# 1. 리액트 시작하기

이건 짚고 넘어갑시다…

# 왜 리액트냐?

# 소개

컴포넌트는, 결국엔 함수!

# 데이터가 들어가면 뷰가 나와요

*We built React to solve one problem: building large applications with data that changes over time.*

# 데이터의 변화

```
{
    "title": "Hello",
    "contents": "Hello World",
    "author": "velopert"
    "likes": 1
}
```

```
<div id="post-1">
    <div class="title">Hello</div>
    <div class="contents">Hello World</div>
    <div class="author">velopert</div>
    <div class="likes">1</div>
</div>
```

# 페이스북의 해결법:
# 다 밀어버리고 새로 만들어!

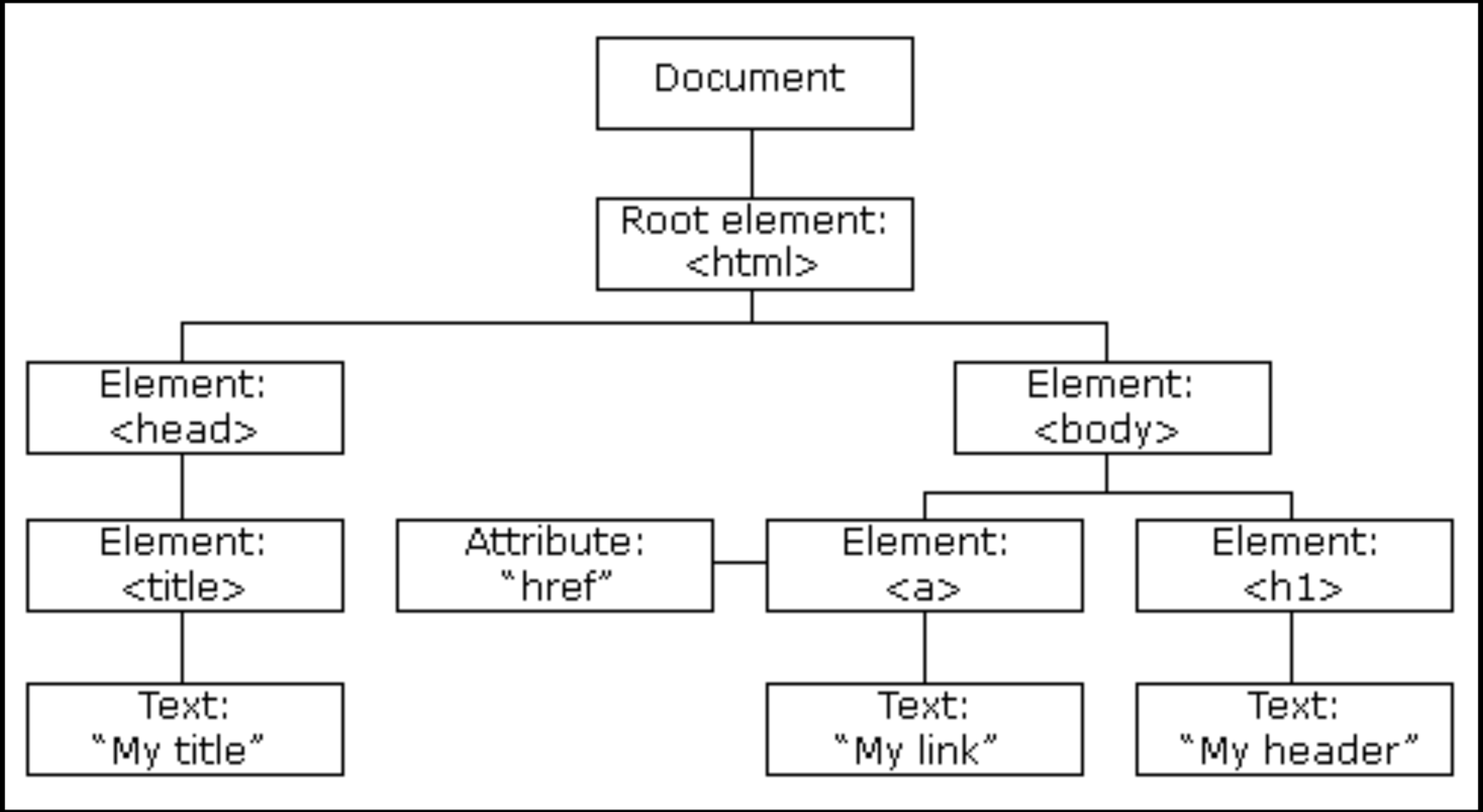
# 과연 이게 될까..?

# Virtual DOM

https://www.youtube.com/watch?v=muc2ZF0QIO4

# DOM?

# Document Object Model


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

# DOM 의 문제점

# 브라우저 레이아웃 엔진

# Repaint & Reflow

```
var style = document.body.style; // 캐싱

style.padding = "20px"; // reflow, repaint
style.border = "10px solid red"; // reflow, repaint

style.color = "blue"; // repaint (레이아웃이 변경되진 않았기 때문에 reflow 안함)
style.backgroundColor = "#ffa"; // repaint

style.fontSize = "1em"; // reflow, repaint

// reflow, repaint
document.body.appendChild(document.createTextNode('hello world!'));
```

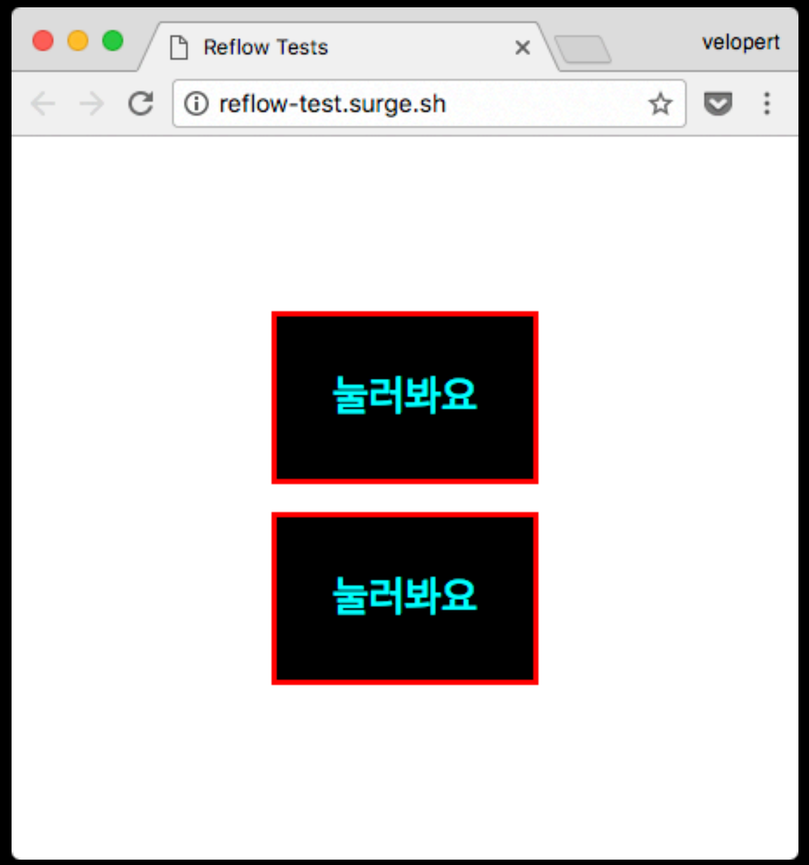
# 브라우저는 바보가 아니다

# Batched DOM Updates

```
var style = document.body.style; // 캐싱

style.padding = "20px"; // reflow, repaint
style.border = "10px solid red"; // reflow, repaint

style.color = "blue"; // repaint (레이아웃이 변경되진 않았기 때문에 reflow 안함)
style.backgroundColor = "#ffa"; // repaint

style.fontSize = "1em"; // reflow, repaint

// reflow, repaint
document.body.appendChild(document.createTextNode('hello world!'));
```

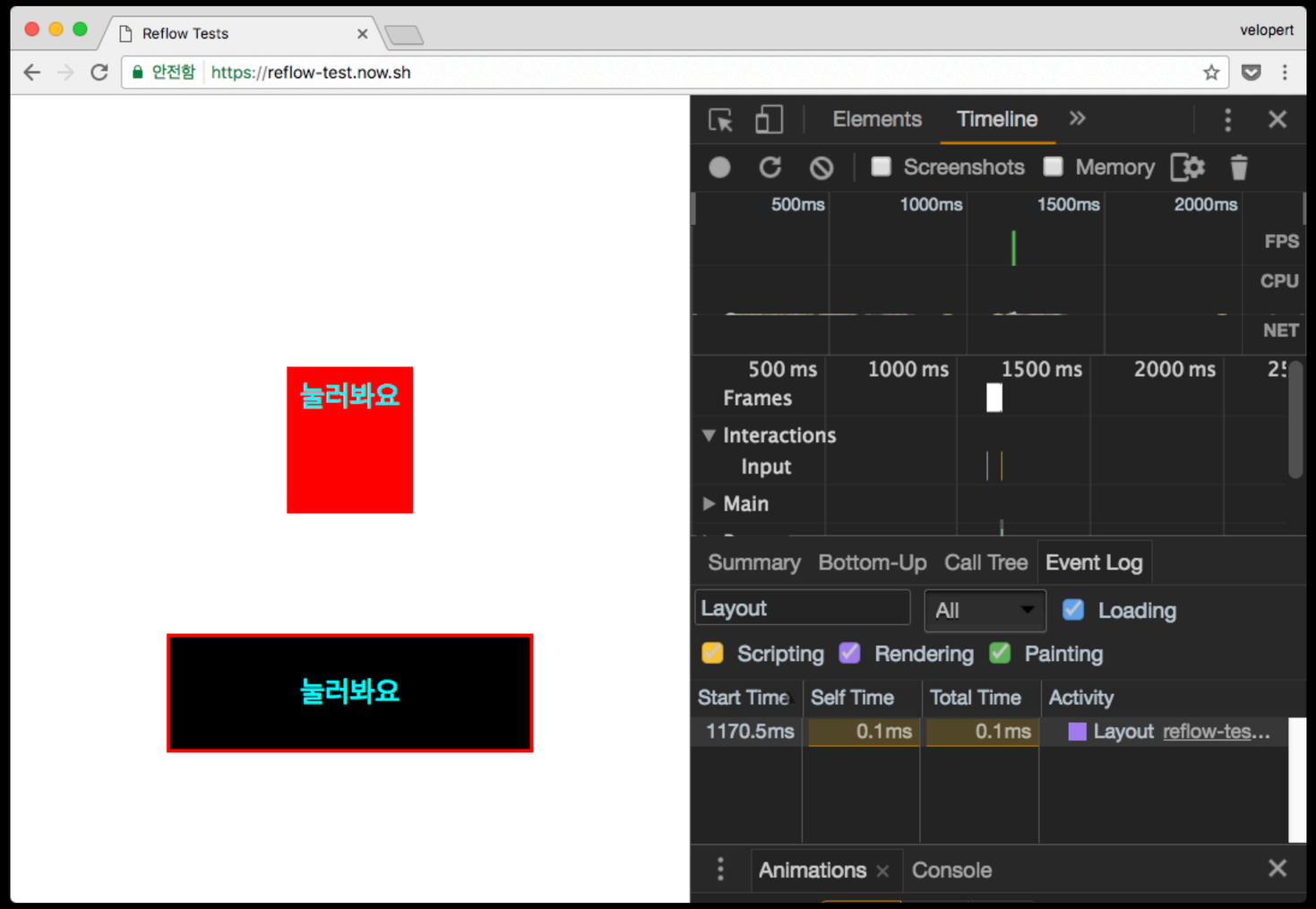
# Forced Reflow

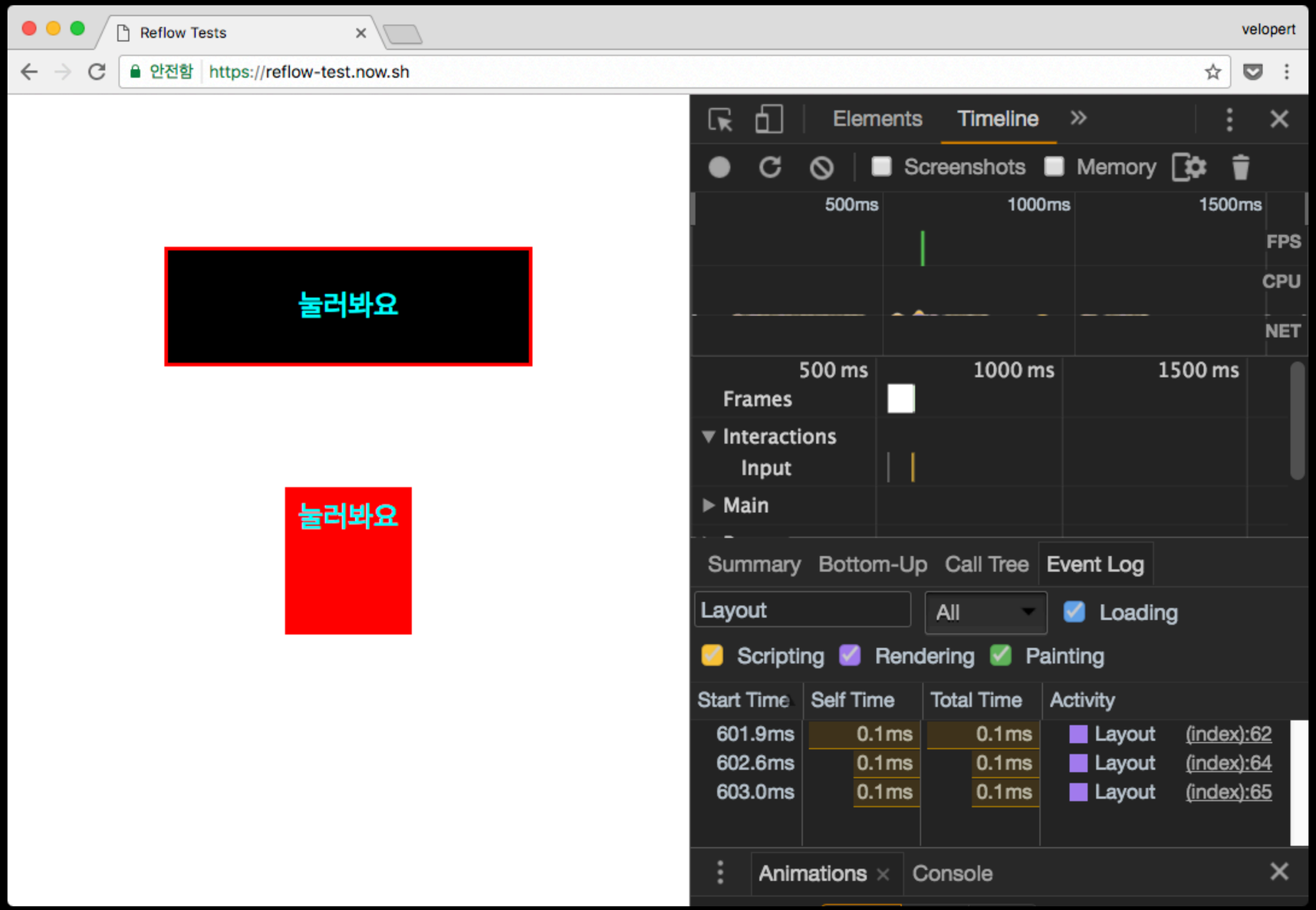
https://gist.github.com/paulirish/5d52fb081b3570c81e3a

