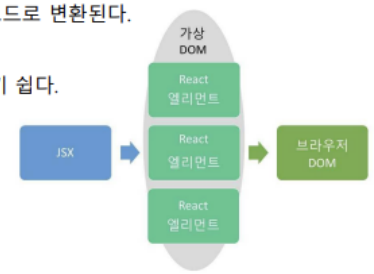
24.03.29

- 과제
- 발표

JSX

JSX (JavaScript XML, formally JavaScript Syntax eXtension)

- XML 과 유사한 구문을 사용하여 DOM(문서 개체 모델) 트리를 생성할 수 있는 JavaScript 확장
- 처음에는 React 와 함께 사용하기 위해 Facebook 에서 만들어졌지만 JSX는 여러 웹 프레임워크 에서 채택
- 구문상의 편의를 위해 JSX는 일반적으로 원래 JSX와 구조적으로 유사한 중첩된 JavaScript 함수 호출로 변환
- JSX(Javascript Syntax eXtension)는 Javascript 확장한 문법이다.
- JSX는 리액트로 프로젝트를 개발할 때 사용되므로 공식적인 자바스크립트 문법은 아니다.
- 브라우저에서 실행하기 전에 바벨을 사용하여 일반 자바스크립트 형태의 코드로 변환된다.
- JSX는 하나의 파일에 자바스크립트와 HTML을 동시에 작성하여 편리하다.
- 자바스크립트에서 HTML을 작성하듯이 하기 때문에 가독성이 높고 작성하기 쉽다.
- JSX는 JavaScript XML을 의미합니다.
- JSX를 사용하면 React에서 HTML을 작성할 수 있습니다.
- JSX를 사용하면 React에서 HTML을 더 쉽게 작성하고 추가할 수 있습니다.



start와 end, 중첩된 개념만 알고있으면 사용할 수 있다

자바스크립트 확장된 문법 → 바로 브라우저에서 실행되진 X

전 처리기 사용 ⇒ 바벨

JS와 HTML을 동시작성

JSX로 가상 DOM을 구성한 후 실제 DOM구성

JavaScript Library

Frontend

- 최근에 많이 사용되는 프론트엔드(frontend) 라이브러리들은 기본적으로 자바스크립트로 HTML 엘리먼트를 동적으로 생성하여 DOM에 추가하는 방식
- 모던(modern)한 라이브러리로 작성된 SPA(Single Page Application)를 브라우저에서 실행 후 소스 보기를 해보면 HTML 코드는 달랑 엘리먼트 하나 밖에 없는 경우가 대부분

```
<body>
  <div id="root"></div>

  <script>
   // 자바스크립트 코드
  </script>
</body>
```

- 이 최상위 엘리먼트 안에 다른 여러 가지 엘리먼트를 채워주는 작업.
- 결국 이 작업을 위해 우리는 다양한 자바스크립트 라이브러리를 사용

<초기>사용자가 이벤트를 처리하면 그 이벤트를 처리하는 용도

자바스크립트가 HTML을 동적으로 생성

JavaScript Library

HTML 엘리먼트를 동적으로 생성하여 DOM에 추가

- <h1> 엘리먼트를 생성하여 기존 엘리먼트에 추가해주는 코드를 순수하게 자바스크립트 만으로 작성

```
<body>
  <div id="root"></div>

  <script type="module">
    const headingElement = document.createElement("h1");
    headingElement.textContent = "안녕, 리액트!";
    headingElement.className = "heading";

    const rootElement = document.getElementById("root");
    rootElement.append(headingElement);
  </script>
</body>
```

- 브라우저에서 실행하면 다음과 같은 HTML 페이지가 렌더링

```
<body>
  <div id="root">
    <h1 class="heading">안녕, 리액트!</h1>
  </div>
</body>
```

안녕, 리액트!

```
const headingElement = document.createElement("h1");
headingElement.textContent = "안녕, 리액트!";
headingElement.className = "heading"
```

```
<body>
<div id="root">
```

```
<h1 class="heading">안녕, 리액트!</h1>
</div>
</body>
```

위에 3줄을 작성하여 밑의 HTML 페이지 작성됨

JavaScript Library

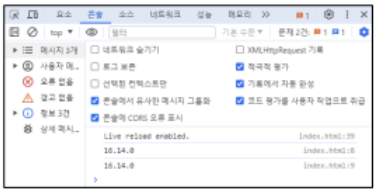
React Raw API

- 동일한 작업을 리액트(React) API를 사용해서 구현
- <script> 태그로 React와 React DOM 패키지를 CDN 주소를 통해 불러옴
- 리액트 관련 패키지를 불러오면 React와 ReactDOM을 브라우저 전역에서 사용
- 리액트는 웹 브라우저 뿐만 아니라 네이티브와 같이 여러 플랫폼에서 돌아가도록 설계된 라이브러리
- React 패키지는 플랫폼과 무방하게 UI 컴포넌트를 생성하기 위해서 사용되고, ReactDOM은 웹 플랫폼에서 UI 컴포넌트를 랜더링하기 위해서 사용

```
<body>
  <div id="root"></div>

  <script src="https://unpkg.com/react@16.14.0/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.14.0/umd/react-dom.development.js"></script>

  <script type="module">
    console.log(React.version);
    console.log(ReactDOM.version);
  </script>
</body>
```



