

Hello,

KDT 웹 개발자 양성 프로젝트

1기!! 37th

with





ko

a

next generation web framework for node.js



Koa 프레임워크 설치 및 실행

- npm install koa

```
const Koa = require('koa');
const app = new Koa();
const PORT = 4500;

app.use(async (ctx, next) => {
  console.log(ctx.request);
  console.log(ctx.response);
  ctx.body = 'Hello, koa world!';
});

app.listen(PORT);
```

Hello, koa world!



pug



pug + 부트스트랩

- 부트스트랩을 이용해서 간단한 채팅 UI 디자인

```
html
  head
    link(href="")
  body.d-flex.text-center.flex-column.align-items-center
    h1.w-100.p-3.bg-primary.text-white.font-bold Tetz 채팅
    div.w-50.h-75.p-5.bg-secondary.text-bottom.overflow-auto
      p.p-2.bg-primary.fw-bold.text-white 채팅 예시
    form.w-50
      input.w-75.m-3.p-3
      a.p-3.btn.btn-primary 보내기
```



WEBSOCKETS





Koa-web socket 사용해 보기

- 웹 소켓 사용을 위한 코드 모듈 추가

```
app.ws.use(  
  route.all('/chat', (ctx) => {  
    ctx.websocket.send('아아~ 들리십니까 여긴 서버입니다!');  
    ctx.websocket.on('message', (message) => {  
      console.log(message.toString());  
    });  
  })  
);
```

- /chat 이라는 경로로 들어오면 웹 소켓 통신을 통해 클라이언트로 Hello, world 라는 문구를 전송
- 클라이언트에서 message 가 오면 message 를 받아서 서버에 출력



IIFE

(Immediately Invoked Function Expression)



IIFE(Immediately Invoked Function Expression)

- 이런 부분을 감추고 싶을 때, IIFE 를 사용합니다

```
// IIFE
(function () {
  const socket = new WebSocket(`ws://${window.location.host}/chat`);

  socket.addEventListener('open', () => {
    socket.send('안녕하세요, 저는 클라이언트에요!');
  });

  socket.addEventListener('message', (event) => {
    console.log(event.data);
  });
})();
```

모든 클라이언트에게 데이터를 보낸다 실시!
➤ socket



MongoDB

연결



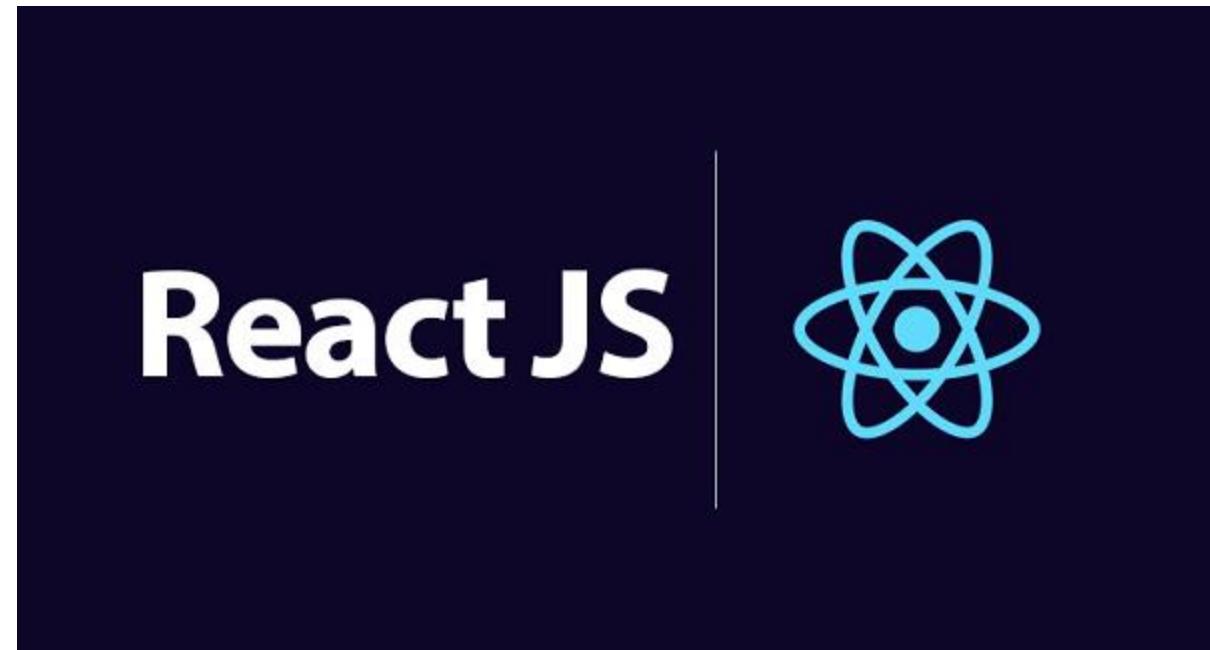
YAY



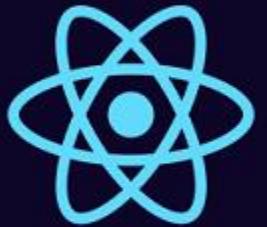
WE DID IT!

SpongeBob SquarePants vector trace by Jassi, iStockPhoto.com

makeameme.org



React JS



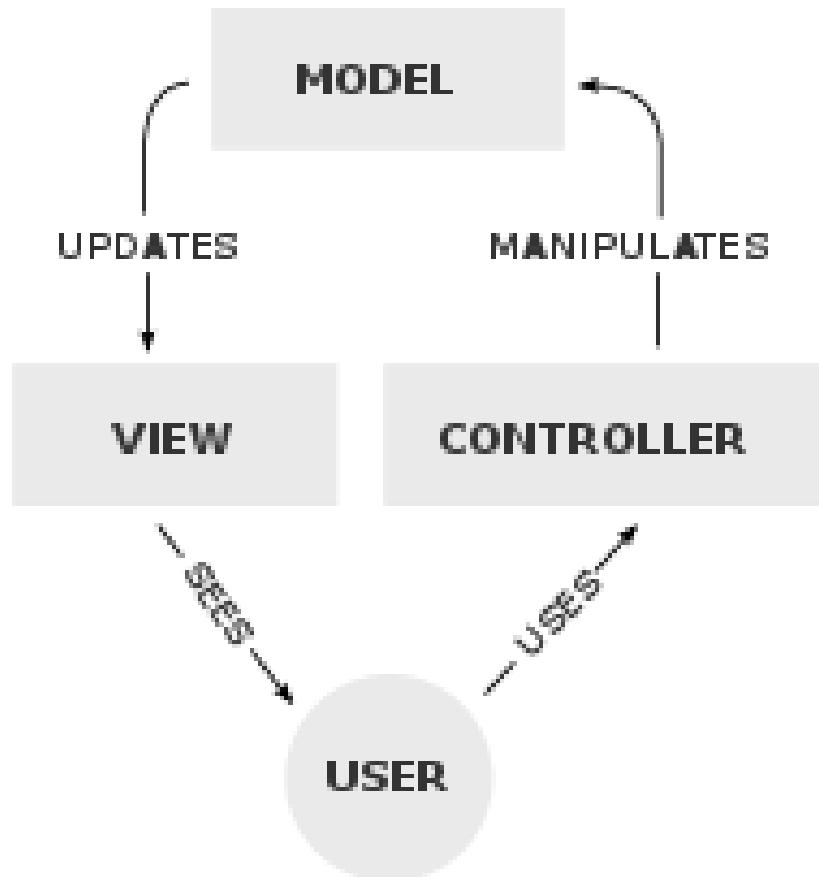
가즈아

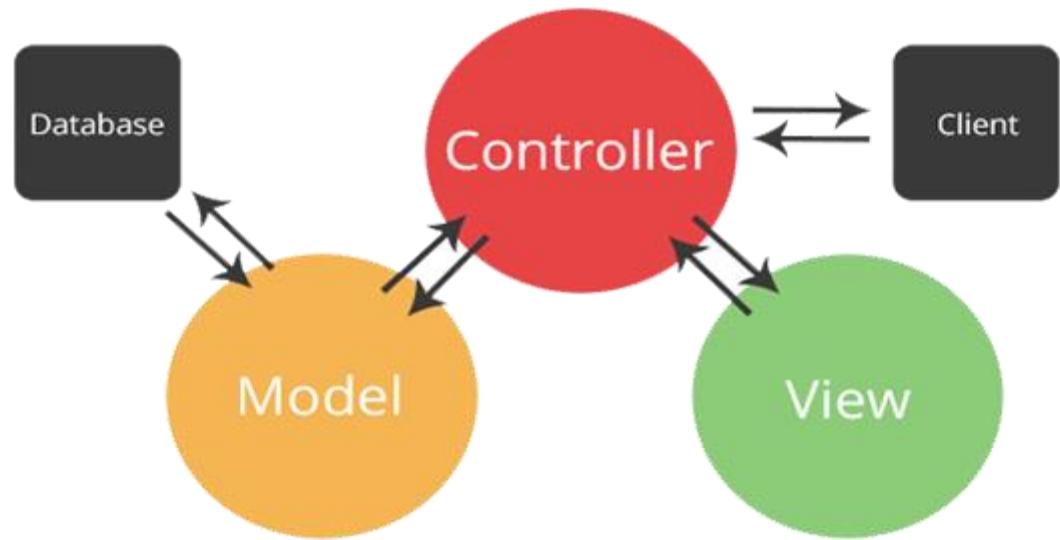






MVC 패턴







MVC?

- Model View Controller 의 약자입니다!
- 웹을 설계에 대한 구조입니다!
- 웹 설계를 3가지 단계로 구분하여 구조적 장점을 가지기 위한 수단 입니다



Model

- 어플리케이션의 데이터를 처리하는 역할을 합니다
- 사용자가 볼 수 없는 곳에서 DB로 부터 데이터를 읽고, 삽입하고, 수정하고, 삭제하는 역할을 합니다
- Controller 와 소통하며, View 와는 소통하지 않습니다
- 우리로 따지면 MongoDB에서 데이터를 읽고, 삽입하고, 수정하는 역할을 합니다 → 단, 이 모든 기능을 하나의 mongo.js에 묶어서 처리하는 방식이 되겠죠!



