

Web Programming

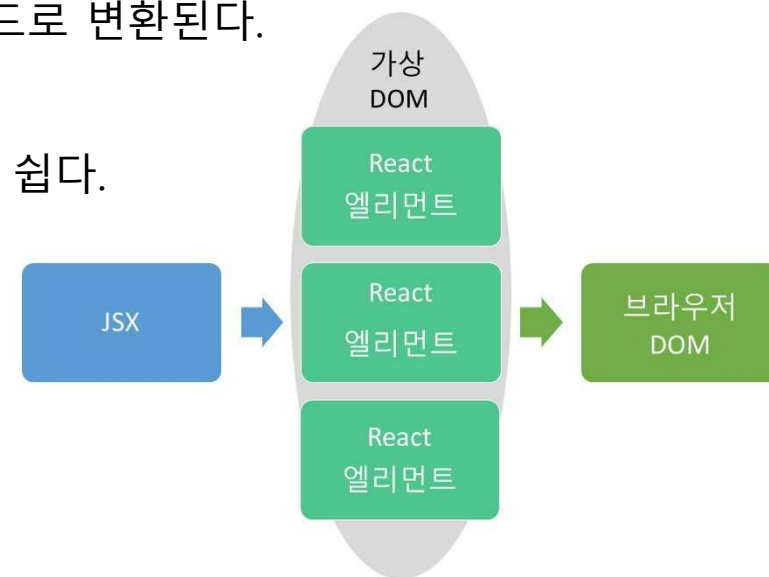
React programming



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.





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 <div> 엘리먼트 하나 밖에 없는 경우가 대부분

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

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

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



JavaScript Library

HTML 엘리먼트를 동적으로 생성하여 DOM에 추가

- <h1> 엘리먼트를 생성하여 기존 <div> 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!



JavaScript Library

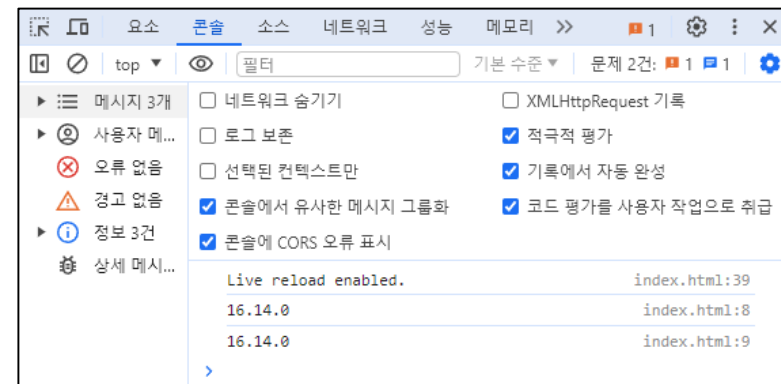
React Raw API

- 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 렌더링하기 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>
```





JavaScript Library

React Raw API

- 리액트 API를 사용해서 <h1> 엘리먼트를 생성하고, 기존 <div> 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>
```

← → ↻ ⓘ 127.0.0.1:5500/public/index.html

안녕, 리액트!



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 <div> 엘리먼트에 리액트 엘리먼트를 추가.
- 순수 자바스크립트 코드와 가장 중요한 차이는 HTML 엘리먼트가 아니라 리액트 엘리먼트를 생성했다는 점.
- JSX 코드는 Babel과 같은 트랜스파일러(transpiler)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.

```
Live reload enabled. index.html:45
▼ Object i index.html:15
  $$typeof: Symbol(react.element)
  key: null
  ▼ props:
    children: "안녕, 리액트!"
    className: "heading"
    ▶ [[Prototype]]: Object
  ref: null
  type: "h1"
  _owner: null
  ▶ _store: {validated: false}
  _self: null
  _source: null
  ▶ [[Prototype]]: Object
```



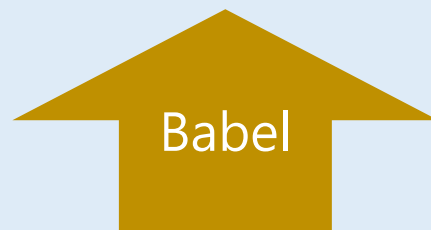
HTML, React Raw API, JSX

HTML

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

React Raw API

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



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.

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

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

JSX

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

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

Without JSX



JSX

Coding JSX

JSX 사용함

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

||

JSX 사용 안함

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



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.

