React 소개 및 구현방법 Demo

김대성

2017. 1. 24 W3C HTML KIG

소개

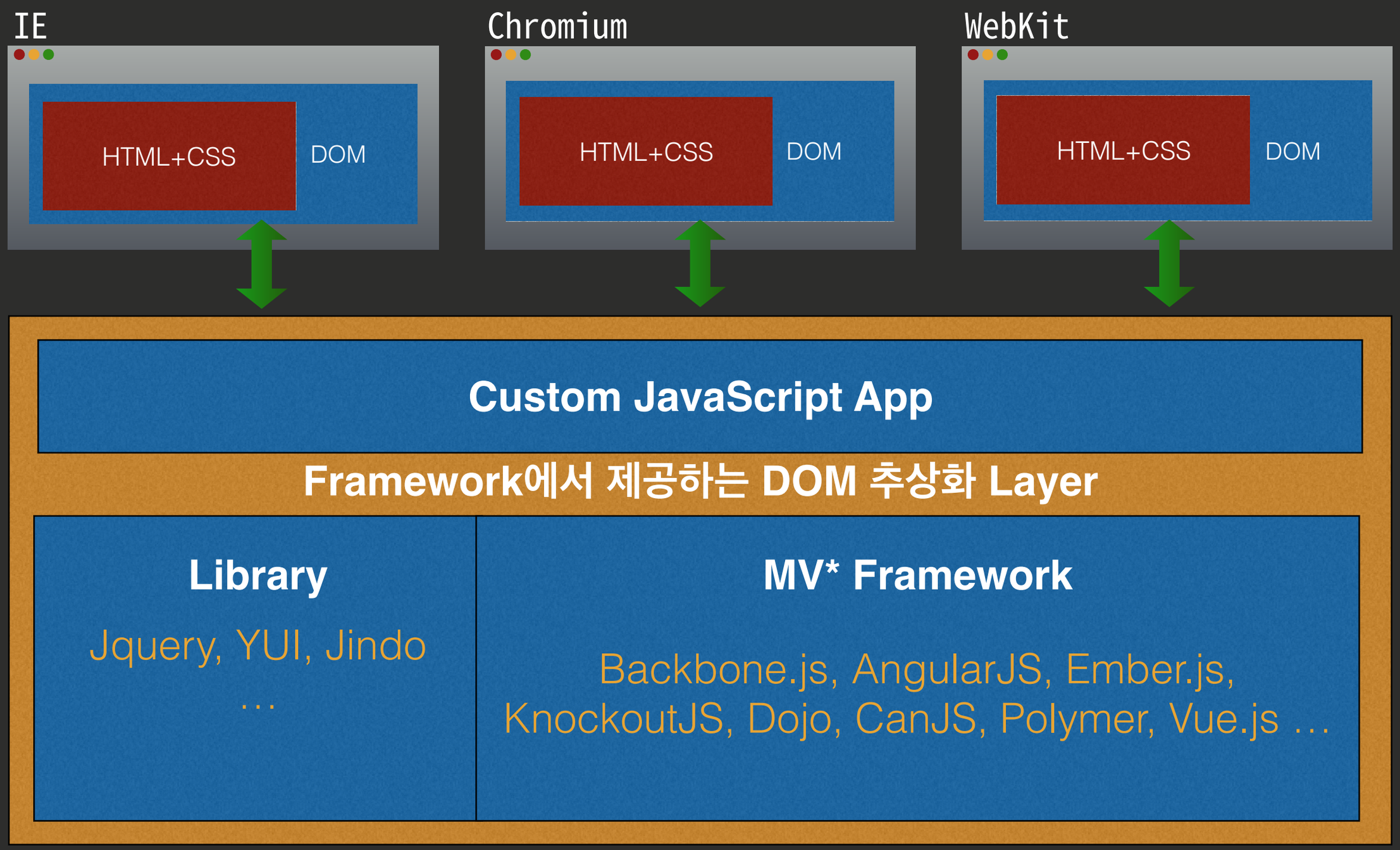
김대성 Frontend engineer

t: @danielkimreally
e: daesung.kim1@yahoo.com

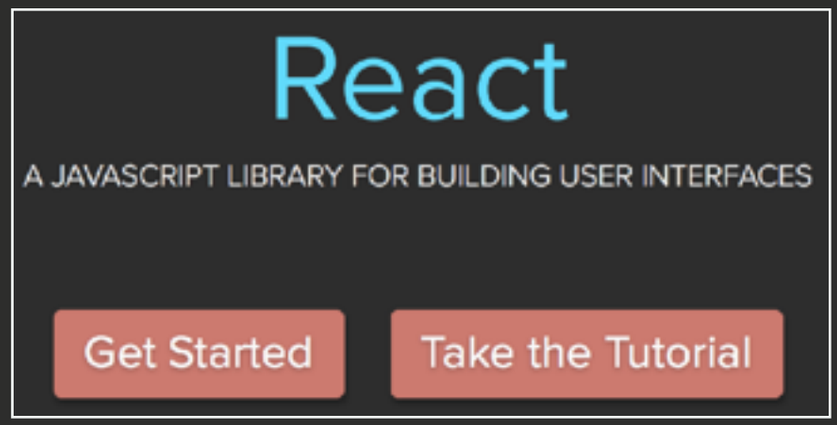
목차

- background
- React Components
- Component Design
- Component 구현 Demo
- Summary

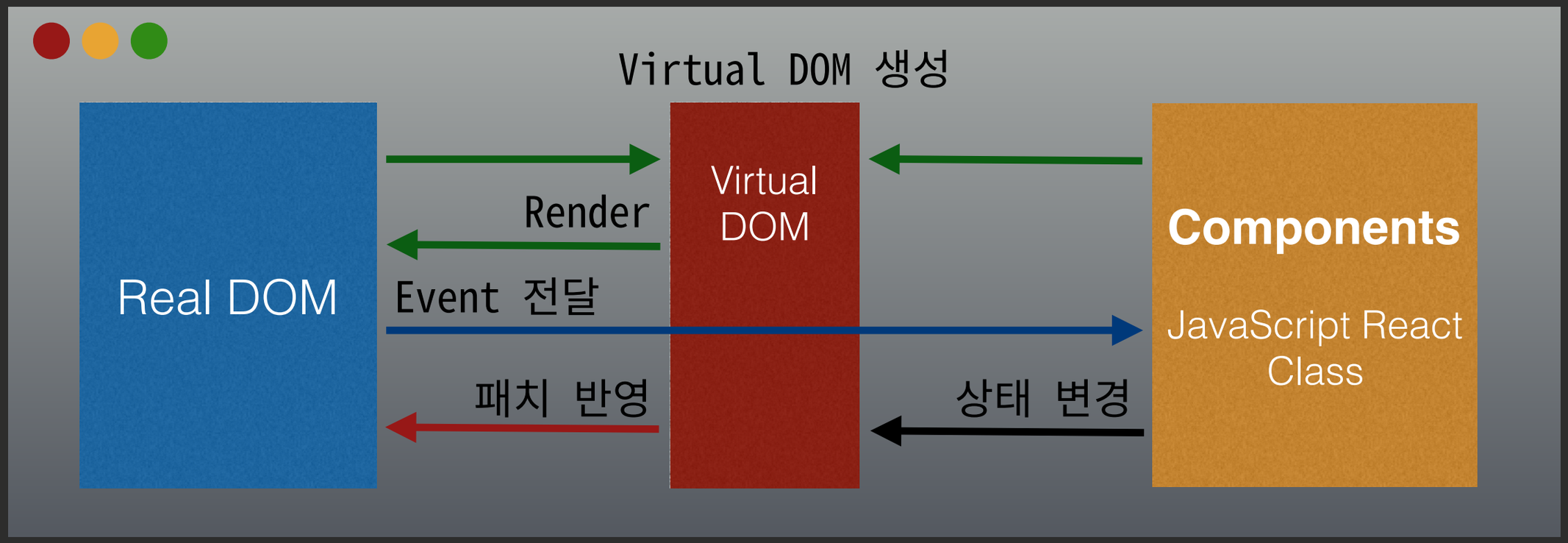
Web UI 구현방법


IE
HTML+CSS
DOM
Chromium
HTML+CSS
DOM
WebKit
HTML+CSS
DOM
Custom JavaScript App
