

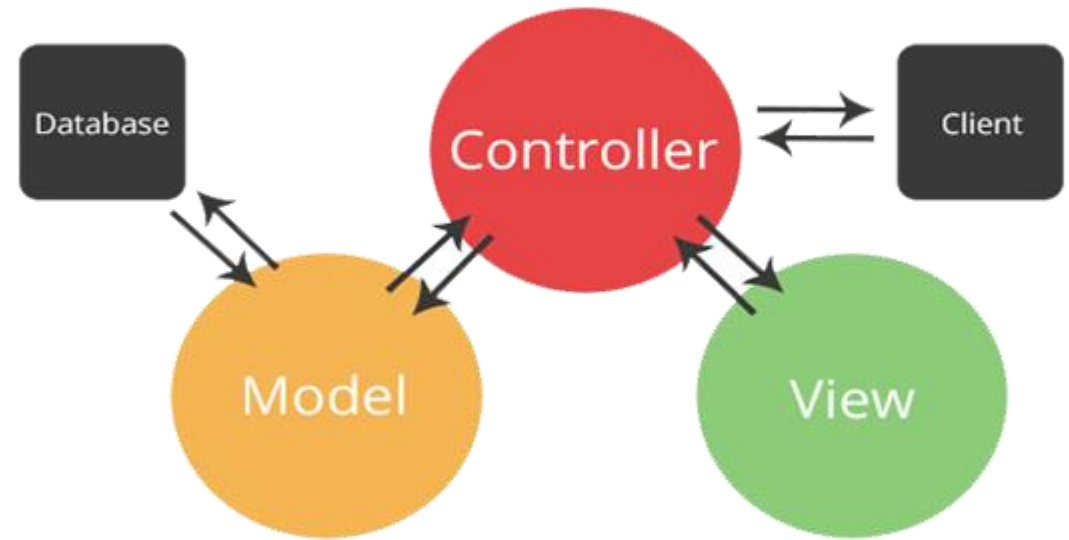
Hello,

KDT 웹 개발자 양성 프로젝트

5기!

with







Mongoose { 🍃 }



# Mongoose 설치

- Npm i mongoose -S

```
lhs@DESKTOP-86MUCGC MINGW64 /d/git/4th_backend (main)
$ npm i mongoose

added 8 packages, and audited 362 packages in 3s

70 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



```
const mongoose = require('mongoose');

const { MONGO_DB_URI } = process.env;

const connect = async () => {
  try {
    await mongoose.connect(MONGO_DB_URI, {
      dbName: 'kdt5',
      useNewUrlParser: true,
    });
    console.log('몽구스 접속 성공');
    mongoose.connection.on('error', (err) => {
      console.error('몽고 디비 연결 에러', err);
    });
    mongoose.connection.on('disconnected', () => {
      console.error('몽고 디비 연결이 끊어졌습니다. 연결을 재시도 합니다!');
      connect();
    });
  } catch (err) {
    console.error(err);
  }
};

connect();
module.exports = connect;
```



# User

# 스키마 생성



# 스키마 설정을 위한 Models 폴더 만들기

```
> controllers  
> models  
> node_modules  
> public  
> routes  
> views
```

- user 스키마를 작성을 위한 user.js 파일 생성

```
const mongoose = require('mongoose');
const { Schema } = mongoose;
const userSchema = new Schema(
  {
    id: {
      type: String,
      required: true,
      unique: true,
    },
    password: {
      type: String,
      required: true,
    },
    createdAt: {
      type: Date,
      default: Date.now,
    },
  },
  {
    collection: 'mongoose-user',
  },
);
module.exports = mongoose.model('User', userSchema);
```







# 컨트롤러 코드 수정



# 몽고 디비 접속 모듈 및 user 스키마 импорт