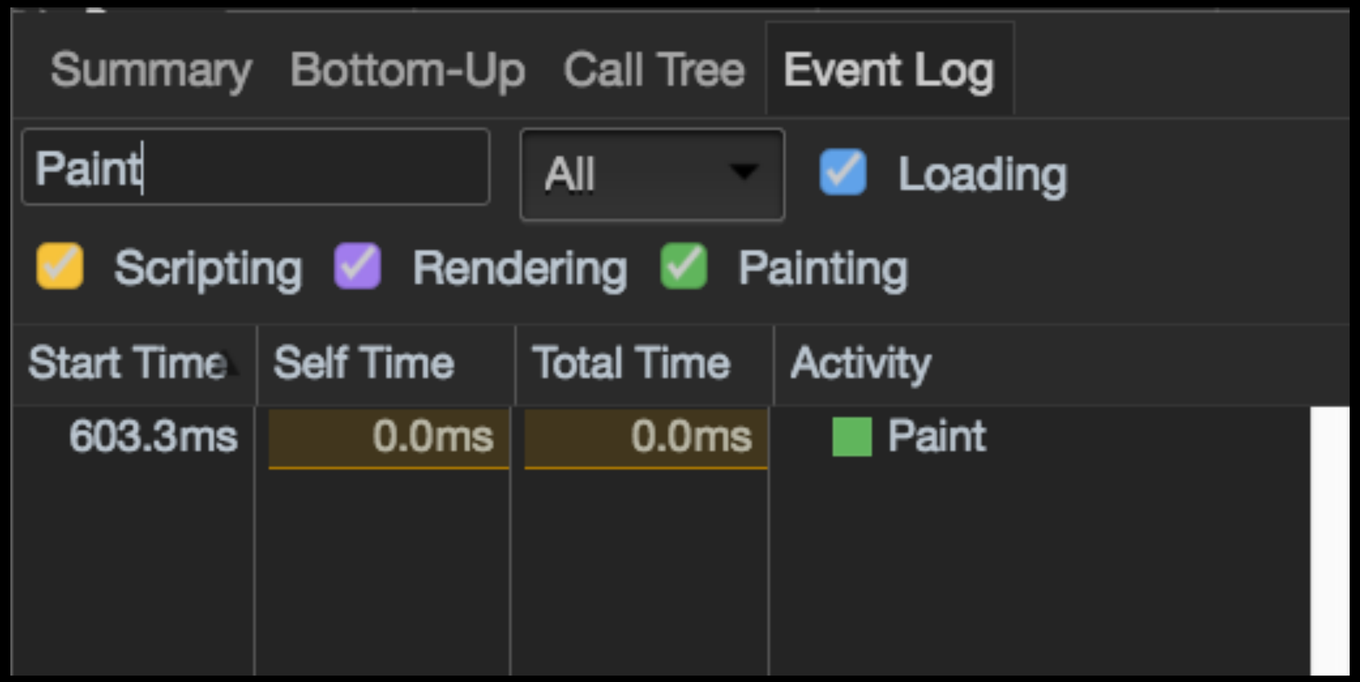



http://reflow-test.surge.sh/

```
<body>
  <div>
    <div class="block" id="one">눌러봐요</div>
    <div class="block" id="two">눌러봐요</div>
  </div>
<script>
var one = document.getElementById('one');
var two = document.getElementById('two');
one.onclick = () => {
  one.style.background = 'red'
  one.style.padding = '0.5rem';
  one.style.margin = '100px'
  one.style.height = '100px';
  // one reflow, one repaint
}
two.onclick = () => {
  two.style.background = 'red'
  two.style.padding = '0.5rem';
  console.log(one.offsetHeight);
  two.style.margin = '100px';
  console.log(one.clientHeight);
  two.style.height = '100px';
  // three reflows, one repaint
}
</script>
```


Reflow Tests
velopert
안전함 https://reflow-test.now.sh
Elements
Timeline
Screenshots
Memory
500ms
1000ms
1500ms
2000ms
FPS
CPU
NET
500 ms
1000 ms
1500 ms
2000 ms
Frames
Interactions
Input
Main
Summary
Bottom-Up
Call Tree
Event Log
Layout
All
Loading
Scripting
Rendering
Painting
Start Time
Self Time
Total Time
Activity
1170.5ms
0.1ms
0.1ms
Layout reflow-tes...
Animations
Console
눌러봐요
눌러봐요

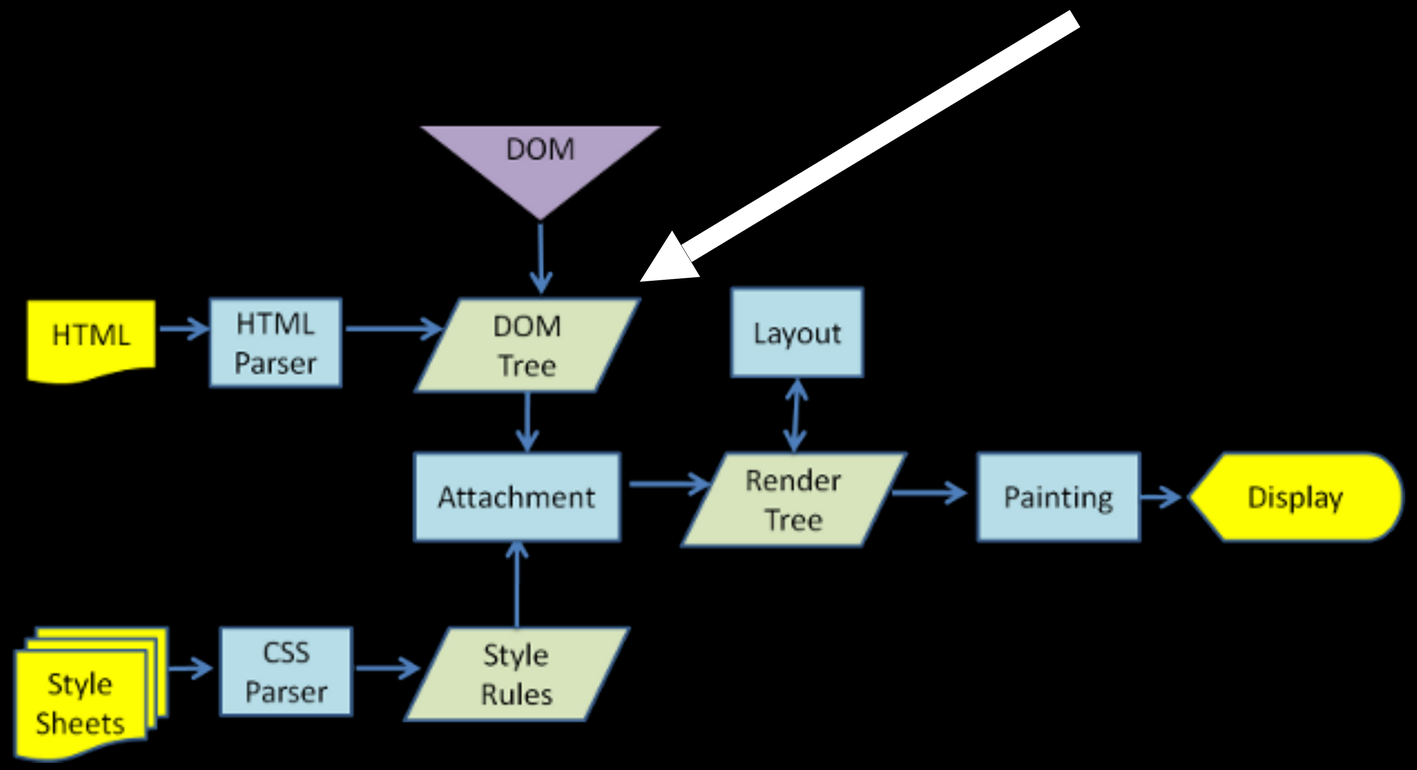

Reflow Tests
velopert
안전함 https://reflow-test.now.sh
Elements
Timeline
Screenshots
Memory
500ms
1000ms
1500ms
FPS
CPU
NET
500 ms
1000 ms
1500 ms
Frames
Interactions
Input
Main
Summary
Bottom-Up
Call Tree
Event Log
Layout
All
Loading
Scripting
Rendering
Painting
Start Time
Self Time
Total Time
Activity
601.9ms 0.1ms 0.1ms Layout (index):62
602.6ms 0.1ms 0.1ms Layout (index):64
603.0ms 0.1ms 0.1ms Layout (index):65
Animations
Console
눌러봐요
눌러봐요


Summary
Bottom-Up
Call Tree
Event Log
Paint
All
Loading
Scripting
Rendering
Painting
Start Time
Self Time
Total Time
Activity
603.3ms
0.0ms
0.0ms
Paint

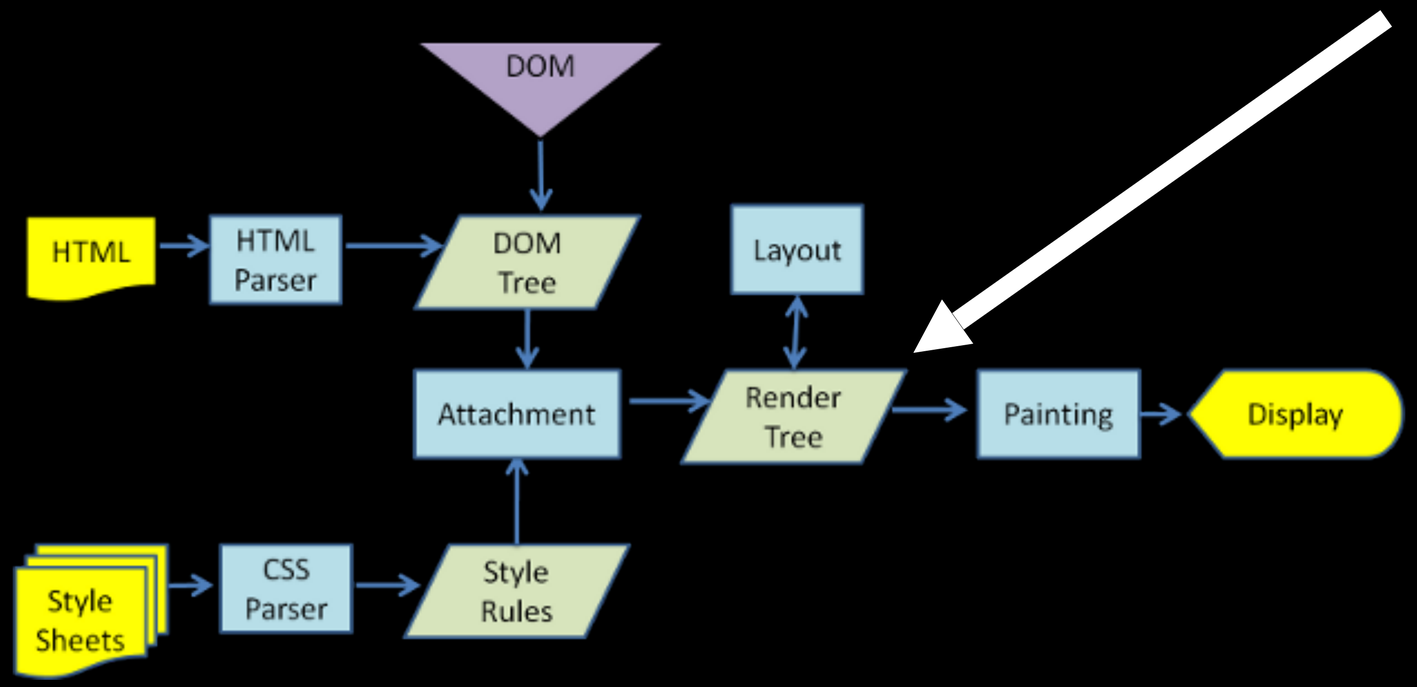
# Reflow 최적화

https://www.sitepoint.com/10-ways-minimize-reflows-improve-performance/

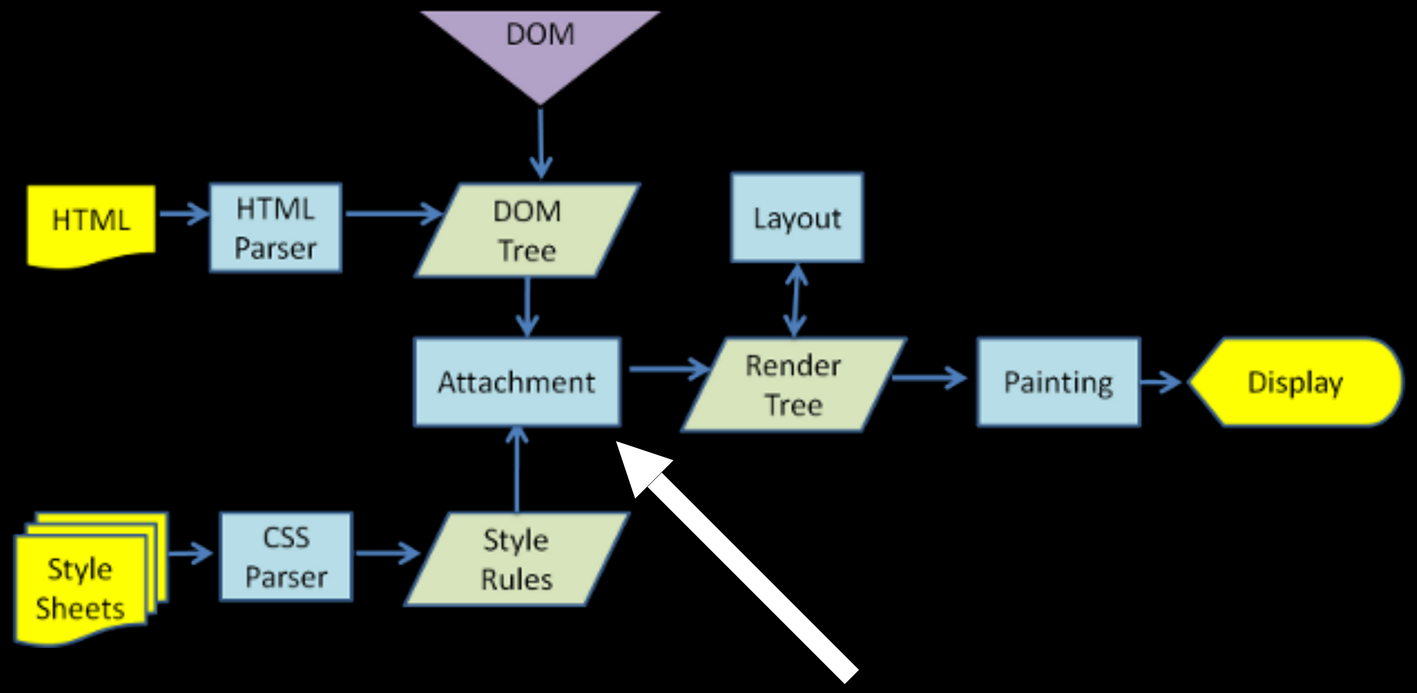
# 브라우저의 작동 방식


DOM
HTML
HTML Parser
DOM Tree
Layout
Attachment
Render Tree
Painting
Display
Style Sheets
CSS Parser
Style Rules