Framework에서 제공하는 DOM 추상화 Layer
Library
Jquery, YUI, Jindo
…
MV* Framework
Backbone.js, AngularJS, Ember.js,
KnockoutJS, Dojo, CanJS, Polymer, Vue.js …

React Framework..?



- 단지 View를 위한 JavaScript Library
- MVC의 View가 아님 (React DOC에서 삭제되었음)
- View를 제외한 나머지 처리는 개발자의 몫

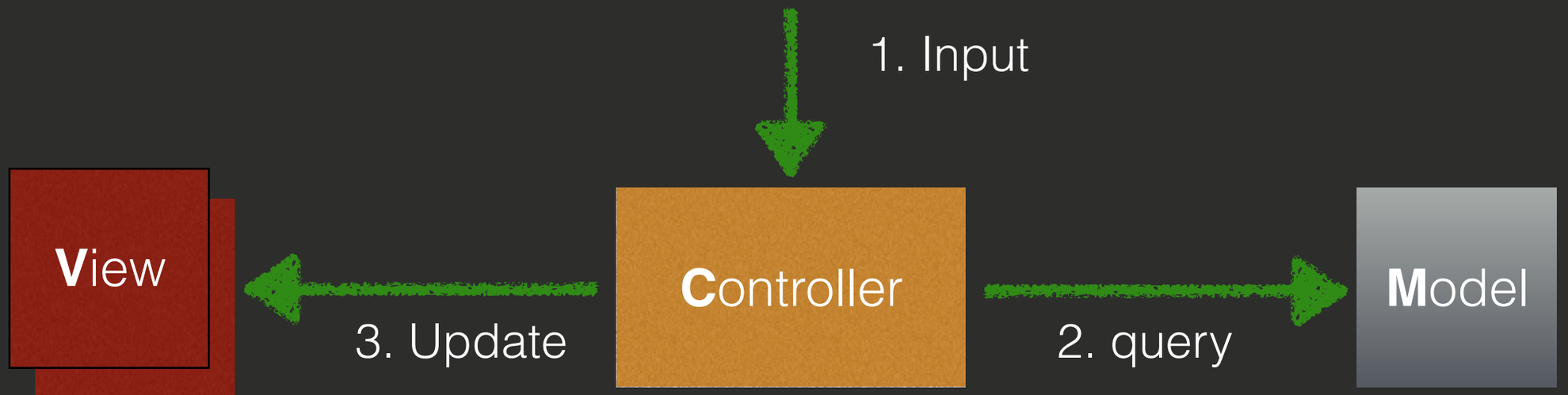


화면출처: https://facebook.github.io/react/

Background

React는 MVC 구현과 어울리지 않음

MVC (Model View Controller)

