log("hey");
    }
    render(){
        let text = "Dev-Server"
        return  (
            <div>
                <h1> Hello React </h1>
                <h2> Welcome to {text}</h2>
                <button onClick={this.sayHey}>Click Me</button>
            </div>
        );
    }
```

3)꼭 닫아야 하는 태그

src/App.js

```
<form>
   First Name : <br/>
   <input type="text"  name="firstname" /> <br/>
</form>
```

4) If-Else 문 사용 불가

JSX 안에서 사용되는 JavaScript 표현에는 If-Else 문이 사용 불가 하다. **대안은 삼항연산자 (condition ? true : false) 표현을 사용한다.**

src/App.js

```
class App extends Component {
  render() {
    return (
      <div>
        {
          1 + 1 === 2 ? (<div>맞아요!</div>):(<div>틀려요!</div>)
        }
      </div>
    );
  }
}
export default App;
```

AND 연산자의 경우 조건식이 true 일 때만 보여주고 false 경우 아무것도 출력하지 않을 때 사용한다.

src/App.js

```
<div>
        {
          1 + 1 === 2 && (<div>맞아요!</div>)
        }
</div>
```

5)JSX에서 css style을 입히고자 할 경우 camelCase 프로퍼티 명명 규칙에 의해 className으로 지정해야 한다.

```
function App() {

  const style = {
      color: 'white',
      backgroundColor: 'red',
      fontSize: '14px',
      fontWeight: '700'
  }
  return (
    <div className="active" style={style}>Hello World!!</div>
  )
}
```

JSX문법 예시)

- 태그 내에서 변수 값을 활용하려면 {변수명} 으로 활용

```
function Component3() {
  const name = "alice";
  return (
    <>
      <h2>hello</h2>
      <h2>hello {name}</h2>
    </>
  );
}

export default Component3;
```

- JavaScript Object로 사용하는 경우

```
function Component3() {
  const name = "alice";
  const student = {
    name: "bob",
    age: 20,
    mobile: "010-1111-1111",
  };
  return (
    <>
      <h2>hello</h2>
      <h2>hello {name}</h2>

      <h2>학생이름: {student.name}</h2>
      <h2>학생나이: {student.age}</h2>
      <h2>학생전화번호: {student.mobile}</h2>
    </>
  );
}

export default Component3;
```

실습해보기

- src/App.jsx 수정해보자

```jsx
function App() {
  const message ="JSX문법 공부하기";

  const student={
    name:"장희정",
    age:15,
    addr:"경기도 성남시",
    phone:"010-8875-8253"
  }

  const cssStyle={
   color:"red" ,
   backgroundColor:"yellow"
  }

  return (
    <>
      <h1 style={cssStyle}>message:{message}</h1>
      <h3>
        name:{student.name}<br/>
        age:{student.age}<br/>
        addr:{student.addr}<br/>
        phone{student.phone}<br/>
      </h3>

    </>
  )
}

export default App
```

JavaScript Object선언

css선언

8253jang@daum.net
장희정

```
18
19  const sayHello = ()=>{
20    console.log("버튼 클릭")
21  }
22
23  return (
24    <>
25      <h1 style={cssStyle}>message:{message}</h1>
26      <h3>
27        name:{student.name}<br/>
28        age:{student.age}<br/>
29        addr:{student.addr}<br/>
30        phone{student.phone}<br/>
31      </h3>
32      <button onClick={sayHello}>클릭</button>
33
34      {/* 조건식 만들어 보기*/}
35      {student.age > 18 ? <div>성인입니다.</div> : <div>미성년자 입니다.</div>}
36        조건식          참인경우              거짓인 경우
37      <hr/>
38        {student.age > 18 && <div style={{color:"red"}}>환영합니다.</div>}
39    </>
40  )
41  }
42
43  export default App
```

함수선언

jsx문법내에서 주석은 {/* */} 이다.

왼쪽 조건이 true일때 오른쪽을 실행한다.

조건부 랜더링

state 값에 따라 화면에 보여줄 요소를 다르게 하는 것

삼항 연산자를 활용하는 것이 일반적이다.

삼항연산자

삼항연산자는 조건에 따라 실행할 내용을 구분할 수 있도록 하는 연산자로서 if, else 와 비슷한 기능을 한다. 표현이 간결하기 때문에 조건부 렌더링에 많이 활용된다.

```
[조건] ? 실행블록1(참인경우) : 실행블록2(거짓인경우)
```

- condition 변수 값에 따라 참 또는 거짓 출력

```
function ConditionalRendering1() {
  const condition = true;
  return (
    <>
      <h2>ConditionalRendering1.jsx</h2>
      {condition ? <h2>참</h2> : <h2>거짓</h2>}
    </>
  );
}
export default ConditionalRendering1;
```

조건부 렌더링 예제

조건을 따질 변수를 state로 지정하여 값이 바뀜에 따라 화면에 보여지는 내용을 바꾼다.
버튼을 클릭할 때마다 isLogin의 반대 값을 isLogin 값으로 저장한다.
로그인상태(isLogin=true ⇒ Logout), 로그아웃상태(isLogin=false ⇒ Login) 출력한다.

```
2  import { useState } from 'react';
3
4  const Ex04_ConditionRendering = () => {
5      const [isLogin, setIsLogin] = useState(false);
6      return (
7          <div>
8              <h3>ConditionRendering Test</h3>
9              <button onClick={()=>setIsLogin(!isLogin)} >
10                 {isLogin? "Logout" : "Login"}
11             </button>
12         </div>
13     );
14 };
15
16 export default Ex04_ConditionRendering;
```