

# [ 1주차 ] 리액트 시작하기

리액트를 위한 ES6 기본 문법 정리

리액트가 하는 일

자바스크립트 컴파일 없이 리액트 시작하기

*Rendering* 이해하기

*Component* 의 이해

JSX 문법

*props* 와 *state*

이벤트 처리

*Lifecycle API*



# Mark Lee

*Software Engineer | Studio XID, Inc.*

*Microsoft MVP*

*TypeScript Korea User Group Organizer*

*Marktube (Youtube)*

# 리액트를 위한 JS 문법 정리

- const let
- template string
- arrow function
- .bind(this)
- const {children} = this.props;
- ...props
- Promise
- async await
- Generator

# var 는 문제아

1. 헷갈리는 함수 레벨 스코프
2. 중복 선언이 가능
3. 생략도 가능
4. 호이스팅

```
1 // var.js
2
3 // 1.
4 (function() {
5   if (true) {
6     var variable = "function scope";
7   }
8
9   console.log(variable);
10 })();
11
12 // 2.
13 (function() {
14   var variable = "function scope";
15   var variable = "duplicated";
16
17   console.log(variable);
18 })();
19
```

# let 은 해결사

1. 블록 레벨 스코프
2. 중복 선언 => syntax error
3. 호이스팅 => syntax error

```
1 // 1.  
2 {  
3     let variable = "block scope";  
4  
5     console.log(variable);  
6 }  
7  
8 // 2.  
9 // {  
10 //     let variable = "block scope";  
11 //     let variable = "duplicated";  
12  
13 //     console.log(variable);  
14 // }  
15  
16 // 3.  
17 {  
18     console.log(variable);  
19     let variable = "hoisted";
```

# let 은 변경 가능, const 는 불가능

- Primitive
- Reference

```
1 // Primitive
2 let a = "a";
3 a = "b";
4 a;
5
6 const c = "c";
7 // c = "d";
8 c;
9
10 // Reference
11 let e = {
12   foo: "foo"
13 };
14 e = {
15   bar: "bar"
16 };
17 e;
18
19 const f = {
20   foo: "foo"
21 };
22 f.foo = "bar";
23 f;
```

# template string

- `문자열`
- `\${자바스크립트 표현식}`

```
1 const name = "Mark";
2
3 console.log("안녕하세요. 제 이름은 " + name + " 입니다.");
4
5 console.log(`안녕하세요. 제 이름은 ${name} 입니다.`);
```

# arrow function

- 자신의 this 를 만들지 않는다.
- 생성자로 사용할 수 없다.
- 항상 익명 함수
- 리턴만 있으면, {} 생략
- 인자가 하나면, () 생략

```
1 function Foo() {
2   this.name = "Mark";
3
4   setTimeout(function() {
5     console.log(this.name);
6   }, 1000);
7
8   setTimeout(() => {
9     console.log(this.name);
10  }, 1000);
11 }
12
13 const foo = new Foo();
14
15 const a = () => {
16   return "리턴";
17 };
18
19 console.log(a());
```

# 함수.bind(디스)

- 함수의 this로 인자로 넣은 "디스"를 사용하는 함수를 만들어 리턴

```
1 const mark = {  
2   name: "Mark"  
3 };  
4  
5 function hello() {  
6   console.log(`안녕하세요 ${this.name}`);  
7 }  
8  
9 const helloMark = hello.bind(mark);  
10  
11 helloMark();  
12  
13 const anna = {  
14   name: "Anna"  
15 };  
16  
17 const helloAnna = hello.bind(anna);  
18  
19 helloAnna();
```

# Destructuring assignment

- 구조 분해 할당
- 배열, 객체

```
1 const foo = {  
2   a: "에이",  
3   b: "비이"  
4 };  
5  
6 const { a, b } = foo;  
7 console.log(a, b);  
8  
9 const bar = [ "씨이", "디이" ];  
10  
11 const [c, d] = bar;  
12 console.log(c, d);  
13  
14 const { a: newA, b: newB } = foo;  
15 console.log(newA, newB);
```

# Spread 와 Rest

- ...
- 배열, 객체
- 1 레벨 깊이 복사

```
1 function sum(a, b, c) {
2     return a + b + c;
3 }
4
5 console.log(sum(1, 2, 3));
6
7 const numbers = [2, 3, 4];
8
9 console.log(sum(...numbers));
10
11 const num = {
12     a: 3,
13     b: 4,
14     c: 5
15 };
16
17 const cloned = { ...num };
18 cloned.a = 10;
19
20 console.log(num);
21 console.log(cloned); // 1 레벨 깊이
22
23 const deep = {
```

# callback

- 과거 비동기 처리를 위한 선택

```
1 function foo(callback) {  
2     setTimeout(() => {  
3         // 로직  
4         callback();  
5     }, 1000);  
6 }  
7  
8 foo(() => {  
9     console.log("end");  
10});  
11 console.log("이것이 먼저 실행");
```

# Promise 객체

- Promise 객체를 만들고, 로직 처리 후 성공과 실패를 알려준다.
- then 과 catch 를 통해 메인 로직에 전달한다.

```
1 function foo() {
2   return new Promise((resolve, reject) => {
3     setTimeout(() => {
4       // 로직
5       resolve();
6     }, 1000);
7   });
8 }
9
10 foo().then(() => {
11   console.log("end");
12 });
13 console.log("이것이 먼저 실행");
```

# async - await

- 기본적으로 Promise 를 사용한다.
- then 과 catch 를 통해 메인 로직에 전달한다.
- async 키워드가 붙은 함수 안에서만 await 키워드를 사용할 수 있다.

```
1 function foo() {  
2   return new Promise((resolve, reject) => {  
3     setTimeout(() => {  
4       // 로직  
5       resolve();  
6     }, 1000);  
7   });  
8 }  
9  
10(async () => {  
11  await foo();  
12  console.log("end");  
13  console.log("이것이 먼저 실행");  
14 })();
```

# Generator 객체

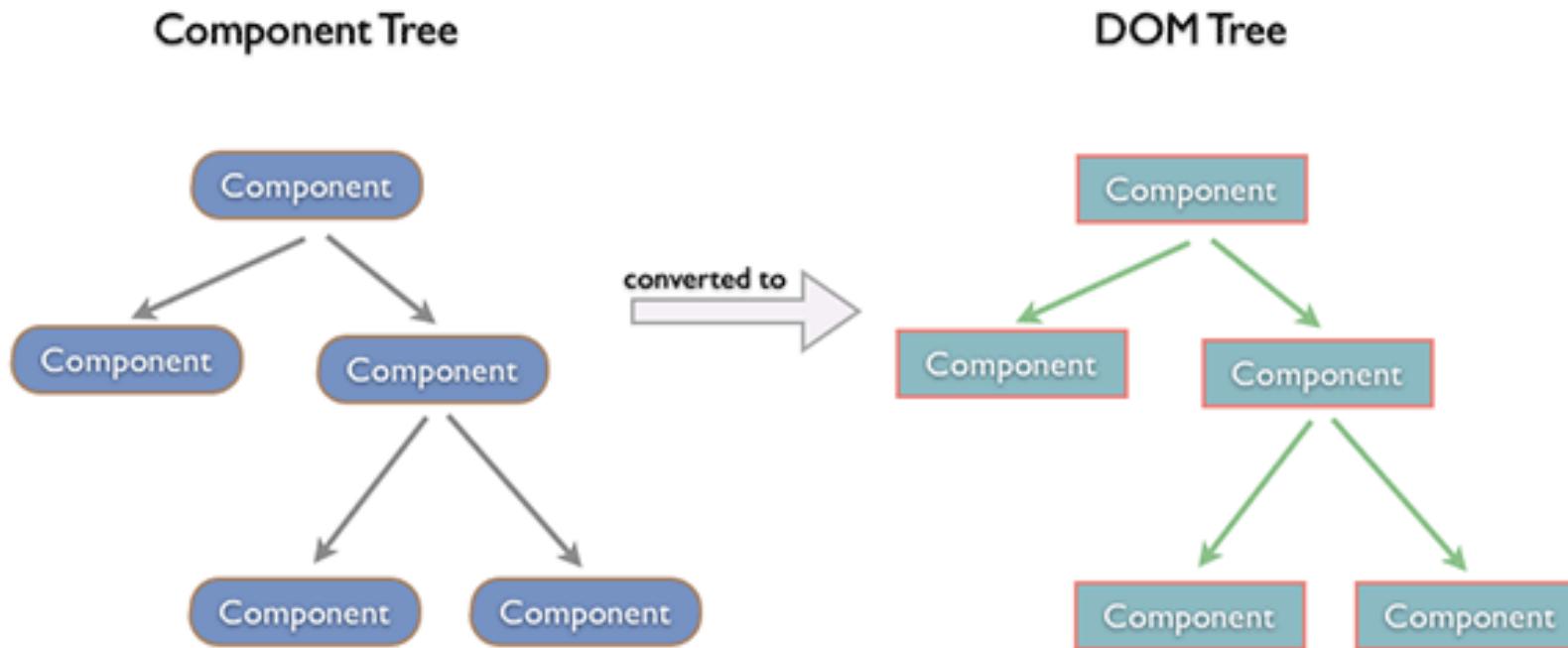
- function\* 으로 만들어진 함수를 호출하면 반환되는 객체이다.
- function\* 에서 yield 를 호출하여, 다시 제어권을 넘겨준다.
- 제너레이터 객체에 next() 함수를 호출하면, 다음 yield 지점까지 간다.

```
1 function foo() {
2   return new Promise((resolve, reject) => {
3     setTimeout(() => {
4       // 로직
5       resolve();
6     }, 1000);
7   });
8 }
9
10 function* bar() {
11   yield foo();
12 }
13
14 console.log(bar().next());
15
16 bar()
17   .next()
18   .value.then(() => {
19     console.log("end");
20   });
21 console.log("이것이 먼저 실행? ");
22
23 function* baz() {
```