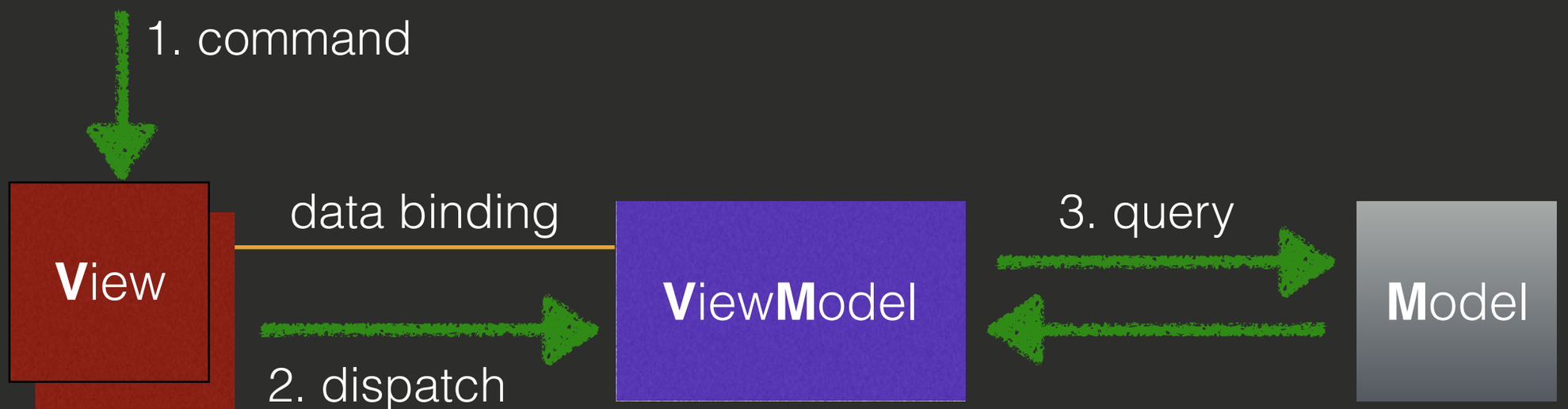


View를 update 하려면

Controller에서 수행해 주어야 함 (View는 수동적)

예) 갱신이 필요한 Templetes에 Model의 Data를 채움

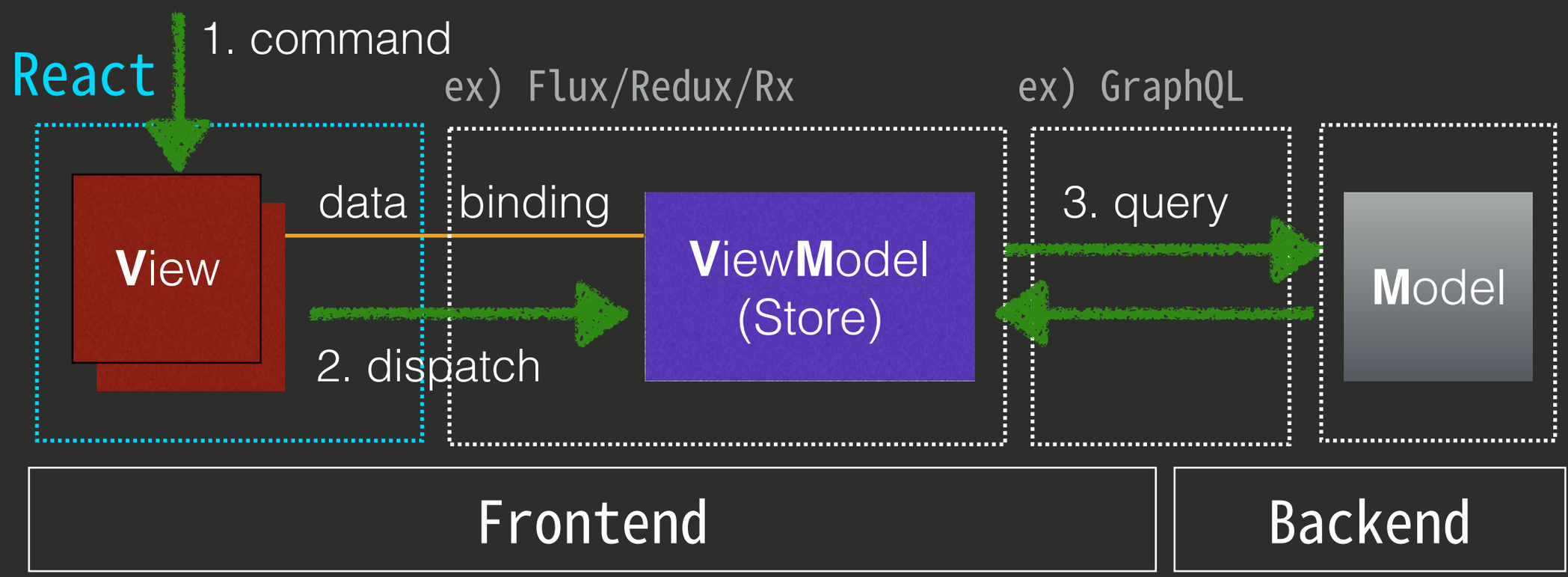
MVVM (Model View ViewModel)