```
require('./mongooseConnect');  
const User = require('../models/user');
```

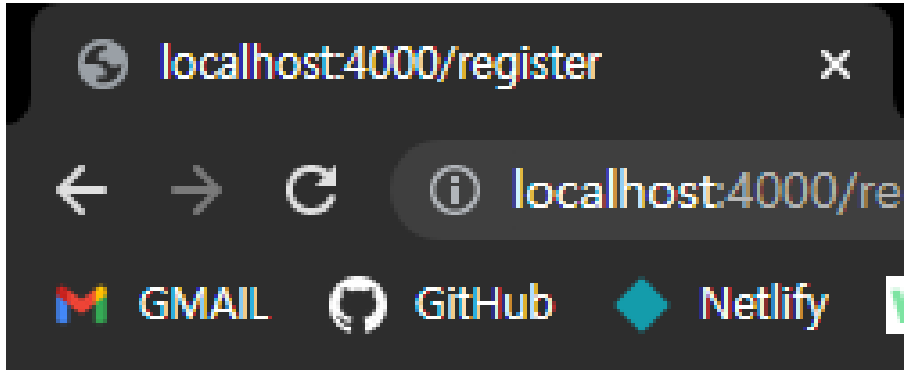
- 클라이언트는 한 번만 접속해도 사용이 가능하니 바로 접속을 시킵시다!

```
[nodemon] starting node app.js  
서버는 4000번에서 실행 중입니다!  
몽고 디비 연결 성공  
□
```



# 새로운 몽구스 컨트롤러 코드

```
const registerUser = async (req, res) => {  
  try {  
    const duplicatedUser = await User.findOne({ id: req.body.id });  
    if (duplicatedUser) return res.status(400).send(REGISTER_DUPLICATED_MSG);  
  
    await User.create(req.body);  
    res.status(200).send(REGISTER_SUCCESS_MSG);  
  } catch (err) {  
    console.error(err);  
    res.status(500).send(REGISTER_UNEXPECTED_MSG);  
  }  
};
```



회원 가입 성공!  
[로그인으로 이동](#)





# 필수 값을 다르게 전달 해보기

```
const registerUser = async (req, res) => {  
  try {  
    const duplicatedUser = await User.findOne({ id: req.body.id });  
    if (duplicatedUser) return res.status(400).send(REGISTER_DUPLICATED_MSG);  
  
    await User.create({ id: req.body.id, password: '' });  
    res.status(200).send(REGISTER_SUCCESS_MSG);  
  } catch (err) {  
    console.error(err);  
    res.status(500).send(REGISTER_UNEXPECTED_MSG);  
  }  
};
```

- 기존 몽고 디비였으면 뭐 그냥 id 라는 키로 데이터를 입력 했겠죠?



서버는 4000번 포트에서 실행 중입니다!

몽구스 접속 성공!

Error: User validation failed: id: Path `id` is required.

at ValidationError.inspect (D:\git\express-board\node\_modules\mongoose\lib\error\validation.js:50:26)

at formatValue (node:internal/util/inspect:782:19)

at inspect (node:internal/util/inspect:347:10)

at formatWithOptionsInternal (node:internal/util/inspect:2167:40)

at formatWithOptions (node:internal/util/inspect:2029:10)

at console.value (node:internal/console/constructor:332:14)

at console.warn (node:internal/console/constructor:365:61)

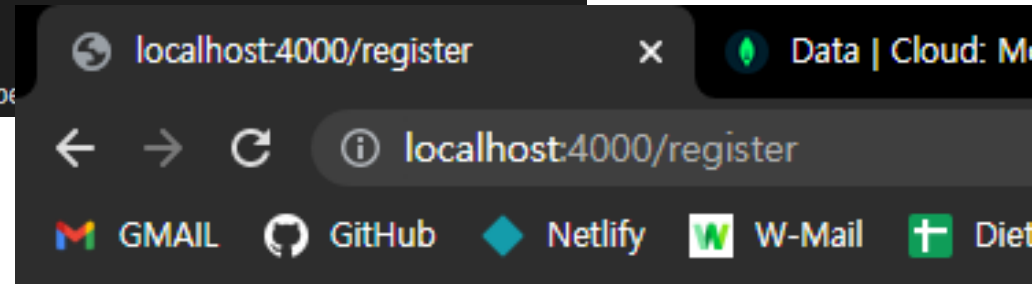
at registerUser (D:\git\express-board\controllers\userController.js:28:13)

at processTicksAndRejections (node:internal/process/task\_queues:96:5) {

errors: {

id: ValidatorError: Path `id` is required.

at validate (D:\git\express-board\node\_modules\mongoose\lib\schematype



회원 가입 실패! 알 수 없는 문제 발생

[회원 가입으로 이동](#)

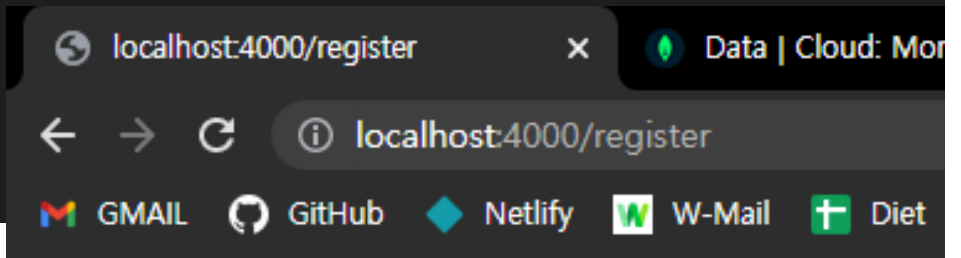


# 컨트롤러 코드에서 중복 체크 제거