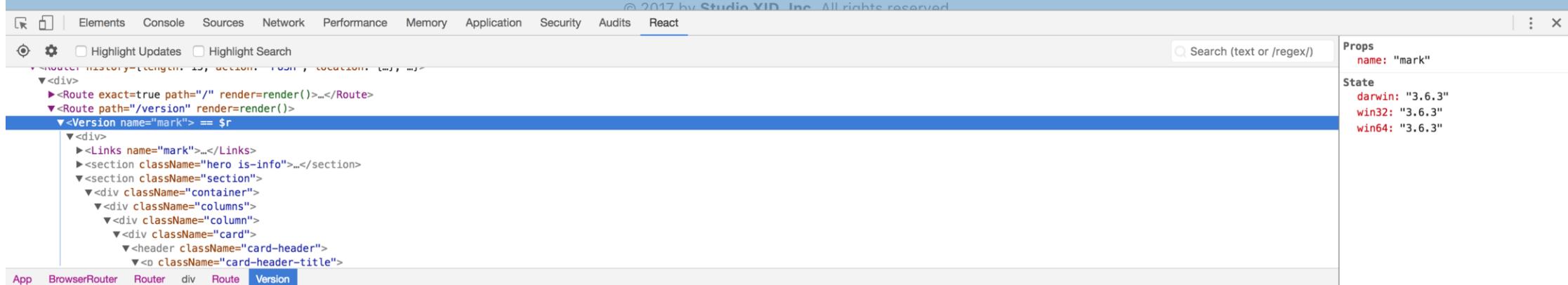
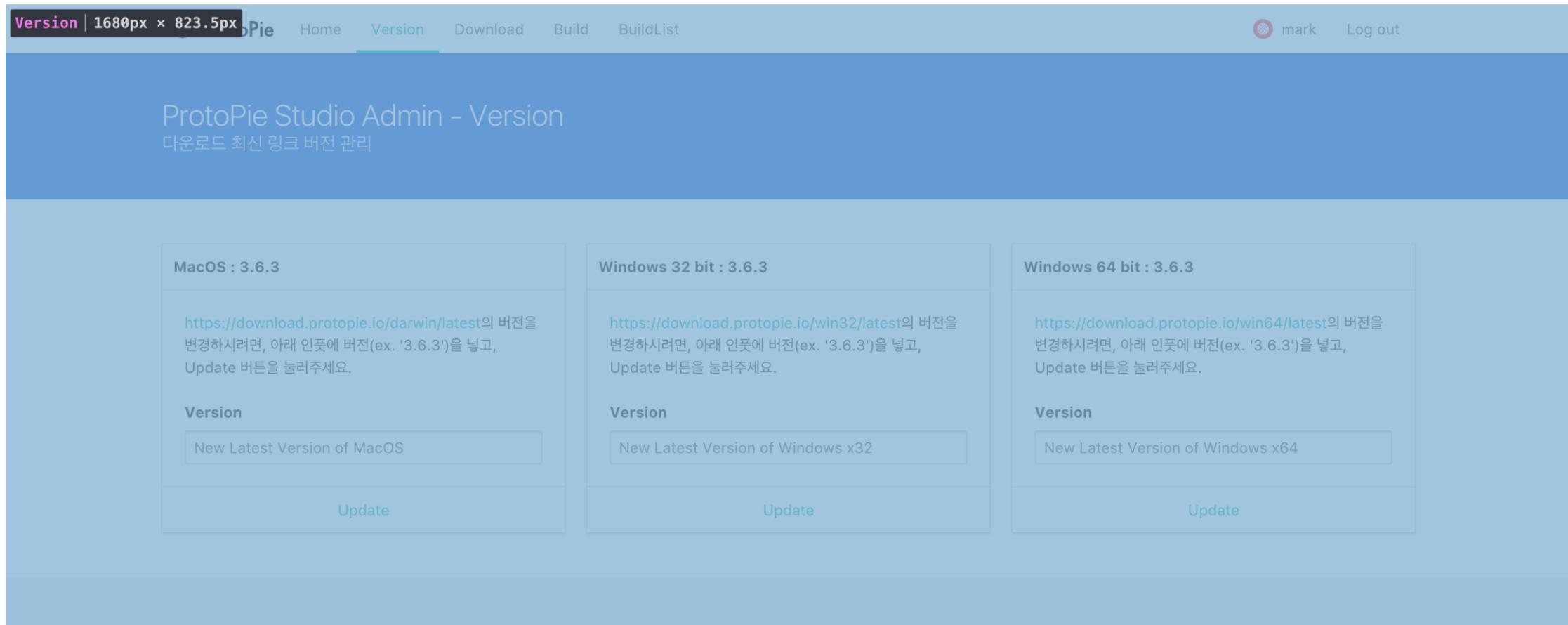
# React Keyword

- View 라이브러리
- Only Rendering & Update
  - NOT included another functionality
- Component Based Development
  - 작업의 단위
- Virtual DOM
  - 이제는 DOM 을 직접 다루지 않음.
- JSX
  - NOT Templates
  - transpile to JS
- CSR & SSR

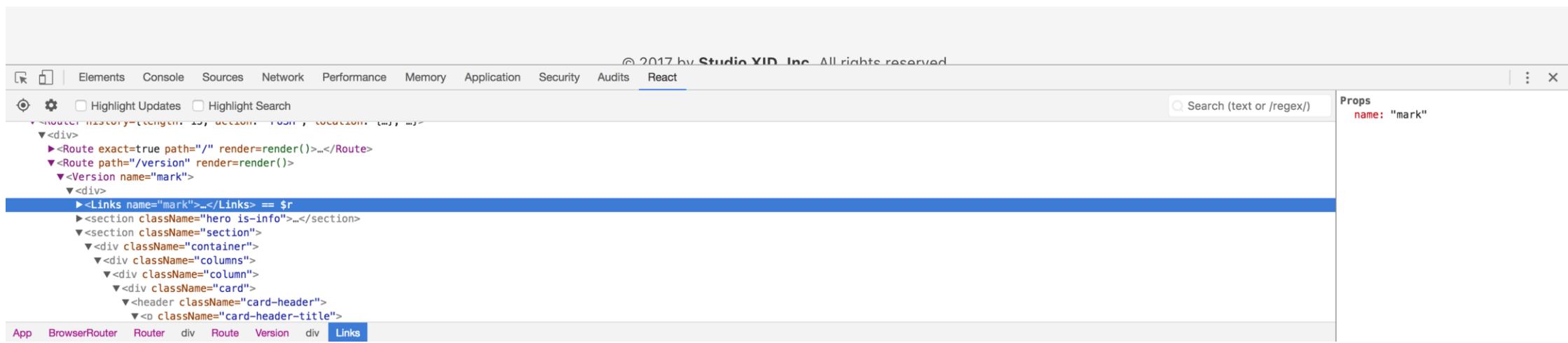
# Component Tree => DOM Tree



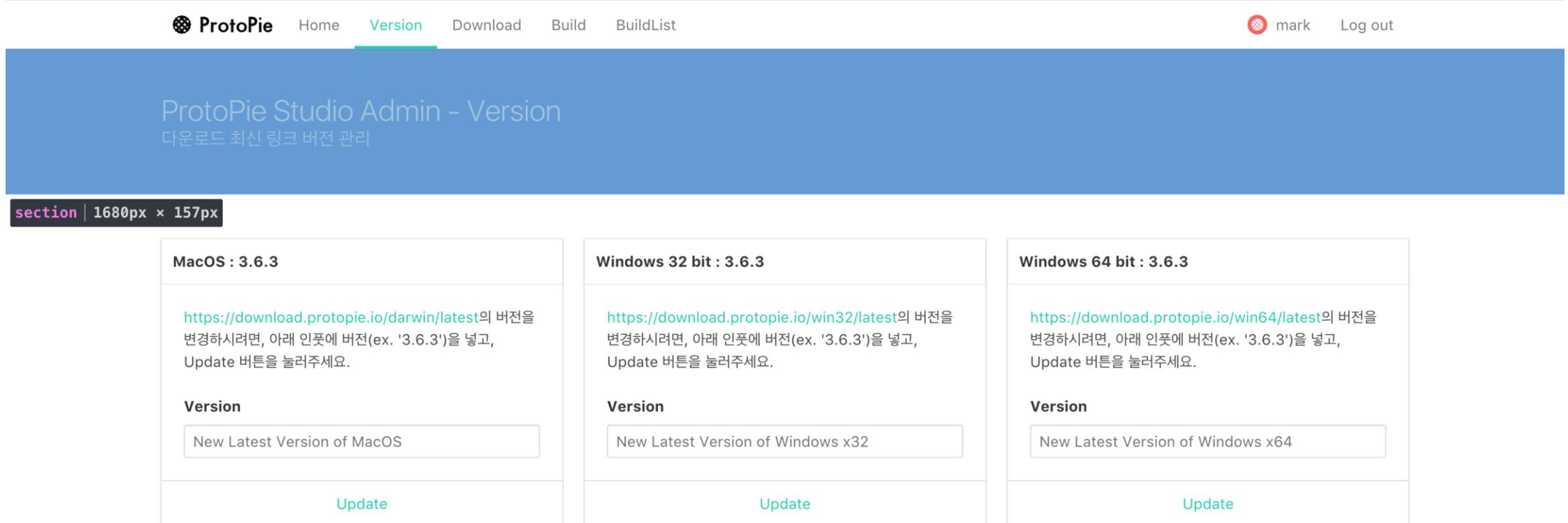
# Component Based Development - Version



# Component Based Development - Links



# Component Based - Title



The screenshot shows the ProtoPie Studio Admin - Version page. It features a blue header bar with the title "ProtoPie Studio Admin - Version" and a sub-header "다운로드 최신 링크 버전 관리". Below the header, there are three separate update cards, each with a title, a description, a "Version" section, and an "Update" button.