State가 변경되면 관련 View들이 Update 수행

예) Excel의 그래프 차트를 업데이트 하려면 데이터 시트 값만 바꿔줌

MVVM 패턴의 Web App



React는 View Layer만 담당

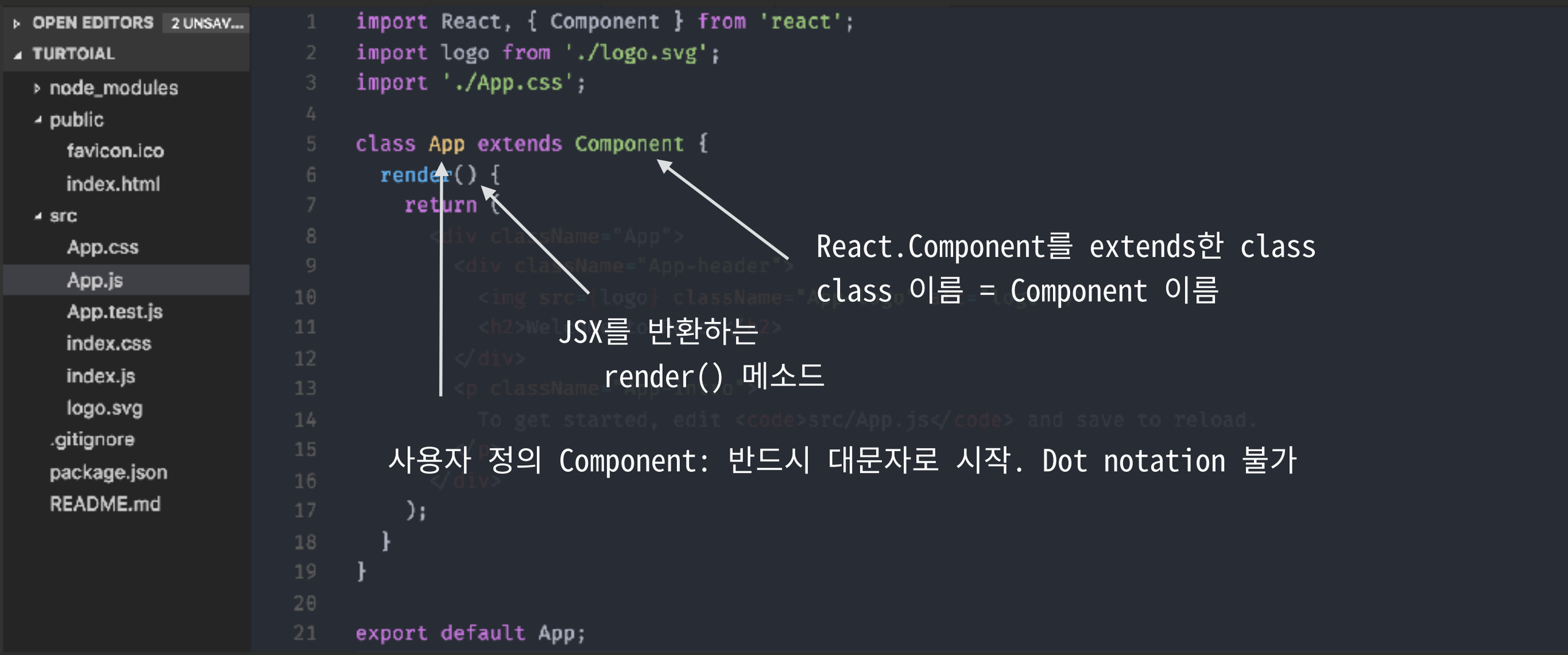
React Components

View를 표시하고 조작하는 JavaScript Class

- React, ReactDOM 객체
- 사용자 정의 Component
- props & state

Components with JSX

JSX !== HTML, React에서 사용하는 DOM 표기방식



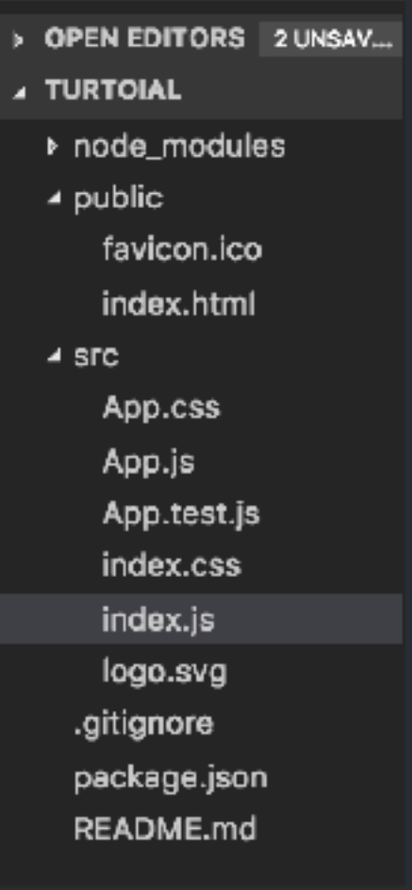
Components with JSX

Babel에 의해 JavaScript 코드로 변환됨 (Transfile)

JSX는 React.createElement(component, props, ...children) 구문의 syntactic sugar

ReactDOM.Render()