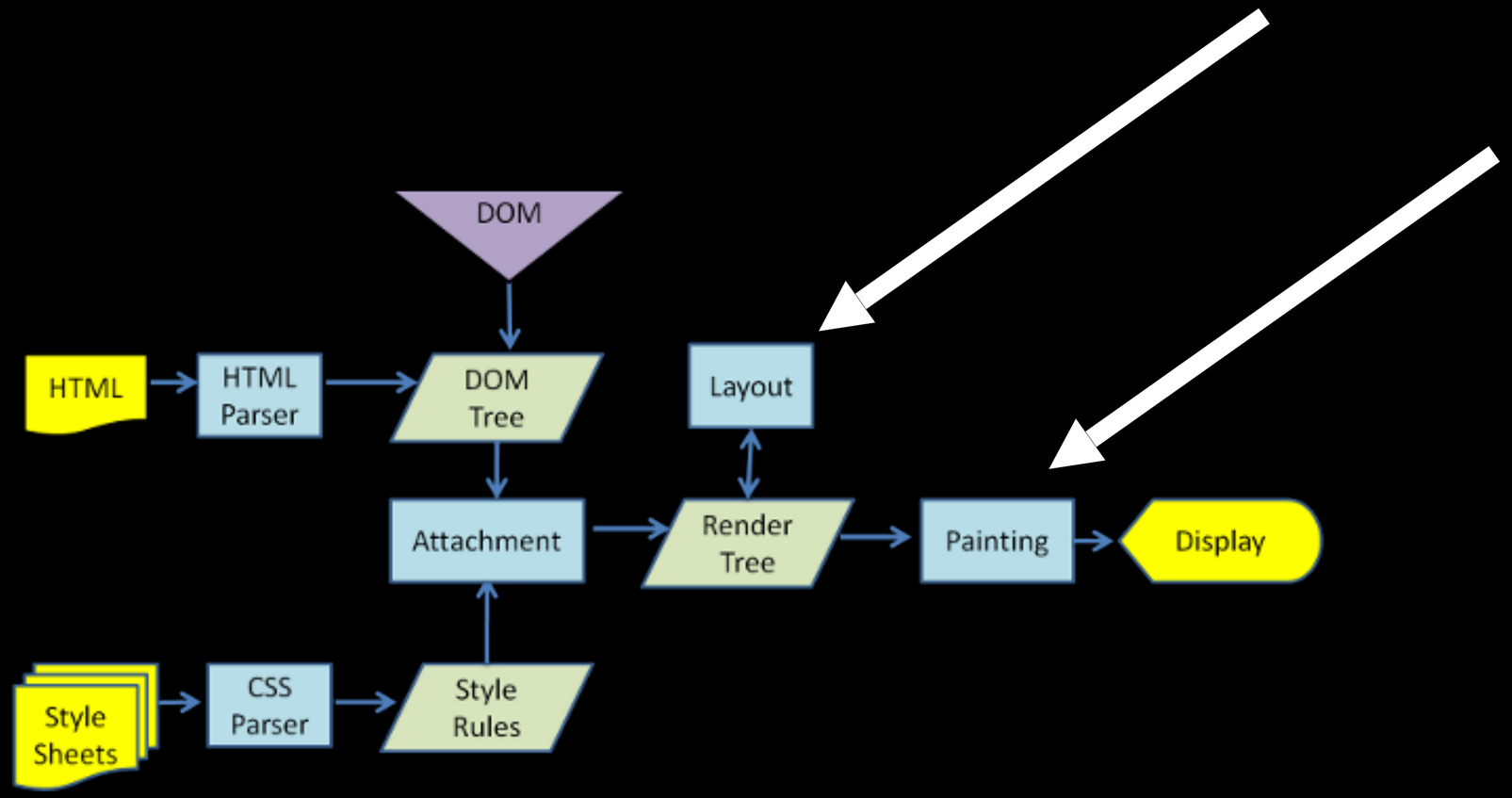
# DOM Tree 생성




# Render Tree 생성

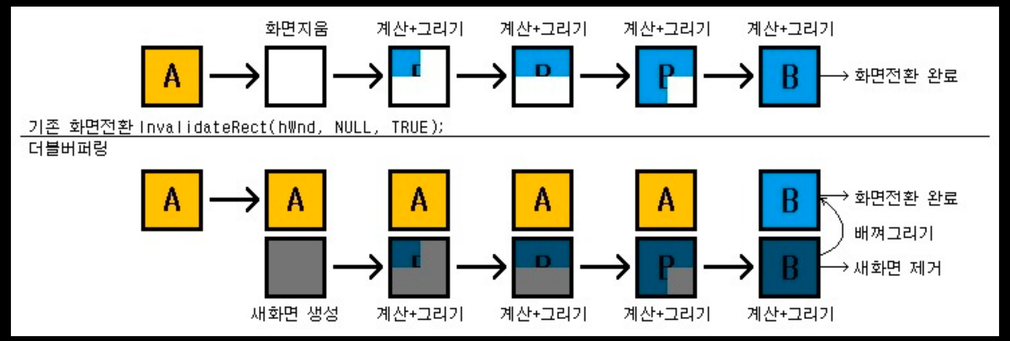



Attachment: 노드의 스타일을 처리하는 과정



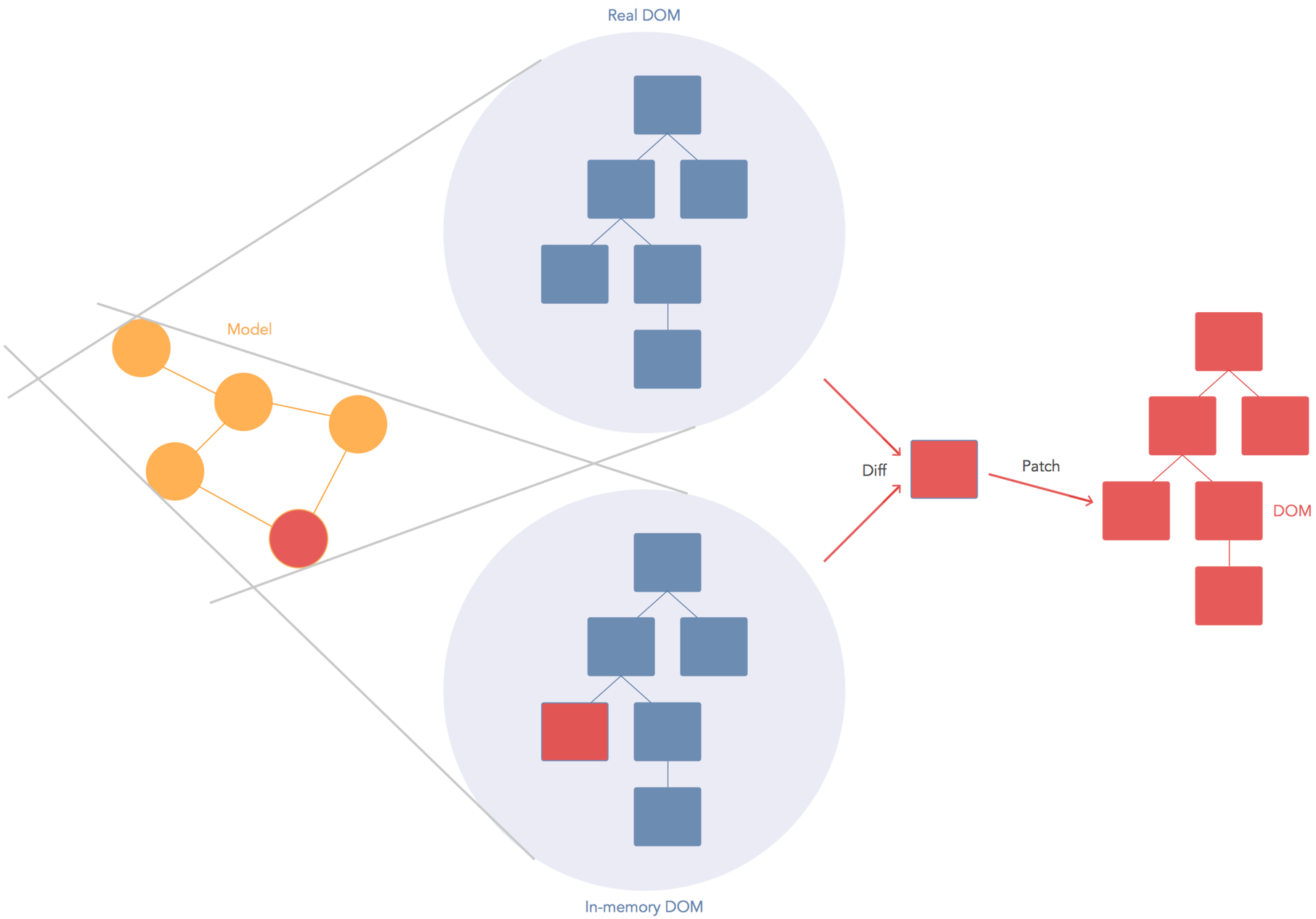

# Layout (reflow) & Painting (repaint)

# Virtual DOM!




(출처 : http://cafe.naver.com/buildgame.cafe)

# 더블 버퍼링




출처: https://auth0.com/blog/face-off-virtual-dom-vs-incremental-dom-vs-glimmer/

# 컴포넌트

# Reconciliation (조화)
# 비교 알고리즘

# 비교 알고리즘

- 원래 트리비교는 최소 O(n^3) 의 비교를 해야합니다
- 이 과정을 최적화하여 O(n)의 비교를 합니다

# 두가지 전제

같은 형태의 엘리먼트들은 비슷한 DOM 트리를 가지고있고,
다른 형태의 엘리먼트들은 서로 다른 DOM 트리를 가지고 있다.

( 리스트를 렌더링 할 때 해당)
엘리먼트에 key 값을 설정 함으로서, 엘리먼트에 고유 값을 주고,
이를 통하여 렌더링시 새로 렌더링하지 않고 유지시킨다.

# 엘리먼트 타입이 다를 경우

```
renderA: <div />
renderB: <span />
```

div 제거 span 추가

# 엘리먼트 타입이 다를 경우

```
renderA: <Header />
renderB: <Content />
```

Header 제거 Content 추가

# 엘리먼트 타입이 같을 경우

```
renderA: <div className="hello"/>
renderB: <div className="world"/>
```

className 만 변경

# 엘리먼트 타입이 같을 경우

```
renderA: <Counter number={0}/>
renderB: <Counter number={1}/>
```

컴포넌트가 사라지지 않음
props 값이 변함
컴포넌트 라이프사이클 호출

# 리스트를 렌더링 할 때

```
renderA:
    <div>
        <span>Hello</span>
    </div>
renderB:
    <div>
        <span>Hello</span>
        <span>World</span>
    </div>
```

**World 삽입**

```
renderA:
    <div>
        <span>Hello</span>
    </div>
renderB:
    <div>
        <span>World</span>
        <span>Hello</span>
    </div>
```

**Hello 의 내용을 World 로 변경**

**그 뒤에 Hello 삽입**

# 해결: **key**

```
renderA:
    <div>
        <span key="hello">Hello</span>
    </div>

renderB:
    <div>
        <span key="hello">Hello</span>
        <span key="world">World</span>
    </div>
```

**World 는 그대로 두고 그 앞에**

**Hello 삽입**

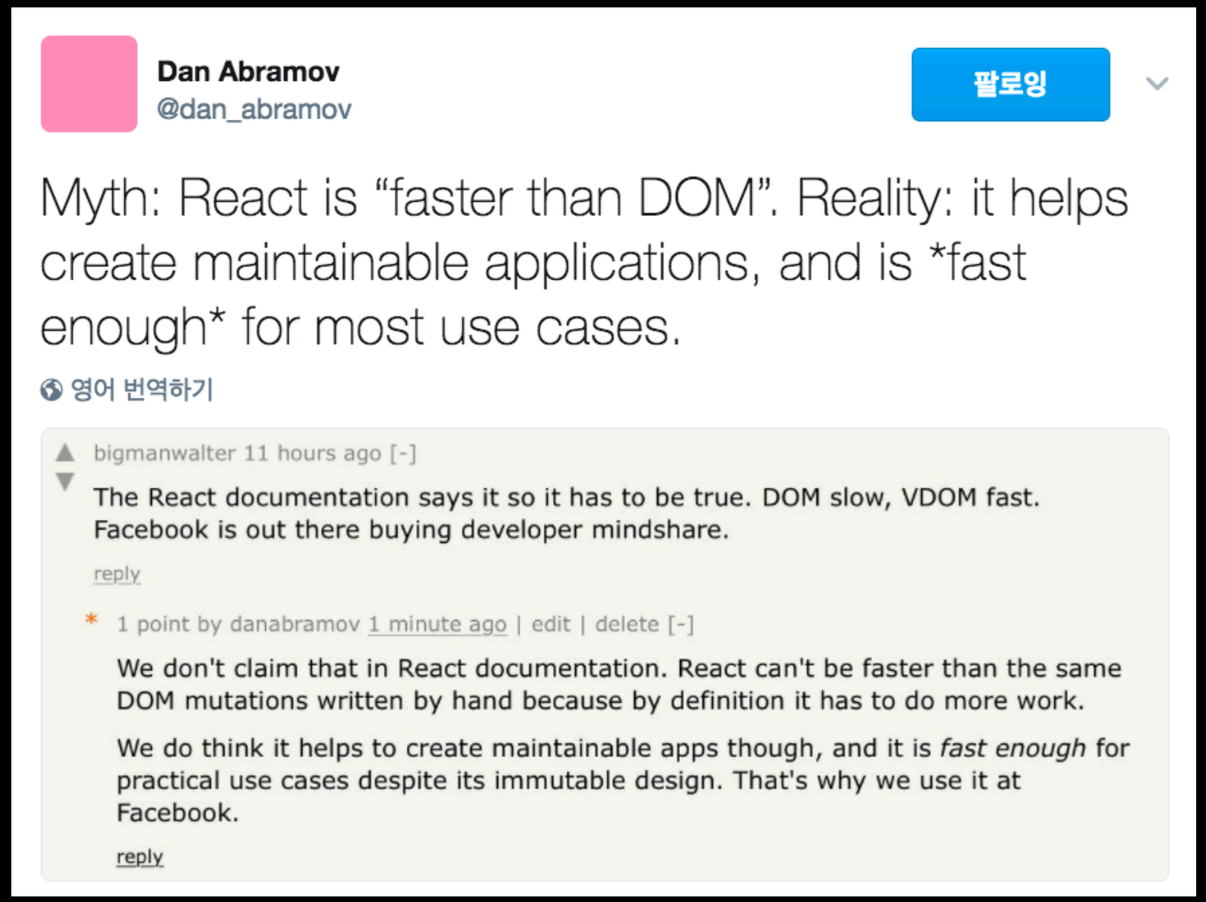
# 요약

- 다르게 생겼으면 그 내부 비교 하지 않음
- key 를 사용해서 리스트 렌더링 성능 최적화

# LifeCycle API
# shouldComponentUpdate

# Browser DOM Update

# 오해: React는 DOM보다 빠르다




번역: React가 DOM 보다 빠르다는건 잘못된 사실이에요. 사실은: 유지보수 가능한 어플리케이션을 만드는것을 도와주고 그리고 대부분의 경우에 ‘충분히 빨라요’

# 쉼.

**Node.js / Yarn / 에디터 사전 설치**: https://git.io/v7hNd

# 작업환경 설정

3. ~ (zsh)

```
~                                          ⬢ 8.9.0
λ node -v
v8.9.0

~                                          ⬢ 8.9.0
λ yarn --version
0.27.5

~                                          ⬢ 8.9.0
λ
```

# 작업환경 설정이 안됐을 경우:

**webpackbin**: http://bit.ly/2xaYm3w

1. ~/fc3 (zsh)

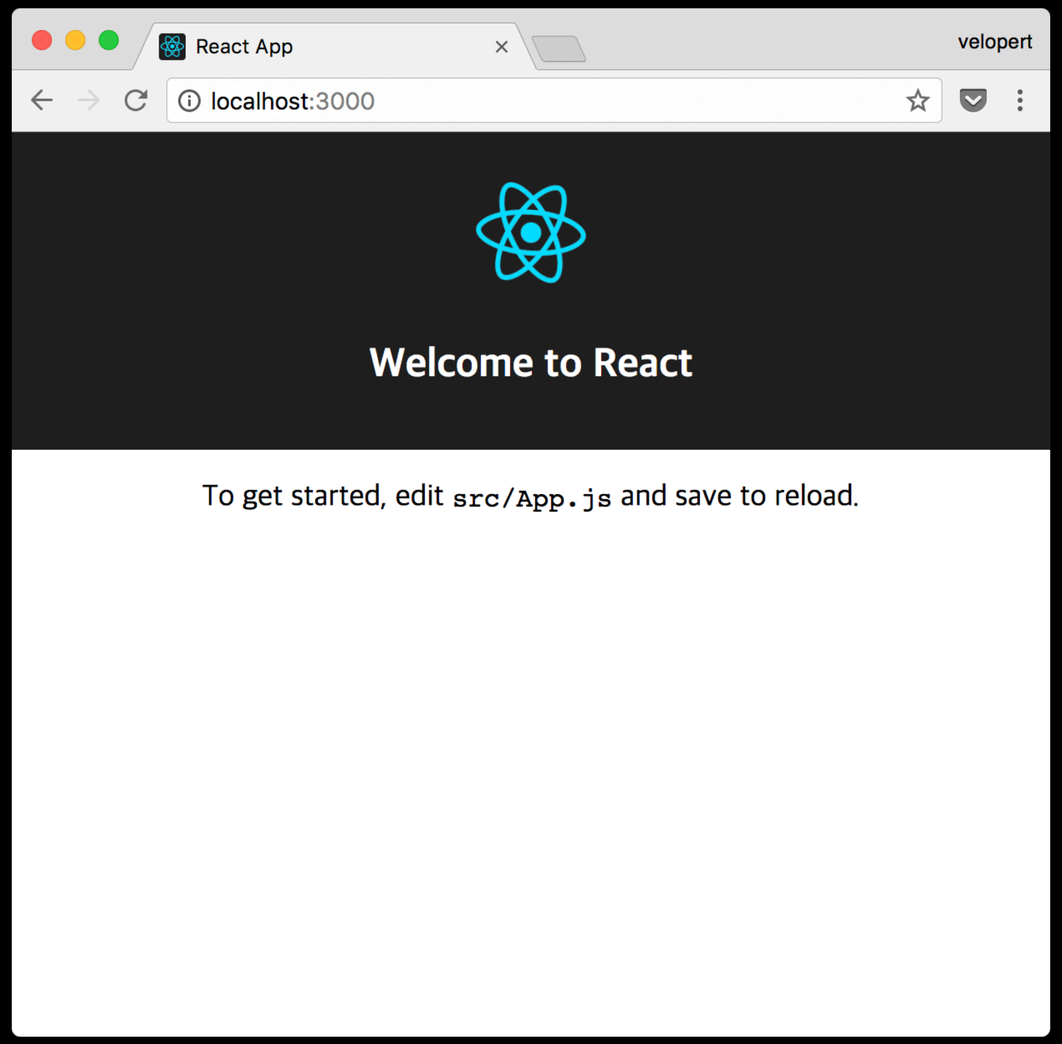
```
~/fc3                                                          ⬢8.9.0
λ yarn global add create-react-app
yarn global v0.27.5
warning package.json: No license field
warning No license field
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Installed "create-react-app@1.4.3" with binaries:
      - create-react-app
warning No license field
Done in 8.92s.

   9s

~/fc3 9s                                                       ⬢8.9.0
λ
```

3. ~/fc3 (zsh)

```
~/fc3                                    ⬢ 8.9.0
λ create-react-app first-day
```

3. ~/fc3/first-day (zsh)

```
    Bundles the app into static files for production.

  yarn test
    Starts the test runner.

  yarn eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd first-day
  yarn start

Happy hacking!

   25s

~/fc3 25s                                  ⬢8.9.0
λ  cd first-day



~/fc3/first-day                                                          ⬢8.9.0
λ yarn start
```


React App
velopert
localhost:3000
Welcome to React
To get started, edit src/App.js and save to reload.