```
const registerUser = async (req, res) => {  
  try {  
    // const duplicatedUser = await User.findOne({ id: req.body.id });  
    // if (duplicatedUser) return res.status(400).send(REGISTER_DUPLICATED_MSG);  
  
    await User.create(req.body);  
    res.status(200).send(REGISTER_SUCCESS_MSG);  
  } catch (err) {  
    console.error(err);  
    res.status(500).send(REGISTER_UNEXPECTED_MSG);  
  }  
};
```



```
MongoServerError: E11000 duplicate key error collection: kdt4.mongoose-user index: id_1 dup key: { id: "11" }  
  at D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\operations\insert.js:53:33  
  at D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\connection_pool.js:308:25  
  at D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\sdam\server.js:213:17  
  at handleOperationResult (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\sdam\server.js:329:20)  
  at Connection.onMessage (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\connection.js:219:9)  
  at MessageStream.<anonymous> (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\connection.js:60:60)  
  at MessageStream.emit (node:events:526:28)  
  at processIncomingData (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\message_stream.js:132:20)  
  at MessageStream._write (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\message_stream.js:33:9)  
  at writeOrBuffer (node:internal/streams/writable:389:12) {  
  index: 0,  
  code: 11000,  
  keyPattern: { id: 1 },  
  keyValue: { id: '11' },  
  [Symbol(errorLabels)]: Set(0) {}  
}
```



회원 가입 실패! 알 수 없는 문제 발생  
[회원 가입으로 이동](#)





Multer 모듈로  
이미지 업로드!



# Multer 모듈 설치

- Npm i multer -S
- 파일을 간단하게 업로드 하게 해주는 multer 모듈을 설치해 봅시다!

```
tetz@DESKTOP-P7Q40LL MINGW64 ~/Desktop/KDT/__수업 자료
$ npm i multer
npm WARN config global `--global`, `--local` are deprecated
added 16 packages, and audited 414 packages in 1s

84 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



# db\_board\_write.ejs 파일 수정

- 이제 파일 업로드 부분이 필요하므로 해당 내용을 추가 합니다

```
<div class="form_img">  
  <h3>이미지 업로드</h3>  
  <input type="file" name="img" />  
</div>
```

- 파일을 업로드 할 때에는 form 데이터가 더 이상 단순 텍스트가이 아니므로 인코딩 타입을 multipart/form-data 을 속성에 추가해 줍니다

```
<form action="/dbBoard/write" method="POST" class="board_form" enctype="multipart/form-data">
```

- 이걸 안써주면 파일 데이터는 안 올라 갑니다 ☹️



```
const dir = './uploads';
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, dir);
  },
  filename: (req, file, cb) => {
    cb(null, file.fieldname + '_' + Date.now());
  },
});
const limits = {
  fileSize: 1024 * 1028 * 2,
};

const upload = multer({ storage, limits });

if (!fs.existsSync(dir)) fs.mkdirSync(dir);
```

Null 은 모듈을 정상적으로 불러왔는지 테스트하기 위한 인자입니다!

파일 업로드를 위한 multer 모듈을 Upload 라는 변수에 담아 줍니다

서버의 최상단 폴더에 uploads 폴더가 있는지 확인하고 없으면 만들어 줍니다!