App Component를 호출하여 UI 생성

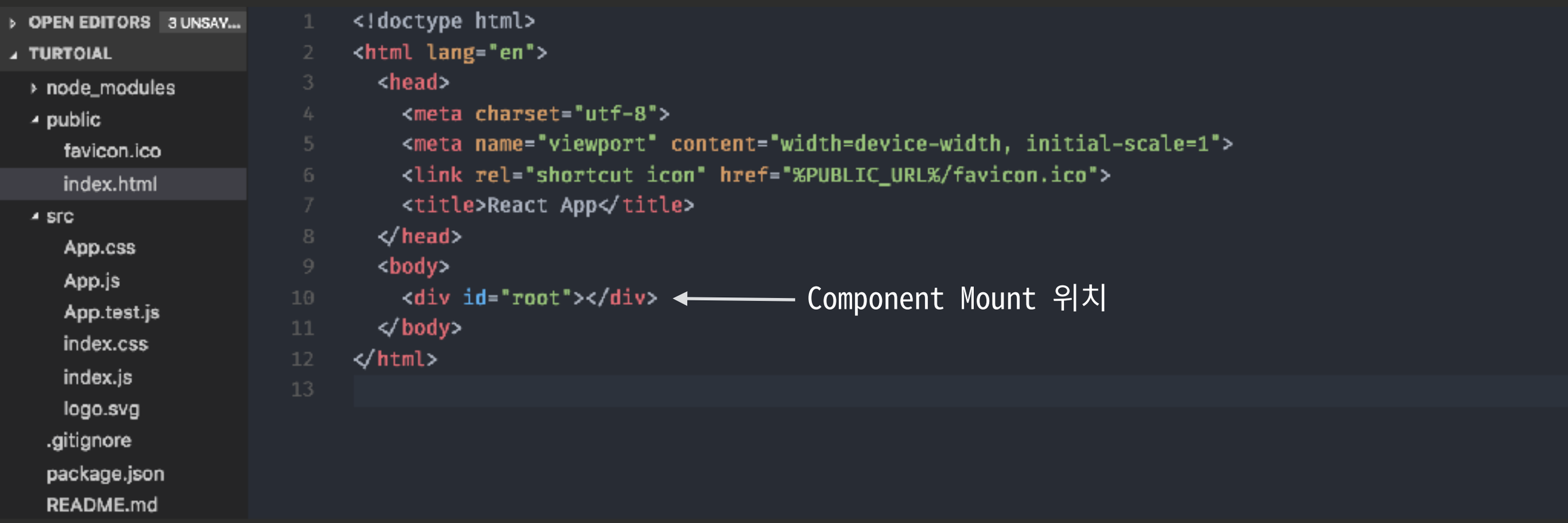



```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3   import App from './App';
4   import './index.css';
5
6   ReactDOM.render(
7     <App />,
8     document.getElementById('root')
9   );
10
```

- Embed된 JSX는 JavaScript 객체
- Props를 통해 미리 정의된 객체, 변수 혹은 참조와 함께 전달 가능

Component Mount

지정된 element에 Mount됨



Props

- Components의 초기 설정
- readonly
- child component에 전달가능
  (속성 값 혹은 methods)
- defaultValue 정의가능
- propsTypes통해 Interface 역할
  (Interface가 같으면 대체가능)

function 정의 (리턴 객체)

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Class 정의 (render method의 리턴 객체)

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

지역변수에 할당가능

```
const element = <Welcome name="Sara" />;
```

다른 Components에서 사용됨

```
function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}
```

States

- 동적인 Components 상태를 객체로 정의 (this.state)
- Mount 이후 UI의 Update를 일으키는 Data를 보관

화면출처: daum.net

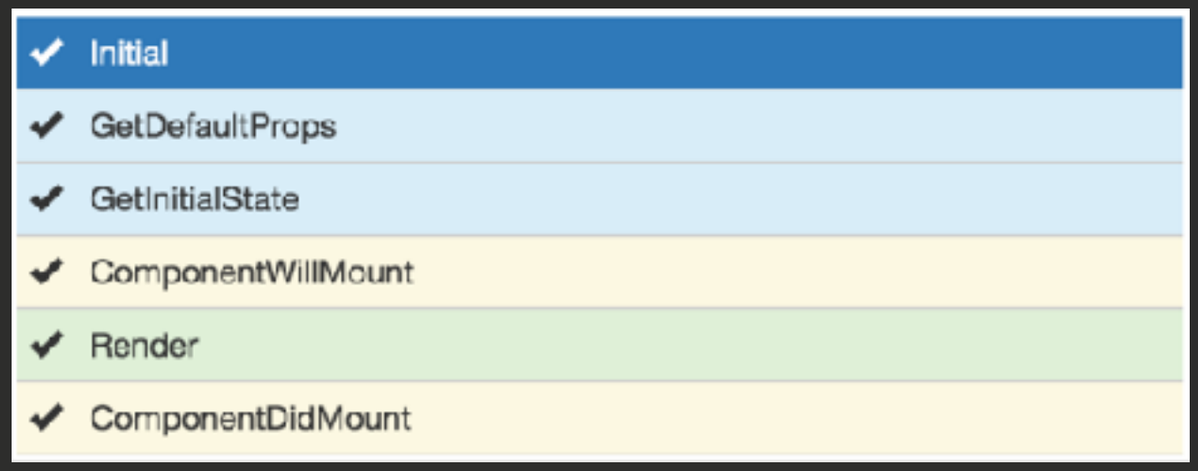
Component 내부의 state 객체 예

```
{
  activeTab: “news",
  currentPage: 1
  articles : [
    { … },
    { … },
    { … },
  ]
}
```