# JSX

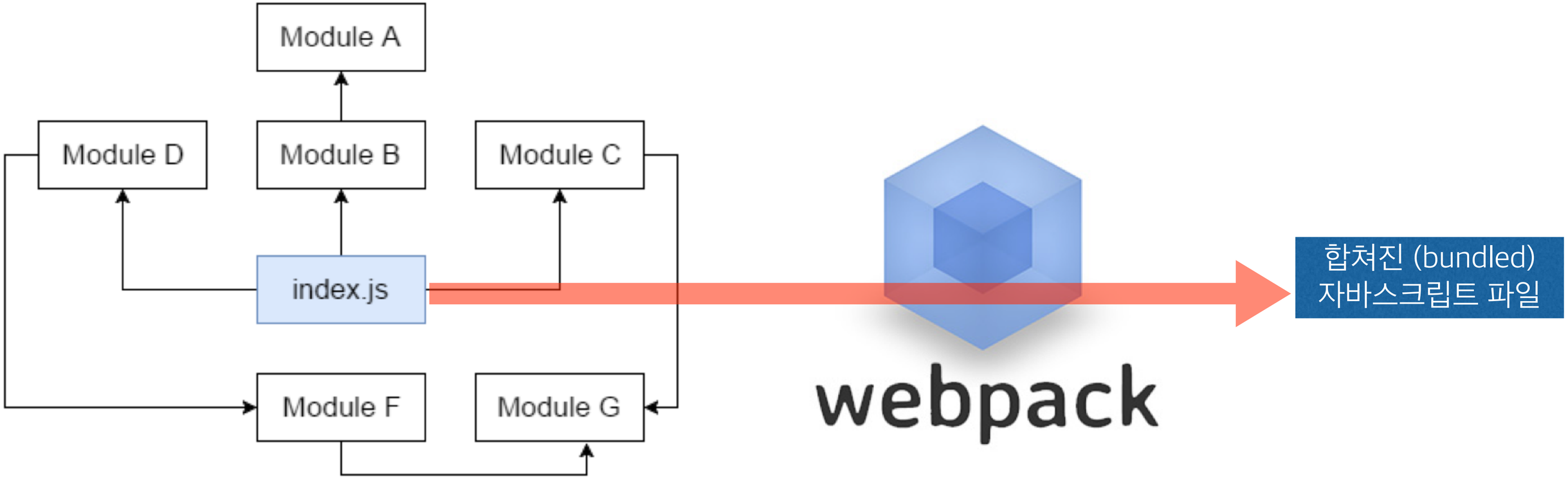
# App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
class App extends Component {
  render() {
    return (
      <div className="App">
        <div className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h2>Welcome to React</h2>
        </div>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}
export default App;
```

# App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
class App extends Component {
  render() {
    return (
      <div className="App">
        <div className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h2>Welcome to React</h2>
        </div>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}
export default App;
```

```
var React = require('react');
var Component = React.Component;
```


Module A
Module D
Module B
Module C
index.js
Module F
Module G
webpack
합쳐진 (bundled)
자바스크립트 파일

# App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
class App extends Component {
  render() {
    return (
      <div className="App">
        <div className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h2>Welcome to React</h2>
        </div>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}
export default App;
```

# App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
class App extends Component {
  render() {
    return (
      <div className="App">
        <div className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h2>Welcome to React</h2>
        </div>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}
export default App;
```

# App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
class App extends Component {
  render() {
    return (
      <div className="App">
        <div className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h2>Welcome to React</h2>
        </div>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}
export default App;
```

# App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
class App extends Component {
  render() {
    return (
      <div className="App">
        <div className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h2>Welcome to React</h2>
        </div>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}
export default App;
```

# JavaScript 의 **class**

```
function Dog(name) {
  this.name = name;
}

Dog.prototype.say = function() {
   console.log(this.name + ': 멍멍');
}
var dog = new Dog('검둥이');
dog.say(); // 검둥이: 멍멍
```

```
class Dog {
  constructor(name) {
    this.name = name;
  }
  say() {
    console.log(this.name + ': 멍멍');
  }
}

const dog = new Dog('흰둥이');
dog.say(); // 흰둥이: 멍멍
```

# JSX: 자바스크립트의 확장문법

```
var a = (
  <div>
    <h1>Awesome <b>React</b></h1>
  </div>
)

```

```
"use strict";

var a = React.createElement(
  "div",
  null,
  React.createElement(
    "h1",
    null,
    "Awesome ",
    React.createElement(
      "b",
      null,
      "React"
    )
  )
);
```

http://bit.ly/2n1PrMy

# JSX 의 장점

# 보기 쉽고 익숙하다

# 에러검사

# 더 높은 활용도

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './index.css';
ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

# ReactDOM.render?

# JSX 문법

# 1) 감싸져있는 엘리먼트

```
import React, {Component} from 'react';
class App extends Component {
    render() {
        return (
                <h1>안녕!</h1>
                <h2>즐거운 리액트</h2>
        );
    }
}
export default App;
```

first-project — node ◂ node /usr/local/bin/yarn start — 79×16

```
Failed to compile.

Error in ./src/App.js
Syntax error: Adjacent JSX elements must be wrapped in an enclosing tag (8:15)

  6 |
  7 |                 <h1>안 녕 !</h1>
> 8 |                 <h2>즐 거 운  리 액 트 </h2>
    |                 ^
  9 |
 10 |             );
 11 |         }

 @ ./src/index.js 13:11-27
```

```
import React, {Component} from 'react';
class App extends Component {
    render() {
        return (
            <div>
                <h1>안녕!</h1>
                <h2>즐거운 리액트</h2>
            </div>
        );
    }
}
export default App;
```

# 2) JavaScript 표현

```
import React, {Component} from 'react';
class App extends Component {
    render() {
        const name = 'velopert';
        return (
            <div>
                <h1>안녕!</h1>
                <h2>즐거운 리액트</h2>
                <p>내 이름은 {name}</p>
            </div>
        );
    }
}
export default App;
```

```
import React, {Component} from 'react';
class App extends Component {
    render() {
        const name = 'velopert';
        return (
            <div>
                <h1>안녕!</h1>
                <h2>즐거운 리액트</h2>
                <p>내 이름은 {name}</p>
            </div>
        );
    }
}
export default App;
```

# var 은 scope 가 함수단위 입니다

```
var a = "hello";
if(true) {
  var a = "bye";
  console.log(a); // bye
}
console.log(a); // bye
```

# const와 let은 scope 가 블록단위 입니다

```
let a = "hello";
if(true) {
  let a = "bye";
  console.log(a); // bye
}
console.log(a); // hello
```

# 3) if문 대신 조건부 연산자

[condition] ? [true] : [false]

1+1 === 2 ? “정답” : “오답”

```
import React, {Component} from 'react';
class App extends Component {
    render() {
        const name = 'velopert';
        return (
            <div>
                <h1>안녕!</h1>
                <h2>즐거운 리액트</h2>
                <p>내 이름은 {name}</p>
                <p>{ 1 + 1 === 2 ? '정답' : '바보'}</p>
            </div>
        );
    }
}
export default App;
```

# 4) inline styling

```
import React, {Component} from 'react';
class App extends Component {
    render() {
        const name = 'velopert';
        const style = {
            color: 'aqua',
            backgroundColor: 'black'
        }
        return (
            <div>
                <h1>안녕!</h1>
                <h2 style={style}>즐거운 리액트</h2>
                <p>내 이름은 {name}</p>
                <p>{ 1 + 1 === 2 ? '정답' : '바보'}</p>
            </div>
        );
    }
}
export default App;
```

# 5) class 대신 className

# App.css

```
.hello {
    color: pink;
}
```

```
import React, {Component} from 'react';
import './App.css';
class App extends Component {
    render() {
        const name = 'velopert';
        const style = {
            color: 'aqua',
            backgroundColor: 'black'
        }
        return (
            <div>
                <h1 className="hello">안녕!</h1>
                <h2 style={style}>즐거운 리액트</h2>
                <p>내 이름은 {name}</p>
                <p>{ 1 + 1 === 2 ? '정답' : '바보'}</p>
            </div>
        );
    }
}
export default App;
```

# 6) 꼭 닫혀야하는 태그

self-closing tag: 

# 7) 주석

```
return (
    <div>
        {/* 주석은 이렇게 작성 */}
        <h1
            className="hello" // 혹은 이렇게 작성
            // 여기에도 작성
            /* 이렇게도 작성 */
        >안녕!</h1>
        <h2 style={style}>즐거운 리액트</h2>
        <p>내 이름은 {name}</p>
        <p>{1 + 1 === 2
                ? '정답'
                : '바보'}</p>
        // 여기에쓰면 렌더링 되버려요
        /* 이것도 마찬가지죠. */
    </div>
);
```

쉼.

# 컴포넌트

내 지갑 :: Bitimulate

velopert

안전함 | https://bitimulate.com/wallet

bitimulate

거래소 내 지갑 랭킹 velopert

- 내 지갑
- 거래내역
- 수익률

# 내 지갑

## 현재 총합 보유 자산

| BTC | USD | KRW |
|---|---|---|
| Ƀ 133.95273689 | $ 1,702,934 | ₩ 1,849,896,675 |

## 화폐별 지갑

| 코인 | 변화율 | 이름 | 보유량 | 거래 대기중 | BTC 가치 |
|---|---|---|---|---|---|
| STRAT | -8.45% | Stratis | 8000.0000000 | 0.0000000000 | 4.5117600000 |
| NMC | +0.50% | Namecoin | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| USD | | Dollar | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| NXT | +9.57% | NXT | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| STR | +33.18% | Stellar | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| BTC | +7.91% | Bitcoin | 8.4922425000 | 0.0000000000 | 8.4922425000 |
| DASH | -12.94% | Dash | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| EMC2 | +99.24% | Einsteinium | 743517.14029 | 0.0000000000 | 117.31213439 |
| FCT | -8.16% | Factom | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| EXP | -12.02% | Expanse | 20000.000000 | 0.0000000000 | 3.6366000000 |

내 지갑 :: Bitimulate

velopert

안전함 | https://bitimulate.com/wallet

bitimulate

거래소 내 지갑 랭킹 velopert

- 내 지갑
- 거래내역
- 수익률

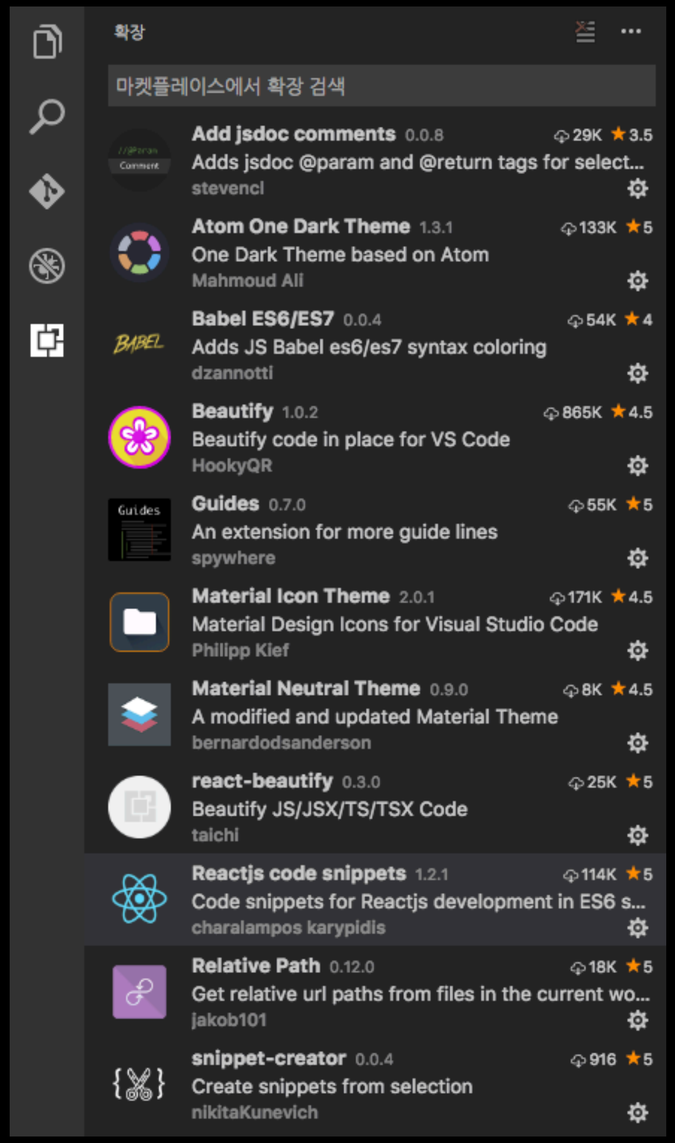
# 내 지갑

## 현재 총합 보유 자산

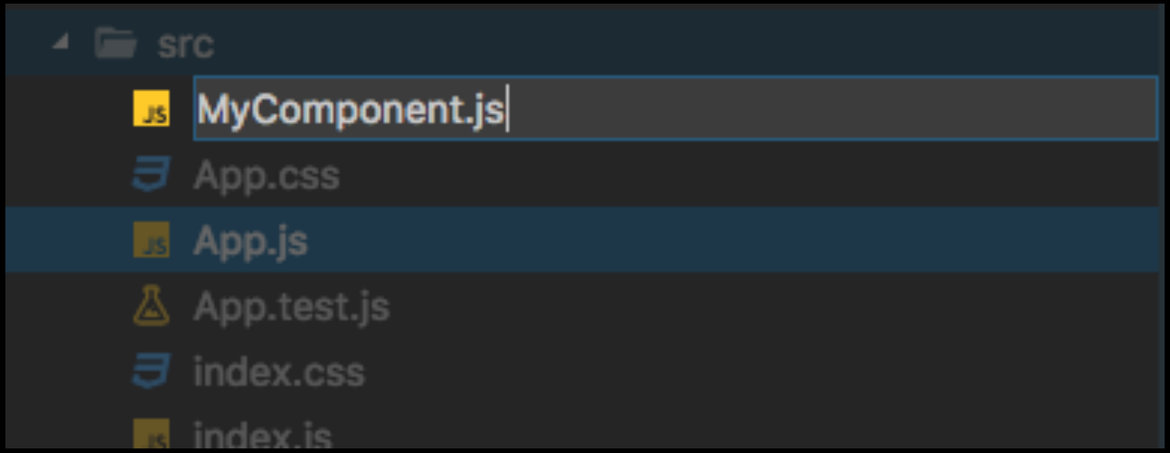
| BTC | USD | KRW |
|---|---|---|
| Ƀ 133.95273689 | $ 1,702,934 | ₩ 1,849,896,675 |