- MacOS : 3.6.3**  
https://download.protopie.io/darwin/latest의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.  
**Version**  
New Latest Version of MacOS  
**Update**
- Windows 32 bit : 3.6.3**  
https://download.protopie.io/win32/latest의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.  
**Version**  
New Latest Version of Windows x32  
**Update**
- Windows 64 bit : 3.6.3**  
https://download.protopie.io/win64/latest의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.  
**Version**  
New Latest Version of Windows x64  
**Update**

At the bottom of the page, a browser developer tools interface is visible, showing the React component tree and props for the selected "section" element.

```
Props read-only
  ▷ children: {...}
  className: "hero is-info"

  <section className="hero is-info">...</section> == $r
```

# Component Based - Content

ProtoPie Home Version Download Build BuildList mark Log out

## ProtoPie Studio Admin - Version

다운로드 최신 링크 버전 관리

MacOS : 3.6.3	Windows 32 bit : 3.6.3	Windows 64 bit : 3.6.3
<p><a href="https://download protopie io/darwin/latest">https://download protopie io/darwin/latest</a>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.</p> <p><b>Version</b></p> <p>New Latest Version of MacOS</p> <p><b>Update</b></p>	<p><a href="https://download protopie io/win32/latest">https://download protopie io/win32/latest</a>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.</p> <p><b>Version</b></p> <p>New Latest Version of Windows x32</p> <p><b>Update</b></p>	<p><a href="https://download protopie io/win64/latest">https://download protopie io/win64/latest</a>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.</p> <p><b>Version</b></p> <p>New Latest Version of Windows x64</p> <p><b>Update</b></p>

**section | 1680px x 405px**

© 2017 by Studio XD Inc. All rights reserved.

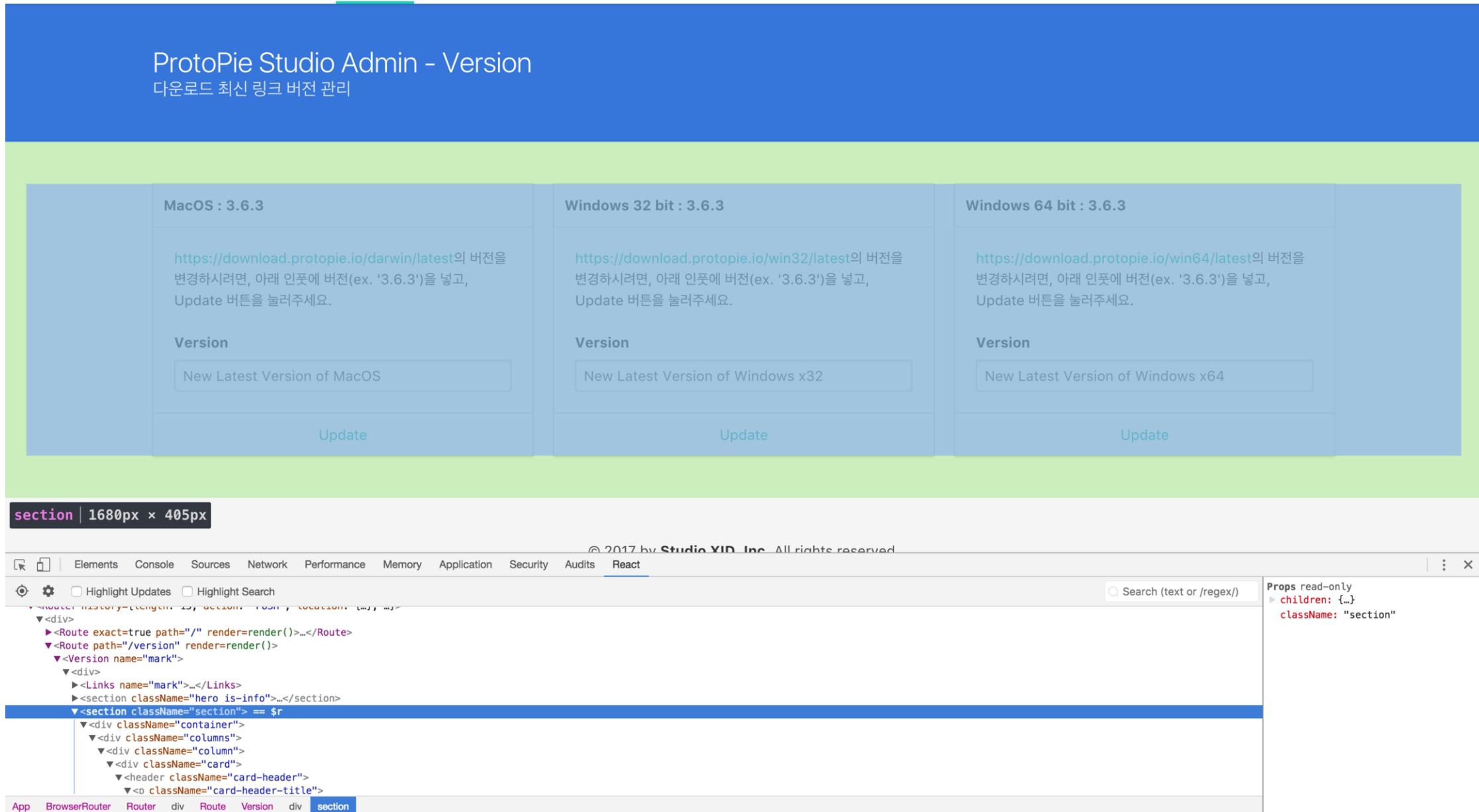
Elements Console Sources Network Performance Memory Application Security Audits React

Highlight Updates Highlight Search

Props read-only  
children: {}  
className: "section"

Search (text or /regex)

App BrowserRouter Router div Route Version div section



# Component Based Development - Card

The screenshot shows the ProtoPie Studio Admin interface for managing versions. It features three main cards:

- MacOS : 3.6.3**

<https://download protopie io/darwin/latest>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.

**Version**  
New Latest Version of MacOS

[Update](#)
- Windows 32 bit : 3.6.3**

<https://download protopie io/win32/latest>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.

**Version**  
New Latest Version of Windows x32

[Update](#)
- Windows 64 bit : 3.6.3**

<https://download protopie io/win64/latest>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.

**Version**  
New Latest Version of Windows x64

[Update](#)

Below the cards, a developer's tool like React DevTools is open, showing the component tree and props for one of the cards. The card component is identified as having a `children` prop and a `className: "column"`.

```
<Route path="/version" render={render}>
  <Version name="mark">
    <div>
      <Links name="mark"/>
      <section className="hero is-info"></section>
      <section className="section">
        <div className="container">
          <div className="columns">
            <div className="column" style={{ border: 1px solid #ccc, padding: 10px }}>
              <header className="card-header">
                <p className="card-header-title">
                  MacOS : 3.6.3
                </p>
              </header>
              <div>
```

# Component Based Development - Card

ProtoPie Studio Admin - Version  
다운로드 최신 링크 버전 관리

**MacOS : 3.6.3**

<https://download.protopie.io/darwin/latest>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.

**Version**

New Latest Version of MacOS

**Update**

**Windows 32 bit : 3.6.3**

<https://download.protopie.io/win32/latest>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.

**Version**

New Latest Version of Windows x32

**Update**

**Windows 64 bit : 3.6.3**

<https://download.protopie.io/win64/latest>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.

**Version**

New Latest Version of Windows x64

**Update**

div | 456px x 333px

© 2017 by Studio XD Inc. All rights reserved.

Elements Console Sources Network Performance Memory Application Security Audits React

Highlight Updates Highlight Search

Search (text or /regex)

Props read-only  
children: {}  
className: "column"

App BrowserRouter Router div Route Version div section div div div

# Component Based Development - Card

ProtoPie Studio Admin - Version  
다운로드 최신 링크 버전 관리

**MacOS : 3.6.3**

<https://download.protopie.io/darwin/latest>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.

**Version**

New Latest Version of MacOS

**Update**

**Windows 32 bit : 3.6.3**

<https://download.protopie.io/win32/latest>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.

**Version**

New Latest Version of Windows x32

**Update**

**Windows 64 bit : 3.6.3**

<https://download.protopie.io/win64/latest>의 버전을 변경하시려면, 아래 인풋에 버전(ex. '3.6.3')을 넣고, Update 버튼을 눌러주세요.

**Version**

New Latest Version of Windows x64

**Update**

div | 456px x 333px

2017 by Studio XD Inc. All rights reserved

Elements Console Sources Network Performance Memory Application Security Audits React

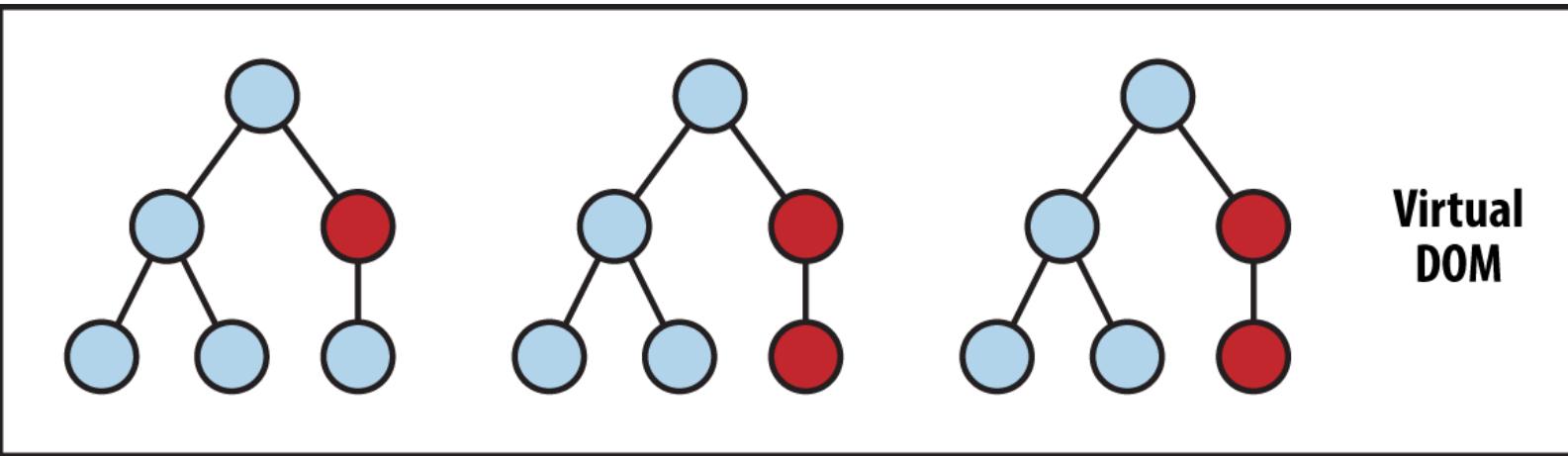
Highlight Updates Highlight Search

Search (text or /regex)