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

Wrong

```
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>  
    </>  
  );  
}
```



JSX

2. 자바스크립트 표현식

- JSX 안에서도 자바스크립트 표현식을 사용할 수 있다.
- 자바스크립트 표현식을 작성하려면 JSX내부에서 코드를 { }로 감싸주면 된다.
- 유효한 모든 [JavaScript 표현식](#)을 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>;
```



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>;  
}
```



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>  
    </>  
  );  
}
```



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>  
    </>  
  );  
}
```

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

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

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



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.
- JSX에서는 class는 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



JSX 요소를 포함하는 간단한 React 컴포넌트 예

```
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>
```

App.js

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

```
    
  </div>
```

```
);
}
```

```
export default App;
```

Report : 자신을 소개하는 페이지를 작성하시오.



<https://babeljs.io/>

BABEL

DocsSetupTry it outVideosBlogDonateTeamGitHub

PRESETS

☒ react

☐ flow

☐ typescript

☒ stage-3

☒ stage-2

☐ stage-1

☐ stage-0

OPTIONS

React RuntimeClassic

Decorators version2021-12

Decorators beforeexport☐

ENV PRESET

☒ Enabled

ELECTRON1.8☐

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

<

```
<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" />

  
</div>
```

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

<

```
/*#__PURE__*/React.createElement("div", {
  className: "App"
}, /*#__PURE__*/React.createElement("h1", null, "Hello, ", name,
"!"), /*#__PURE__*/React.createElement("p", {
  className: "intro"
}, "This is an introduction paragraph."),
/*#__PURE__*/React.createElement("ul", null, items.map((item,
index) => /*#__PURE__*/React.createElement("li", {
  key: index
}, item))), /*#__PURE__*/React.createElement("button", {
  onClick: () => {
    alert('Button is clicked!');
  }
}, "Click me"), /*#__PURE__*/React.createElement("input", {
  type: "text",
  placeholder: "Enter some text"
}), /*#__PURE__*/React.createElement("img", {
  src: "https://i.imgur.com/MK3eW3Am.jpg",
  alt: "placeholder"
})));
```



JSX Code 작성해보기

```
import React from "react";
```

```
function Book(props) {
```

```
  return (
```

```
    <div>
```

```
      <h1>{'이 책의 이름은 ${props.name}입니다.'}</h1>
```

```
      <h2>{'이 책은 총 ${props.numberOfPage}페이지로 이뤄져 있습니다.'}</h2>
```

```
    </div>
```

```
  );
```

```
}
```

```
export default Book;
```

src/JSXCode/Book.jsx

```
import React from "react";
```

```
import Book from "./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 BookLibrary;
```

src/JSXCode/BookLibrary.jsx



JSX Code 작성해보기

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
//import App from './App';
import reportWebVitals from './reportWebVitals';

import BookLibrary from './JSXCode/BookLibrary';

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

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

src/index.js

이 책의 이름은 처음 만난 **React**입니다.

이 책은 총 300페이지로 이뤄져 있습니다.

이 책의 이름은 처음 만난 **JSX**입니다.

이 책은 총 400페이지로 이뤄져 있습니다.

이 책의 이름은 처음 만난 **Component**입니다.

이 책은 총 500페이지로 이뤄져 있습니다.



JSX Code 작성해 보기

```
import React from "react";

function Book(props) {
  return React.createElement(
    'div',
    null,
    [
      React.createElement(
        'h1',
        null,
        `이 책의 이름은 ${props.name}입니다.`
      ),
      React.createElement(
        'h2',
        null,
        `이 책은 총 ${props.numberOfPage}페이지로 이뤄져 있습니다.`
      )
    ]
  )
}
```

```
import React from "react";

function Book(props) {
  return (
    <div>
      <h1>{`이 책의 이름은 ${props.name}입니다.`}</h1>
      <h2>{`이 책은 총 ${props.numberOfPage}페이지로 이뤄져 있습니다.`}</h2>
    </div>
  );
}

export default Book;
```




Elements

HTML Elements

- The HTML element is everything from the start tag to the end tag:
<tagname>Content goes here... </tagname>
- HTML elements can be nested (this means that elements can contain other elements).
- All HTML documents consist of nested HTML elements.
- The following example contains four HTML elements (<html>, <body>, <h1> and <p>):

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
<p>	My first paragraph.	</p>
 	<i>none</i>	<i>none</i>