## 화폐별 지갑

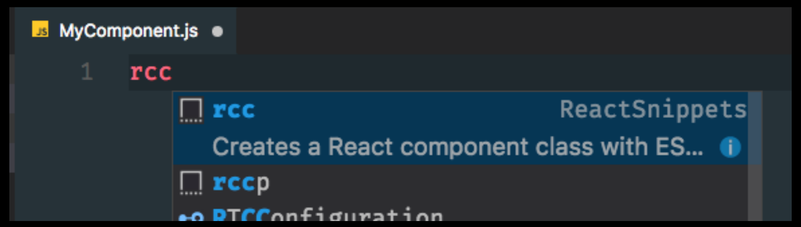
| 코인 | 변화율 | 이름 | 보유량 | 거래 대기중 | BTC 가치 |
|---|---|---|---|---|---|
| STRAT | -8.45% | Stratis | 8000.0000000 | 0.0000000000 | 4.5117600000 |
| NMC | +0.50% | Namecoin | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| USD | | Dollar | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| NXT | +9.57% | NXT | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| STR | +33.18% | Stellar | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| BTC | +7.91% | Bitcoin | 8.4922425000 | 0.0000000000 | 8.4922425000 |
| DASH | -12.94% | Dash | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| EMC2 | +99.24% | Einsteinium | 743517.14029 | 0.0000000000 | 117.31213439 |
| FCT | -8.16% | Factom | 0.0000000000 | 0.0000000000 | 0.0000000000 |
| EXP | -12.02% | Expanse | 20000.000000 | 0.0000000000 | 3.6366000000 |




# Reacts Code Snippets




MyComponent.js 생성





**rcc** 라고 적으면 자동완성이 됩니다

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

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

# props

위에서 아래로

부모 – – – – 자식

부모 - - 데이터 - - 자식

부모 – – **props** – – 자식

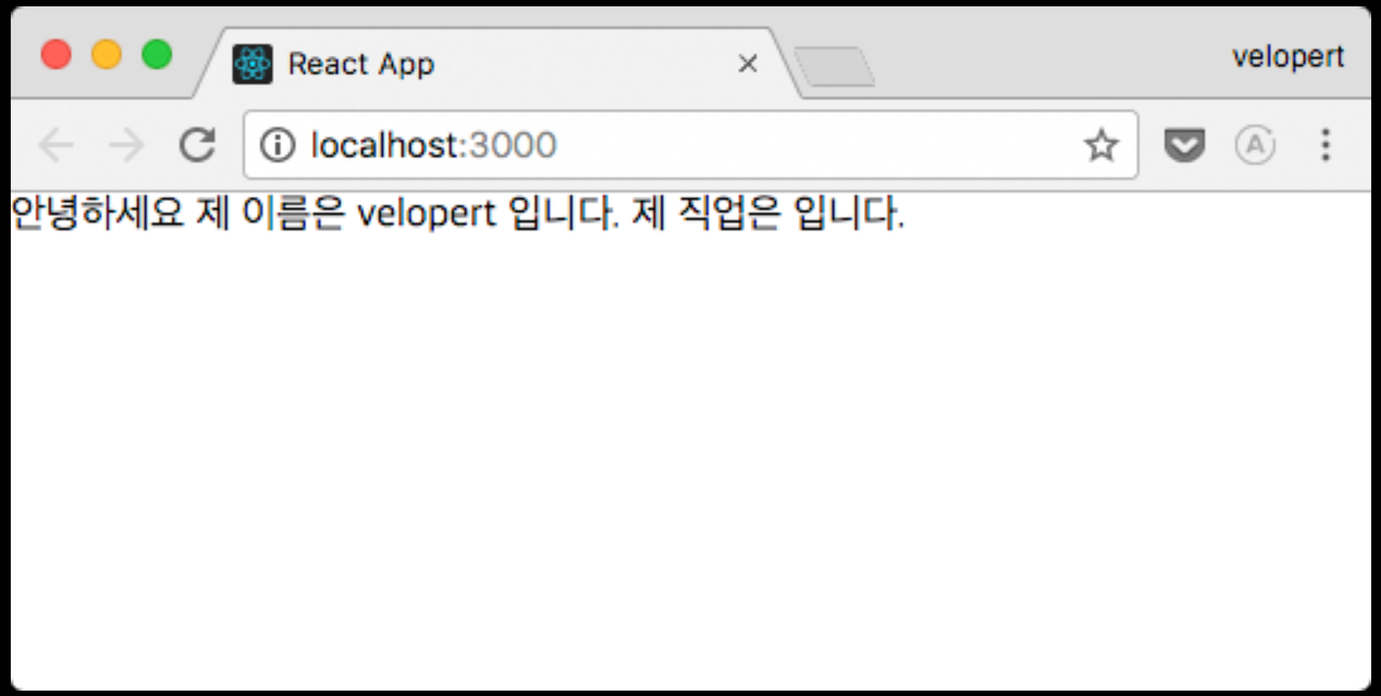
```
<MyComponent name="velopert"/>
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    render() {
        return (
            <div>
                안녕하세요 제 이름은 {this.props.name} 입니다.
            </div>
        );
    }
}
export default MyComponent;
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    render() {
        const { name, job } = this.props;
        return (
            <div>
                안녕하세요 제 이름은 {name} 입니다.
                제 직업은 {job} 입니다.
            </div>
        );
    }
}
export default MyComponent;
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    render() {
        const { name, job } = this.props;
        return (
            <div>
                안녕하세요 제 이름은 {name} 입니다.
                제 직업은 {job} 입니다.
            </div>
        );
    }
}
export default MyComponent;
```

```
const name = this.props.name;
const job = this.props.job;
```


React App
velopert
localhost:3000
안녕하세요 제 이름은 velopert 입니다. 제 직업은 입니다.

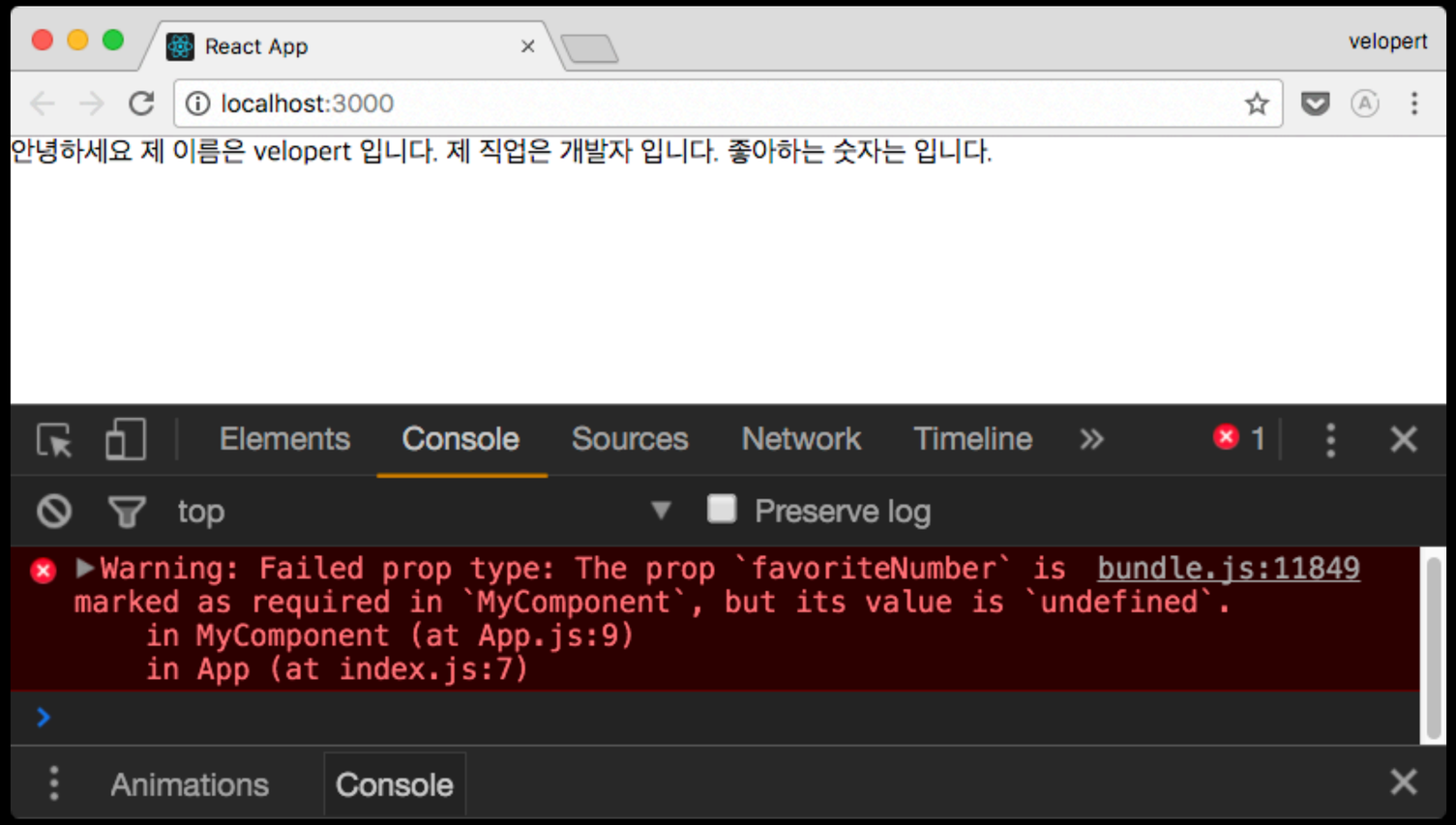
# 기본값 설정하기: `defaultProps`

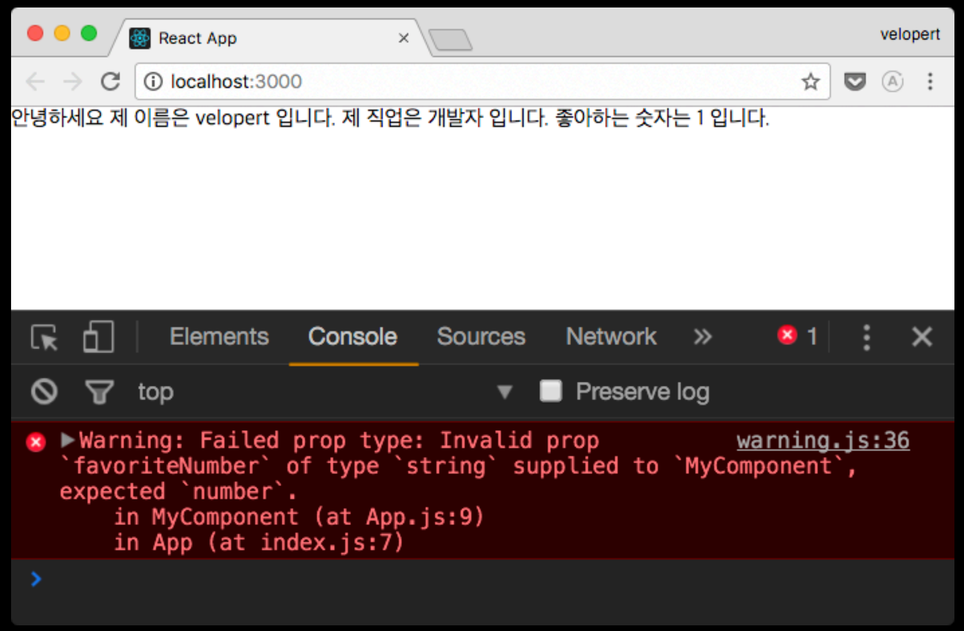
```
import React, { Component } from 'react';
class MyComponent extends Component {
    static defaultProps = {
        job: '개발자'
    }
    render() {
        const { name, job } = this.props;
        return (
            <div>
                안녕하세요 제 이름은 {name} 입니다.
                제 직업은 {job} 입니다.
            </div>
        );
    }
}
export default MyComponent;
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    render() {
        const { name, job } = this.props;
        return (
            <div>
                안녕하세요 제 이름은 {name} 입니다.
                제 직업은 {job} 입니다.
            </div>
        );
    }
}
MyComponent.defaultProps = {
  job: '개발자'
}
export default MyComponent;
```

# props 검증하기: **propTypes**

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';

class MyComponent extends Component {
    static defaultProps = {
        job: '개발자'
    }
    static propTypes = {
        name: PropTypes.string,
        job: PropTypes.string,
        favoriteNumber: PropTypes.number.isRequired
    }
    render() {
        const { name, job, favoriteNumber } = this.props;
        return (
            <div>
                안녕하세요 제 이름은 {name} 입니다.
                제 직업은 {job} 입니다.
                좋아하는 숫자는 {favoriteNumber} 입니다.
            </div>
        );
    }
}
export default MyComponent;
```


React App
velopert
localhost:3000
안녕하세요 제 이름은 velopert 입니다. 제 직업은 개발자 입니다. 좋아하는 숫자는 입니다.
Elements
Console
Sources
Network
Timeline
1
top
Preserve log
Warning: Failed prop type: The prop `favoriteNumber` is marked as required in `MyComponent`, but its value is `undefined`.
in MyComponent (at App.js:9)
in App (at index.js:7)
bundle.js:11849
Animations
Console




```
~~<MyComponent name="velopert" favoriteNumber="1" />~~
<MyComponent name="velopert" favoriteNumber={1} />
```

# PropTypes 종류:

- array: 배열
- bool: true or false
- func: 함수
- number: 숫자
- object: 객체
- string: 문자열
- element: div 태그, 리액트 컴포넌트 등
- node: 렌더링 될 수 있는 모든것
- oneOf([배열]): 배열 내부의 값들
- any: 모든 것
- **더보기**: https://facebook.github.io/react/docs/typechecking-with-proptypes.html

# 모든 컴포넌트마다 PropTypes 를 지정해주는것이 **좋은 습관**

어떤 컴포넌트가 어떤 props 를 요구하는지 명확하게 알려주어 유지보수에 도움이 됩니다
협업할때는 더더욱 필수입니다.

# 변하는 값: **state**

컴포넌트 고유의 것.