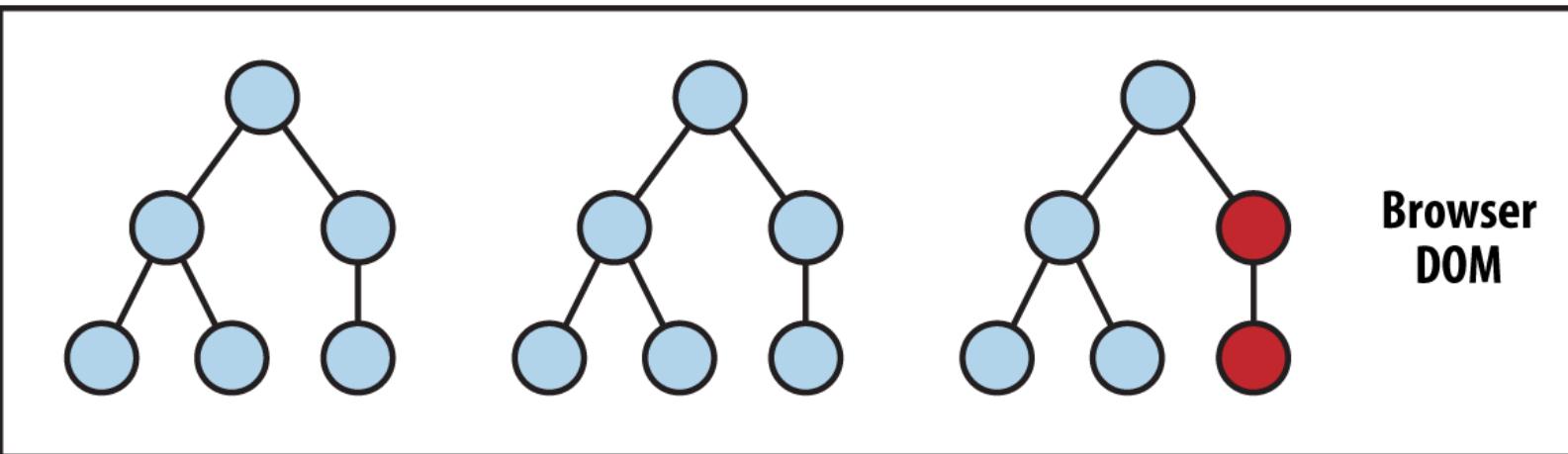
Props read-only  
children: {...}  
className: "column"