CDM 방식 (url이 있으면 네트워크가 연결되어있을때 그 정보를 가져옴)

JavaScript Library

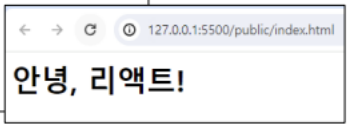
React Raw API

- 리액트 API를 사용해서 <h1> 엘리먼트를 생성하고, 기존 엘리먼트 안에 추가

```
<body>
  <div id="root"></div>

  <script src="https://unpkg.com/react@16.14.0/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.14.0/umd/react-dom.development.js"></script>

  <script type="module">
    const headingElement = React.createElement(
      "h1",
      { className: "heading" },
      "안녕, 리액트!"
    );
    const rootElement = document.getElementById("root");
    ReactDOM.render(headingElement, rootElement);
    console.log(headingElement);
  </script>
</body>
```



리엑트는 className으로 HTML의 속성을 지정해준다

JavaScript Library

React Raw API

- React.createElement() 메서드를 이용해서, 리액트 엘리먼트를 생성.
- 첫 번째 인자로 엘리먼트 이름을 넘기며, 두 번째 인자로 속성을 넘기고, 마지막 인자로 엘리먼트의 자식으로 들어갈 값을 넘김.
- ReactDOM.render() 메서드를 이용해서, 브라우저 DOM 상의 엘리먼트에 리액트 엘리먼트를 추가.
- 순수 자바스크립트 코드와 가장 중요한 차이는 HTML 엘리먼트가 아니라 리액트 엘리먼트를 생성했다는 점.
- JSX 코드는 Babel과 같은 트랜스파일러(transfiler)를 통해 브라우저가 실행할 수 있는 형태의 자바스크립트로 변환.
- 결국 브라우저는 JSX가 아닌 리액트 API로 작성된 코드를 실행하기 때문에, 이와 같이 리액트 API를 직접 사용해서 코딩을 해도 같은 효과를 냄.

엘리먼트 = 노드 하나

그에 대한 이름을 먼저 기술하고 그 다음 속성, 그 다음으로 자식을 기술하면된다

JSX을 사용하면 기술 하기 편함

바벨이 실제 순수한 자바스크립트 코드로 만들어준다

HTML, React Raw API, JSX

HTML

```
<h1 id="greeting">Hello, world!</h1>
```

React Raw API

```
React.createElement("h1", { id: "greeting" }, "Hello, world!")
```

Babel

JSX

```
const element = <h1 id="greeting">Hello, world!</h1>
```

JSX

Coding JSX

- JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
- JSX converts HTML tags into react elements.
- You are not required to use JSX, but JSX makes it easier to write React applications.

JSX

```
const myElement = <h1>I Love JSX!</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Without JSX

```
const myElement = React.createElement('h1', {}, 'I do not use JSX!');

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

렌더를 사용해서 HTML과 자바스크립트, DOM을 바로 추가할 수 있다

JSX을 사용하는건 필수가 아니다 하지만 쓰기에는 편하다

JSX

Coding JSX

JSX 사용함

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

||

JSX 사용 안함

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

자바스크립트 표현식을 사용할 수 있다

JSX

1. 반드시 부모 요소 하나가 감싸는 형태여야 한다. One Top Level Element

- Virtual DOM에서 컴포넌트 변화를 감지할 때 효율적으로 비교할 수 있도록 컴포넌트 내부는 하나의 DOM 트리 구조로 이루어져야 한다는 규칙이 있기 때문
- 태그가 비어있다면 XML처럼 /> 를 이용해 바로 닫아주어야 한다.
- JSX 태그는 자식을 포함할 수 있다.

Wrong

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

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

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

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

무조건 div안에 자식으로 div를 넣어줘야함

JSX

2. 자바스크립트 표현식

- JSX 안에서도 자바스크립트 표현식을 사용할 수 있다.
- 자바스크립트 표현식을 작성하려면 JSX내부에서 코드를 { }로 감싸주면 된다.
- 유효한 모든 JavaScript 표현식을 넣을 수 있다.