```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        number: 0
    }
    render() {
        const {number} = this.state;
        return (
            <div>
                <h1>{number}</h1>
                <button onClick={()=>{this.setState({number: number + 1})}}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        number: 0
    }
    render() {
        const {number} = this.state;
        return (
            <div>
                <h1>{number}</h1>
                <button onClick={()=>{this.setState({number: number + 1})}}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```

```
constructor(props) {
    super(props);
    this.state = {
        number: 0
    }
}
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        number: 0
    }
    render() {
        const {number} = this.state;
        return (
            <div>
                <h1>{number}</h1>
                <button onClick={()=>{this.setState({number: number + 1})}}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        number: 0
    }
    render() {
        const {number} = this.state;
        return (
            <div>
                <h1>{number}</h1>
                <button onClick={()=>{this.setState({number: number + 1})}}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        number: 0
    }
    render() {
        const {number} = this.state;
        return (
            <div>
                <h1>{number}</h1>
                <button onClick={()=>{this.setState({number: number + 1})}}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```

```
function() {
    this.setState({
        number: number + 1
    });
}
```

⇒ https://github.com/tonsky/FiraCode

```
_createClass(MyComponent, [{
    key: 'render',
    value: function render() {
        var _this2 = this;

        var number = this.state.number;

        return _react2.default.createElement(
            'div',
            null,
            _react2.default.createElement(
                'h1',
                null,
                number
            ),
            _react2.default.createElement(
                'button',
                { onClick: function onClick() {
                    _this2.setState({ number: number + 1 });
                } },
                '\uB098\uB97C \uB20C\uB7EC\uBCF4\uC138\uC694.'
            )
        );
    }
}]);
```

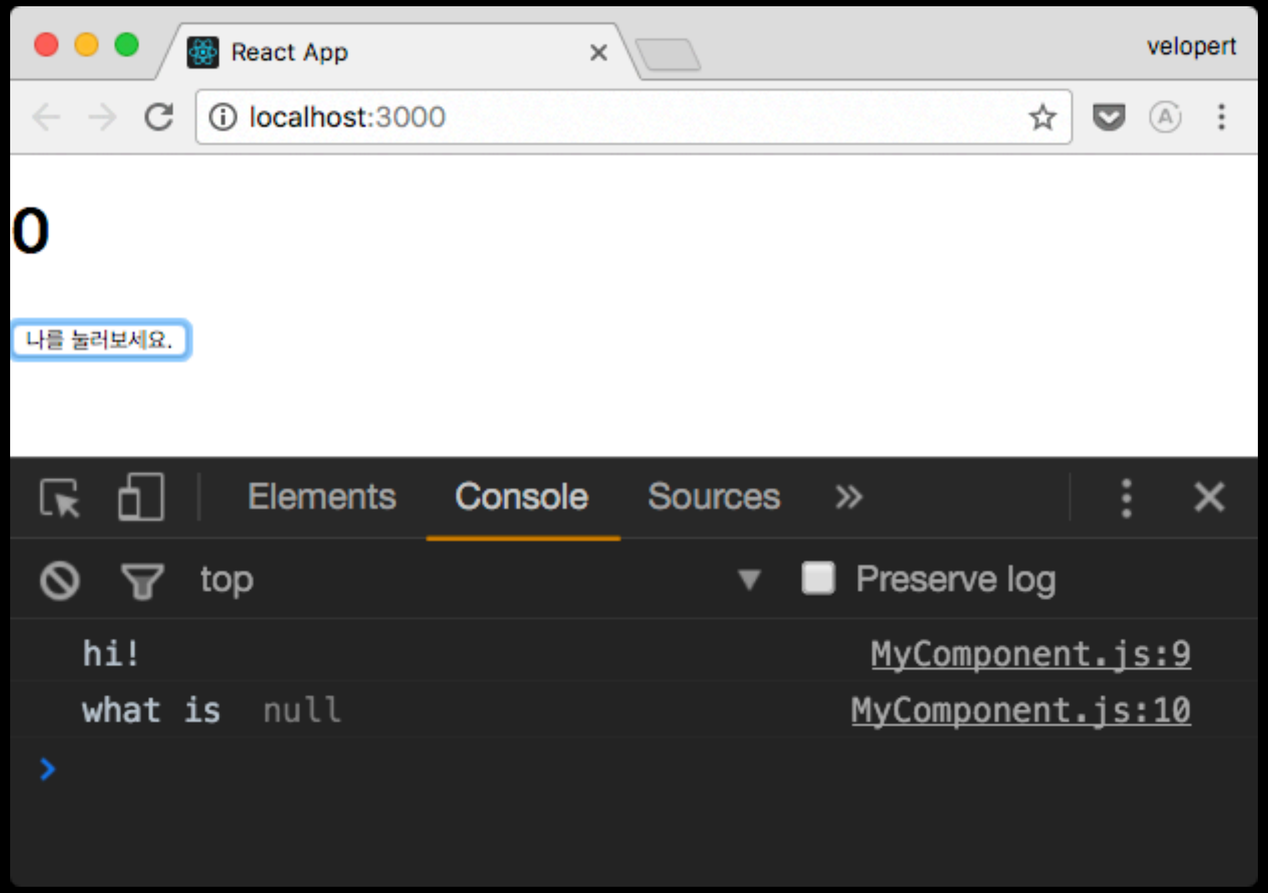
state 를 업데이트할 땐 언제나

**`this.setState({…})`**

함수가 실행되면, 리렌더링을 트리거합니다

# 임의 메소드 만들기

```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        number: 0
    }
    handleClick() {
        console.log('hi!');
        console.log('what is ', this);
    }
    render() {
        const {number} = this.state;
        return (
            <div>
                <h1>{number}</h1>
                <button onClick={this.handleClick}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```


React App
velopert
localhost:3000
0
나를 눌러보세요.
Elements
Console
Sources
top
Preserve log
hi!
MyComponent.js:9
what is null
MyComponent.js:10

```
var obj = {
    prop: 'Hello',
    sayHello: function() {
        console.log( this.prop );
    }
};
obj.sayHello(); // Logs "Hello"
```

```
var reference = obj.sayHello;
reference(); // logs "undefined"
```

```
var obj = {
    prop: 'Hello',
    sayHello: function() {
        console.log(this.prop);
    }
};
var newFunction = obj.sayHello.bind(obj);
newFunction(); // logs "Hello"
```

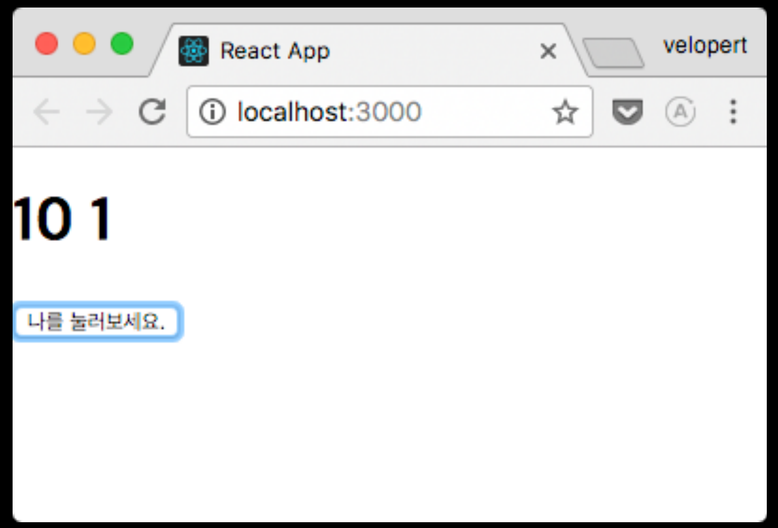
```
<button onClick={this.handleClick.bind(this)}>
  나를 눌러보세요.
</button>
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        number: 0
    }
    constructor(props) {
        super(props);
        this.handleClick = this.handleClick.bind(this);
    }
    handleClick() {
        console.log('hi!');
        console.log('what is ', this);
    }
    render() {
        const {number} = this.state;
        return (
            <div>
                <h1>{number}</h1>
                <button onClick={this.handleClick}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```

```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        number: 0
    }
    handleClick = () => {
        this.setState({
            number: this.state.number + 1
        });
    }
    render() {
        const {number} = this.state;
        return (
            <div>
                <h1>{number}</h1>
                <button onClick={this.handleClick}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```

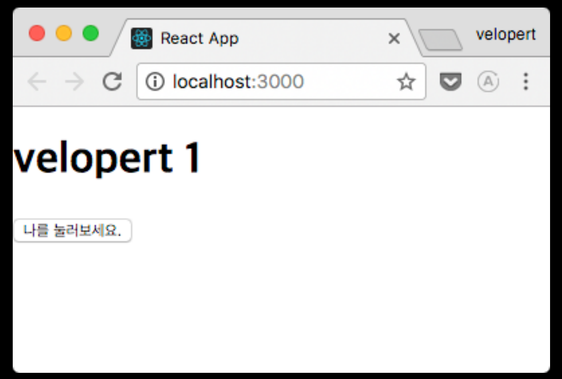
**setState({…})**
안에 있는 값들만 처리합ㄴ니다.

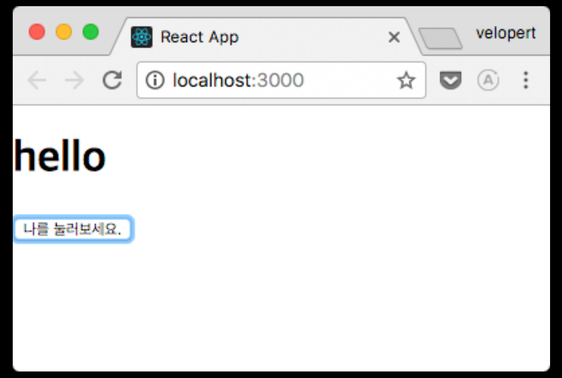
```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        number: 0,
        theNumber: 1
    }
    handleClick = () => {
        this.setState({
            number: this.state.number + 1
        });
    }
    render() {
        const {number, theNumber} = this.state;
        return (
            <div>
                <h1>{number} {theNumber}</h1>
                <button onClick={this.handleClick}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```

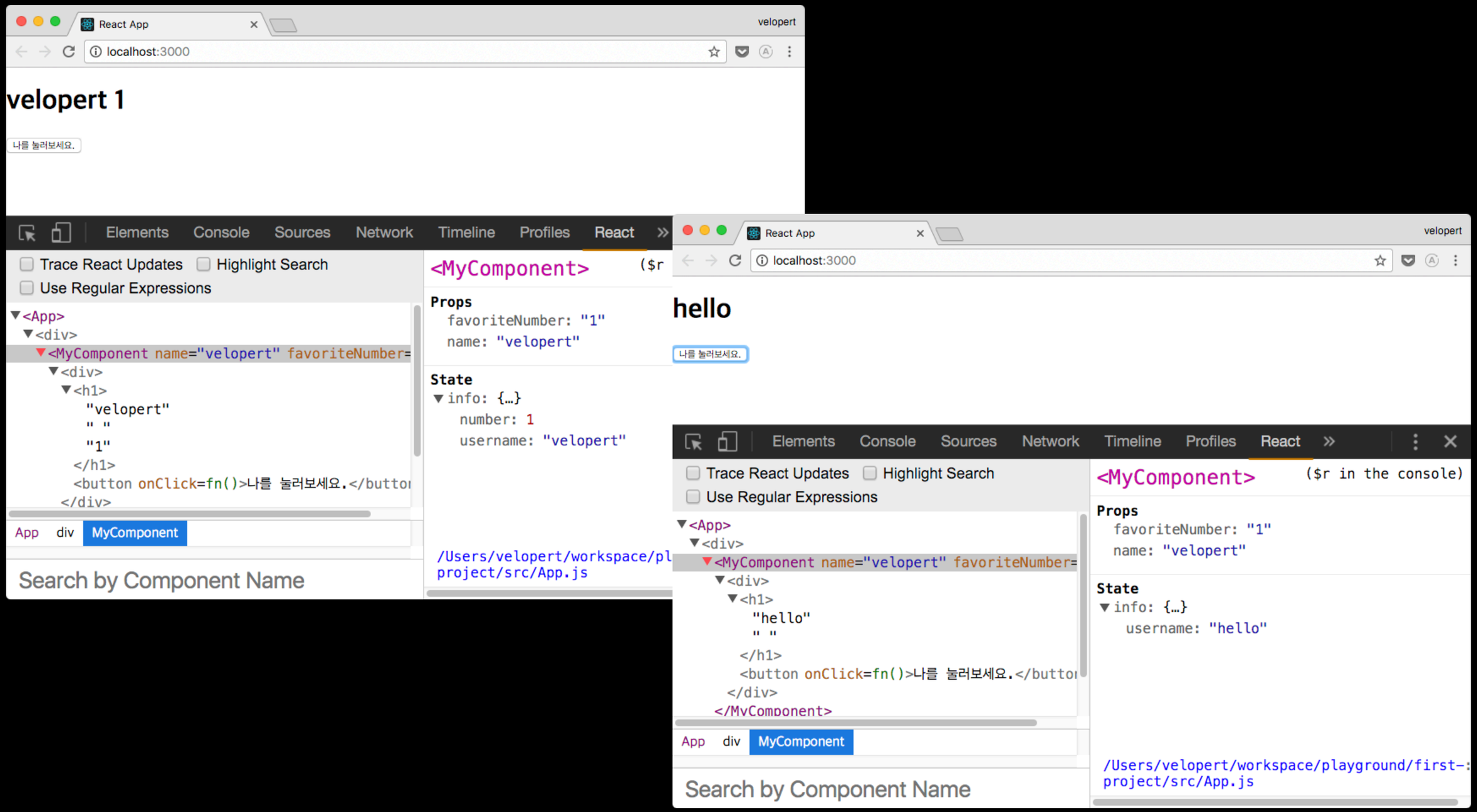


nested 된 값은 단일 처리 불가

```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        info: {
            username: 'velopert',
            number: 1
        }
    }
    handleClick = () => {
        this.setState({
            info: {
                username: 'hello'
            }
        });
    }
    render() {
        const {info} = this.state;
        return (
            <div>
                <h1>{info.username} {info.number}</h1>
                <button onClick={this.handleClick}>
                    나를 눌러보세요.
                </button>
            </div>
        );
    }
}
export default MyComponent;
```




React App
velopert
localhost:3000
velopert 1
나를 눌러보세요.
Elements
Console
Sources
Network
Timeline
Profiles
React
Trace React Updates
Highlight Search
Use Regular Expressions
<App>

<MyComponent name="velopert" favoriteNumber=

<h1>
"velopert"
" "
"1"
</h1>
<button onClick=fn()>나를 눌러보세요.</butto

App
div
MyComponent
Search by Component Name
<MyComponent>
($r
Props
favoriteNumber: "1"
name: "velopert"
State
info: {…}
number: 1
username: "velopert"
/Users/velopert/workspace/pl
project/src/App.js
React App
velopert
localhost:3000
hello
나를 눌러보세요.
Elements
Console
Sources
Network
Timeline
Profiles
React
Trace React Updates
Highlight Search
Use Regular Expressions
<App>

<MyComponent name="velopert" favoriteNumber=

<h1>
"hello"
" "
</h1>
<button onClick=fn()>나를 눌러보세요.</butto

</MyComponent>
App
div
MyComponent
Search by Component Name
<MyComponent>
($r in the console)
Props
favoriteNumber: "1"
name: "velopert"
State
info: {…}
username: "hello"
/Users/velopert/workspace/playground/first-
project/src/App.js

```
handleClick = () => {
    this.setState({
        info: {
            ...this.state.info,
            username: 'hello'
        }
    });
}
```

... : ES6 spread 문법, 객체를 풀어서 그 자리에 대입한다

컴포넌트를 만드는 또 다른 방법

# 함수형 컴포넌트

# src/Container.js

```
import React from 'react';
const Container = (props) => {
    return (
        <div>
            <h1>{props.title}</h1>
            <div>
                {props.children}
            </div>
        </div>
    );
};
export default Container;
```

# src/App.js

```
import React, {Component} from 'react';
import MyComponent from './MyComponent';
import Container from './Container';
class App extends Component {
    render() {
        return (
            <Container title="Welcome">
                <MyComponent />
            </Container>
        );
    }
}
export default App;
```

# src/Container.js

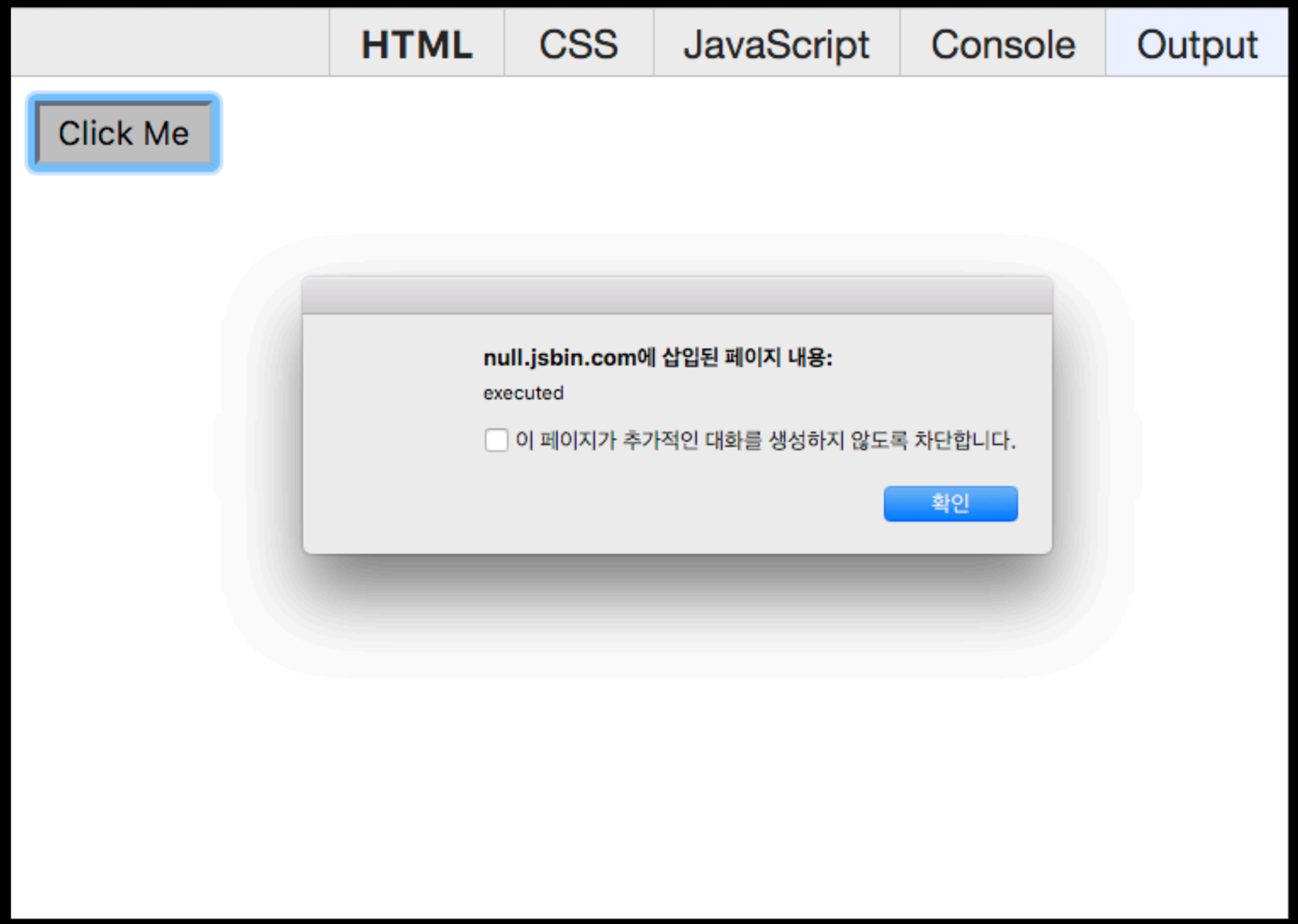
```
import React from 'react';
const Container = ({children, title}) => {
    return (
        <div>
            <h1>{title}</h1>
            <div>
                {children}
            </div>
        </div>
    );
};
export default Container;
```

쉼.

# 이벤트 핸들링

# 이벤트:
# 유저와 DOM 의 상호작용

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Events in HTML</title>
</head>
<body>
  <button onclick="alert('executed')">
    Click Me
  </button>
</body>
</html>
```