App BrowserRouter Router div Route Version div section div div div

# Virtual DOM - diff 로 변경

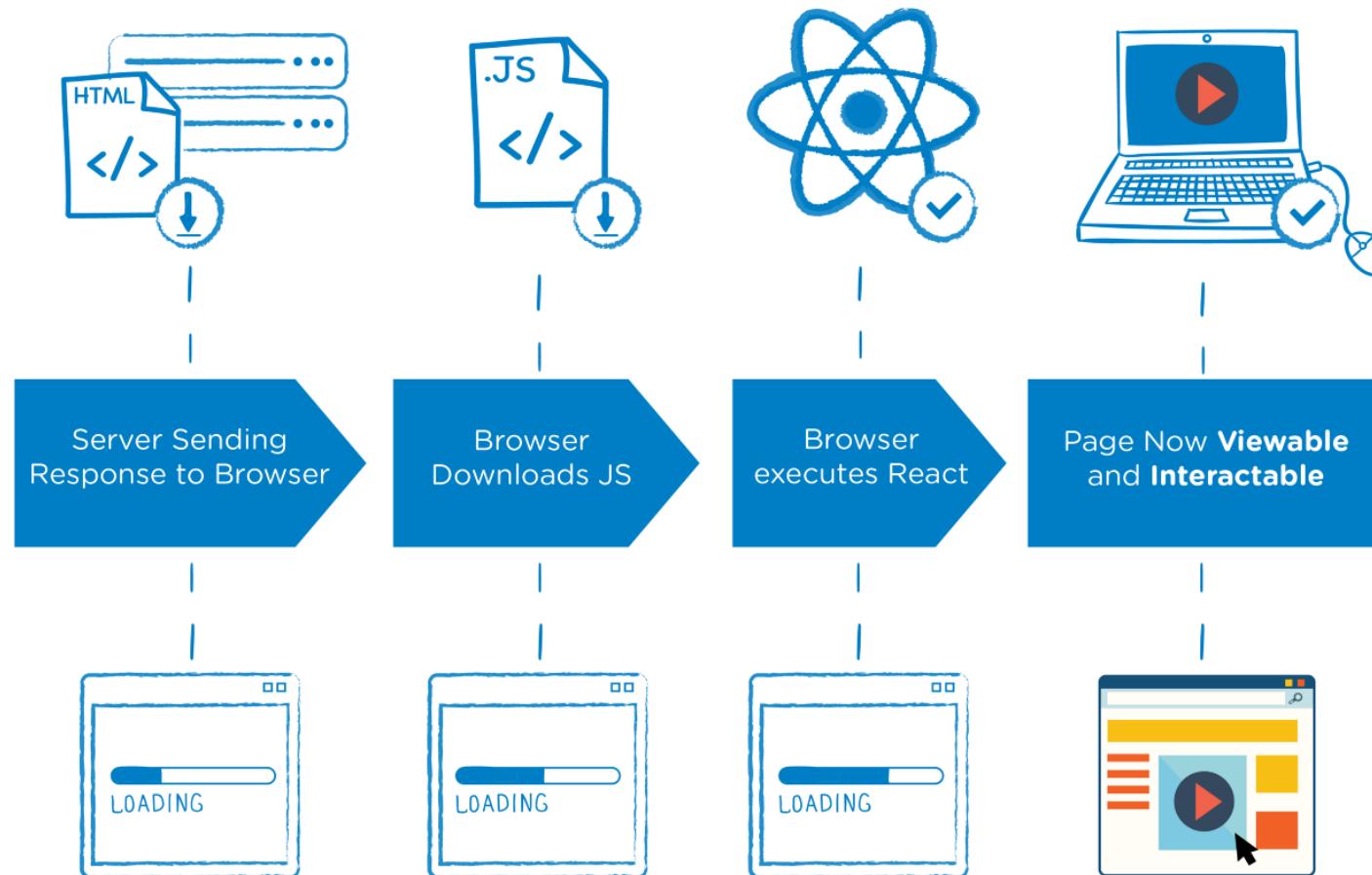


State Change → Compute Diff → Re-render



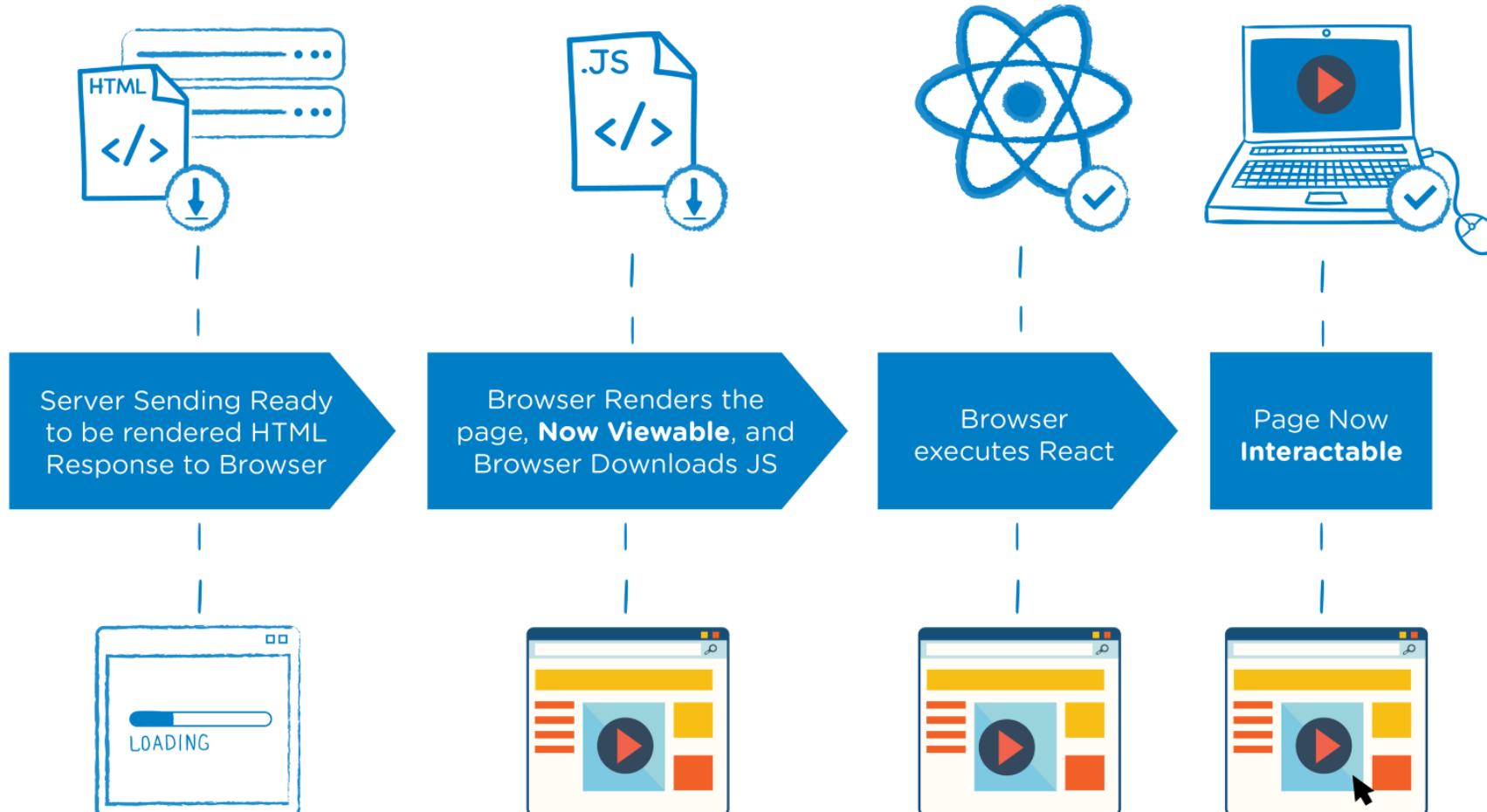
# React Client Side Rendering

CSR



# React Server Side Rendering

SSR

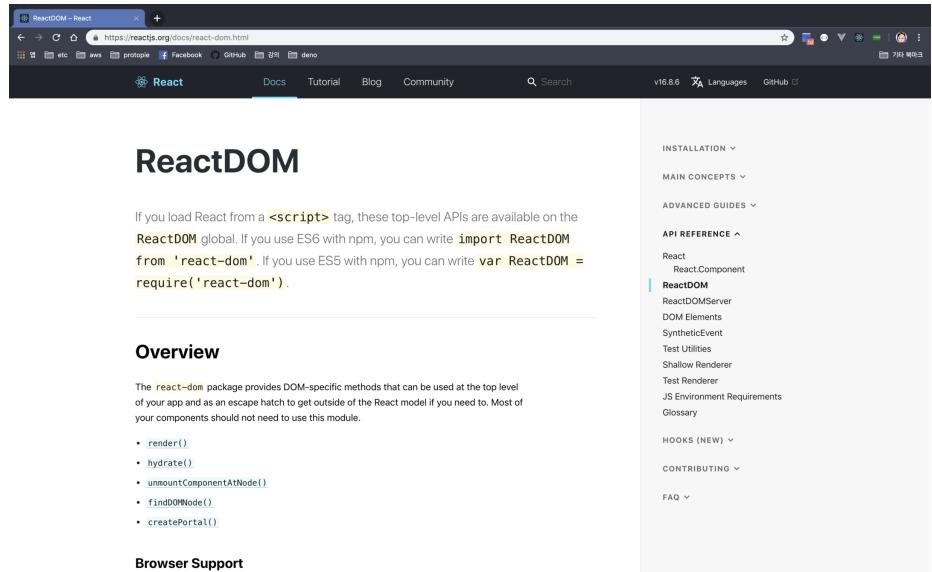


# 리액트가 하는 일

리액트의 핵심 노출된 모듈 2개로 리액트가 하는 일 알아보기

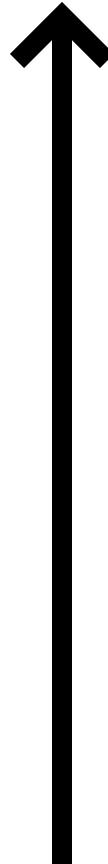
```
import React from 'react';
import ReactDOM from 'react-dom';
```

# < DOM Element >



<https://reactjs.org/docs/react-dom.html>

<https://ko.reactjs.org/docs/react-dom.html>



"만들어진 리액트 컴포넌트"를  
실제 DOM에 연결할 때  
ReactDOM을 이용합니다.

{ React 컴포넌트 }

# { React 컴포넌트 } 만들기

리액트 컴포넌트를 만들 때 사용하는 API 모음

The screenshot shows a browser window with the React.js documentation at <https://reactjs.org/docs/react-api.html>. The page title is "React Top-Level API". The content discusses the entry point to the React library and how to use top-level APIs. It includes sections for "Overview" and "Components". A sidebar on the right contains links to various React API reference pages like React, React.Component, and ReactDOM.

**React Top-Level API**

React is the entry point to the React library. If you load React from a `<script>` tag, these top-level APIs are available on the `React` global. If you use ES6 with npm, you can write `import React from 'react'`. If you use ES5 with npm, you can write `var React = require('react')`.

## Overview

### Components

React components let you split the UI into independent, reusable pieces, and think about each piece in isolation. React components can be defined by subclassing `React.Component` or `React.PureComponent`.

- `React.Component`
- `React.PureComponent`

If you don't use ES6 classes, you may use the `create-react-class` module instead. See Using React without ES6 for more information.

React components can also be defined as functions which can be wrapped:

INSTALLATION ▾  
MAIN CONCEPTS ▾  
ADVANCED GUIDES ▾  
API REFERENCE ▾

- React
  - React.Component
  - ReactDOM
  - ReactDOMServer
  - DOM Elements
  - SyntheticEvent
  - Test Utilities
  - Shallow Renderer
  - Test Renderer
  - JS Environment Requirements
  - Glossary
- HOOKS (NEW) ▾
- CONTRIBUTING ▾
- FAQ ▾

<https://reactjs.org/docs/react-api.html>

<https://ko.reactjs.org/docs/react-api.html>

# Use React, ReactDOM Library with CDN

CDN 을 통한 리액트 라이브러리 사용

The screenshot shows a browser window displaying the React.js documentation at <https://reactjs.org/docs/cdn-links.html>. The page title is "CDN Links". The main content explains that both React and ReactDOM are available over a CDN, providing development versions via Unpkg:

```
<script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
```

It notes that these are for development only and provides links to minified and optimized production versions:

```
<script crossorigin src="https://unpkg.com/react@16/umd/react.production.min.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>
```

To load a specific version, it suggests replacing the version number.

### Why the crossorigin Attribute?

If you serve React from a CDN, we recommend to keep the `crossorigin` attribute set:

```
<script crossorigin src="..."></script>
```

We also recommend to verify that the CDN you are using sets the `Access-Control-Allow-Origin: *` HTTP header.

The sidebar on the right contains navigation links for installation, main concepts, advanced guides, API reference, hooks, contributing, and FAQ.

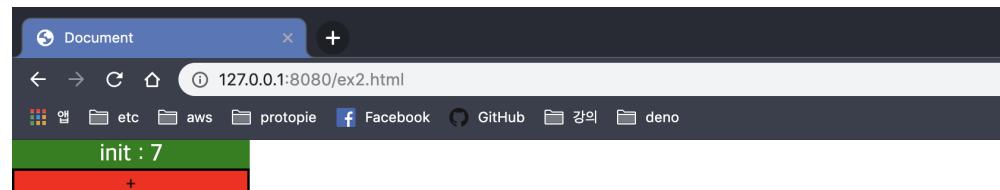
<https://reactjs.org/docs/cdn-links.html>

<https://ko.reactjs.org/docs/cdn-links.html>

```
1 <!-- ex1.html : CDN 을 통해 React, ReactDOM 가져오기 -->
2 <!DOCTYPE html>
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
8     <title>Document</title>
9   </head>
10  <body>
11    <script
12      crossorigin
13      src="https://unpkg.com/react@16/umd/react.development.js"
14    ></script>
15    <script
16      crossorigin
17      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
18    ></script>
19    <script type="text/javascript">
20      // Global 에 React 와 ReactDOM 객체가 생성
21      console.log(React);
22      console.logReactDOM();
23    </script>
24  </body>
25 </html>
```

# 고전 프론트엔드

HTML로 문서 구조를 잡고,  
CSS로 스타일을 입히고,  
JavaScript로 DOM을 조작합니다.

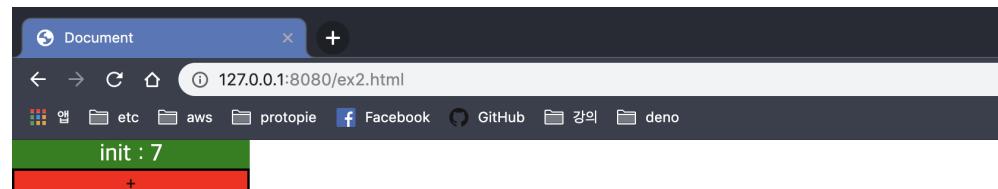


```
1 <!-- ex2.html : 고전 프론트엔드 -->
2 <!DOCTYPE html>
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
8     <title>Document</title>
9     <style>
10       * {
11         margin: 0;
12         padding: 0;
13         border: 0;
14       }
15       #root p {
16         color: white;
17         font-size: 20px;
18         background-color: green;
19         text-align: center;
20         width: 200px;
21       }
22       #btn_plus {
23         background-color: red;
24         border: 2px solid #000000;
25         font-size: 15px;
26         width: 200px;
27       }
28     </style>
29   </head>
30   <body>
31     <div id="root"></div>
32     <button id="btn_plus">+</button>
33
34     <script type="text/javascript">
35       const root = document.querySelector("#root");
36       const btn_plus = document.querySelector("#btn_plus");
37 
```

# 컴포넌트를 활용한 프론트엔드

컴포넌트를 정의하고,

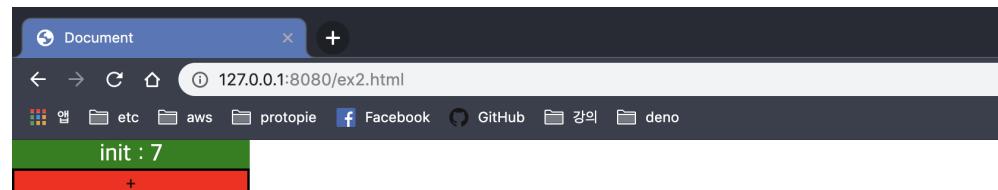
실제 DOM 에 컴포넌트를 그려준다.



```
1 <!-- ex3.html : 컴포넌트를 만들고, 실제 DOM 에 그린다. -->
2 <!DOCTYPE html>
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
8     <title>Document</title>
9     <style>
10       * {
11         margin: 0;
12         padding: 0;
13         border: 0;
14       }
15       #root p {
16         color: white;
17         font-size: 20px;
18         background-color: green;
19         text-align: center;
20         width: 200px;
21       }
22       #btn_plus {
23         background-color: red;
24         border: 2px solid #000000;
25         font-size: 15px;
26         width: 200px;
27       }
28     </style>
29   </head>
30   <body>
31     <div id="root"></div>
32     <button id="btn_plus">+</button>
33
34     <script type="text/javascript">
35       // react 라이브러리가 하는 일
36       const component = {
37         message: "init",
```

# React 프론트엔드

컴포넌트를 정의하고,  
실제 DOM 에 컴포넌트를 그려준다.



# 브라우저 지원

## 브라우저 지원

React는 Internet Explorer 9과 상위 버전을 포함한 모든 주요 브라우저를 지원합니다. 그러나 IE 9과 IE 10과 같은 구형 브라우저는 폴리필(polyfill)이 필요합니다.