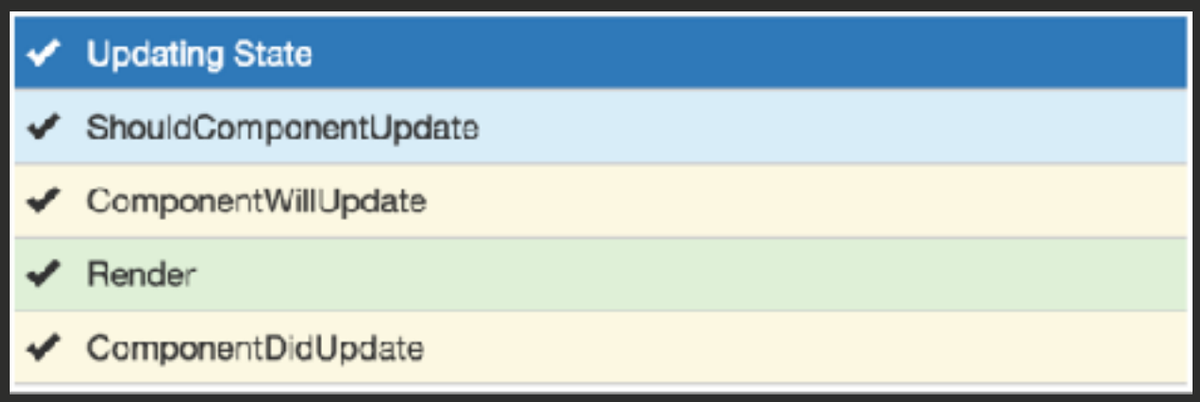
이벤트 핸들러로 state만 Update하면 React가 View를 그려줌

Component Lifecycle

초기화



State가 변경될때



Unmount

출처: http://busypeoples.github.io/post/react-component-lifecycle/

Components Design

- 시안분석 및 요구사항 확인
- Props과 State 구분
- 단일 책임 원칙에 의한 Component 분리

Component Design

설계의 목적: 하나의 components로 여러 서비스/스크린에서 재사용

Desktop

Mobile

화면출처: daum.net

Component 공통 Spec

- Editorial Tool에서 편집 가능한
- 여러 Vertical 서비스의 공용 모듈
- 서비스별 Customize 가능
- Screen에 따른 View 지원
- i18n 지원
- static contents 제공 (장애시)
- 지정가능한 API
- ...

Props & State 결정

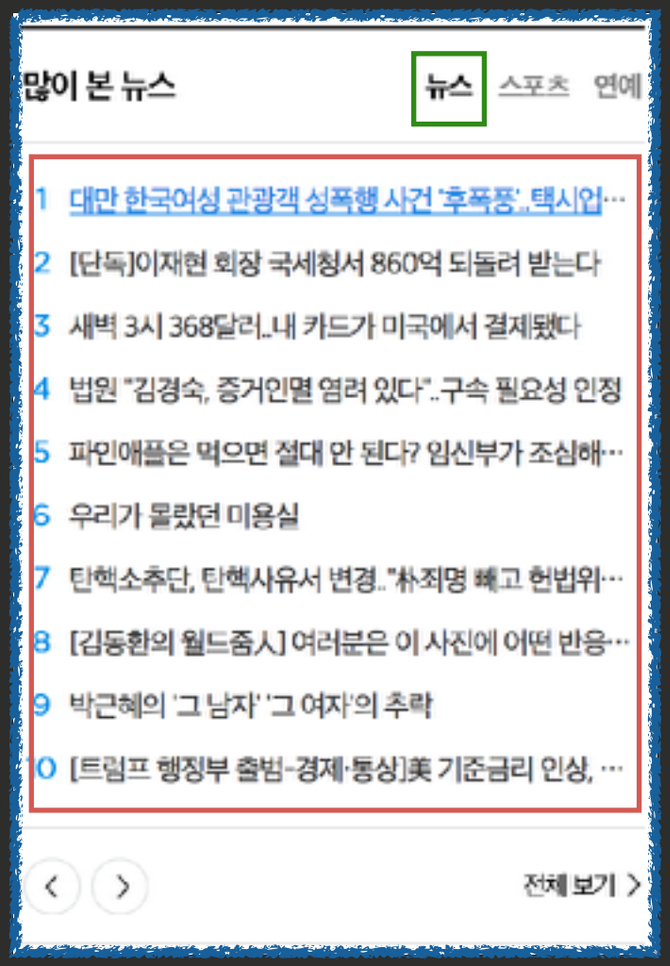
```
ReactDOM.render(
  <Ranking
    tabList={["뉴스", "스포츠", "연예"]}
    activeTab="연예"
  />,
  document.getElementById('root')
);
```