Controller

- Model 과 View 의 상호작용을 컨트롤 합니다
- Model 로 부터 전달 받은 데이터를 가공하여 View 에게 전달합니다
- 또한 View 부터 들어온 사용자 요청을 Model 에 전달 합니다!
- 사용자 요청은 전부 Controller 가 처리 합니다!
 - View 에서 삭제 버튼을 누름 → 해당 내용을 Controller 에 전달 → Controller 는 해당 내용을 Model 이 알아 들을 수 있는 언어로 변경하여 Model 에 전달 → DB에 서 해당 내용 삭제 → 해당 내용은 다시 Controller 를 통해 View 전달 → 사용자가 보는 화면이 변경

View



- Model 부터 받은 정보를 Controller 가 받아서 전달하면 그려주는 역할을 합니다
- 사용자가 보는 화면을 담당
- 사용자의 요청을 Controller 로 전달 합니다



MVC의 장단점

- 장점
 - 역할이 명확히 구분 되어 있어서 유지 보수가 쉽다
 - 기능 추가 및 확장, 유지 보수가 쉽다
 - 프론트 백이 서로 독립적으로 개발 이 가능
- 단점
 - MVC 구조에 대한 이해를 바탕으로 설계가 필요하다
 - MVC 를 명확하게 나누기가 어려운 부분이 있어서 구조 상으로 복잡해 지는 경우가 많다



MVC의 현재

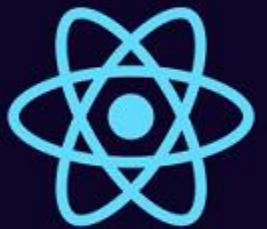
- 장점
 - 역할이 명확히 구분 되어 있어서 유지 보수가 쉽다
 - 기능 추가 및 확장이 쉽다
 - 서로간의 협업에 좋다
- 단점
 - MVC 구조에 대한 이해를 바탕으로 설계가 필요하다
 - MVC 를 명확하게 나누기가 어려운 부분이 있어서 구조 상으로 복잡해 지는 경우가 많다





A FEW
MOMENTS LATER

React JS



가즈아

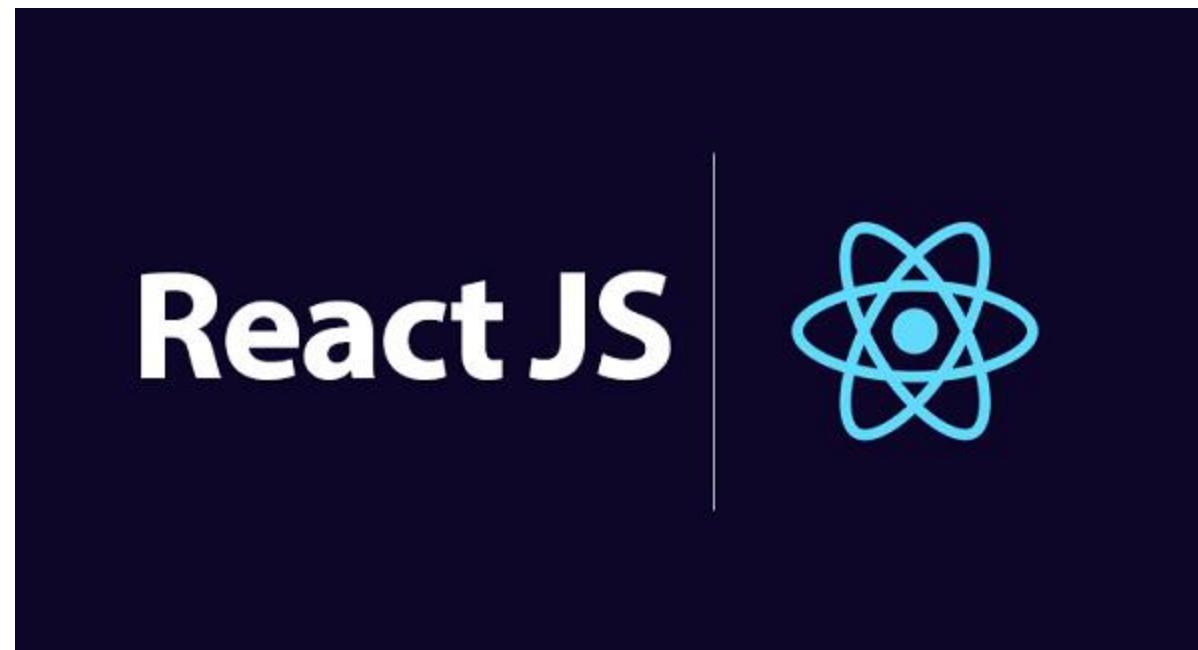




Why?



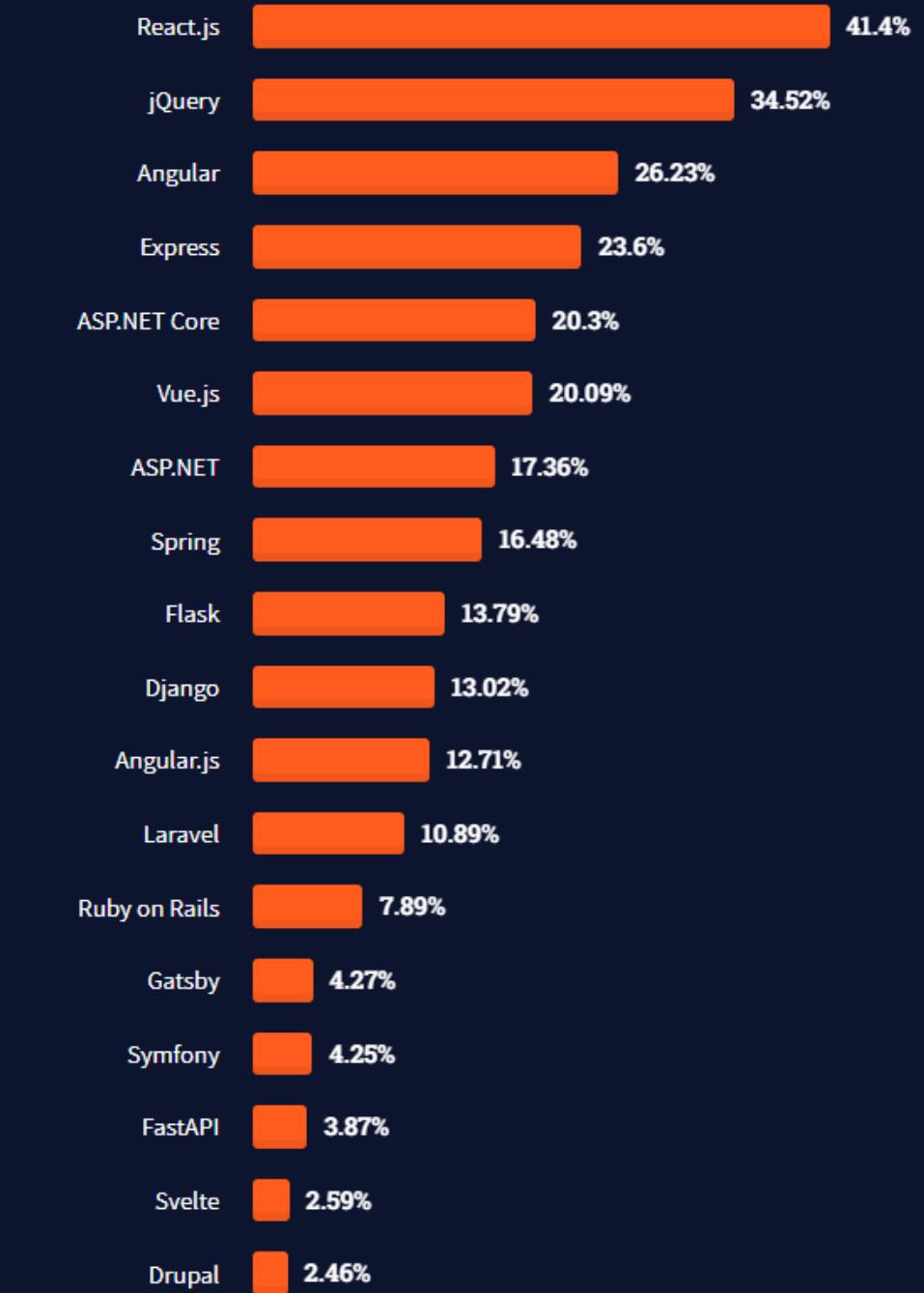
Why?





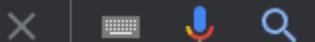
생태계







site: stackoverflow.com react



전체 뉴스 동영상 이미지 쇼핑

검색결과 약 34,900,000개 (0.48초)

<https://stackoverflow.com/questions/tagged/reactjs>

Newest 'reactjs' Questions - Stack Overflow

React is a JavaScript library for building user interfaces. ... I'm getting an error while trying to run a landing **page** component.

React · Bountied 19 · Page 4 · Active

site: stackoverflow.com vue

전체 뉴스 이미지 동영상 쇼핑 더보기 도구

검색결과 약 1,930,000개 (0.40초)

site stackoverflow.com angular

전체 뉴스 동영상 이미지 쇼핑 더보기 도구

검색결과 약 27,700,000개 (0.48초)

site stackoverflow.com laravel

전체 동영상 뉴스 이미지 지도 더보기 도구

검색결과 약 13,700,000개 (0.48초)







기립 박수로 설명할 수 있는 그분



실제로 확인

JOBKOREA x | 지역 선택

경력 ▾ 학력 ▾ 기업형태 ▾ 고용형태 ▾ 연봉 ▾ 조건추가 ▾

채용정보 기업정보

총 2,142건

JOBKOREA x | 지역 선택

경력 ▾ 학력 ▾ 기업형태 ▾ 고용형태 ▾ 연봉 ▾ 조건추가 ▾

채용정보 기업정보

총 4,346건

JOBKOREA x

경력 ▾ 학력 ▾ 기업형태 ▾ 고용형태 ▾ 연봉 ▾

채용정보 기업정보

총 412건

JOBKOREA x

경력 ▾ 학력 ▾ 기업형태 ▾ 고용형태 ▾ 연봉 ▾

채용정보 기업정보

총 1,006건

JOBKOREA x

경력 ▾ 학력 ▾ 기업형태 ▾ 고용형태 ▾ 연봉 ▾

채용정보 기업정보

총 1,797건





홈 > 확장 프로그램 > Wappalyzer - Technology profiler



Wappalyzer - Technology profiler

[Chrome에서 삭제](#) wappalyzer.com

★★★★★ 1,873 ⓘ | [개발자 도구](#) | 사용자 1,000,000+명

<https://chrome.google.com/webstore/detail/wappalyzer-technology-pro/gppongmhjkpfnbhagpmjfkannfbllamg>



lyzer - Technology profile x NAVER +

Diet NSS CUE ND GD GC GooPh GMail Daum 네이버 My N NBAM FAN YO

매일 쓰는 브라우저 보안이 걱정된다면, 안전하고 빠른 최신 브라우저 웨일로 업데이트 하세요. 다운로드

NAVER

메일 카페 블로그 지식iN 쇼핑 LIVE Pay TV 사전 뉴스 증권 부동산 지도 VIBE 도서 웹툰 더보기 ▾

푸름웰니스 한예슬도, 마스크도, 가을스타일도 스타일핏클래식 9,800원 무 / 료 / 배 / 송 네이버를 아이디 Something wrong or missing?

연합뉴스 > 野 "외교참사 삼진 아웃"...대국민 사과·외교라인 교체... 뉴스홈 · 연예 · 스포츠 · 경제

뉴스스탠드 > 구독한 언론사 · 전체언론사

Wappalyzer 기술 MORE INFO Export

분석 Google Analytics

보안 HSTS

기타 Open Graph webpack

JavaScript 라이브러리 jQuery 1.12.4 core-js 3.21.1

Generate sales leads
Find new prospects by the technologies they use. Reach out to customers of Shopify, Magento, Salesforce and others.