# 글쓰기 라우터에 파일 업로드 코드 추가!

- Multer 모듈은 현재 upload 에 들어 있습니다!
- 해당 모듈을 글쓰기 라우터에 isLogin 함수를 넣어 주었던 것 처럼, 미들 웨어로 넣어주면 됩니다!

```
// 글 쓰기  
router.post('/write', isLogin, upload.single('img'), writeArticle);
```



```
const writeArticle = async (req, res) => {  
  try {  
    const client = await MongoClient.connect();  
    const board = client.db('kdt5').collection('board');  
  
    console.log(req.file);  
  
    const newArticle = {  
      USERID: req.session.userId,  
      TITLE: req.body.title,  
      CONTENT: req.body.content,  
      IMAGE: req.file ? req.file.filename : null,  
    };  
    await board.insertOne(newArticle);  
    res.redirect('/dbBoard');  
  } catch (err) {  
    console.error(err);  
    res.status(500).send(err.message + UNEXPECTED_MSG);  
  }  
};
```

파일이 업로드 되어서  
Req.file 값이 있을 때에만  
IMAGE 프로퍼티에게 파일의 이름  
을 넣어주기



# AWS

# 서버 세팅!

## EC2 대시보드

EC2 글로벌 보기

이벤트

태그

제한

### ▼ 인스턴스

인스턴스 New

인스턴스 유형

시작 템플릿

스팟 요청

Savings Plans

예약 인스턴스 New

전용 호스트

용량 예약

### ▼ 이미지

AMI New

AMI 카탈로그

### ▼ Elastic Block Store

볼륨 New

스냅샷 New

수명 주기 관리자 New

### ▼ 네트워크 및 보안

보안 그룹

탄력적 IP

배치 그룹

## 리소스

EC2 글로벌 보기



아시아 태평양 (서울) 리전에서 다음 Amazon EC2 리소스를 사용하고 있음:

인스턴스(실행 중)	0	로드 밸런서	0	배치 그룹	0
보안 그룹	1	볼륨	0	스냅샷	0
인스턴스	0	전용 호스트	0	키 페어	0
탄력적 IP	0				

**i** AWS Launch Wizard for SQL Server를 사용하여 AWS에서 Microsoft SQL Server Always On 가용성 그룹을 손쉽게 크기 조정, 구성 및 배포할 수 있습니다. 자세히 알아보기

## 인스턴스 시작

시작하려면 클라우드의 가상 서버인 Amazon EC2 인스턴스를 시작하십시오.

인스턴스 시작 ▲

서버 마이그레이션

인스턴스 시작

템플릿으로 인스턴스 시작

## 예약된 이벤트



아시아 태평양 (서울)

예약된 이벤트 없음

## 서버 마이그레이션

## 서비스 상태



AWS Health 대시보드

리전

아시아 태평양 (서울)

상태

이 서비스가 정상적으로 작동 중입니다.

## 영역

영역 이름

영역 ID

ap-northeast-2a

apne2-az1

ap-northeast-2b

apne2-az2

ap-northeast-2c

apne2-az3

ap-northeast-2d

apne2-az4







## 키 페어 생성



키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

아래에 키 페어의 이름을 입력합니다. 메시지가 표시되면 프라이빗 키를 사용자 컴퓨터의 안전하고 액세스 가능한 위치에 저장합니다. 나중에 인스턴스에 연결할 때 필요합니다. [자세히 알아보기](#)

키 페어 이름

키 페어 이름 입력

이름은 최대 255개의 ASCII 문자를 포함할 수 있습니다. 선행 또는 후행 공백은 포함할 수 없습니다.

키 페어 유형

- ☒ RSA  
RSA 암호화된 프라이빗 및 퍼블릭 키 페어
- ☐ ED25519  
ED25519 암호화된 프라이빗 및 퍼블릭 키 페어(Windows 인스턴스에는 지원되지 않음)

프라이빗 키 파일 형식

- ☒ .pem  
OpenSSH와 함께 사용
- ☐ .ppk  
PuTTY와 함께 사용

취소

키 페어 생성



# 퍼블릭 IP 복사

## i-0b3c7dfe69bcf91e7 (board)에 대한 인스턴스 요약 정보

less than a minute 전에 업데이트됨

[🔄](#) [연결](#) [인스턴스 상태 ▼](#) [작업 ▼](#)