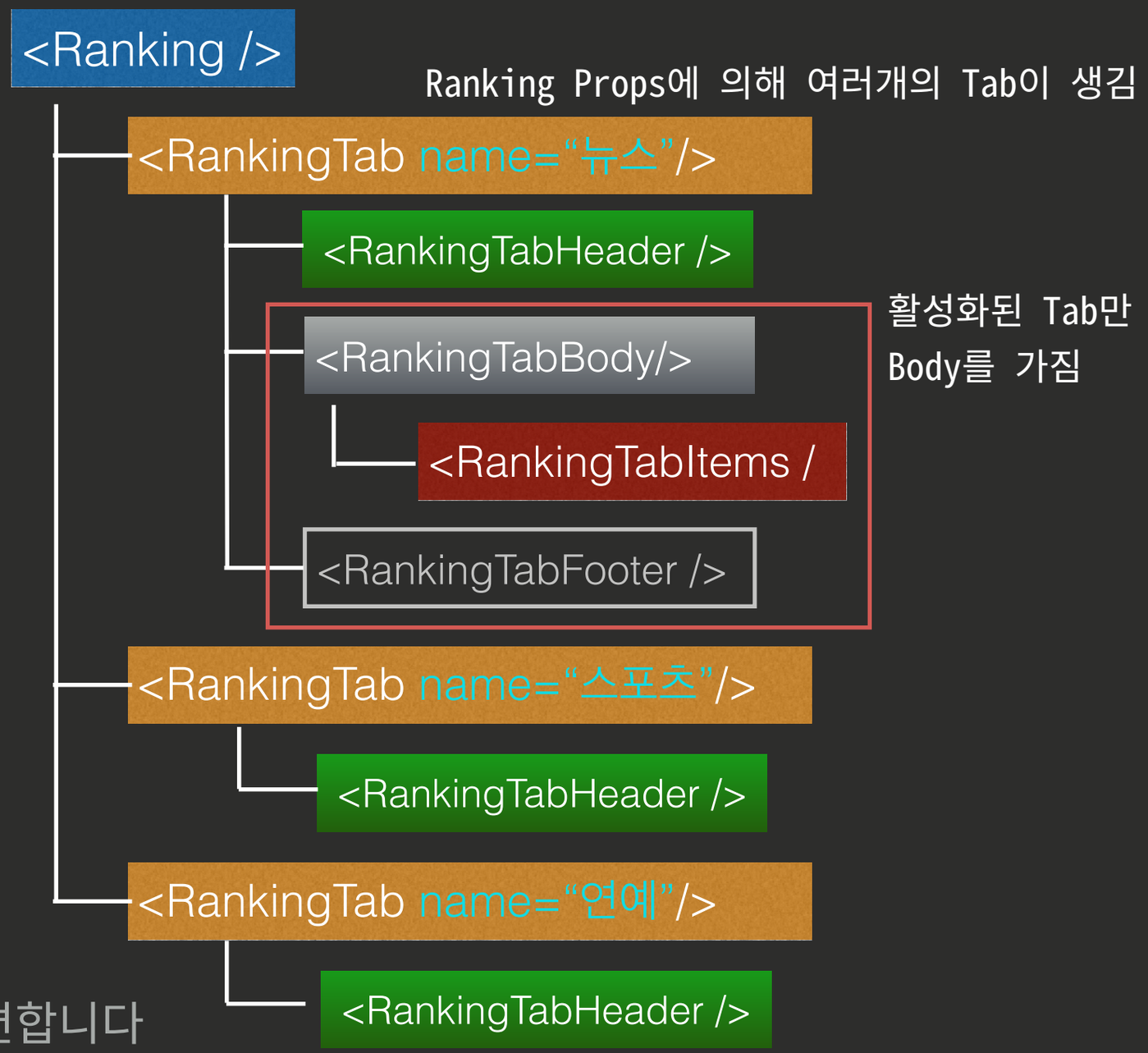
Props)
mount시 결정되고 유지됨

State)
실행중에 활성화된 Tab 보관

```
{
  activeTab : '',
  stories : [
    {
      title: '안정환 "아내 이혜원 때문에 벌금 천만원 낸 적 있어"',
      type: '연예',
      rankByMostPopular: 7,
      visited: false
    },
...
```