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;
```

```
function App() {
        const name = 'GodDaeHee';
        return (
                <div>
                        <div>Hello</div>
                        <div>{name}!</div>
                </div>
        );
}
```

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);
```

```
const myElement = <h1>React is {5 + 5} times better with JSX</h1>;
```

소프트웨어 Web programmi

var = expression

수식결과를 변수에 넣는다

var = expression(식)

식 에는

1. 숫자, 문자 = constant / literal
2. 함수 = funtion
3. 연산자 = variable
4. 변수 = operater

허헝...안외웠는데 외우어양대....

```
'ez';
Hello, {name}</h1>;
```

중괄호 안에 들어갈 수 있는게 표현식이다

```
const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);
```

표현식 - 함수 사용

JSX

3. JSX도 표현식이다

- 컴파일이 끝나면, JSX 표현식이 JavaScript 객체로 인식된다.
- 즉, JSX를 if 구문 및 for loop 안에 사용하고, 변수에 할당하고, 인자로서 받아들이고, 함수로부터 반환할 수 있다.
- JSX는 자바스크립트 문법을 확장시킨 것, 따라서 모든 자바 스크립트 문법을 지원한다.
- 자바스크립트에 추가로 XML과 HTML 섞어서 사용하면 된다
- xml, html 코드를 사용 시 중간에 자바스크립트 코드를 사용하고 싶으면 중괄호 {}를 사용하여 묶어주면 된다.

```
function getGreeting(user) {
  if (user) {
    return <h1>Hello, {formatName(user)}!</h1>;
  }
  return <h1>Hello, Stranger.</h1>;
}
```

JSX의 표현식 자체도 자바스크립트의 객체로 표현된다

JSX

4. if문(for문) 대신 삼항 연산자(조건부 연산자) 사용

- if 구문과 for 루프는 JavaScript 표현식이 아니기 때문에 JSX 내부 자바스크립트 표현식에서는 사용할 수 없다.
- 그렇기 때문에 조건부에 따라 다른 렌더링 시 JSX 주변 코드에서 if문을 사용하거나, {}안에서 삼항 연산자(조건부 연산자)를 사용 한다.

```
function App() {
  let desc = '';
  const loginYn = 'Y';
  if(loginYn === 'Y') {
    desc = <div>GodDaeHee 입니다.</div>;
  } else {
    desc = <div>비회원 입니다.</div>;
  }
  return (
    <>
      {desc}
    </>
  );
}
```

```
function App() {
  const loginYn = 'Y';
  return (
    <>
      <div>
        {loginYn === 'Y' ? (
          <div>GodDaeHee 입니다.</div>
        ) : (
          <div>비회원 입니다.</div>
        )}
      </div>
    </>
  );
}
```

JSX에서는 if문이랑 for문 사용이안됨

표현식(expression)이아니라 control statement이다

진짜 너무너무써야겠다 하면 위에서 필요한 자바스크립트 코드를 작성하고 변수에 할당한다

그 후 JSX에 넣어주면됨

JSX

4. if문(for문) 대신 삼항 연산자(조건부 연산자) 사용

```
// 조건이 만족하지 않을 경우 아무것도 노출되지 않는다.
function App() {
  const loginYn = 'Y';
  return (
    <>
      <div>
        {loginYn === 'Y' && <div>GodDaeHee 입니다.</div>}
      </div>
    </>
  );
}
```

```
//즉시 실행 함수
function App() {
  const loginYn = 'Y';
  return (
    <>
      {
        (() => {
          if(loginYn === "Y"){
            return (<div>GodDaeHee 입니다.</div>);
          }else{
            return (<div>비회원 입니다.</div>);
          }
        })()
      }
    </>
  );
}
```

```
const App = () => {
  const i = 1;

  return (
    <div>
      <h1>{ i === 1 ? 'true' : 'false' }</h1>
    </div>
  );
}
```

```
<div>
  <h1>true</h1>
</div>
```

소프트웨어 Web programm

```c
#include <stdio.h>

int main()
{
    int a=0, b=1;
    if(a == 0 && ++b > 5) {
        printf("ok");
    }
    printf(" A = %d B = %d", a, b);
    return 0;
}
```

```
input
A = 0 B = 2

..Program finished with exit code 0
ress ENTER to exit console.
```

논리연산 or일때

앞의 조건이 참이면 컴퓨터는 뒤의 조건을 확인하지 X

논리연산 and일때

앞의 조건이 참이면 뒤의 조건확인

(() ⇒ 어쩌구

)()

JSX

5. JSX 속성 정의

- 속성에 따옴표를 이용해 문자열 리터럴을 정의할 수 있다.
- 속성에 중괄호를 이용해 자바스크립트 표현식을 포함시킬 수 있다.

```
const element = <div tabIndex="0"></div>;
```