<b>인스턴스 ID</b> i-0b3c7dfe69bcf91e7 (board)	<b>퍼블릭 IPv4 주소</b> 43.200.173.233   <a href="#">개방 주소법</a>	<b>프라이빗 IPv4 주소</b> 172.31.46.71
<b>IPv6 주소</b> -	<b>인스턴스 상태</b> 🟢 실행 중	<b>퍼블릭 IPv4 DNS</b> ec2-43-200-173-233.ap-northeast-2.compute.amazonaws.com   <a href="#">개방 주소법</a>
<b>호스트 이름 유형</b> IP 이름: ip-172-31-46-71.ap-northeast-2.compute.internal	<b>프라이빗 IP DNS 이름(IPv4만 해당)</b> ip-172-31-46-71.ap-northeast-2.compute.internal	<b>탄력적 IP 주소</b> -
<b>프라이빗 리소스 DNS 이름 응답</b> IPv4(A)	<b>인스턴스 유형</b> t2.micro	<b>AWS Compute Optimizer 찾기</b> 📄 권장 사항을 위해 AWS Compute Optimizer에 옵트인합니다.   <a href="#">자세히 알아보기</a>
<b>자동 할당된 IP 주소</b> 43.200.173.233 [퍼블릭 IP]	<b>VPC ID</b> vpc-0c95088cff8de042c	<b>Auto Scaling 그룹 이름</b> -
<b>IAM 역할</b> -	<b>서브넷 ID</b> subnet-0ce7e077099429665	



# 키페어 권한 설정하기

- 먼저 터미널 또는 Git-bash 를 이용해서 키페어 파일을 저장한 폴더로 이동
- `chmod 400 키페어이름.pem` (키페어 권한 설정)

```
Ths@DESKTOP-86MUCGC MINGW64 ~/Desktop/업무 /KDT_4th/서버
$ chmod 400 tetz.pem :
```

- Chmod → Change Mode
  - 나 | 그룹 | 전체
  - Read : 4 / write : 2 / excute : 1 의 합으로 권한 표기
  - 400 → 나한테만 읽기 권한 / 754 → 나는 읽기쓰기실행, 그룹은 읽기쓰기, 전체는 읽기만



# SSH 를 이용 서버 접속

```
Ths@DESKTOP-86MUCGC MINGW64 ~/Desktop/업무 /KDT_4th/서버
$ ssh -i tetz.pem ec2-user@13.209.97.89
The authenticity of host '13.209.97.89 (13.209.97.89)' can't be established.
ED25519 key fingerprint is SHA256:FIAY0etf1r4t43ULPHrNiQUSUF0R7KzabefcbTpNPf8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.209.97.89' (ED25519) to the list of known hosts.

  _ | _ | _ )
  _ | (   /   Amazon Linux 2 AMI
 _ | \ _ | _ |

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 1 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-34-194 ~]$ D|
```

- `ssh -i 키페어이름.pem ec2-user@퍼블릭IP주소`
- 다음 질문에서 Yes 입력



# EC2에 Node.js 설치

- [https://docs.aws.amazon.com/ko\\_kr/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html](https://docs.aws.amazon.com/ko_kr/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html)
- Nvm 을 설치
  - `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash`
  - Nvm 활성화
  - `. ~/.nvm/nvm.sh`



# EC2에 깃 설치

- 리눅스 명령어로 git 설치
- `sudo yum install git`

```
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-31-46-71 ~]$ sudo yum install git  
Loaded plugins: extras, suggestions, langpacks, prior
```

- 다운 로드 질문이 나오면 y 누르고 엔터

```
Total download size: 9.3 M  
Installed size: 40 M  
Is this ok [y/d/N]: y  
Downloading packages:
```



# 서버에 git clone 진행

- git clone 복사한 주소

```
[ec2-user@ip-172-31-46-71 app]$ git clone https://github.com/xenosign/backend.git
Cloning into 'backend'...
remote: Enumerating objects: 5017, done.
remote: Counting objects: 100% (5017/5017), done.
remote: Compressing objects: 100% (3920/3920), done.
remote: Total 5017 (delta 896), reused 4994 (delta 873), pack-reused 0
Receiving objects: 100% (5017/5017), 22.58 MiB | 16.41 MiB/s, done.
Resolving deltas: 100% (896/896), done.
[ec2-user@ip-172-31-46-71 app]$
```