Component 계층화

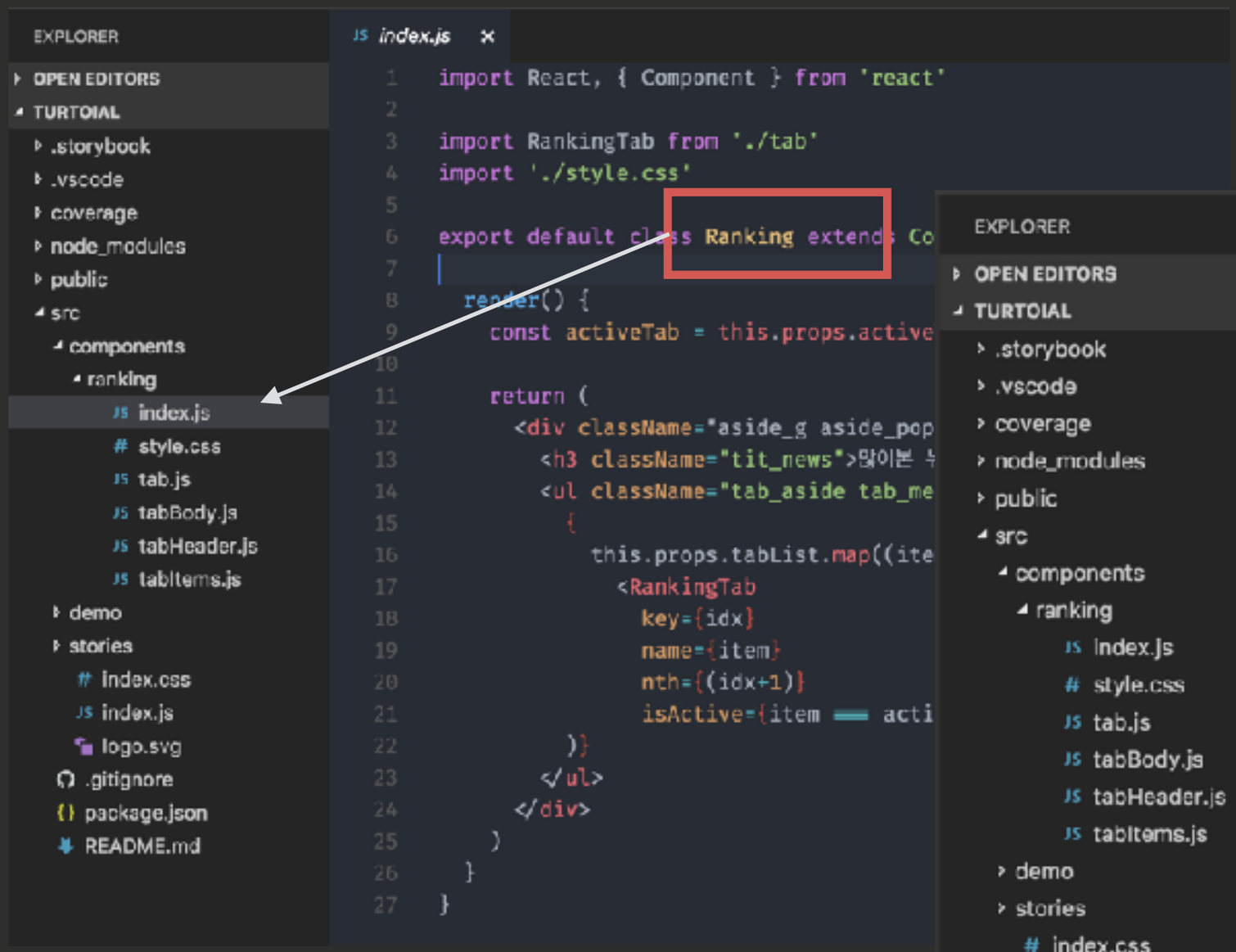


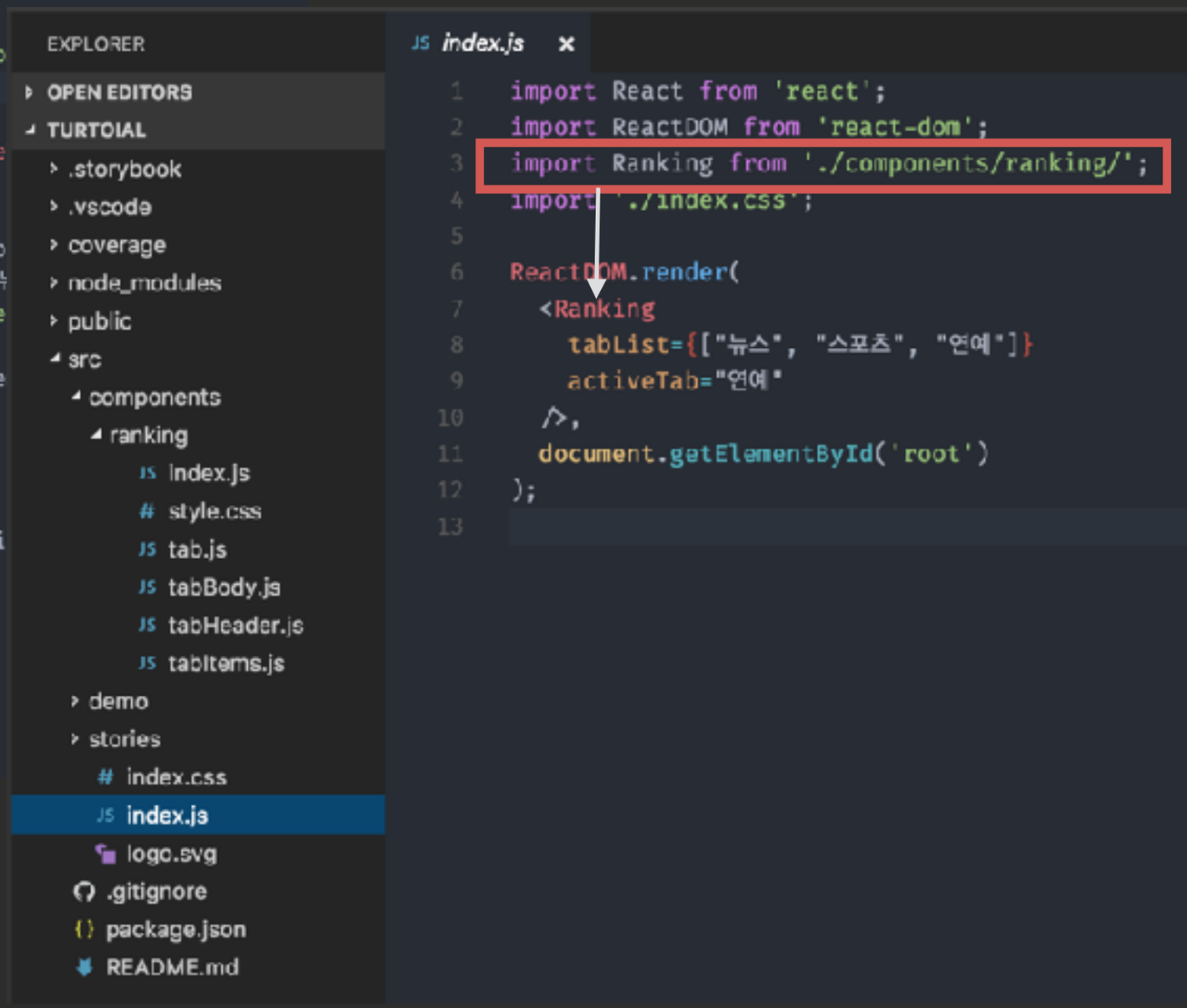


HTML markup 부터 작업하시면 편합니다

Component 구현 Demo

Directory 구조





coding session

- Markup 및 CSS는 daum.net으로 부터 Copy하여 준비
- 환경세팅: https://github.com/facebookincubator/create-react-app
- 코드: https://github.com/daesungkim1/react-tutorial

구현과정 정리

Static View를 구현

1. 계층 구조 확인
2. 자식 Components에 제공할 Props을 정의하며 render() 완성
3. 완성된 View확인
4. Components 계층구조 검증 (PropsTypes, DefaultProps 추가)

View Update 구현

1. 최소한의 State를 추가
2. render()에 state로 data를 가져오고 setState()로 Update확인

Unittest 코드 작성

1. Components가 의도한 Markup을 Render했는지 검증 (jest & enzyme)

Summary

그밖에…

- Flux, Redux (States Management Library)
- RxJs / GraphQL / Relay / Apollo

END