### 주의

우리는 ES5 메서드를 지원하지 않는 구형 브라우저를 지원하지 않지만, 페이지에 es5-shim과 es5-sham과 같은 폴리필을 포함한다면 앱이 구형 브라우저에서도 동작할 수 있습니다. 이 길을 선택한다면 스스로 해결해야 합니다.

```
1 <!-- ex4.html : React 로 컴포넌트를 만들고, 실제 DOM 에 그린다. -->
2 <!DOCTYPE html>
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
8     <title>Document</title>
9     <style>
10       * {
11         margin: 0;
12         padding: 0;
13         border: 0;
14       }
15       #root p {
16         color: white;
17         font-size: 20px;
18         background-color: green;
19         text-align: center;
20         width: 200px;
21       }
22       #btn_plus {
23         background-color: red;
24         border: 2px solid #000000;
25         font-size: 15px;
26         width: 200px;
27       }
28     </style>
29   </head>
30   <body>
31     <div id="root"></div>
32     <button id="btn_plus">+</button>
33
34     <script
35       crossorigin
36       src="https://unpkg.com/react@16/umd/react.development.js"
37     ></script>
```

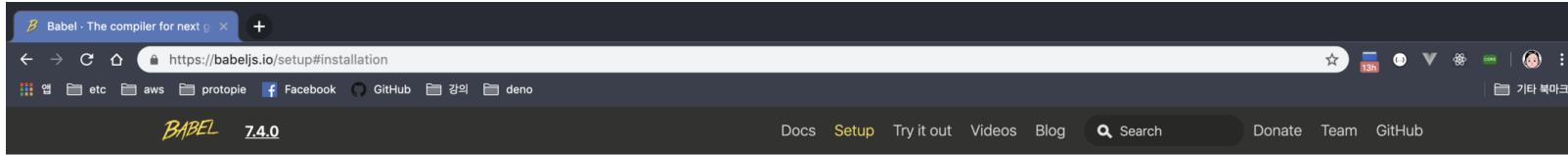
# React.createElement

순수 JavaScript (그렇다면 순수하지 않은 것은??)

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <title>Document</title>
8   </head>
9   <body>
10    <div id="root"></div>
11    <script
12      crossorigin
13      src="https://unpkg.com/react@16/umd/react.development.js"
14    ></script>
15    <script
16      crossorigin
17      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
18    ></script>
19    <script type="text/javascript">
20      // React.createElement(
21      //   type, // 태그 이름 문자열 | React 컴포넌트 | React.Fragment
22      //   [props], // 리액트 컴포넌트에 넣어주는 데이터 객체
23      //   [...children] // 자식으로 넣어주는 요소들
24      // );
25
26      // 1. 태그 이름 문자열 type
27      // ReactDOM.render(
28      //   React.createElement('h1', null, `type 이 "태그 이름 문자열" 입니다.`),
29      //   document.getElementById('root')
30    </script>
```

# JSX

JSX 문법으로 작성된 코드는 순수한 JavaScript로 컴파일하여 사용한다.  
누가 해주나요?? => babel



## Using Babel

How to use Babel with your tool of choice.

### 1 Choose your tool (try CLI)

#### Prototyping

In the browser

#### Babel built-ins

CLI Require hook

#### Build systems

Broccoli Browserify Brunch Duo Gobble Grunt Gulp jspm Make MSBuild RequireJS Rollup Sprockets Webpack Fly Start

#### Frameworks

Ember Meteor Rails Sails

#### Test frameworks

AVA Jasmine Jest Karma Lab Mocha

#### Utilities

Connect Nodemon

#### Language APIs

C# / .NET Node Ruby

#### Template engines

Pug

#### Editors and IDEs

WebStorm

# JSX

## JSX 문법 => React.createElement

```
1 <!-- ex6.html : React.createElement => JSX -->
2 <!DOCTYPE html>
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
8     <title>Document</title>
9   </head>
10  <body>
11    <div id="root"></div>
12    <script
13      crossorigin
14      src="https://unpkg.com/react@16/umd/react.development.js"
15    ></script>
16    <script
17      crossorigin
18      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
19    ></script>
20    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
21
22    <script type="text/babel">
23      //   React.createElement(
24      //     type, // 태그 이름 문자열 | React 컴포넌트 | React.Fragment
25      //     [props], // 리액트 컴포넌트에 넣어주는 데이터 객체
26      //     [...children] // 자식으로 넣어주는 요소들
27      //   );
28
```

# 왜 JSX 을 쓰나요??

- React.createElement VS JSX
  - 가독성 완승
- babel 과 같은 컴파일 과정에서 문법적 오류를 인지하기 쉬움

# JSX 문법

- 최상위 요소가 하나여야 합니다.
- 최상위 요소 리턴하는 경우, ()로 감싸야 합니다.
- 자식들을 바로 랜더링하고 싶으면, <>자식들</>를 사용합니다. => Fragment
- 자바스크립트 표현식을 사용하려면, {표현식}를 이용합니다.
- if 문은 사용할 수 없습니다.
  - 삼항 연산자 혹은 &&를 사용합니다.
- style 을 이용해 인라인 스타일링이 가능합니다.
- class 대신 className 을 사용해 class 를 적용할 수 있습니다.
- 자식요소가 있으면, 꼭 닫아야 하고, 자식요소가 없으면 열면서 닫아야 합니다.
  - <p>어쩌구</p>
  - <br />

# JSX 문법

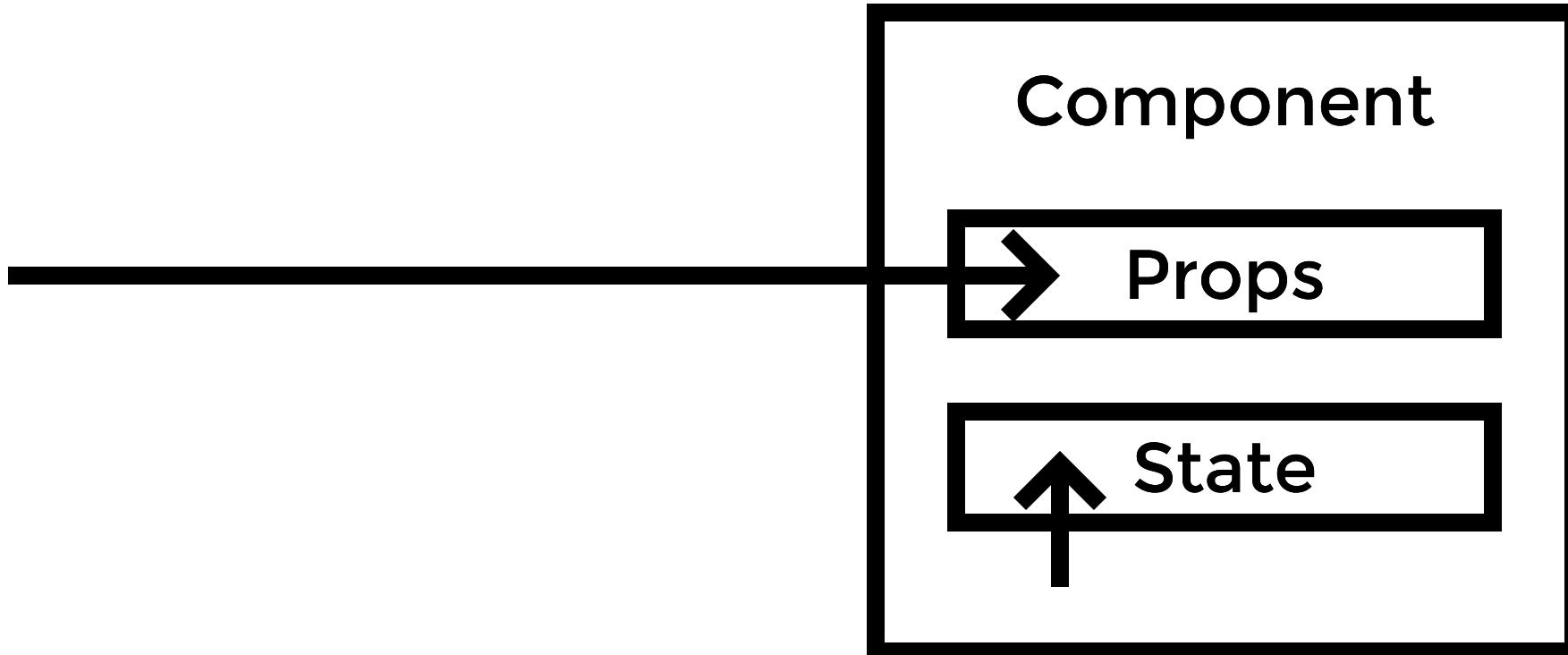
```
1 <!-- ex7.html : JSX 문법 -->
2 <!DOCTYPE html>
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
8     <title>Document</title>
9   </head>
10  <body>
11    <div id="root"></div>
12    <script
13      crossorigin
14      src="https://unpkg.com/react@16/umd/react.development.js"
15    ></script>
16    <script
17      crossorigin
18      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
19    ></script>
20    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
21
22    <script type="text/babel">
23      // 1. 최상위 요소가 하나여야 합니다.
24      // 2. 최상위 요소 리턴하는 경우, ( ) 로 감싸야 합니다.
25      //   const Comp1 = props => {
26      //     return (
27      //       <h1>제목</h1>
28      //       <h2>부제목</h2>
29      //     );
30      //   }
31      //   const Comp2 = props => {
32      //     return (
33      //       <div>
34      //         <h1>제목</h1>
```

# Props 와 State

Props 는 컴포넌트 외부에서 컴포넌트에게 주는 데이터입니다.

State 는 컴포넌트 내부에서 변경할 수 있는 데이터입니다.

둘 다 변경이 발생하면, 랜더가 다시 일어날 수 있습니다.



# class 컴포넌트 extends React.Component

```
1 <!-- ex8.html : 클래스로 리액트 컴포넌트 만들기 --><!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <title>Document</title>
8   </head>
9   <body>
10    <div id="root"></div>
11    <script
12      crossorigin
13      src="https://unpkg.com/react@16/umd/react.development.js"
14    ></script>
15    <script
16      crossorigin
17      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
18    ></script>
19    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
20
21    <script type="text/babel">
22      class Component extends React.Component {
23        state = {
24          count: 0,
25        };
26        render() {
27          return (
28            <div>
29              <h1>
30                {this.props.message} 이것은 클래스를 상속하여 만든 컴포넌트
31                입니다.
32              </h1>
33              <p>count는 {this.state.count} 입니다.</p>
34            </div>
```

# <Component p="프롭스" />

*props* 설정

## this.props.p 로 접근

*props* 사용