- Clone 된 폴더로 이동
- Cd 폴더 명

```
[ec2-user@ip-172-31-46-71 app]$ cd backend/
[ec2-user@ip-172-31-46-71 backend]$ ls
app.js  dist  package.json  package-lock.json
[ec2-user@ip-172-31-46-71 backend]$
```



# Github Repo 는 퍼블릭!

- 퍼블릭이 아니면 id 와 pw 를 입력하라고 나옵니다
- 여기서 pw 는 여러분의 github 비번이 아니라 access token 입니다
- access token 발급법
  - <https://curryyou.tistory.com/344>





# Node module 설치

- Npm install

```
[ec2-user@ip-172-31-46-71 backend]$ npm install
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: undefined,
npm WARN EBADENGINE   required: { node: '16.x', npm: '8.11.x' },
npm WARN EBADENGINE   current: { node: 'v16.17.0', npm: '8.15.0' }
npm WARN EBADENGINE }
( [REDACTED] ) : reify:has: http fetch GET 200 https://registr
```



# 실행 테스트

- Node app.js

```
[ec2-user@ip-172-31-34-194 4th_backend2]$ node app.js
서버는 undefined번에서 실행 중입니다!
node:events:491
  throw er; // Unhandled 'error' event
  ^

Error: connect ECONNREFUSED 127.0.0.1:3306
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1278:16)
    -----
    at Protocol._enqueue (/home/ec2-user/app/4th_backend2/node_modules/mysql/lib/protocol/Protocol.js:144:48)
    at Protocol.handshake (/home/ec2-user/app/4th_backend2/node_modules/mysql/lib/protocol/Protocol.js:51:23)
    at Connection.connect (/home/ec2-user/app/4th_backend2/node_modules/mysql/lib/connection.js:116:18)
    at Object.<anonymous> (/home/ec2-user/app/4th_backend2/controllers/dbConnect.js:2)
    at Module._compile (node:internal/modules/cjs/loader:1155:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1209:10)
    at Module.load (node:internal/modules/cjs/loader:1033:32)
    at Function.Module._load (node:internal/modules/cjs/loader:868:12)
```

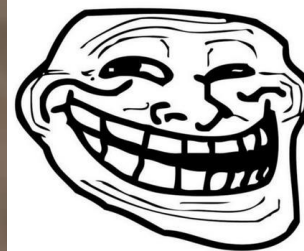


**지금 시작합니다**





TV TOKYO



**problem?**

해치웠나..!



# MongoDB

# Network Access

# MongoDB Atlas 는 IP를 가려서 받습니다!



DEPLOYMENT

Database

Data Lake PREVIEW

SERVICES

Triggers

Data API

Data Federation

Search

SECURITY

Database Access

**Network Access**

Advanced

New On Atlas

Goto

효석'S ORG - 2022-09-06 > PROJECT 0

Network Access

IP Access List

Peering

Private Endpoint

You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment
15.164.94.246/32	Amazon
3.34.177.57/32	AWS
0.0.0.0/0 (includes your current IP address)	
115.136.110.37/32 (includes your current IP address)	

# MongoDB Atlas 는 IP를 가려서 받습니다!



- 모든 IP에서 접근은 사실 좀 위험하긴 합니다! 하지만 편했죠!? 😊
- AWS 서버만 접속 가능하게 하려면 AWS 의 공인 IP만 등록해 주면 됩니다

## Add IP Access List Entry

Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more.](#)

Access List Entry:

15.164.94.246

Comment:

AWS



This entry is temporary and will be deleted in

6 hours ▼

Cancel

Confirm







# DOTENV

# 파일 옮기기!



# 파일 질라 설치!

- .env 파일은 직접 업로드를 해줘야 하므로 파일 질라 설치
- <https://filezilla-project.org/>

## Overview

Welcome to the homepage of FileZilla®, the free FTP solution. The *FileZilla Client* not Public License.

We are also offering *FileZilla Pro*, with additional protocol support for WebDAV, Amazon

Last but not least, *FileZilla Server* is a free open source FTP and FTPS Server.