Wappalyzer - Technology profile x k 카카오 +

/page/

W-Mail Diet NSS CUE ND GD GC GooPh GMail Daum 네이버 My N NBAM FAN YOUTUBE

kakao 카카오 뉴스 기술과 서비스 약속과 책임

25 오늘의 카카오
9월 25일 일요일 소식입니다

보도자료 카카오, '카카오클라우드 스쿨' 2기 모집

보도자료 카카오엔터테인먼트,

Wappalyzer

기술 MORE INFO Export

분석 기타

- Naver Analytics Open Graph
- webpack
- Kakao 1.43.0

JavaScript 프레임워크

- Vue.js
- Nuxt.js

웹 서버

- Nuxt.js

보안

- HSTS

프로그래밍 언어

- Node.js

웹 프레임워크

- Nuxt.js

정적 사이트 생성기

- Nuxt.js



astcampus 기업 강의 × Wappalyzer - Technology profile × LINE | 라인은 언제나 사용자와 함께 × +

line.me/ko/ MAIL GitHub Netlify W-Mail Diet NSS CUE ND GD GC GooPh GMail Daum 네이버 My N NBAM FAN YOUTUBE

NE

Life on LINE 커뮤니케이션 앱 서비스

Life on LINE

Wappalyzer 기술 MORE INFO Export

분석 CDN

- Google Analytics Amazon Cloudfront

JavaScript 프레임워크 태그 관리자

- React Google Tag Manager

Gatsby 정적 사이트 생성기

- Gatsby 2.32.8

보안

- HSTS

기타 JavaScript 라이브러리

- core-js 3.9.0

PaaS

- Open Graph
- webpack
- Amazon Web Services

The screenshot shows a browser window with several tabs open. The active tab is for the LINE website at line.me/ko/. The page features a large banner with the text 'Life on LINE' overlaid on a background image of a person holding a smartphone. Below the banner, there are navigation links for 'Life on LINE', '커뮤니케이션 앱', and '서비스'. To the right of the banner, a sidebar titled 'Wappalyzer' provides a technical analysis of the site's stack. The '기술' (Technologies) section lists various tools used in the site's construction, including Google Analytics, Amazon Cloudfront, React, Google Tag Manager, Gatsby (version 2.32.8), HSTS, core-js (version 3.9.0), Open Graph, webpack, and Amazon Web Services. The '분석' (Analysis) section also lists Google Analytics and Amazon Cloudfront.



Wappalyzer - Technology profile x 쿠팡! +

src=1042016&spec=10304903&addtag=900&ctag=HOME&lptag=쿠팡&itime=20220925194850&pageType=HOME&pageValue=HOME&wPcid=16483577321785948861150&wRef=www.google.com&wTime=20220... ↗ ☆ 14

W-Mail Diet NSS CUE ND GD GC GooPh GMail Daum 네이버 My N NBAM FAN YO...

오늘 밤 12시까지 주문해도
로켓배송은 내일 도착!

카테고리

coupan

전체 찾고 싶은 상품을 검색해보세요!

로켓배송 로켓프레시 biz 쿠팡비즈 로켓직구 글드박스 와우회원할인 이벤트/쿠폰

대상 인기상품 모음전

청정원부터 그레이트코屋里

Lightly 곤약밥 힌미 80% 115g 카놀라유 청정원 낙곱새전골 HOME:ings

Facebook Pixel 2.9.84 Akamai

JavaScript 프레임워크 광고

RequireJS 2.1.14 Criteo

Handlebars 1.3.0

JavaScript 라이브러리

core-js 3.12.1

Akamai Bot Manager

HSTS

jQuery UI 1.11.4

jQuery 1.11.1

Open Graph

Nginx

리버스 프록시

Criteo



Wappalyzer - Technology profile | 베민 - Google 검색 | 대한민국 1등 배달앱, 배달의민족 | +

W-Mail Diet NSS CUE ND GD GC GooPh GMail Daum 네이버 My N NBAM FAN YOUTUBE

Wappalyzer 기술 MORE INFO Export

분석 프로그래밍 언어

Google Analytics Node.js

JavaScript 프레임워크 CDN

styled-components 5.3.0 Amazon S3

React Amazon Cloudfront

이슈 트래커 개발

Sentry styled-components 5.3.0

웹 프레임워크 JavaScript 라이브러리

Next.js 10.2.3 core-js 3.15.2

web-vitals

배달의민족

f blog

따뜻한 라떼 한 잔이 생각날 때



Wappalyzer - Technology profile |  베민 - Google 검색 |  당근마켓 - Google 검색 |  당신 근처의 당근마켓

Y-Mail + Diet NSS CUE ND GD GC GooPh GMail Daum HIOLHI My N M NRAM FAN YO

[MORE INFO](#) [EXPORT](#)

당근마켓

중고거래

알바

부동산 직거래

물품이나

분석

N Naver Analytics

 [Google Analytics](#)

 Facebook Pixel 2.9.84

 AppsFlyer

프로그래밍 언어

 Ruby 50% sure

태그 관리자

 [Google Tag Manager](#)

JavaScript 라이브러리

 [jQuery](#) 1.12.4

 core-js 3.0.0

보안

 HSTS

웹 프레임워크

 [Ruby on Rails](#) 50% sure

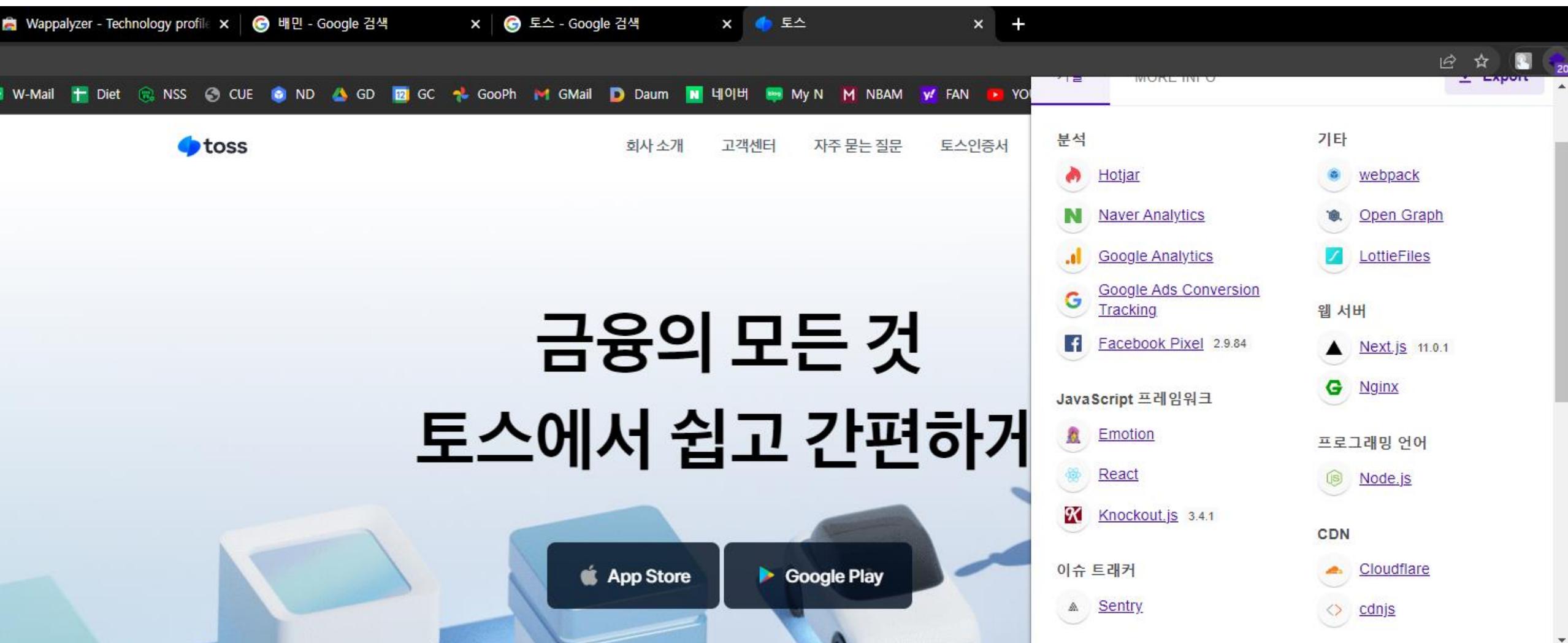
기타

 [Open Graph](#)

Something wrong or missing?

당신 근처의 당근마켓

중고 거래부터 동네 정보까지, 이웃과 함께해
가깝고 따뜻한 당신의 근처를 만들어요.





Virtual DOM

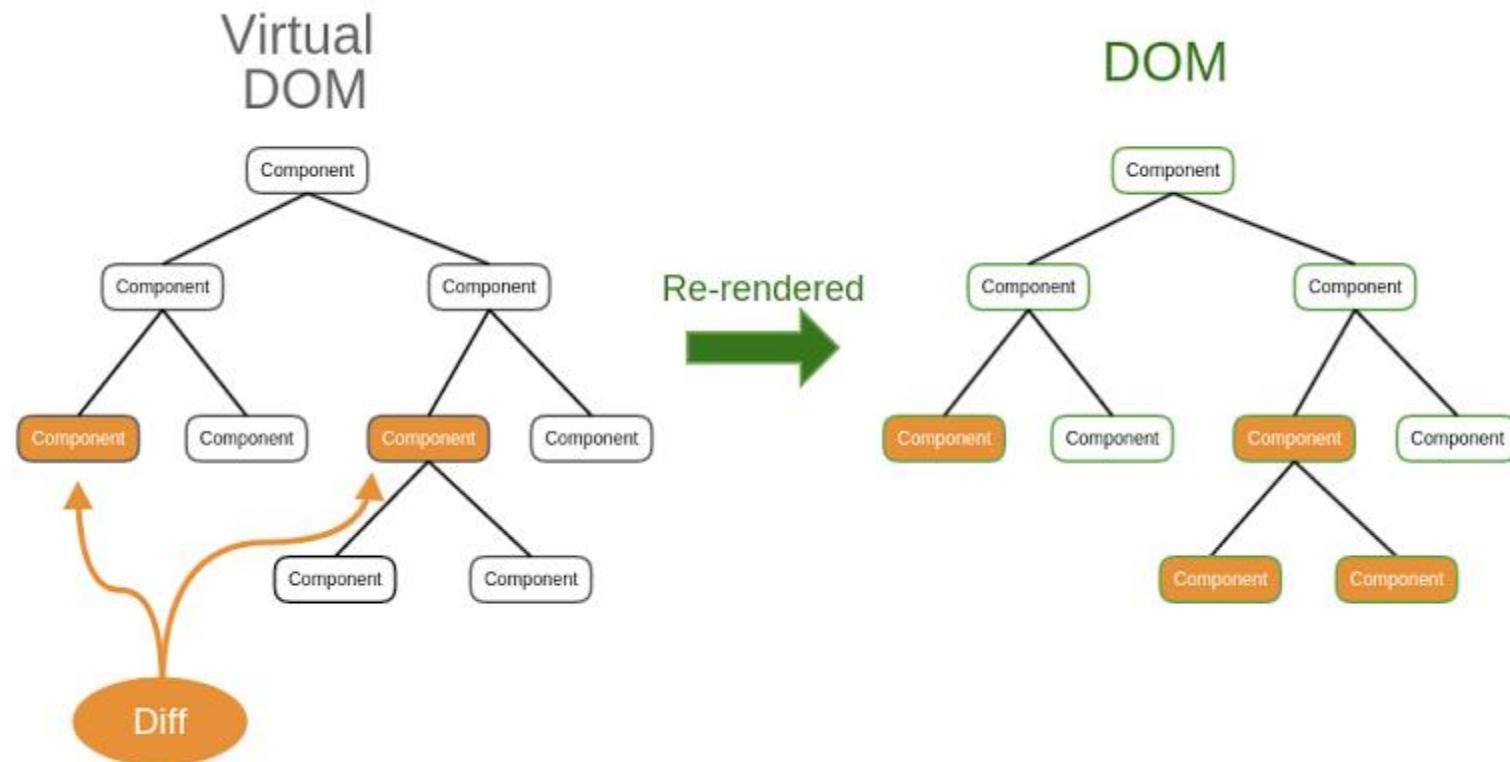














Virtual DOM

부드럽고 빠르다!