React Elements

Elements

- Elements are the smallest building blocks of React apps.
- DOM elements의 가상 표현

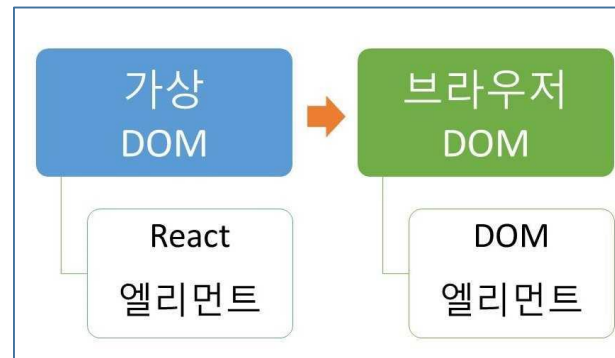
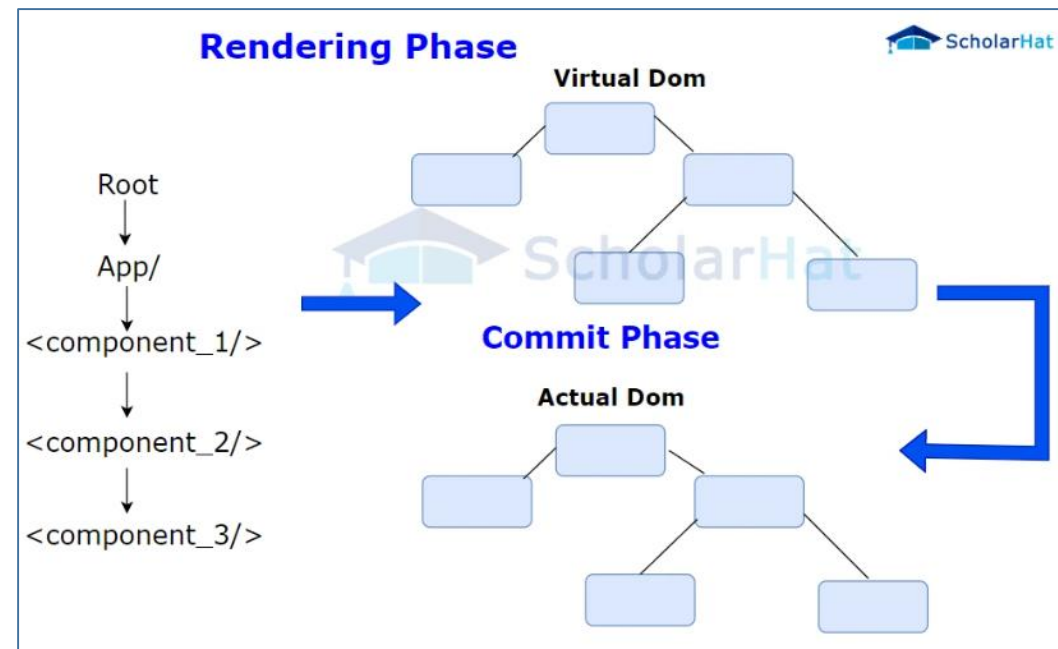
Descriptor

- 화면에 나타나는 요소를 기술하는 자바스크립트 객체



DOM elements

- 최종적으로 나타나는 형태는 DOM elements





React Elements

Elements

- 가상 DOM은 자바스크립트 객체입니다. 가장 DOM은 리액트 엘리먼트로 이루어져 있고 브라우저 DOM은 DOM 엘리먼트로 이루어져 있다.
- 리액트 앱을 이루는 가장 기본적인 단위
- 리액트 엘리먼트는 브라우저 DOM을 만들기 위해 개발자와 브라우저 DOM을 이어주는 오작교
- 엘리먼트(element)는 View에 렌더링 할 내용을 React에 알려주기 위한 수단으로, React 애플리케이션을 구성하는 가장 작은 블록
- React 엘리먼트는 HTML 태그의 이름을 값으로 가지는 type 필드와 그 외 속성들을 값으로 전달받는 props 필드로 구성된 객체(object) 형태로 정의되며, React는 이 객체를 읽어들이어 DOM을 구성하고 최신 상태로 업데이트하는데 사용
- React 엘리먼트는 일반 객체(plain object)로 손쉽게 생성할 수 있지만, 불변 객체(immutable object)이기 때문에 일단 생성된 후에는 상태나 속성을 변경할 수 없다.
- 따라서 React에서 UI를 업데이트하는 방법은 새로운 엘리먼트를 생성하고, 이를 render() 메소드에 전달

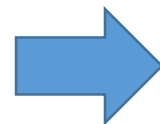


React Elements

React elements

- 자바 스크립트 객체 형태로 존재

```
{
  type: 'button',
  props: {
    className: 'bg-green',
    children: {
      type: 'b',
      props: {
        children: 'Hello, element!'
      }
    }
  }
}
```