HTML
CSS
JavaScript
Console
Output
Click Me
null.jsbin.com에 삽입된 페이지 내용:
executed
이 페이지가 추가적인 대화를 생성하지 않도록 차단합니다.
확인

```
<button onClick={()=>{this.setState({number: number + 1})}}>
  나를 눌러보세요.
</button>
```

주의사항

# 1) 이벤트 이름은 camelCase

```
onclick → onClick
onkeyup → onKeyUp
…
```

주의사항

# 2) 함수형태의 객체를 전달

~~onClick={handleClick()}~~

onClick={handleClick}

주의사항

# 3) DOM 요소에만 이벤트 설정 가능

```
<MyComponent onClick={doSomething}/>
```

```
import React from 'react';
const MyComponent = ({onClick}) ⇒ {
    return (
        <button onClick={onClick}>
            클릭
        </button>
    );
};
export default MyComponent;
```

# 이벤트 종류

- Clipboard
- Composition
- Keyboard
- Focus
- Form
- Mouse
- Selection
- Touch
- UI
- Wheel
- Media
- Image
- Animation
- Transition

https://facebook.github.io/react/docs/events.html

```
import React, { Component } from 'react';
class MyComponent extends Component {
    handleChange = (e) => {
        e.persist();
        console.log(e);
    }
    handleClick = (e) => {
        e.persist();
        console.log(e);
    }
    render() {
        const { handleChange, handleClick } = this;
        return (
            <div>
                <input onChange={handleChange} type="text" name="last-name" placeholder="성" />
                <input onChange={handleChange} type="text" name="first-name" placeholder="이름" />
                <button onClick={handleClick}>등록</button>
            </div>
        );
    }
}
export default MyComponent;
```

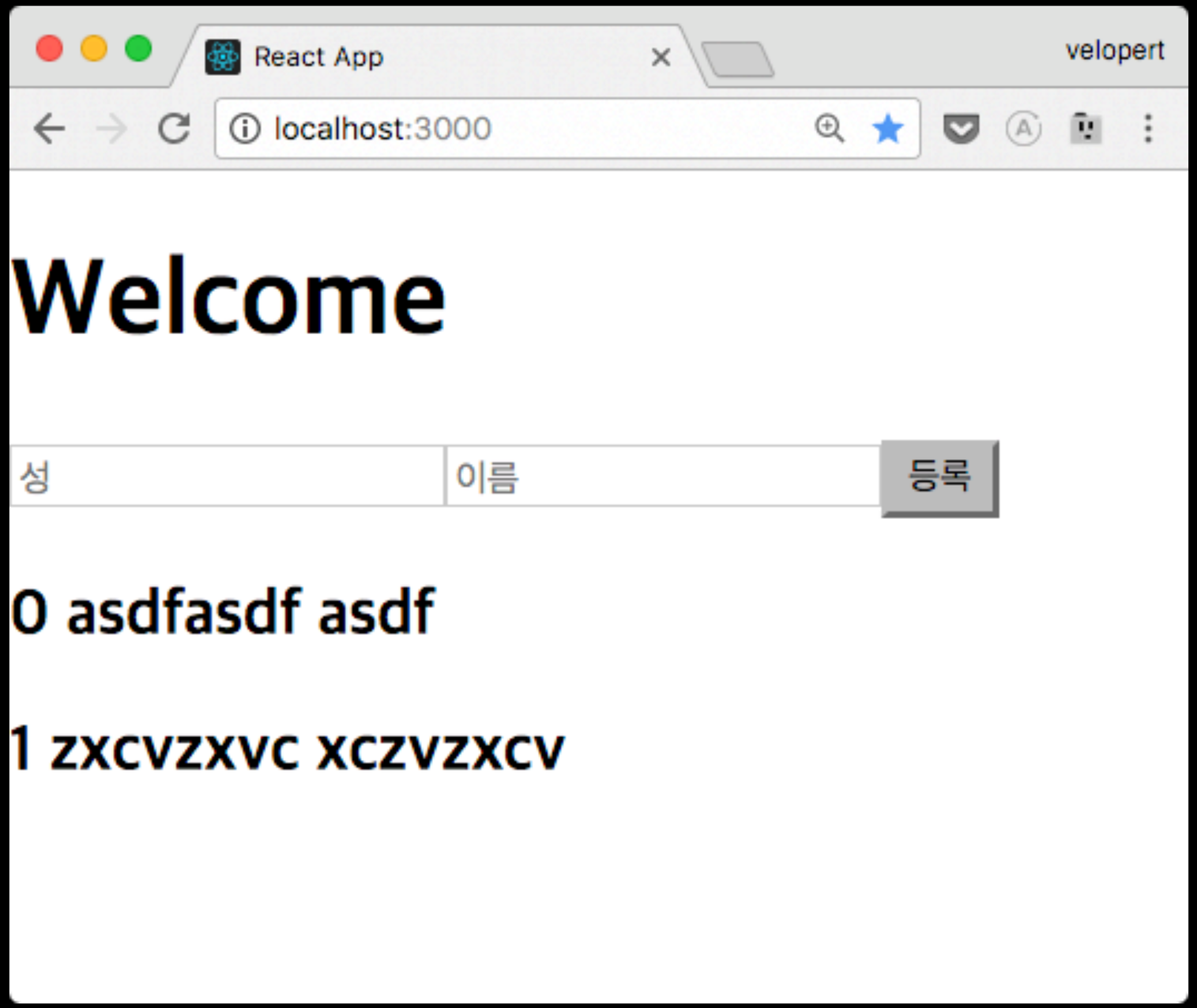
```
▼ Proxy {dispatchConfig: Object, _targetInst:
  ReactDOMComponent, nativeEvent: Event, type:
  "change", target: input…} i
  ▶ [[Handler]]: Object
  ▼ [[Target]]: SyntheticEvent
      bubbles: true
      cancelable: false
      currentTarget: null
      defaultPrevented: false
    ▶ dispatchConfig: Object
      eventPhase: 3
    ▶ isDefaultPrevented: function ()
    ▶ isPersistent: function ()
    ▶ isPropagationStopped: function ()
      isTrusted: true
    ▶ nativeEvent: Event
    ▼ target: input
        accept: ""
        accessKey: ""
        align: ""
        alt: ""
        assignedSlot: null
      ▶ attributes: NamedNodeMap
        autocapitalize: "sentences"
        autocomplete: ""
        autofocus: false
```

```
name: "lastname"
value: "hello"
```

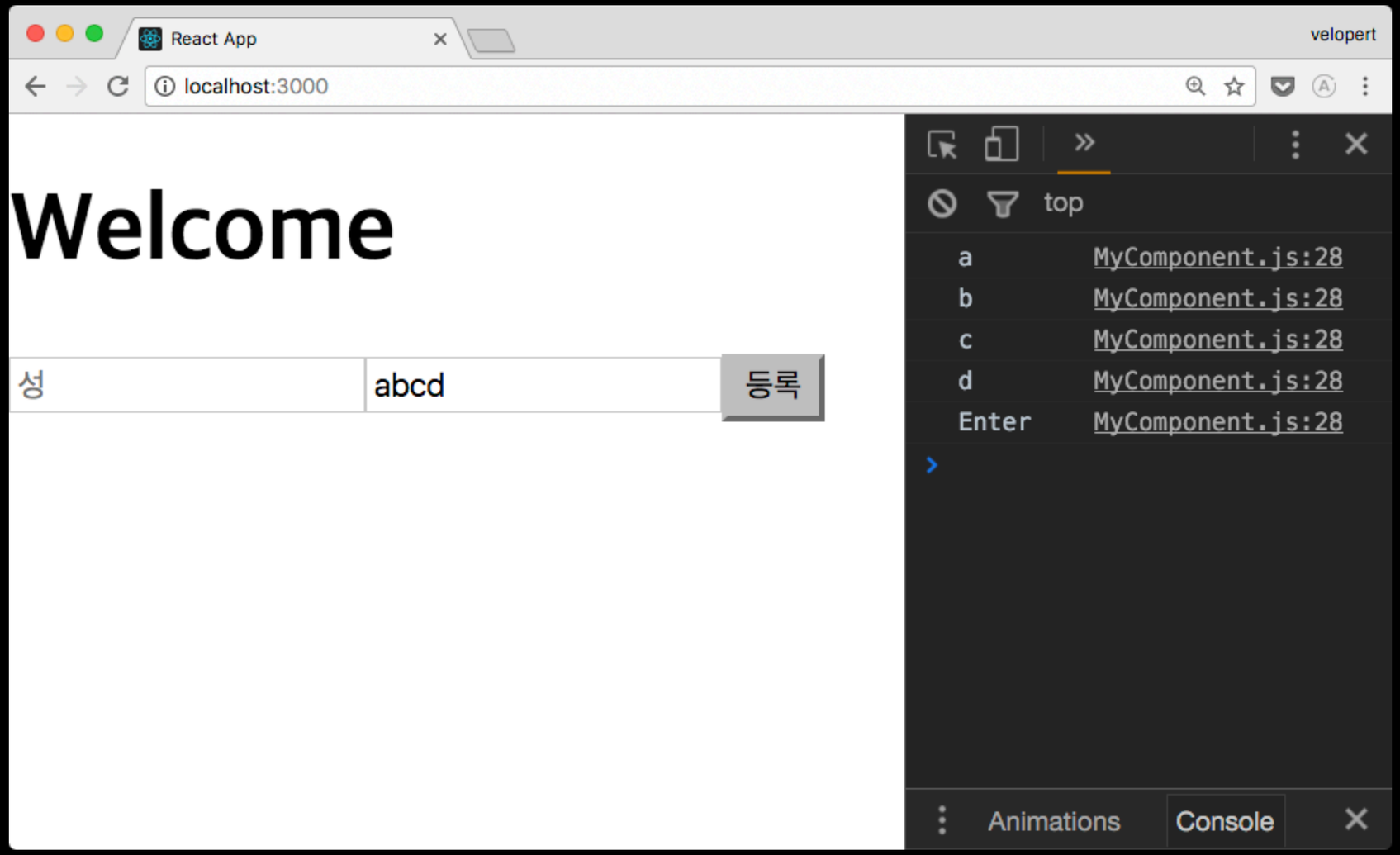
```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        lastname: '',
        firstname: ''
    }
    handleChange = (e) ⇒ {
        this.setState({
            [e.target.name]: e.target.value
        });
    }
    ( … )
```

```
render() {
    const { handleChange, handleClick } = this;
    const { lastname, firstname } = this.state;
    return (
        <div>
            <input
                onChange={handleChange}
                value={lastname}
                type="text"
                name="lastname"
                placeholder="성" />
            <input
                onChange={handleChange}
                value={firstname}
                type="text"
                name="firstname"
                placeholder="이름" />
            <button onClick={handleClick}>등록</button>
            <h2>{lastname} {firstname}</h2>
        </div>
    );
}
```

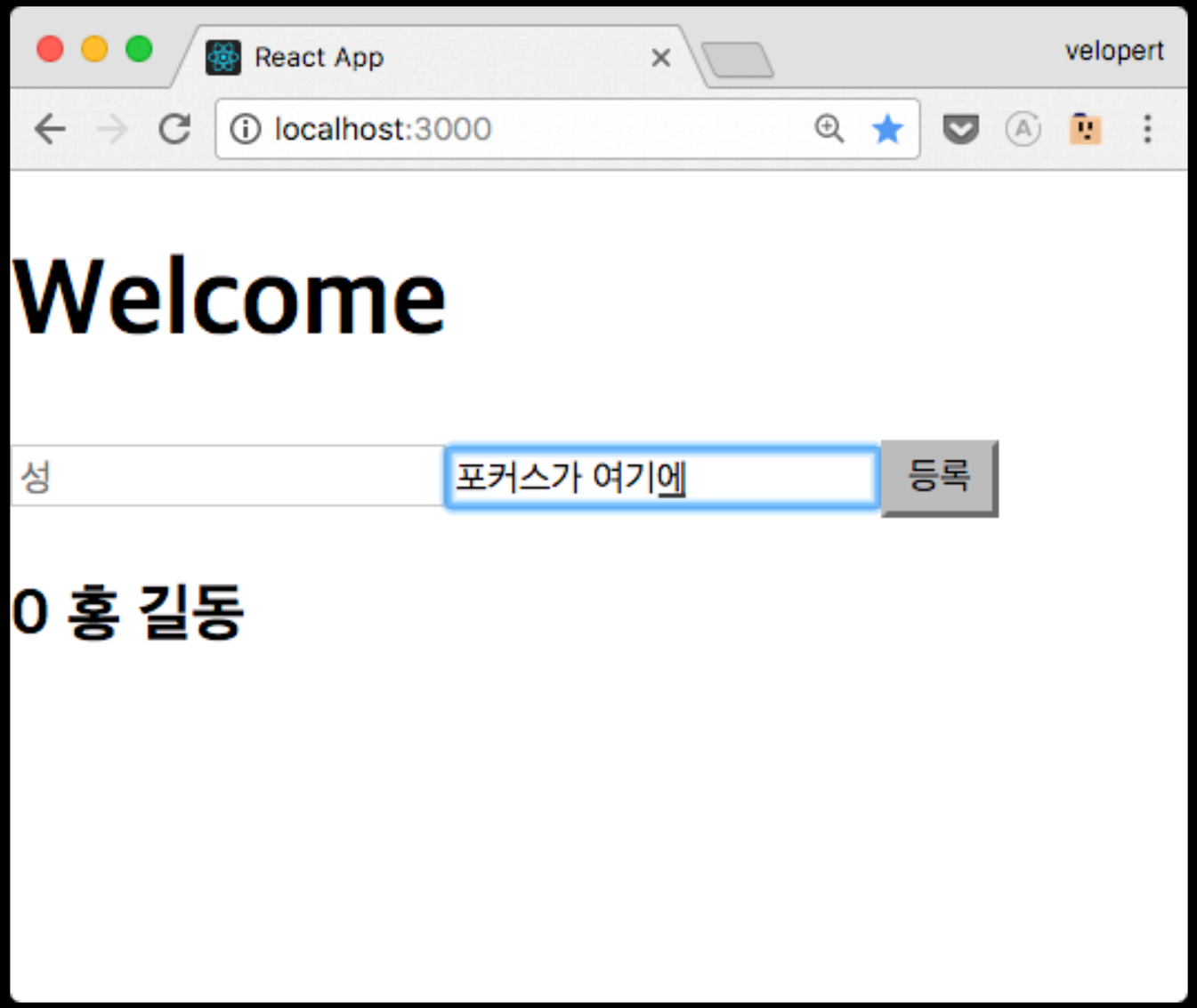
```
import React, { Component } from 'react';
class MyComponent extends Component {
    state = {
        lastname: '',
        firstname: '',
        names: []
    }

    ( ... )

    handleClick = (e) => {
        const { lastname, firstname, names } = this.state;
        this.setState({
            lastname: '',
            firstname: '',
            names: [ ... names, `${lastname} ${firstname}`]
        });
    }

    render() {
        const { handleChange, handleClick } = this;
        const { lastname, firstname, names } = this.state;
        return (
            <div>
                ( ... )
                <button onClick={handleClick}>등록</button>
                {names.map((name, index) => <h3 key={name}>{index} {name}</h3>)}
            </div>
        );
    }
}
export default MyComponent;
```