아키텍쳐에 대한 고민 ↓



거인의 어깨를 빌려라

성공 공식을 읽다

배연국 지음

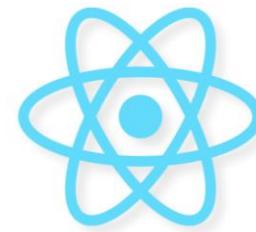
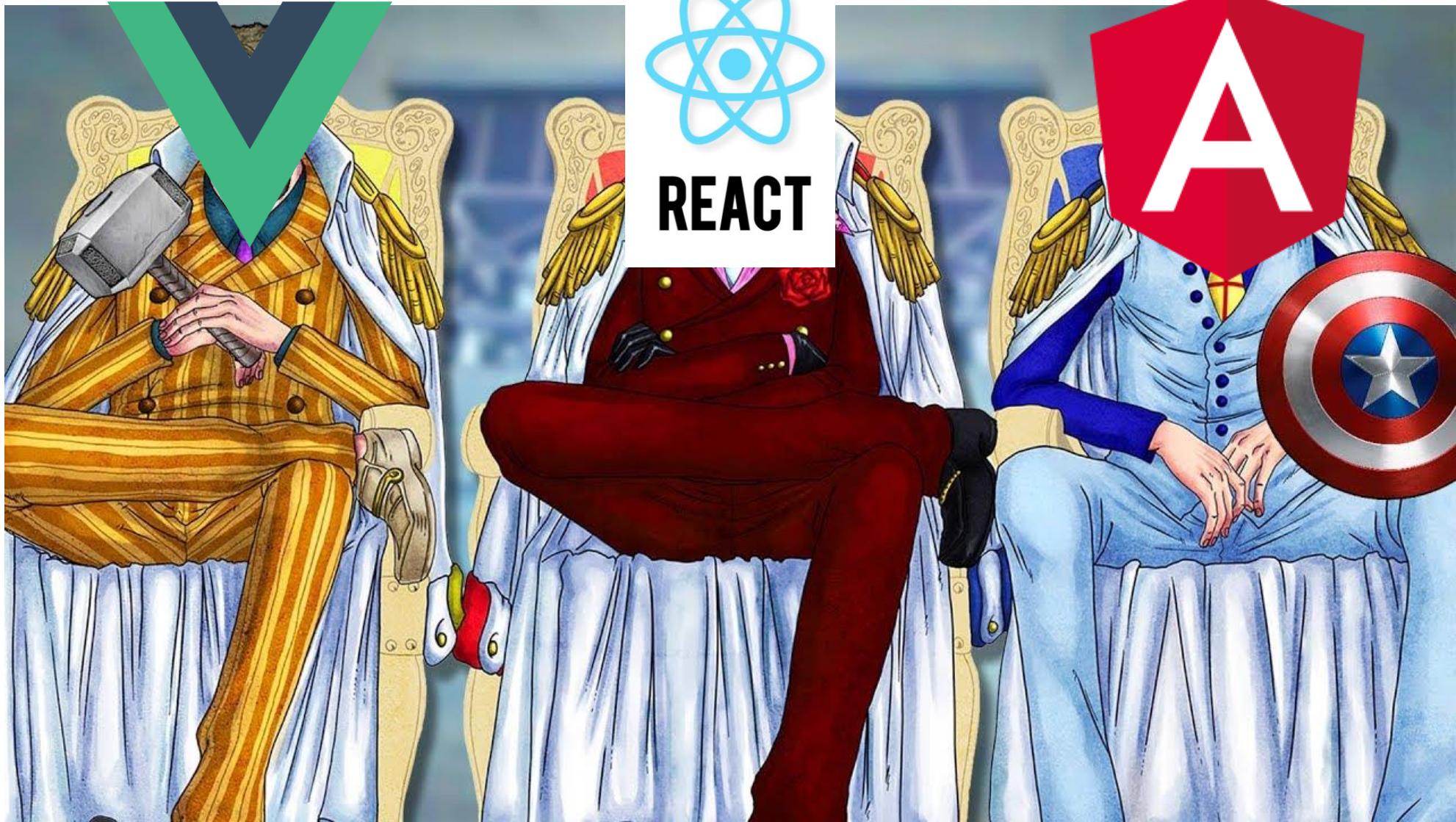
당신을 성공으로 안내할
영웅들의 감동 스토리!



지상사



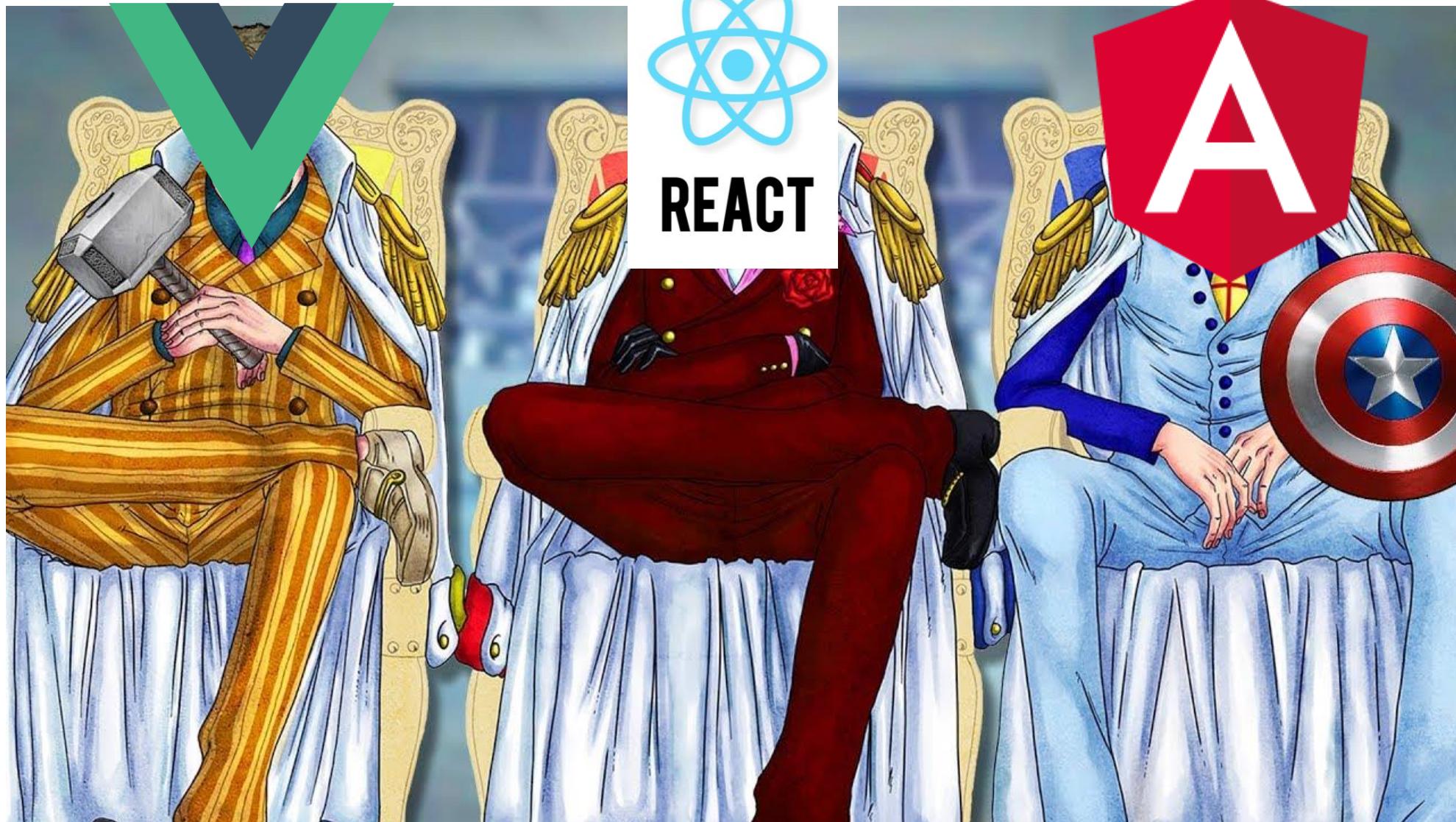
facebook

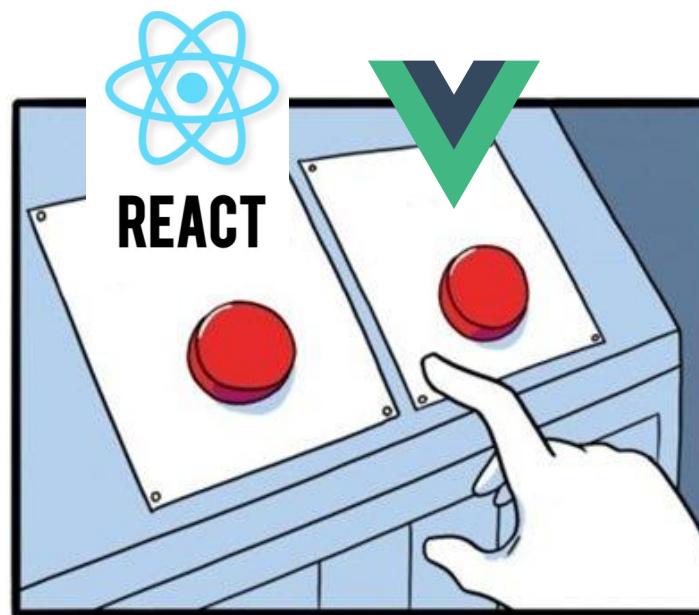




facebook

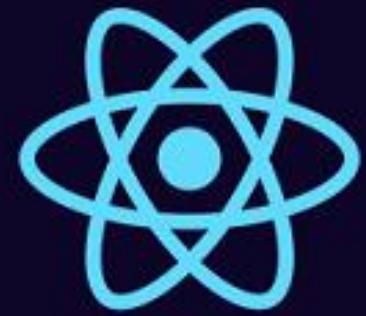
Google







React JS





일단

시작하기!



<https://ko.reactjs.org/>

A screenshot of the React.js Korean documentation homepage. The background is dark with a large, faint React logo watermark. The title "React" is in a large, light blue sans-serif font at the top center. Below it is a subtitle in Korean: "사용자 인터페이스를 만들기 위한 JavaScript 라이브러리". At the bottom left is a blue button with white text that says "시작하기". To its right is another button with white text that says "자습서 읽어보기 >".

React

사용자 인터페이스를 만들기 위한 JavaScript 라이브러리

시작하기

자습서 읽어보기 >



여러분이 사용하고 있는 코드 편집기를 사용하기 원한다면, [이 HTML 파일을 다운로드](#)하고 편집한 다음 브라우저의 로컬 파일 시스템에서 열 수도 있습니다. 런타임 코드 변환이 느리므로 간단한 데모에만 이 코드를 사용하는 것이 좋습니다.

웹사이트에 React를 추가하기

[1분 안에 HTML 페이지에 React를 추가할 수 있습니다.](#) 그리고 조금씩 React의 비중을 늘리거나 몇 개의 동적 위젯에 포함할 수 있습니다.

새 React 앱 만들기

React 프로젝트를 시작한다면 [스크립트 태그를 사용한 간단한 HTML 페이지를 만드는](#) 것이 최고의 방법일 것입니다. 설치하는 데 1분밖에 걸리지 않습니다!

그러나 애플리케이션이 커진다면 보다 통합된 설정을 고려하는 것이 좋습니다. 대규모 애플리케

설치 ^

시작하기

[웹 사이트에 React 추가하기](#)

[새로운 React 앱 만들기](#)

[CDN 링크](#)

[배포 채널](#)

주요 개념 ▼

고급 안내서 ▼

API 참고서 ▼

hook ▼





CDN 링크

React와 ReactDOM 모두 CDN을 통해 사용할 수 있습니다.

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```



리액트 시작하기!

- 먼저 리액트용 폴더를 하나 만들어 봅시다!
- Index.html 파일을 하나 만들고 index.js 도 만듭시다!
- Index.html 은 ! 를 사용해서 기본 코드 만들기 → CDN 의 링크 연결 하기!
- Index.js 파일도 불러오기! Defer 옵션 붙여서!
- Body 태그 자식 요소로 하나의 div 를 선언하고 id 는 app 으로 부여!