DOM element

```
<button class='bg-green'>
  <b>
    Hello, element!
  </b>
</button>
```



React Elements 생성

createElement

- createElement를 사용하면 React 엘리먼트를 생성할 수 있다.
- JSX를 작성하는 대신 사용할 수 있다.

```
const element = createElement(type, props, ...children)
```

type

- type 인수는 유효한 React 컴포넌트여야 한다.
- 예를 들어 태그 이름 문자열 (예: 'div', 'span') 또는 React 컴포넌트(함수, 클래스, Fragment 같은 특수 컴포넌트)가 될 수 있다.

props

- props 인수는 객체 또는 null이어야 한다. Null을 전달하면 빈 객체와 동일하게 처리된다.
- React는 전달한 props와 일치하는 프로퍼티를 가진 엘리먼트를 생성한다.

...children

- 0개 이상의 자식 노드. React 엘리먼트, 문자열, 숫자, 포탈, 빈 노드(null, undefined, true, false) 그리고 React 노드 배열을 포함한 모든 React 노드가 될 수 있다.



React Elements 생성

엘리먼트 생성하기

- JSX가 마음에 들지 않거나 프로젝트에서 사용할 수 없는 경우, createElement를 대안으로 사용할 수 있다.
- JSX 없이 엘리먼트를 생성하려면 type, props, children와 함께 createElement를 호출한다

```
import { createElement } from 'react';

function Greeting({ name }) {
  return createElement(
    'h1',
    { className: 'greeting' },
    'Hello ',
    createElement('i', null, name),
    '. Welcome!'
  );
}
```

```
function Greeting({ name }) {
  return (
    <h1 className="greeting">
      Hello <i>{name}</i>. Welcome!
    </h1>
  );
}
```



React Elements 생성

엘리먼트 생성하기

```
import { createElement } from 'react';

function Greeting({ name }) {
  return createElement(
    'h1',
    { className: 'greeting' },
    'Hello ',
    createElement('i', null, name),
    '. Welcome!'
  );
}

export default function App() {
  return createElement(
    Greeting,
    { name: 'Taylor' }
  );
}
```

```
function Greeting({ name }) {
  return (
    <h1 className="greeting">
      Hello <i>{name}</i>. Welcome!
    </h1>
  );
}

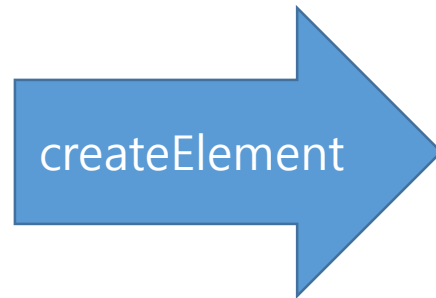
export default function App() {
  return <Greeting name="Taylor" />;
}
```

Hello Taylor. Welcome!



React Elements 생성

```
function Button(props) {  
  return (  
    <button className={`bg-${props.color}`}>  
      <b>  
        {props.children}  
      </b>  
    </button>  
  )  
}  
  
function ConfirmDialog(props) {  
  return (  
    <div>  
      <p>내용을 확인하셨으면 확인 버튼을 눌러주세요.</p>  
      <Button color='green'>확인</Button>  
    </div>  
  )  
}
```





React Elements 생성

```
{
  type: 'div',
  props: {
    children: [
      {
        type: 'p',
        props: {
          children: '내용을 확인하셨으면 확인 버튼을 눌러주세요!'
        }
      },
      {
        type: Button,
        props: {
          color: 'green',
          children: '확인'
        }
      }
    ]
  }
}
```



React Elements 생성

```
{
  type: 'div',
  props: {
    children: [
      {
        type: 'p',
        props: {
          children: '내용을 확인하셨으면 확인 버튼을 눌러주세요!'
        }
      },
      {
        type: button,
        props: {
          className: 'bg-green',
          children: {
            type: 'b',
            props: {
              children: '확인'
            }
          }
        }
      }
    ]
  }
}
```



Reference

- <https://reactjs-kr.firebaseio.com/docs/introducing-jsx.html>
 - <https://ko.legacy.reactjs.org/docs/jsx-in-depth.html>
 - <https://transform.tools/html-to-jsx>
 - <https://ko.react.dev/reference/react/createElement>
 - <https://wikidocs.net/197615>
-
- 명품 웹 프로그래밍, 황기태, 생능출판사
 - 리액트 & 리액트 네이티브 통합 교과서, 아담 보두치, 강경일, 신희철, 에이콘
 - Do it! 리액트 모던 웹 개발 with 타입스크립트, 전예홍, 이지스퍼블리싱