Support is available through our [forums](#), the [wiki](#) and the [bug and feature request track](#)

In addition, you will find documentation on how to compile FileZilla and nightly builds

## Quick download links

**Download  
FileZilla Client**

All platforms



**Download  
FileZilla Server**

All platforms





## Download FileZilla Client for Windows (64bit x86)

The latest stable version of FileZilla Client is 3.62.2

Please select the file appropriate for your platform below.

🔹 Windows (64bit x86) 





**Download  
FileZilla Client**



This installer may include bundled offers. Check below for more options.

The 64bit versions of Windows 8.1, 10 and 11 are supported.

🔹 **More download options**

Other platforms:    


Not what you are looking for?

[➔ Show additional download options](#)

## Download FileZilla Client for macOS

The latest stable version of FileZilla Client is 3.62.2

Please select the file appropriate for your platform below.





🔹 macOS 

**Download  
FileZilla Client**



Requires macOS 10.13.2 or newer

🔹 **More download options**

Other platforms:    

Not what you are looking for?

[➔ Show additional download options](#)



FileZilla

파일(F) 편집(E) 보기(V) 전송(T) 서버(S) 북마크(B) 도움말(H)

호스트: 사용자명(U): 비밀번호(W): 포트(P): 빠른 연결(Q)

로컬 사이트: C:\Users\lhs\ Users

- Users
  - All Users
  - Default
  - Default User
  - lhs
  - Public
- Windows
- XboxGames
- D: (새 볼륨)
- E: (Sony HDD)
- F: (LG External HDD)

리모트 사이트:

파일명	크기	파일 유형	최종 수정
..			
.config		파일 폴더	2022-09-09 오후 ...
.docker		파일 폴더	2022-03-25 오후 ...
.ssh		파일 폴더	2022-12-03 오후 ...
.vscode		파일 폴더	2022-03-24 오전 ...
3D Objects		파일 폴더	2022-03-23 오후 ...
AppData		파일 폴더	2022-03-23 오후 ...
Application Data		파일 폴더	2022-11-15 오후 ...
Contacts		파일 폴더	2022-03-23 오후 ...
Cookies		파일 폴더	2022-03-25 오전 ...
Desktop		파일 폴더	2022-11-30 오후 ...
Documents		파일 폴더	2022-11-26 오후 ...
Downloads		파일 폴더	2022-12-03 오후 ...
Favorites		파일 폴더	2022-03-23 오후 ...
Links		파일 폴더	2022-03-23 오후 ...

14 파일 및 29 디렉터리. 총 크기: 12,986,243 바이트

서버/로컬 파일    방향    리모트 파일    크기    우선 ...    상태

대기 파일    전송 실패    전송 성공

대기열: 비었음



# 사이트 관리자

항목 선택(S):

- 내 사이트
  - AWS-tetz
  - 새 사이트

새 사이트(N)

새 폴더(f)

새 북마크(M)

이름 바꾸기(R)

삭제(D)

복제(I)

일반 고급 전송 설정 문자셋

프로토콜(t): SFTP - SSH File Transfer Protocol

호스트(H): 13.209.97.89 포트(P):

로그온 유형(L): 키 파일

사용자(U): ec2-user

키 파일(K): C:\Users\Wls\Desktop\업무용 키 파일 찾기...  
찾아보기...

배경색(B): 없음

비고(M):

연결(C)

확인(O)

취소



AWS-tetz - sftp://ec2-user@13.209.97.89 - FileZilla

파일(F) 편집(E) 보기(V) 전송(T) 서버(S) 북마크(B) 도움말(H)

호스트(H):

사용자명(U):

비밀번호(W):

포트(P):

빠른 연결(Q)

상태: Connected to 13.209.97.89

상태: 디렉터리 목록 조회...