React App
velopert
localhost:3000
Welcome
성
이름
등록
0 asdfasdf asdf
1 zxcvzxvc xczvzxcv

```
// 메소드 추가
handleKeyPress = (e) => {
    console.log(e.key);
}
// 렌더함수 변경
const { handleChange, handleClick, handleKeyPress } = this;
( ... )
<input
     onChange={handleChange}
     value={firstname}
     type="text"
     name="firstname"
     placeholder="이름"
     onKeyPress={handleKeyPress}
/>
```


React App
velopert
localhost:3000
Welcome
성
abcd
등록
top
a MyComponent.js:28
b MyComponent.js:28
c MyComponent.js:28
d MyComponent.js:28
Enter MyComponent.js:28
Animations
Console

```
handleKeyPress = (e) => {
    if(e.key === 'Enter') this.handleClick();
}
```


React App
velopert
localhost:3000
Welcome
성
포커스가 여기에
등록
0 홍 길동

# **ref:** DOM에 직접 접근

```
<div ref={ref=>{this.element = ref}}></div>
```

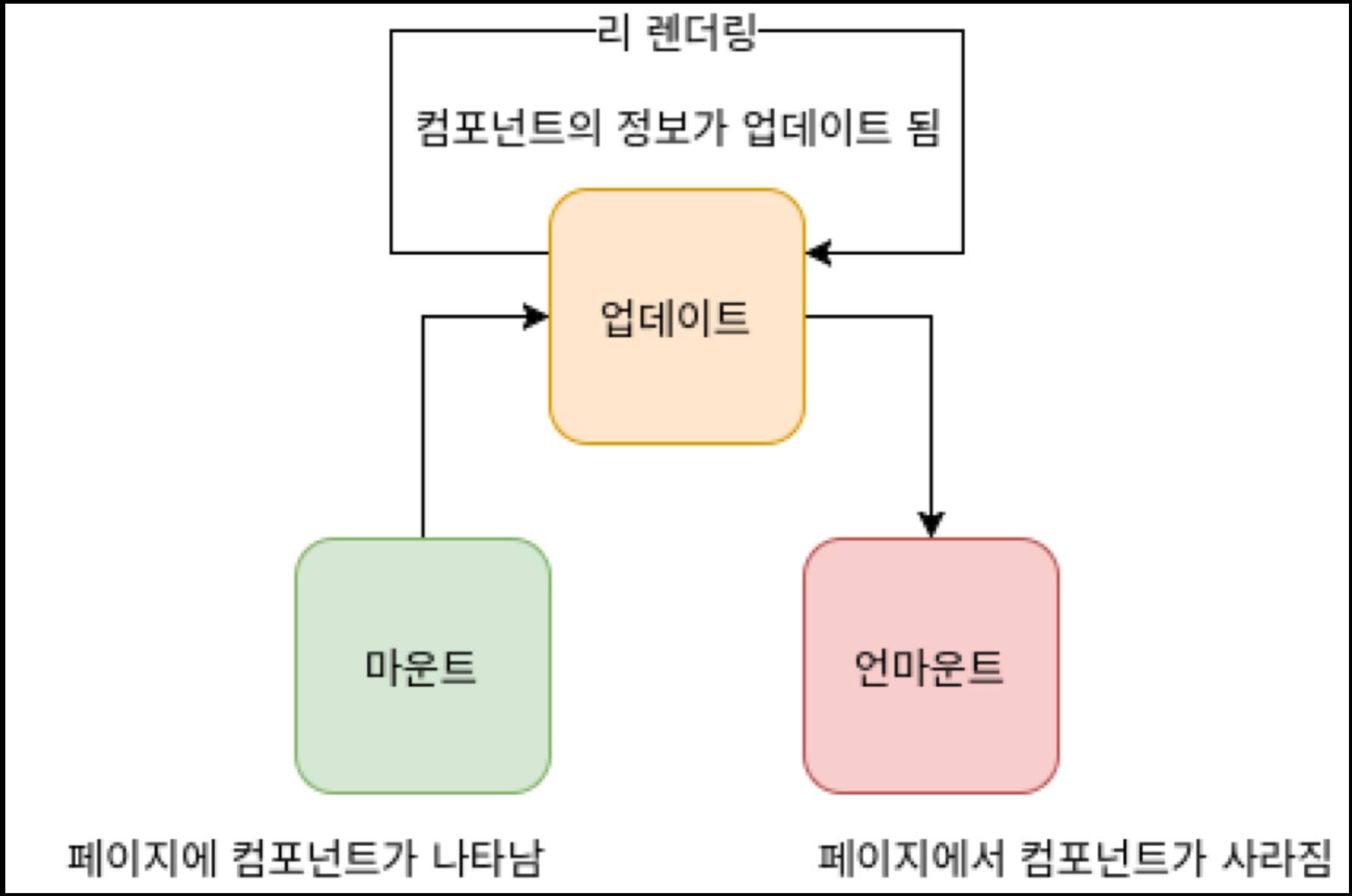
```
<input
    onChange={handleChange}
    value={lastname}
    type="text"
    name="lastname"
    placeholder="성"
    ref={ref⇒this.lastnameInput=ref}
/>
```

```
handleClick = (e) => {
    const { lastname, firstname, names } = this.state;
    this.setState({
        lastname: '',
        firstname: '',
        names: [...names, `${lastname} ${firstname}`]
    });
    this.lastnameInput.focus();
}
```

# 전체코드

https://gist.github.com/velopert/e41829893446f110adb3a015dd67d83e

# 컴포넌트의 LifeCycle 메소드


리 렌더링
컴포넌트의 정보가 업데이트 됨
업데이트
마운트
언마운트
페이지에 컴포넌트가 나타남
페이지에서 컴포넌트가 사라짐

# 마운트

constructor()
component**Will**Mount()
render()
component**Did**Mount()

# 업데이트

component**Will**ReceiveProps()
shouldComponentUpdate()
component**Will**Update()
render()
component**Did**Update()

# 언마운트

component**Will**Unmount()

# `render() { … }`

컴포넌트의 모양새를 정의해줌

컴포넌트 생성

`constructor→componentWillMount→`**`render`**`→componentDidMount`

컴포넌트 업데이트

`componentWillReceiveProps→shouldComponentUpdate`
`→componentWillUpdate→`**`render`**`→componentDidUpdate`

# constructor(props) { … }

컴포넌트의 생성자 메소드
초기 멤버변수 및 state 정의 - class property 로도 할 수 있음
내부에서 super(props) 실행해 주어야함

컴포넌트 생성

**constructor**→componentWillMount→render→componentDidMount

# `componentWillMount() { … }`

DOM에 나타나기 전에 실행되는 메소드
한번만 실행되며, this.props / this.state 사용 가능
setState 사용 가능, DOM 에 접근 불가능
서버사이드에서도 실행됨

컴포넌트 생성

`constructor`→**`componentWillMount`**→`render`→`componentDidMount`

# `componentDidMount() { … }`

첫 렌더링 후 실행되는 메소드
다른 자바스크립트 라이브러리 / 프레임워크 함수 호출
이벤트 등록 / setTimeout 및 네트워크 요청 / 비동기작업은 여기서 합니다
DOM 에 접근 가능

컴포넌트 생성

`constructor→componentWillMount→render→`**`componentDidMount`**

# `componentWillReceiveProps(nextProps) {…}`

부모로부터 새 props를 전달 받게될때 실행됩니다
props 값에 따라 state 에 변화를 주어야 할때 여기서 작업을 하면 됩니다
setState 메소드를 실행 가능하며, 새 props 는 nextProps 키워드로 접근가능합니다

컴포넌트 업데이트

**`componentWillReceiveProps`**→`shouldComponentUpdate`
→`componentWillUpdate`→`render`→`componentDidUpdate`

# shouldComponentUpdate(nextProps, nextState) {…}

props 나 state 가 변경되면 이 메소드가 실행됩니다
true / false 값을 반환해야합니다.
false 를 반환하면 업데이트가 중지됩니다.
컴포넌트를 최적화할때 중요한 역할을 합니다.
기본값으론 true를 반환합니다

컴포넌트 업데이트

componentWillReceiveProps→**shouldComponentUpdate**
→componentWillUpdate→render→componentDidUpdate

# `componentWillUpdate(nextProps, nextState) {…}`

shouldComponentUpdate 가 true 를 반환했을때 실행됩니다
업데이트를 실행하기 전에 실행됩니다
DOM 조작 및 setState 를 하면 정상적으로 작동하지 않습니다

컴포넌트 업데이트

`componentWillReceiveProps→shouldComponentUpdate`
`→`**`componentWillUpdate`**`→render→componentDidUpdate`

# `componentDidUpdate(prevProps, prevState) {…}`

리렌더링이 마친 후 실행됩니다
DOM 관련 처리를 해도 됩니다
prevProps 혹은 prevState 를 사용하여
이전에 지니고있던 데이터에 접근할수있습니다

컴포넌트 업데이트

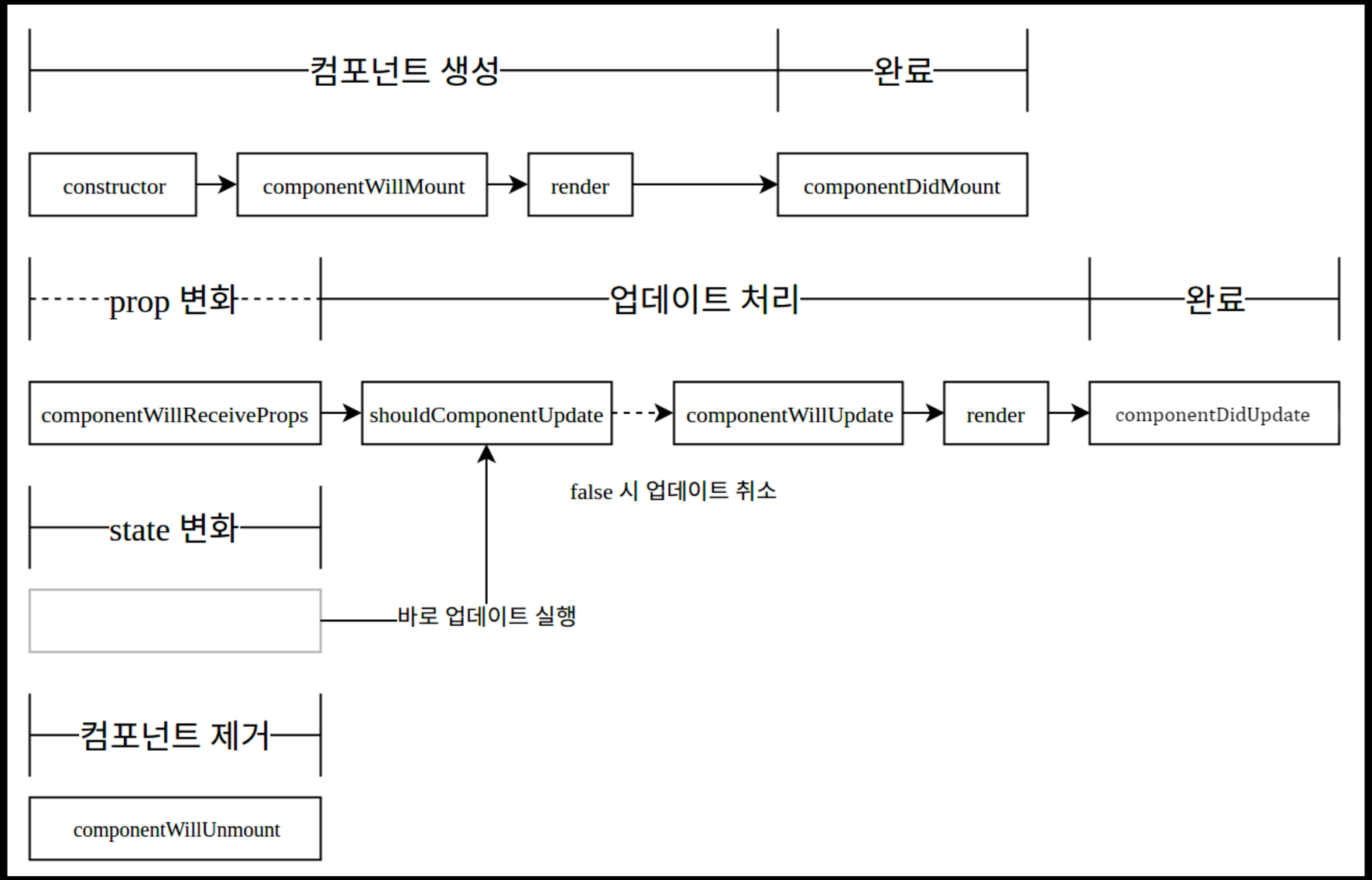
`componentWillReceiveProps→shouldComponentUpdate`
`→componentWillUpdate→render→`**`componentDidUpdate`**

# componentWillUnmount

컴포넌트가 DOM에서 제거 될 때 실행됩니다.
여기서 타이머 / 이벤트 등을 제거합니다

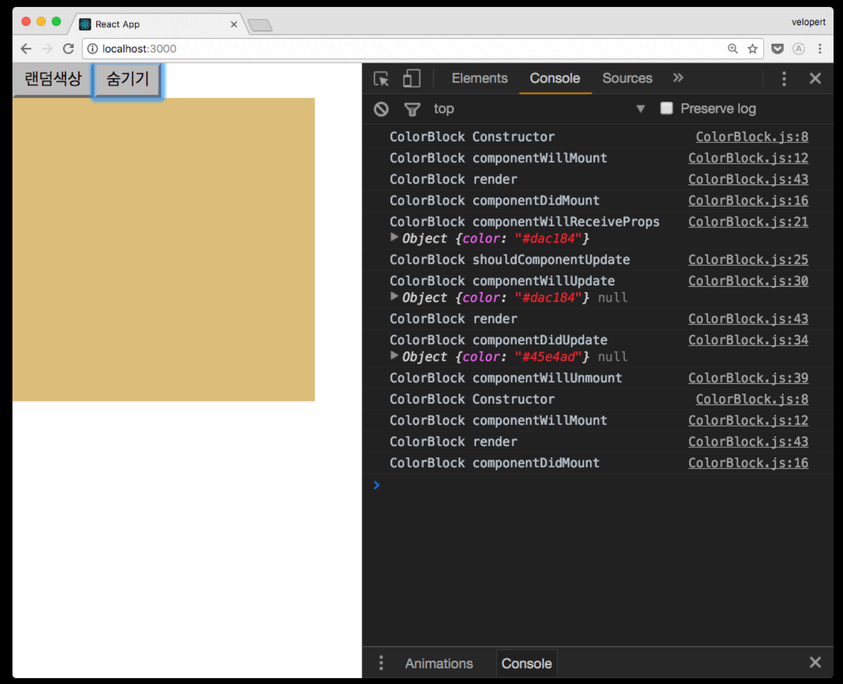
컴포넌트 언마운트

**componentWillUnmount**


컴포넌트 생성
완료
constructor
componentWillMount
render
componentDidMount
prop 변화
업데이트 처리
완료
componentWillReceiveProps
shouldComponentUpdate
componentWillUpdate
render
componentDidUpdate
false 시 업데이트 취소
state 변화
바로 업데이트 실행
컴포넌트 제거
componentWillUnmount

# LifeCycle API 실습

https://gist.github.com/velopert/3126fa38067183b7e03ce7f64ba056a6



참고: https://reactarmory.com/guides/lifecycle-simulators

쉼.