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>REACT</title>
  <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
  <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
  <script defer src="index.js"></script>
</head>

<body>
  <div id="app">
  </div>
</body>

</html>
```



Index.js

- 그럼 일단 리액트 코드를 한번 따라서 입력해 봅시다!

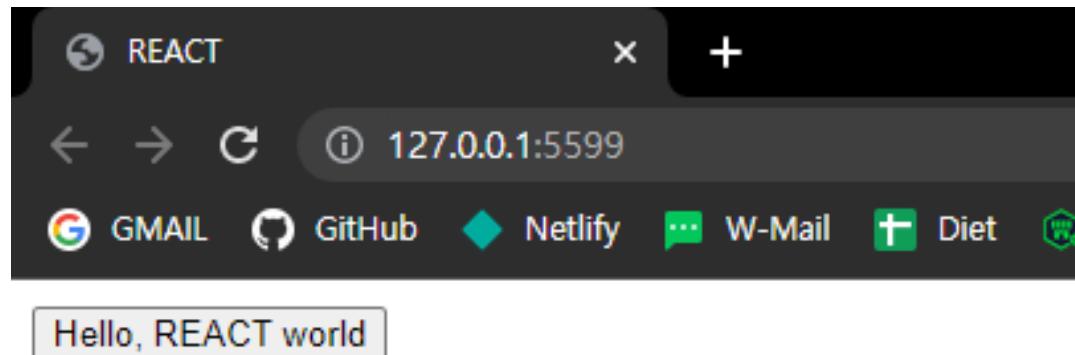
```
// import React from 'react';
// import ReactDOM from 'react-dom';
// 함수형 컴포넌트
function HelloWorldButton() {
  return React.createElement(
    "button",
    { onClick: () => {} },
    "Hello, REACT world"
  );
}

const e = React.createElement;
const domContainer = document.querySelector("#app");
const root = ReactDOM.createRoot(domContainer);
root.render(e(HelloWorldButton));
```



JS, DOM? REACT?

- 일단 보면, JS 에서 DOM 을 컨트롤 하는 코드와 비슷합니다!
- 기본 HTML 에서 부족 했던 점을 채우고자 JS 가 나왔고, JS 의 부족한 점을 채우고자 나온 것이 REACT 이기 때문입니다
- JS 에 몇몇 기능을 추가한 버전이라 생각하면 쉽습니다!





More REACT!?

- 그럼 나중에 배울 개념을 약간 더해서 코드를 추가해 봅시다!

```
function HelloWorldButton() {
  const [isClick, setClickState] = React.useState("It isn't clicked");
  console.log(isClick);
  return React.createElement(
    "button",
    { onClick: () => setClickState("It's clicked") },
    isClick
  );
}

const e = React.createElement;
const domContainer = document.querySelector("#app");
const root = ReactDOM.createRoot(domContainer);
root.render(e(HelloWorldButton));
```



More REACT!?

- 클릭이 되면 버튼의 문구가 변경 되도록 수정!

```
function HelloWorldButton() {
  const [isClick, setClickState] = React.useState(false);
  const text = isClick ? "It's clicked" : "Hello, React world";

  return React.createElement(
    "button",
    { onClick: () => setClickState(!isClick) },
    text
  );
}

const e = React.createElement;
const domContainer = document.querySelector("#app");
const root = ReactDOM.createRoot(domContainer);
root.render(e(HelloWorldButton));
```

REACT?!



- 뭔가 컴포넌트 별로 상태를 관리하려고 하는듯 하네요?
- 아마도 컴포넌트 별로 다시 그려주는 작업을 하다보니 이런게 있는게 아닌가 싶습니다!?
- 그럼 좀 더 배워보시죠!



JSX

(JavaScript XML)



자 이러한 상황을 가정해 봅시다!

- 방금 만든 코드를 약간 수정하여 아래와 같이 만들어 봅시다!
- 단순히 버튼의 문구를 바꾸는게 아니라, 버튼 태그 안에 div 요소도 하나 넣고, 그 div 안에 span 태그의 content로 text를 넣어 봅시다!



```
function HelloWorldButton() {
  const [isClick, setClickState] = React.useState(false);
  const text = isClick ? "It's clicked" : "Hello, React world";

  return React.createElement(
    "button",
    { onClick: () => setClickState(!isClick) },
    React.createElement("div", null, React.createElement("span", null, text))
  );
}

const e = React.createElement;
const domContainer = document.querySelector("#app");
const root = ReactDOM.createRoot(domContainer);
root.render(e(HelloWorldButton));
```



그런데 이걸 HTML로 쓰면!?

```
<button>
  <div>
    <span>
      Hello, REACT world!
    </span>
  </div>
</button>
```

```
return React.createElement(
  "button",
  { onClick: () => setClickState(!isClick) },
  React.createElement("div", null, React.createElement("span", null, text))
);
```





그런데, 짜잔!

JSX 가 이것을 해결해 줄 겁니다!



- 개발자들이 과거 DOM 방식으로 html 을 그리다보니 이게 바로바로 납득이 안갔습니다
- 그래서 만든 문법이! JSX(Javascript XML) 입니다!





```
function HelloWorldButton() {
  const [isClick, setClickState] = React.useState(false);
  const text = isClick ? "It's clicked" : "Hello, React world";

  return React.createElement(
    "button",
    { onClick: () => setClickState(!isClick) },
    React.createElement("div", null, React.createElement("span", null, text))
  );
}

const e = React.createElement;
const domContainer = document.querySelector("#app");
const root = ReactDOM.createRoot(domContainer);
root.render(e(HelloWorldButton));
```



```
// 함수형 컴포넌트
function HelloWorldButton() {
  const [isClick, setClickState] = React.useState(false);
  const text = isClick ? "It's clicked" : "Hello, React world";
  return (
    <button onClick={() => setClickState(!isClick)}>
      <div>
        <span>{text}</span>
      </div>
    </button>
  );
}

const domContainer = document.querySelector("#app");
const root = ReactDOM.createRoot(domContainer);
root.render(<HelloWorldButton />);
```





JSX

(JavaScript XML)



Javascript XML

- JS 에 XML 을 추가한 문법입니다
- 보통 리액트에서만 사용됩니다!
- Html 문서 구조를 JS 에서도 사용이 가능합니다! 따라서 JS 내부에서도 html 을 짜듯 코드 구성이 가능해 집니다!
- 이를 읽기 위해서는 **Babel** 이라는 **컴파일러(번역기)**가 필요합니다!



Babel



Babel

- <https://babeljs.io/>
- JS 의 컴파일러 입니다
- 예전에 ES6 가 나오고나서 몇몇 브라우저가 ES6를 지원하지 않아서 + ES6 문법과 ES5 문법의 충돌이 잦아서 ES6 문법을 ES5 문법으로 변환해 주던 기능을 하던 친구입니다!
- 그래서 예전 이름이 Six to Five 였었죠 😊

Babel



- 그런데 요즘은 ES6 는 브라우저 레벨에서 지원을 많이 하다보니, 다른 추가적인 언어들에 대한 컴파일러 역할을 많이 합니다!
- 그중 대표적인 것이 REACT 입니다!

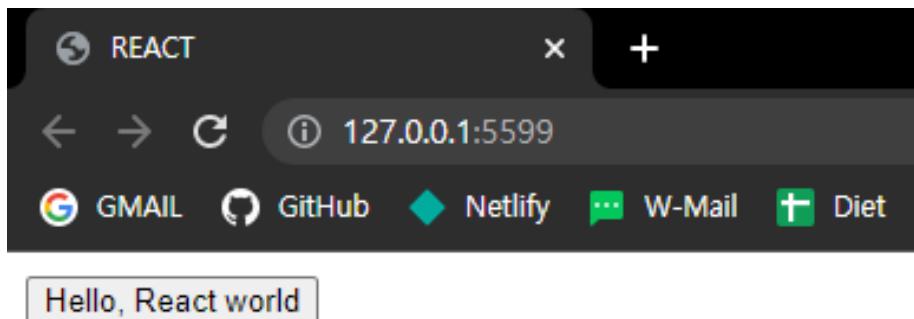


Babel 도 가져 옵시다!

- <https://babeljs.io/setup#installation>

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
```

- 그리고 JSX 문법이 적용 된, 페이지를 불러와 봅시다!





Webpack



Webpack

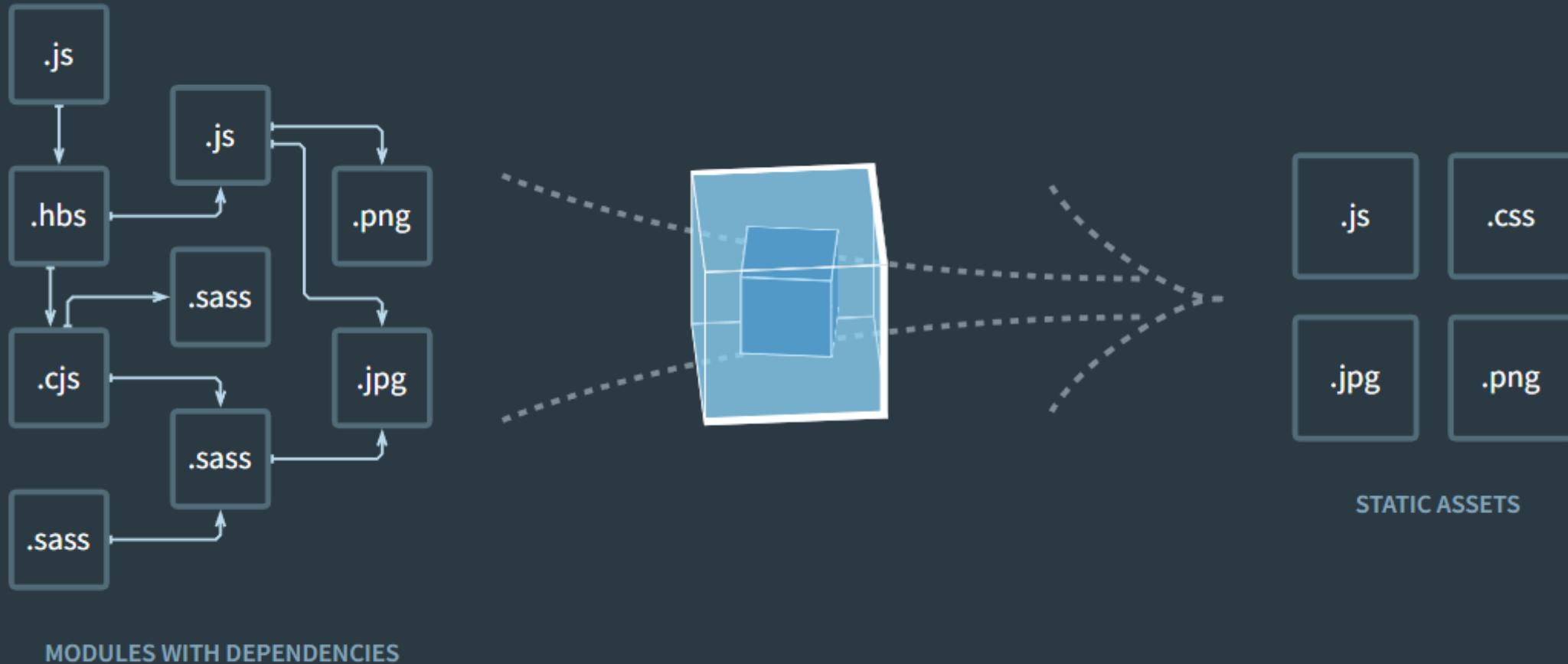
- <https://webpack.js.org/>
- JS 는 사실 별 기능이 없던 언어 입니다!
- 그래서 브랜던 아이크가 자바를 배껴서 단 10일만에 만들 수 있었습니다
- 그런데 웹이 커지고, 점점 더 요구하는게 많아지다보니 기능이 추가되기 시작 했습니다
- 그런데, 이건 웹이죠? 즉 통신을 기반으로 한 서비스 입니다!