```
class Component extends React.Component {
  render() {
    return (
      <div>{this.props.p}</div>
    );
  }
}
```

# <Component p="프롭스" />

*props* 설정

## this.props.p 로 접근

*props* 사용

```
class Component extends React.Component {  
  render() {  
    return (  
      <div>{this.props.p}</div>  
    );  
  }  
}
```

# defaultProps

```
<!-- ex9.html : defaultProps 설정 -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Document</title>
  </head>
  <body>
    <div id="root"></div>
    <script
      crossorigin
      src="https://unpkg.com/react@16/umd/react.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
    ></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <script type="text/babel">
      class Component extends React.Component {
        static defaultProps = {
          message: '안녕하세요!!!',
        };
        render() {
          return (
            <div>
              {this.props.message} 이것은 클래스를 상속하여 만든 컴포넌트
              입니다.
            </div>
          );
        }
      }
    </script>
  </body>
</html>
```

# state = {}; constructor

*state 초기값 설정*

```
class Component extends React.Component {  
  state = {  
    s: '스테이트'  
  };  
  render() {  
    return (  
      <div>{this.state.s}</div>  
    );  
  }  
}
```

```
class Component extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = {s: '스테이트'};  
  }  
  render() {  
    return (  
      <div>{this.state.s}</div>  
    );  
  }  
}
```

# this.state.s로 접근

*state 사용*

# this.setState({s: '새 스테이트'});

*state 값 업데이트*

```
class Component extends React.Component {
  state = {
    s: '스테이트'
  };
  render() {
    return (
      <div onClick={() => {
        this.setState({s: '새 스테이트'});
      }}>{this.state.s}</div>
    );
  }
}
```

# Event Handling

HTML DOM 에 클릭하면 이벤트가 발생하고, 발생하면 그에 맞는 변경이 일어나도록 해야합니다.  
JSX 에 이벤트를 설정할 수 있습니다.

```
1 class Comp extends React.Component {
2   render() {
3     return (
4       <div>
5         <button onClick={() => {
6           console.log('clicked');
7         }}>클릭</button>
8       </div>
9     );
10   }
11 }
```

# Event Handling

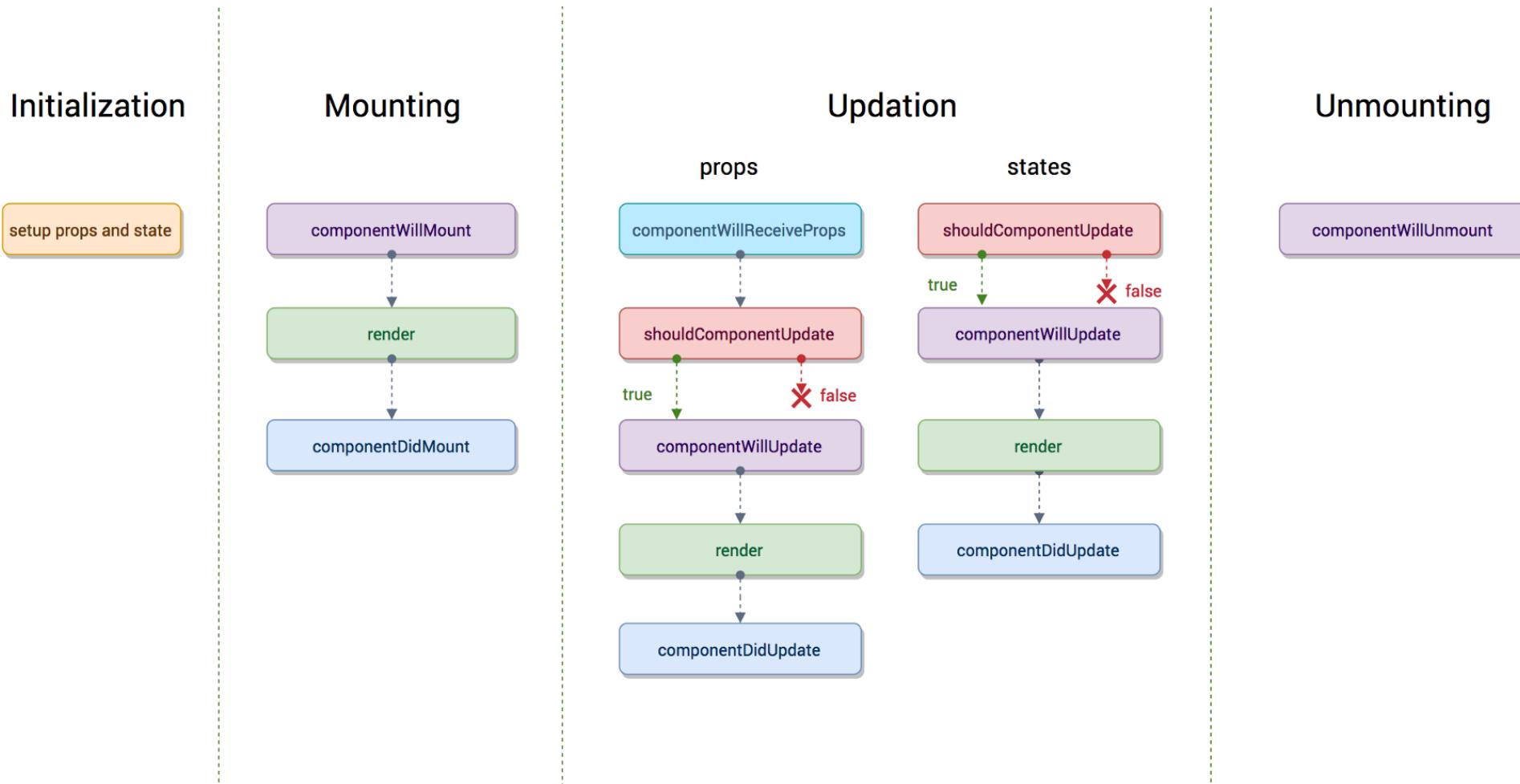
- camelCase 로만 사용할 수 있습니다.
  - onClick
- 이벤트에 연결된 자바스크립트 코드는 함수입니다.
  - 이벤트={함수} 와 같이 씁니다.
- 실제 DOM 요소들에만 사용 가능합니다.
  - 리액트 컴포넌트에 사용하면, 그냥 props 로 전달합니다.

```
1 <!-- ex10.html : 이벤트를 이용하여 state 바꾸기 -->
2 <!DOCTYPE html>
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
8     <title>Document</title>
9     <style>
10       * {
11         margin: 0;
12         padding: 0;
13         border: 0;
14       }
15     </style>
16   </head>
17   <body>
18     <div id="root"></div>
19
20     <script
21       crossorigin
22       src="https://unpkg.com/react@16/umd/react.development.js"
23     ></script>
24     <script
25       crossorigin
26       src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
27     ></script>
28     <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
29
30     <script type="text/babel">
31       class Component extends React.Component {
32         state = { message: 'init', count: 0 };
33         render() {
34           return (
35             <>
36               <p
37                 style={{
```

# Component Lifecycle

리액트 컴포넌트는 탄생부터 죽음까지  
여러지점에서 개발자가 작업이 가능하도록  
메서드를 오버라이딩 할 수 있게 해준다.

# Declarative 디클레러티브



Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

# Component 생성 및 마운트

```
<!-- ex11.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Document</title>
    <style>
      * {
        margin: 0;
        padding: 0;
        border: 0;
      }
    </style>
  </head>
  <body>
    <div id="root"></div>

    <script
      crossorigin
      src="https://unpkg.com/react@16/umd/react.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
    ></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <script type="text/babel">
      class App extends React.Component {
        _interval;

        constructor(props) {
```

constructor

componentWillMount

render

componentDidMount

# Component props, state 변경

```
componentWillReceiveProps(nextProps) {
  console.log(
    `App componentWillReceiveProps : ${JSON.stringify(nextProps)}`,
  );
}

shouldComponentUpdate(nextProps, nextState): boolean {
  console.log(
    `App shouldComponentUpdate : ${JSON.stringify(
      nextProps,
    )}, ${JSON.stringify(nextState)}`,
  );
  return true;
}

componentWillUpdate(nextProps, nextState) {
  console.log(
    `App componentWillUpdate : ${JSON.stringify(
      nextProps,
    )}, ${JSON.stringify(nextState)}`,
  );
}

componentDidUpdate(prevProps, prevState) {
  console.log(
    `App componentDidUpdate : ${JSON.stringify(
      prevProps,
    )}, ${JSON.stringify(prevState)}`,
  );
}
```

componentWillReceiveProps

shouldComponentUpdate

componentWillUpdate

render

componentDidUpdate

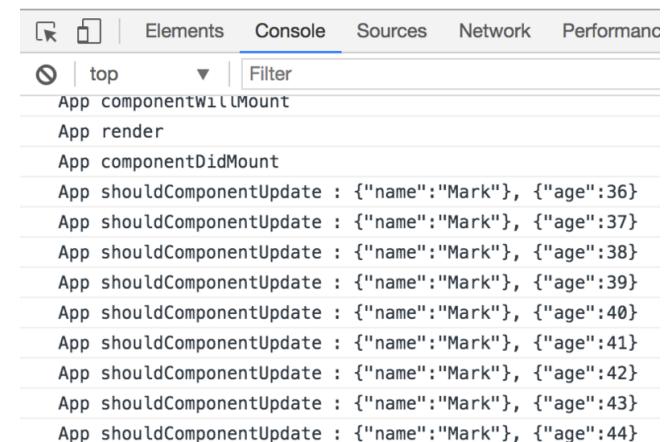
# componentWillReceiveProps

- **props** 를 새로 지정했을 때 바로 호출됩니다.
- 여기는 **state** 의 변경에 반응하지 않습니다.
  - 여기서 **props** 의 값에 따라 **state** 를 변경해야 한다면,
    - **setState** 를 이용해 **state** 를 변경합니다.
    - 그러면 다음 이벤트로 각각 가는것이 아니라 한번에 변경됩니다.

# shouldComponentUpdate

Hello Mark - 35

- props 만 변경되어도
- state 만 변경되어도
- props & state 둘다 변경되어도
- newProps 와 newState 를 인자로 해서 호출
- return type 이 boolean 입니다.
  - true 면 render
  - false 면 render 가 호출되지 않습니다.
  - 이 함수를 구현하지 않으면, 디폴트는 true



```
App componentWillMount
App render
App componentDidMount
App shouldComponentUpdate : {"name":"Mark"}, {"age":36}
App shouldComponentUpdate : {"name":"Mark"}, {"age":37}
App shouldComponentUpdate : {"name":"Mark"}, {"age":38}
App shouldComponentUpdate : {"name":"Mark"}, {"age":39}
App shouldComponentUpdate : {"name":"Mark"}, {"age":40}
App shouldComponentUpdate : {"name":"Mark"}, {"age":41}
App shouldComponentUpdate : {"name":"Mark"}, {"age":42}
App shouldComponentUpdate : {"name":"Mark"}, {"age":43}
App shouldComponentUpdate : {"name":"Mark"}, {"age":44}
```

# componentWillUpdate

- 컴포넌트가 재 랜더링 되기 직전에 불립니다.
- 여기선 `setState` 같은 것을 쓰면 아니됩니다.

# componentDidUpdate

- 컴포넌트가 재 랜더링을 마치면 불립니다.

npx: 91개의 패키지를 4.178초만에 설치했습니다.

Creating a new React app in `/Users/mark/fastcampus/react-camp/tic-tac-toe`.

Installing packages. This might take a couple of minutes.

Installing `react`, `react-dom`, and `react-scripts`...