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

- 스타일 적용
  - 리액트에서 DOM 요소에 스타일을 적용할 때는 문자열 형태로 넣는 것이 아니라 객체 형태로 넣어줘야 한다.
  - 자바스크립트 코드니까 중괄호가 있는데 객체형태이므로 또 중괄호가 생겨 이중 중괄호 형태가 된 것이다.
  - 또한 스타일 이름 중에서 background-color처럼 -문자가 포함되는 이름은 -문자를 없애고 카멜 표기법으로 작성한다.
  - background-color는 backgroundColor로 작성한다.

```
<p style="color: blue">Lorem ipsum dolor.</p>
```

```
<p style={{color: blue}}>Lorem ipsum dolor.</p>
```

JSX

6. React DOM은 HTML 어트리뷰트 이름 대신 camelCase를 사용한다.

1. JSX 스타일링
   - JSX에서 자바스크립트 문법을 쓰려면 {}를 써야 하기 때문에, 스타일을 적용할 때에도 객체 형태로 넣어 주어야 한다.
   - 카멜 표기법으로 작성해야 한다. (font-size => fontSize)

```
function App() {
        const style = {
                backgroundColor: 'green',
                fontSize: '12px'
        }
        return (
                <div style={style}>Hello, GodDaeHee!</div>
        );
}
```

JSX

6. React DOM은 HTML 어트리뷰트 이름 대신 camelCase를 사용한다.

2. class 대신 className
   - 일반 HTML에서 CSS 클래스를 사용할 때에는 class 라는 속성을 사용한다.
   - JSX에서는class는 className이 되고 tabindex는 tabIndex가 된다.

```
function App() {
        const style = {
                backgroundColor: 'green',
                fontSize: '12px'
        }
        return (
                <div className="testClass">Hello, GodDaeHee!</div>
        );
}
```

JSX

7. JSX 내에서 주석 사용 방법

- JSX 내에서 {/*...*/} 와 같은 형식을 사용 한다.
- 시작태그를 여러줄 작성시에는, 내부에서 // 의 형식을 사용할 수 있다.

```
function App() {
  return (
    <>
      {/* 주석사용방법 */}
      <div>Hello, GodDaeHee!</div>
    </>
  );
}
```

```
function App() {
  return (
    <>
      <div
      // 주석사용방법
      >Hello, GodDaeHee!</div>
    </>
  );
}
```

개발자가 JSX를 작성하기만 하면, 리액트 엔진은 JSX를 기존 자바스크립트로 해석하여 준다.
이를 '선언형 화면' 기술이라고 한다

시작태그에서는 //를 사용할 수 있다

JSX 요소를 포함하는 간단한 React 컴포넌트 예

App.js

```
import React from 'react';

function App() {
  const name = 'John Doe';
  const items = ['Apple', 'Banana', 'Cherry'];

  return (
    <div className="App">
      <h1>Hello, {name}!</h1>
      <p className="intro">This is an introduction paragraph.</p>

      <ul>
        {items.map((item, index) => <li key={index}>{item}</li>)}
      </ul>

      <button onClick={() => { alert('Button is clicked!'); }}>
        Click me
      </button>

      <input type="text" placeholder="Enter some text" />

      <img src="https://i.imgur.com/MK3eW3Am.jpg" alt="placeholder" />
    </div>
  );
}

export default App;
```

```
PS D:\react\my-app> cd ..
PS D:\react> npx create-react-app book
```

npx create-react-app book

book파일 열기

jsxCode 생성

new file Book.js

import React form “react”; 6.9k (gzipped:2.7k)

```
function Book(props) {
    return(
        <div>
            <h1>{`이 책의 이름은 ${props.name}입니다`}</h1>
            <h1>{`이 책은  ${props.numberOfPage}페이지로 이루어져있습니다`}</h1>
        </div>
```

BookLibrary.js생성

import React form “react”; 6.9k (gzipped:2.7k)

import React form “./Book”;

```
function BookLibrary(props) {
    return(
        <div>
            <Book name=”처음 만난 react” numberOfPage={300} />
            <Book name=”처음 만난 JSX” numberOfPage={400} />
            <Book name=”처음 만난 Component” numberOfPage={500} />
        </div>

export default BoolLibrary;
```

index.js키고

import moduleName from ‘./jsxCode/BookLibrary’

npm start

이 책의 이름은 처음 만난 React입니다.

이 책은 총 300페이지로 이뤄져 있습니다.

이 책의 이름은 처음 만난 JSX입니다.

이 책은 총 400페이지로 이뤄져 있습니다.

이 책의 이름은 처음 만난 Component입니다.

이 책은 총 500페이지로 이뤄져 있습니다.

JSX예제.txt