Webpack

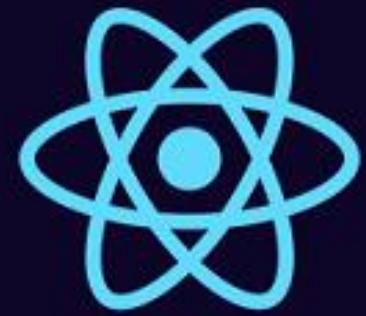
- 그 모든 기능을 다 빼려넣고 무언가를 만들자니 용량의 문제가 생깁니다!
- 그래서 필요한 기능을 필요한 순간에 모듈 형태로 불러와서 사용을 하고, 배포 할 때에는 필요 없는 기능은 다 빼고 빌드를 하는 방식이 사용 되었습니다
- 그런데 이게 모든 브라우저가 지원을 안해 줍니다!
- 그래서 등장 한 것이 Webpack 입니다!
- 의존성이 있는 모듈을 모아서 하나의 파일로 만들어 주는 역할을 하죠!

bundle your images





React JS





REACT?!

- 그럼 React 하려면 항상 바벨 추가하고? 빌드할 때에는 Webpack 설정 해 줘야 해요?
- 우리에겐 페북느님(요즘은 Meta) 계십니다!





Create-react-app



<https://ko.reactjs.org/docs/create-a-new-react-app.html>

Create React App

Create React App은 React 배우기에 간편한 환경입니다. 그리고 시작하기에 최고의 방법은 새로운 싱글 페이지 애플리케이션입니다.

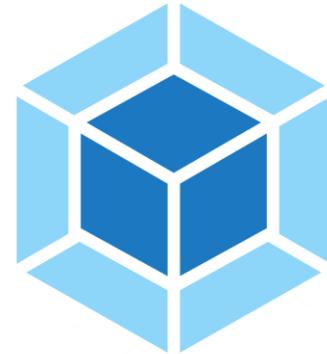
이것은 개발 환경을 설정하고, 최신 JavaScript를 사용하게 해주며, 좋은 개발 경험과 프로덕션 앱 최적화를 해줍니다. Node 14.0.0 혹은 상위 버전 및 npm 5.6 혹은 상위 버전이 필요합니다. 새로운 프로젝트를 만들기 위해 아래의 명령어를 실행합니다.

```
npx create-react-app my-app  
cd my-app  
npm start
```





BABEL



webpack



A FEW
MOMENTS LATER



Create-react-app



Create-react-app

- 이제 create-react-app 을 이용해서 react app 을 만들어 봅시다!
- React app 을 만들 폴더로 이동 합시다!
- Npx create-react-app 원하는 앱 이름
- 입력하신 앱 이름으로 폴더가 생기니 그 부분 유의 하세요!



Create-react-app

- 최초 설치면 시간이 걸릴 겁니다!

```
lhs@DESKTOP-86MUCGC MINGW64 /d/git
$ npx create-react-app my-app
Need to install the following packages:
  create-react-app
Ok to proceed? (y) y

Creating a new React app in D:\git\my-app.

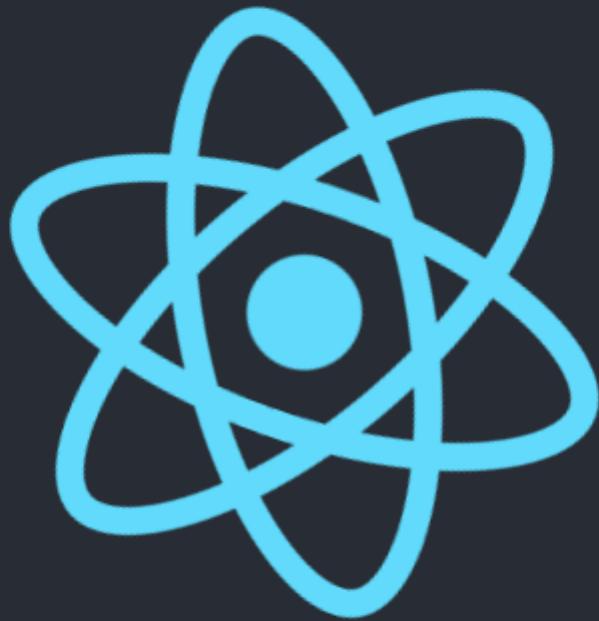
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

[#####.....] \ idealTree:@babel/core: sill fetch manifest klonan@^2.0.4
```

Create-react-app



- 설치가 완료 되면!?
- Npm start
- 3000번 포트에서 알아서 실행이 됩니다!



Edit `src/App.js` and save to reload.

[Learn React](#)



NPX?

NPM?



NPX?

- NPX 는 Node Package eXute 를 뜻 합니다!
- Node 실행을 위한 명령어이며 npm 과는 달리 최신 버전의 패키지를 임시로 설치해서 실행하는 용도로 사용됩니다
- 한번만 임시로 설치해서 해당 Node 를 실행시키고 사라집니다!
- 따라서 npm 에 대한 의존성이 없어서 다른 Node.js 버전으로 이동도 가능하고 좀 더 자유로운 코드 공유가 가능합니다
- 또한 한번만 쓰고 마는 코드 Create-react-app 같은 경우에 유용합니다

NPX?



- 또한 한번만 쓰고 마는 코드 Create-react-app 같은 경우에 유용합니다
- React 를 위해서 많은 패키지가 필요하지만 그럴때마다 Local 에 그런 패키지를 전부 설치하면 낭비가 심하겠죠?
- 그래서 한번만 설치하고 할 일이 끝나면 삭제되어 사라집니다!



프로젝트 폴더

살펴보기



Package.json

```
"dependencies": {  
    "@testing-library/jest-dom": "^5.16.5",  
    "@testing-library/react": "^13.4.0",  
    "@testing-library/user-event": "^13.5.0",  
    "react": "^18.2.0",  
    "react-dom": "^18.2.0",  
    "react-scripts": "5.0.1",  
    "web-vitals": "^2.1.4"  
},  
"scripts": {  
    "start": "react-scripts start",  
    "build": "react-scripts build",  
    "test": "react-scripts test",  
    "eject": "react-scripts eject"  
},
```



public



Index.html

```
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
</body>
```



src



Index.js

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

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



app.js

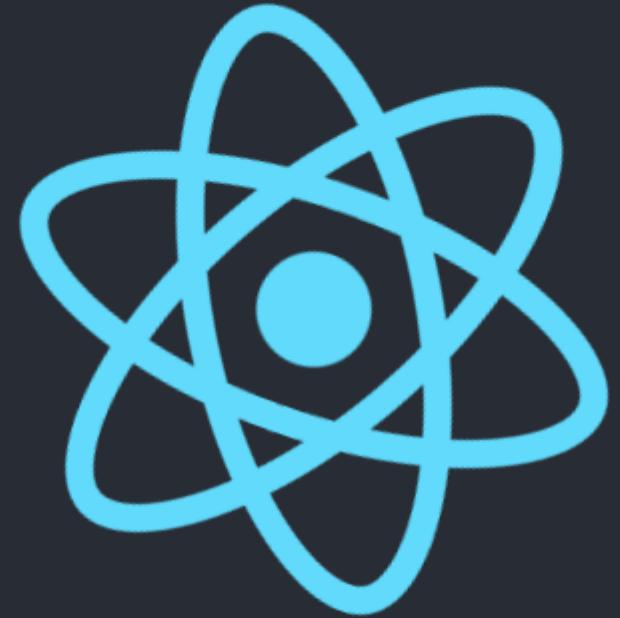
```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```



개발자 도구로 보기



Edit `src/App.js` and save to reload.

[Learn React](#)

Network						
Filter		Invert		Hide data URLs		
All	Fetch/XHR	JS	CSS	Img	Media	Font
<input type="checkbox"/> 3rd-party requests						
	1000 ms	2000 ms	3000 ms	4000 ms		
Name	Status	Type	Initiator	Size	Time	
localhost	200	document	Other	1.3 kB	2 ms	
bundle.js	200	script	(index)	360 kB	47 ms	
logo.6ce24c58023cc2f8fd88fe9...	200	svg+xml	react-dom.deve...	1.7 kB	3 ms	
manifest.json	200	manifest	Other	936 B	3 ms	
favicon.ico	200	x-icon	Other	3.9 kB	3 ms	
logo192.png	200	png	Other	5.7 kB	3 ms	
ws	101	websocket	WebSocketClien...	0 B	Pending	
js.js	200	script	content.js:32	1.4 kB	3 ms	
dom.js	200	script	content.js:32	2.0 kB	2 ms	
js.js	200	script	content.js:32	1.4 kB	2 ms	
dom.js	200	script	content.js:32	2.0 kB	1 ms	
js.js	200	script	content.js:32	1.4 kB	2 ms	



ws	101	websoc...	WebSocketCle...	0 B	Pendir
js.js	200	script	content.js:32	1.4 kB	3 m
dom.js	200	script	content.js:32	2.0 kB	2 m
js.js	200	script	content.js:32	1.4 kB	2 m
dom.js	200	script	content.js:32	2.0 kB	1 m
js.js	200	script	content.js:32	1.4 kB	2 m

12 requests | 382 kB transferred | 1.7 MB resources | Finish: 6.28 s



1.7M!?????

- 사진 하나 빙빙 도는 사이트가
- 1.7M !?????
- 너무 크다고 생각하지 않나요?





1.7M!?????

- 기본 프로젝트가 1.7m 씩이나 하는 이유는 React에서 필요한 필수 모듈 등이 전부다 들어와 있기 때문입니다!
- 즉, 아까 배웠던 Webpack이 압축하기 전 이라서 그렇습니다!!



빌드 해보기



빌드 하기

- 프로젝트 폴더에 가서 빌드를 해봅시다!
- Npm run build

```
lhs@DESKTOP-86MUCGC MINGW64 /d/git/my-app (main)
$ npm run build

> my-app@0.1.0 build
> react-scripts build

Creating an optimized production build...
Compiled successfully.
```



빌드 결과물 확인

- 만들어진 build 폴더에 가서 결과물 확인을 해봅시다!