```
> fsevents@1.2.9 install /Users/mark/fastcampus/react-camp/tic-tac-toe/node_modules/chokidar/node_modules/fsevents  
> node install
```

[fsevents] Success: "/Users/mark/fastcampus/react-camp/tic-tac-toe/node\_modules/chokidar/node\_modules/fsevents/lib/binding/Release/node-v64-darwin-x64/fse.node" is installed via remote

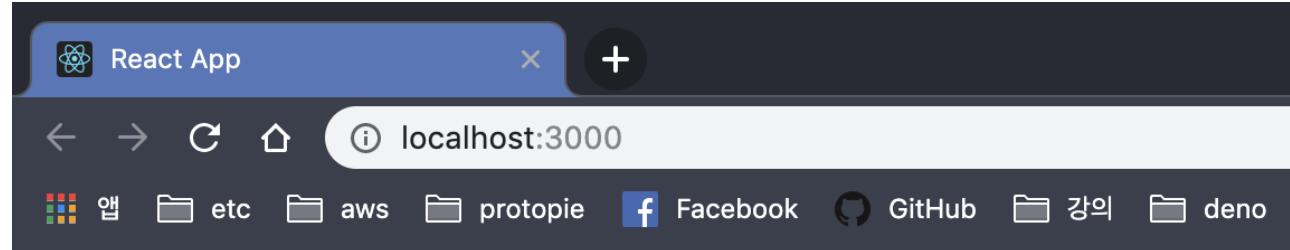
```
> fsevents@1.2.9 install /Users/mark/fastcampus/react-camp/tic-tac-toe/node_modules/jest-haste-map/node_modules/fsevents  
> node install
```

[fsevents] Success: "/Users/mark/fastcampus/react-camp/tic-tac-toe/node\_modules/jest-haste-map/node\_modules/fsevents/lib/binding/Release/node-v64-darwin-x64/fse.node" is installed via remote

```
> core-js@2.6.9 postinstall /Users/mark/fastcampus/react-camp/tic-tac-toe/node_modules/babel-runtime/node_modules/core-js  
> node scripts/postinstall || echo "ignore"
```

```
> core-js-pure@3.1.4 postinstall /Users/mark/fastcampus/react-camp/tic-tac-toe/node_modules/core-js-pure  
> node scripts/postinstall || echo "ignore"
```

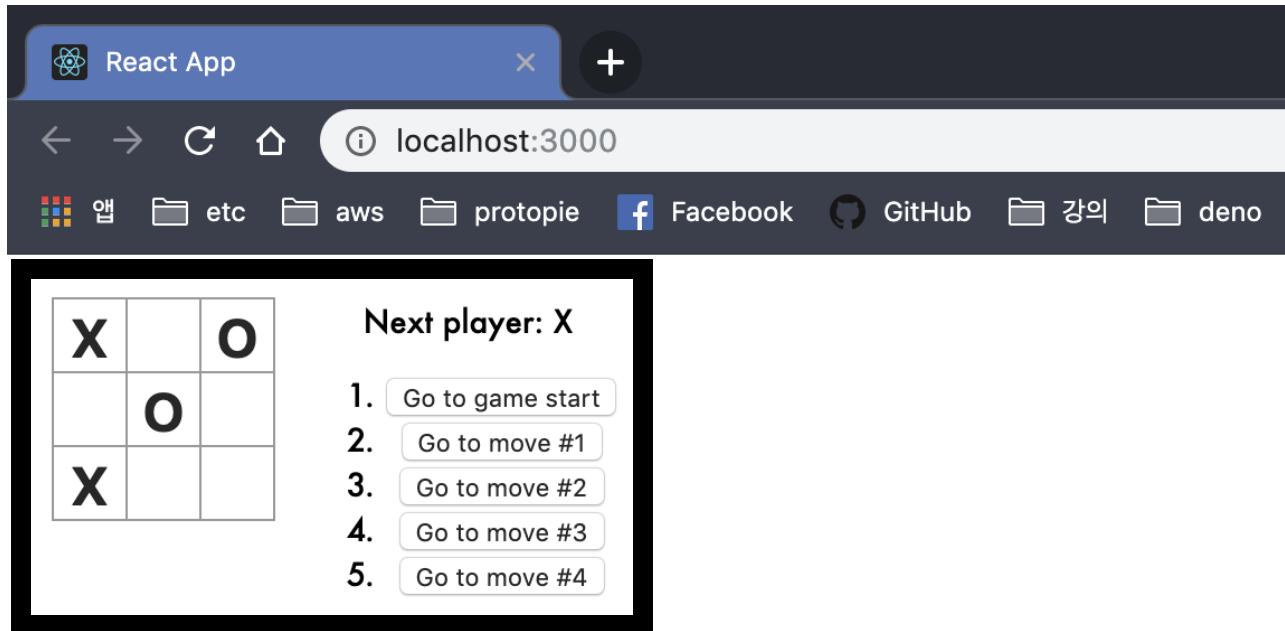
```
+ react@16.8.6  
+ react-dom@16.8.6
```



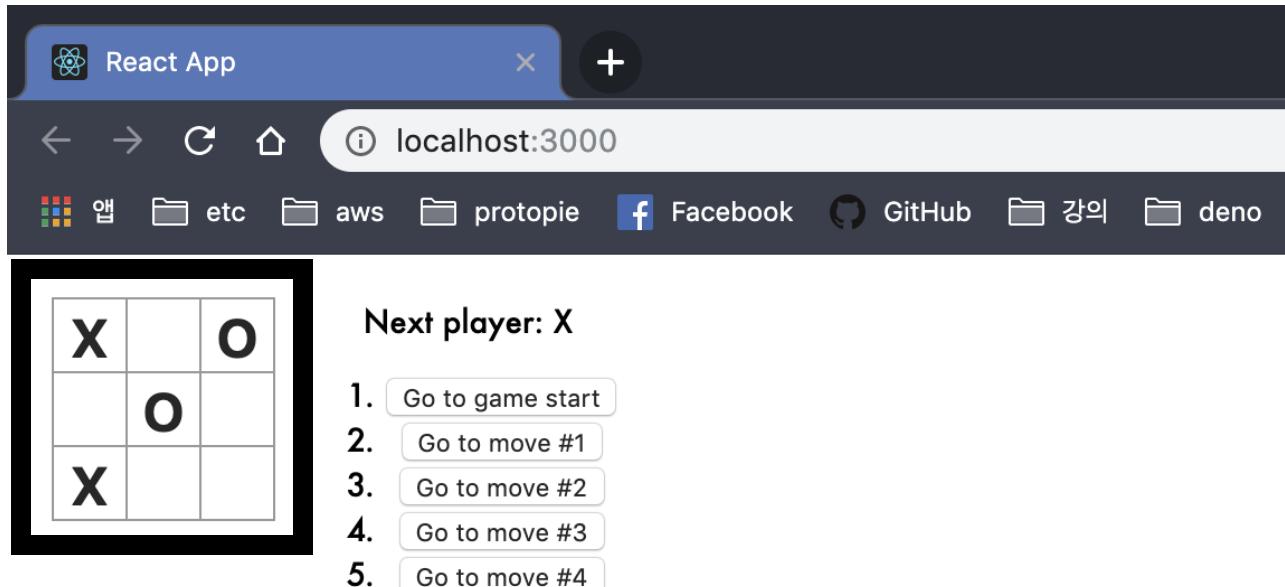
X		O
	O	
X		

Next player: X

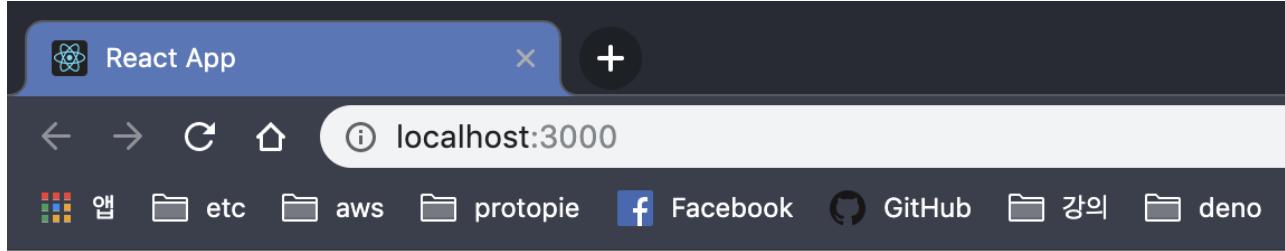
1. Go to game start
2. Go to move #1
3. Go to move #2
4. Go to move #3
5. Go to move #4



Game Componenet



## Board Componenet



Square Componenet

## Git Repository

<https://github.com/2woongjae/before-reactjs>

<https://github.com/2woongjae/react-syntax>

<https://github.com/2woongjae/what-is-react>

<https://github.com/2woongjae/tic-tac-toe>