```
build > index.html > ...
1  <!doctype html><html lang="en"><head><meta charset="utf-8"/><link rel="icon" href="/favicon.ico"/><meta name="viewport" content="width=device-width,initial-scale=1"/><me

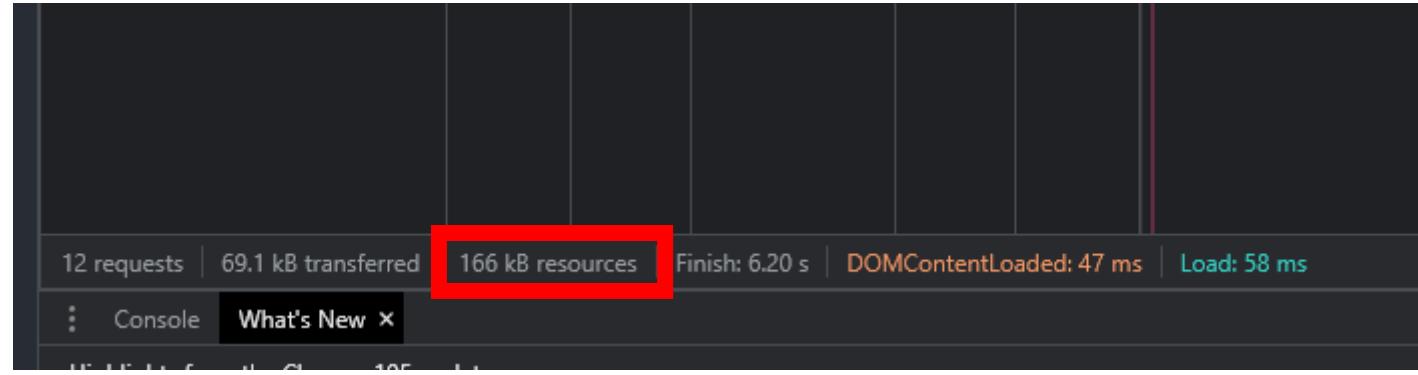
/*! For license information please see main.02159/66.js.LICENSE.txt */
!function(){"use strict";var e={463:function(e,n,t){var r=t(791),l=t(296);function a(e){for(var n="https://reactjs.org/docs/error-decoder.html?invariant="+e,t+="&args[]="+encodeURIComponent(arguments[t]));return"Minified React error #"+e+"; visit "+n+" for the full message or use the non-minified dev environment for helpful warnings."}var o=new Set,u={};function i(e,n){s(e,n),s(e+"Capture",n)}function s(e,n){for(u[e]=n,e=0;e<n.length;e++)o.add(n[e])}var c!=("undefined"==="t"?"undefined"===""typeof window.document||"undefined"===""typeof window.document.createElement),f=Object.prototype.hasOwnProperty,d=/^
[.:A-Z_a-z\u00C0-\u00D6\u00D8-\u00F6\u00F8-\u02FF\u0370-\u037D\u037F-\u1FFF\u200C-\u200D\u2070-\u218F\u2C00-\u2FEF\u3001-\uD7FF\uF900-\uFDCF\uFDF0-\uFFFD]
[.:A-Z_a-z\u00C0-\u00D6\u00D8-\u00F6\u00F8-\u02FF\u0370-\u037D\u037F-\u1FFF\u200C-\u200D\u2070-\u218F\u2C00-\u2FEF\u3001-\uD7FF\uF900-\uFDCF\uFDF0-\uFFFD\]-.
0-9\u00B7\u0300-\u036F\u203F-\u2040]*$/;p={};m={}:function h(e,n,t,r,l,a,o){this.acceptsBooleans=2===""n||3===""n||4===""n,this.attributeName=r,this.attributeNamespa
this.propertyName=e,this.type=n,this.sanitizeURL=a,this.removeEmptyString=o}var v={};"children dangerouslySetInnerHTML defaultValue defaultChecked innerHTML su
suppressHydrationWarning style".split(" ").forEach((function(e){v[e]=new h(e,0,!1,e,null,!1,!1)})),[["acceptCharset","accept-charset"],["className","class"],["
"http-equiv"]].forEach((function(e){var n=e[0];v[n]=new h(n,1,!1,e[1],null,!1,!1)})),[["contentEditable","draggable","spellCheck","value"].forEach((function(e){
toLowerCase(),null,!1,!1)}),["autoReverse","externalResourcesRequired","focusable","preserveAlpha"].forEach((function(e){v[e]=new h(e,2,!1,e,null,!1,!1)})),["a
autoFocus autoComplete controls default defer disabled disablePictureInPicture disableRemotePlayback formNoValidate hidden loop noModule noValidate open playsInlin
scoped seamless itemScope".split(" ").forEach((function(e){v[e]=new h(e,3,!1,e.toLowerCase(),null,!1,!1)})),["checked","multiple","muted","selected"].forEach((
!0,e,null,!1,!1)}),["capture","download"].forEach((function(e){v[e]=new h(e,4,!1,e,null,!1,!1)})),["cols","rows","size","span"].forEach((function(e){v[e]=new
["rowSpan","start"].forEach((function(e){v[e]=new h(e,5,!1,e.toLowerCase(),null,!1,!1)}));var g=/[\-\:](\w)/g;function y(e){return e[1].toUpperCase()}function
hasOwnProperty(n)?v[n]:null;(null!==l.type:r||(2<n.length)||"o"!=n[0]&&"0"!=n[0]||"n"!=n[1]&&"N"!=n[1])&&(function(e,n,t,r){if(null===""n||"undefined"
r){if(null!==t&&0===""t.type) return!1;switch(typeof n){case"function":case"symbol":return!0;case"boolean":return!r&&(null!==t?!.acceptsBooleans:"data-")!==(e=e.t
"aria-")!==e;default:return!1}}(e,n,t,r))return!0;if(r) return!1;if(null!==t) switch(t.type){case 3:return!n;case 4:return!1===""n;case 5:return isNaN(n);case 6:re
(n,t,l,r)&&(t=null),r||null===""l?function(e){return!!f.call(m,e)||!f.call(p,e)&&(d.test(e)?m[e]===""0:(p[e]===""0,!1))}(n)&&(null===""t?e.removeAttribute(n):e.setAttrib
});}});}};
```



빌드 결과물 확인

- 웹팩이 열심히 일을 해서 코드를 줄여 놨네요!
- 그럼 한번 실제로 빌드 결과물을 개발자 도구로 용량이 어찌 되는지 봅시다!
- 빌드 된 프로젝트 결과물의 경우 그냥 html 파일을 켠다고 읽을 수 없습니다!
- 서버 환경에서 켜주어야 하기 때문이죠! 그런데 Live server 도 안먹습니다!
- Npx serve -s build

```
Serving!
- Local:          http://localhost:3000
- On Your Network: http://192.168.0.8:3000
Copied local address to clipboard!
```





webpack



Facebook 신의 가호에서 벗어나기



Package.json 에 보면

```
"scripts": {  
  "start": "react-scripts start",  
  "build": "react-scripts build",  
  "test": "react-scripts test",  
  "eject": "react-scripts eject"  
},
```

- 다른건 대충 알겠는데… eject 는 뭐죠?
- 일단 한번 해보시죠 → npm run eject



Package.json 에 보면

```
lhs@DESKTOP-86MUCGC MINGW64 /d/git/my-app (main)
$ npm run eject

> my-app@0.1.0 eject
> react-scripts eject

NOTE: Create React App 2+ supports TypeScript, Sass, CSS Modules and more without ejecting: https://reactjs.org/blog/2018/10/01/create-react-app-v2.html

? Are you sure you want to eject? This action is permanent. » (y/N)
```

- 뭔가 무서운 일을 하려나 봐요!
- 이건 영원한 작업이라고 협박을 하네요!? 일단 GO!



```
"dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
},
"scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
},
```

```
1 {  
2   "name": "my-app",  
3   "version": "0.1.0",  
4   "private": true,  
5   "dependencies": {  
6     "@babel/core": "^7.16.0",  
7     "@pmmwh/react-refresh-webpack-plugin": "^0.5.3",  
8     "@svgr/webpack": "^5.5.0",  
9     "@testing-library/jest-dom": "^5.16.5",  
0     "@testing-library/react": "^13.4.0",  
1     "@testing-library/user-event": "^13.5.0",  
2     "babel-jest": "^27.4.2",  
3     "babel-loader": "^8.2.3",  
4     "babel-plugin-named-asset-import": "^0.3.8",  
5     "babel-preset-react-app": "^10.0.1",  
6     "bfj": "^7.0.2",  
7     "browserslist": "^4.18.1",  
8     "camelcase": "^6.2.1",  
9     "case-sensitive-paths-webpack-plugin": "^2.4.0",  
0     "css-loader": "^6.5.1",  
1     "css-minimizer-webpack-plugin": "^3.2.0",  
2     "dotenv": "^10.0.0",  
3     "dotenv-expand": "^5.1.0",  
4     "eslint": "^8.3.0",  
5     "eslint-config-react-app": "^7.0.1",  
6     "eslint-webpack-plugin": "^3.1.1",  
7     "file-loader": "^6.2.0",  
8     "fs-extra": "^10.0.0",  
9     "html-webpack-plugin": "^5.5.0",  
0     "identity-obj-proxy": "^3.0.0",  
1     "jest": "^27.4.3",  
2     "jest-resolve": "^27.4.2",  
3     "jest-watch-typeahead": "^1.0.0",  
4     "mini-css-extract-plugin": "^2.4.5",  
5     "postcss": "^8.4.4",  
6     "postcss-flexbugs-fixes": "^5.0.2",  
7     "postcss-loader": "^6.2.1",  
8     "postcss-normalize": "^10.0.1",  
9     "postcss-preset-env": "^7.0.1",  
0     "prompts": "^2.4.2",  
1     "react": "^18.2.0",  
2     "react-app-polyfill": "^3.0.0",  
3     "react-dev-utils": "^12.0.1",  
4     "react-dom": "^18.2.0",  
5     "react-refresh": "^0.11.0",  
6   }  
7 }
```

Eject



- 페북에서 기본으로 제공하는 설정을 포기하고 스스로 설정을 커스터마이징 할 때 사용하는 명령어입니다
- 따라서, 페북에서 세팅해 둔 모듈들이 전부 튀어 나와서 package.json에 등록이 된 것을 볼 수 있습니다!
- 요 부분은 앞서 말씀드린 것 처럼 좀 더 리액트가 익숙해 지시면 그 때 다시 말씀 드릴게요!



Component



Component

- 리액트의 핵심 개념입니다!
- 앞서 말씀드린 Virtual DOM 의 핵심이기도 합니다!
- 기존의 웹 서비스들은 웹페이지에서 정말 작은 부분이 업데이트 되어도 페이지의 전체를 리로딩 해줘야 했습니다!
- 하지만 React 는 컴포넌트 단위로 페이지 새로 고침이 가능하여 리소스 절약이 가능하고, 사용자에게 부드럽고 빠른 경험을 제공합니다!



Component

- 그리고 독립적으로 구성하여 재사용이 편리 합니다
- 데이터는 속성(props)으로 받고, 상태(state)에 따라 View를 변화 합니다
- 즉, 쇼핑몰에서 상품 목록 페이지를 하나하나 다 그려줄 필요 없이 하나의 컴포넌트만 만들고 변화되는 데이터만 props로 전달하여 재사용 하면 됩니다!



어디든지

언제든 일주일

게스트 추가



호스트 되기



기상천외한 숙소



국립공원



통나무집



풍차



상징적 도시



섬



해변 근처



초소형 주택



디자인



캠핑카



A자형 주택



호숫가



북극



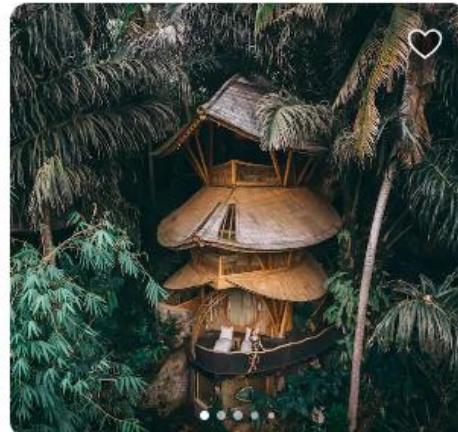
멋진 수영장



동글



서핑



Abiansemal, 인도네시아

5,275km

4월 13일~18일

₩483,768 /박

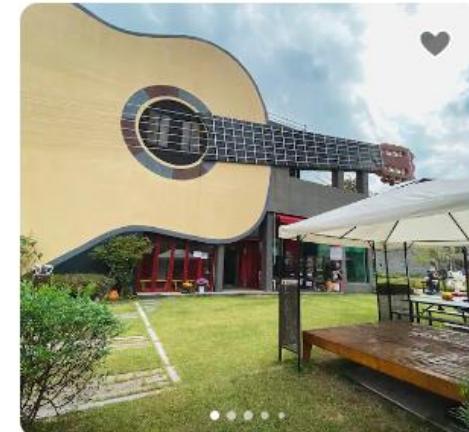


Abiansemal, 인도네시아

5,275km

12월 13일~19일

₩1,550,904 /박

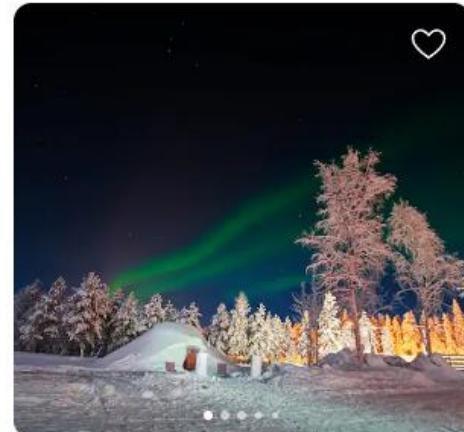


Sindun-myeon, Icheon-si, 한국

46km

10월 3일~8일

₩107,673 /박



Pelkosenniemi, 핀란드

6,604km

12월 28일 ~ 1월 2일

₩202,046 /박



Tambon Nong Kae, 태국

3,866km

10월 4일~9일

₩153,815 /박

★ 4.95



Component

- 그럼 일단 한번 써봅시다!
- Src 의 App.js 로 가봅시다!
- 중간의 코드는 날려 줍시다!

```
import './App.css';

function App() {
  return (
    <div className="App">
      </div>
    );
}

export default App;
```



새로운 Component 만들기

- Src 폴더 아래에 components 폴더 만들기!
- Components 폴더에 mainHeader.js 만들기!

```
function mainHeader() {  
  return (  
    <h1>Hello, Component world!</h1>  
  )  
}  
  
export default mainHeader;
```



새로운 Component 삽입!

- App.js 에 가서 mainHeader 를 삽입해 봅시다!
- Import 로 mainHeader 불러오기! 그리고 App 사이에 넣어주기!

```
import './App.css';
import mainHeader from
'./components/mainHeader';

function App() {
  return (
    <div className="App">
      <mainHeader />
    </div>
  );
}

export default App;
```



Error 발생!

```
✖ ▶ Warning: <mainHeader /> is using incorrect casing. Use react-dom.development.js:86
  PascalCase for React components, or lowercase for HTML elements.
    at mainHeader
    at div
    at App

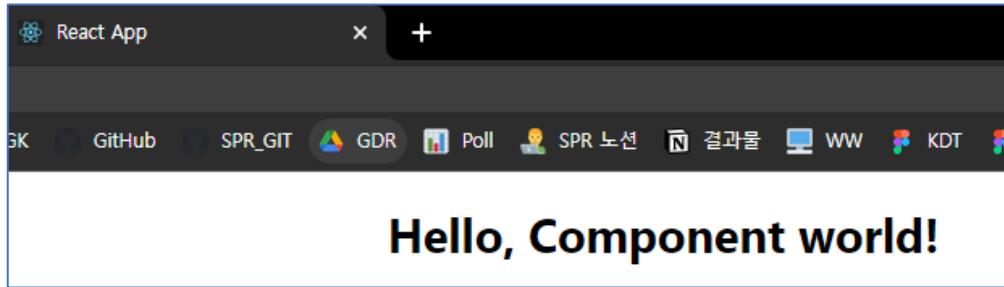
✖ ▶ Warning: The tag <mainHeader> is unrecognized in this react-dom.development.js:86
  browser. If you meant to render a React component, start its name with an uppercase
  letter.
    at mainHeader
    at div
    at App
```

- PascalCase 기억 나시나요?
- 컴포넌트는 JS에서 클래스 또는 생성자 함수와 비슷한 역할(정의 → 재사용)을 하다보니 PascalCase 를 사용합니다!



PascalCase 로 변경

- 파일명, 컴포넌트명을 전부 PasacalCase 로 변경하여 적용해 봅시다!



- 페이지를 새로 고침 않아도 적용이 되어 있죠?
- 이것이 바로 리액트의 장점인 Virtual DOM 의 효과입니다!
- 컴포넌트 레벨에서의 변화가 있으면 페이지가 자동으로 해당 부분만을 리랜더링 하는 것이죠!

실습, 컴포넌트 만들기



- 이미지 파일을 불러오는 ImgComponent 와 버튼 클릭을 하면 네이버 페이지로 이동하는 BtnToNaver 컴포넌트를 만들어 App 에 추가해 봅시다!
- 이미지는 파일을 리액트 폴더에 넣어서 사용합시다! (외부 링크 X)
 - 힌트는 최초의 App 컴포넌트 코드를 기억해 보세요!
- 버튼은 A 태그로 만들어도 무방합니다!



Component 의

종류

Component 의 종류!



- 컴포넌트는 크게 2가지로 나뉩니다.
- 클래스형 컴포넌트 vs 함수형 컴포넌트



클래스형 컴포넌트

- 예전에 최초로 사용 되었던 컴포넌트입니다!
- 컴포넌트 자체가 JS의 Class 와 유사하기 때문에 자연스럽게 사용 했었죠!
- 오래 된 만큼 state 와 라이프 사이클이라는 리액트의 장점을 사용 가능!
- 다만, 메모리 자원도 더 먹고 느리다는 단점이 있습니다
- 그리고 render 라는 함수를 사용해야만 그릴 수 있습니다!
- 최근에는 함수형 컴포넌트에게 완전히 밀리고 있습니다!



```
import React, {Component} from 'react';

class ClassComponent extends Component {
  render() {
    return(
      <h1>Class Component 입니다.</h1>
    );
  }
}

export default ClassComponent;
```



함수형 컴포넌트

- JS에서 익숙하게 사용하였던 함수를 컴포넌트화 시킨 것입니다!
- 아무래도 구조 자체가 클래스에 비해 단순하여 코드도 단순하고 빠르게 배울 수 있습니다
- 메모리도 덜 먹고 빠릅니다!(render 함수가 빠짐)
- 다만 예전에는 state 와 라이프사이클 기능 사용이 불가능하여 제한적으로 사용 → 최근에는 Hooks라는 기능의 도입으로 같은 역할 수행 가능!
- 최근에는 대부분 이거만 씁니다!



```
const FunctionComponent = () => {
  return <div>Functional Component 입니다
</div>
};

export default FunctionComponent;
```



함수형 컴포넌트를 클래스형 컴포넌트로!

- MainHeader.js 를 클래스형으로 변경해 봅시다!
- 먼저 클래스형 컴포넌트는 리액트의 컴포넌트 클래스를 상속 받아 사용해야 합니다!
- 함수 선언을 클래스로 변경하고 리턴은 render() {} 함수 내부에 넣어 줍시다!



```
import React from "react";

class MainHeader extends React.Component {
  render() {
    return (
      <h1>Hello, Component world!</h1>
    )
  }
}

export default MainHeader;
```

```
import React, { Component } from "react";

class MainHeader extends Component {
  render() {
    return (
      <h1>Hello, Component world!</h1>
    )
  }
}

export default MainHeader;
```

실습, 이전 실습을 클래스형 컴포넌트로 구현



- 이전 실습 내용을 클래스형 컴포넌트로 변경 하시면 됩니다! 😊



디버그의 편의성



기존 JS 에서는

- 에러 메시지를 보려면 Console 창을 띄워서 봐야 했지만
- React 는 치명적 버그일 경우 바로 화면에 띄워 버립니다!

```
Compiled with problems:          Hello, Compo
                               ERROR
[eslint]
src\components\BtnToNaver.js
Line 17:16:  'BtnToNaver' is not defined  no-undef
Search for the keywords to learn more about each error.
```



Strict!

- 즉, React 는 기존 JS의 문제점을 보완하고자 프레임워크 레벨에서 자체적으로 Strict 모드를 강제합니다!
- 실수하면 바로 어디서 실수 했는지, 그리고 상당히 디테일한 리포트를 제공하기 때문에 장점이 많습니다!





State

State



- 리액트에서 컴포넌트에 대한 상황을 처리하는 것을 의미 합니다
- 사용하는 이유는? → State 가 변경되면 해당 컴포넌트는 바로 다시 렌더링
이 되기 때문에 컴포넌트의 유동성 관리가 쉽습니다!
- 리액트에서 컴포넌트의 유동성을 담당하며, 컴포넌트 안에서만 관리가 되기
때문에 독립적입니다



클래스형 컴포넌트의

State



클래스형 컴포넌트의 State

- 클래스형 컴포넌트는 클래스가 기반이 되기 때문에 이전 클래스에서 배웠던 생성자(함수에서 선언한 변수 개념)에 state 값을 지정합니다
- Super 를 사용해서 초기값을 지정하고, this.state 라는 객체에 변경하고자 하는 값을 저장합니다.
- 그리고 this.setState 메소드를 이용하여 this.state 라는 객체에 저장 된 값을 변경 합니다
- 변경이 일어나면 컴포넌트는 알아서 다시 렌더링 되어 변경 값이 반영



```
import React, { Component } from "react";

class ClassState extends Component {
  constructor(props) {
    super(props);

    this.state = {
      message: ""
    };
  }

  render() {
    const { message } = this.state;
    const onClickEnter = () => { this.setState({ message: "안녕하세요!" }); };
    const onClickLeave = () => { this.setState({ message: "안녕히가세요!" }); };
    return (
      <div>
        <button onClick={onClickEnter}>입장</button>
        <button onClick={onClickLeave}>퇴장</button>
        <h1>{message}</h1>
      </div>
    );
  }
}

export default ClassState;
```

```
import React, { Component } from "react";

class ClassState extends Component {
  // 현재 버전
  state = {
    message: 0
  };

  render() {
    const { message } = this.state;
    const onClickEnter = () => { this.setState({ message: "안녕하세요!" }); };
    const onClickLeave = () => { this.setState({ message: "안녕히가세요!" }); };
    return (
      <div>
        <button onClick={onClickEnter}>입장</button>
        <button onClick={onClickLeave}>퇴장</button>
        <h1>{message}</h1>
      </div>
    );
  }
}

export default ClassState;
```





함수형 컴포넌트의

State



함수형 컴포넌트의 State

- 함수형 컴포넌트의 초기에는 리액트의 핵심 기능인 State 기능을 쓸 수 없습니다!
- 하지만 16.8 버전 이후 부터는 useState라는 메소드를 제공하여 함수형 컴포넌트에서도 State 사용이 가능해 졌습니다
- useState를 이용해서 state의 값을 초기화하고 state를 변경 할 수 있는 함수를 지정 → 지정한 함수를 이용해서 state를 변경 → 리렌더링



```
import React, { useState } from 'react';

function FuntionalState() {
    const [message, setMessage] = useState("");
    const onClickEnter = () => { setMessage("안녕하세요~"); };
    const onClickLeave = () => { setMessage("안녕히가세요."); };

    return (
        <div>
            <button onClick={onClickEnter}>입장</button>
            <button onClick={onClickLeave}>퇴장</button>
            <h1>{message}</h1>
        </div>
    );
}

export default FuntionalState;
```

실습, 버튼을 클릭하면 버튼 내용을 변경!



- '클릭해 주세요'라는 내용을 가진 버튼을 클릭하면 '다시 클릭해 주세요'라고 내용을 변경
- '다시 클릭해 주세요' 상태에서 다시 버튼을 클릭하면 '클릭해 주세요'로 변경이 되도록 리액트 앱을 만들어 주세요!