상태: Listing directory /home/ec2-user

상태: "/home/ec2-user" 디렉터리 목록 조회 성공

로컬 사이트: C:\Users\lhs\

Users

All Users

Default

Default User

lhs

Public

Windows

XboxGames

D: (새 볼륨)

E: (Sony HDD)

F: (LG External HDD)

리모트 사이트: /home/ec2-user

home

ec2-user

파일명	크기	파일 유형	최종 수정
..		파일 폴더	2022-09-09 오후 ...
.config		파일 폴더	2022-03-25 오후 ...
.docker		파일 폴더	2022-12-03 오후 ...
.ssh		파일 폴더	2022-03-24 오전 ...
.vscode		파일 폴더	2022-03-23 오후 ...
3D Objects		파일 폴더	2022-03-23 오후 ...
AppData		파일 폴더	2022-11-15 오후 ...
Application Data		파일 폴더	2022-03-23 오후 ...
Contacts		파일 폴더	2022-03-25 오전 ...
Cookies		파일 폴더	2022-11-30 오후 ...
Desktop		파일 폴더	2022-11-26 오후 ...
Documents		파일 폴더	2022-12-03 오후 ...
Downloads		파일 폴더	2022-03-23 오후 ...
Favorites		파일 폴더	2022-03-23 오후 ...
Links		파일 폴더	2022-03-23 오후 ...

14 파일 및 29 디렉터리. 총 크기: 12,986,243 바이트

파일명	크기	파일 유형	최종 수정	권한	소유자/그룹
..		파일 폴더	2022-12-03 ...	drwxrwxr-x	ec2-user ec...
.npm		파일 폴더	2022-12-03 ...	drwxrwxr-x	ec2-user ec...
.nvm		파일 폴더	2022-12-03 ...	drwx-----	ec2-user ec...
.ssh		파일 폴더	2022-12-03 ...	drwxrwxr-x	ec2-user ec...
app		파일 폴더	2020-07-15 ...	-rw-r--r--	ec2-user ec...
.bash_	18	Bash Logo...	2020-07-15 ...	-rw-r--r--	ec2-user ec...
.bash_p	193	Bash Profil...	2022-12-03 ...	-rw-r--r--	ec2-user ec...
.bashrc	428	Bash RC ...		-rw-r--r--	ec2-user ec...

3 파일 및 4 디렉터리. 총 크기: 639 바이트

서버/로컬 파일

방향

리모트 파일

크기

우선 ...

상태

대기 파일

전송 실패

전송 성공

대기열: 비었음



리모트 사이트: /home/ec2-user/app/4th\_backend2

File Explorer View:

- /? /
  - /? home
    - ec2-user
      - /? .npm
      - /? .nvm
      - /? .ssh
      - app
        - 4th\_backend2

File List Table:

파일명	크기	파일 유형	최종 수정	권
..				
.git		파일 폴더	2022-12-03 ...	dn
.vscode		파일 폴더	2022-12-03 ...	dn
controllers		파일 폴더	2022-12-03 ...	dn
node_modules		파일 폴더	2022-12-03 ...	dn
public		파일 폴더	2022-12-03 ...	dn
routes		파일 폴더	2022-12-03 ...	dn
views		파일 폴더	2022-12-03 ...	dn
.env	70	ENV 파일	2022-12-03 ...	-rv
.eslintrc.js	288	JavaScript ...	2022-12-03 ...	-rv
.gitattributes	66	텍스트 문서	2022-12-03 ...	-rv
.gitignore	18	텍스트 문서	2022-12-03 ...	-rv

A red arrow points to the .env file in the file list.



# MySQL

# 처리!





# 실행 테스트

- Node app.js

```
[ec2-user@ip-172-31-34-194 4th_backend2]$ node app.js
서버는 undefined번에서 실행 중입니다!
node:events:491
  throw er; // Unhandled 'error' event
  ^

Error: connect ECONNREFUSED 127.0.0.1:3306
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1278:16)
    -----
    at Protocol._enqueue (/home/ec2-user/app/4th_backend2/node_modules/mysql/lib/protocol/Protocol.js:144:48)
    at Protocol.handshake (/home/ec2-user/app/4th_backend2/node_modules/mysql/lib/protocol/Protocol.js:51:23)
    at Connection.connect (/home/ec2-user/app/4th_backend2/node_modules/mysql/lib/connection.js:116:18)
    at Object.<anonymous> (/home/ec2-user/app/4th_backend2/controllers/dbConnect.js:2)
    at Module._compile (node:internal/modules/cjs/loader:1155:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1209:10)
    at Module.load (node:internal/modules/cjs/loader:1033:32)
    at Function.Module._load (node:internal/modules/cjs/loader:868:12)
```

# 엇!?



- mysql 서버에 접속이 안된다고 뜹니다! 이걸 서버에 mysql 을 설치를 안해서 그렇습니다!
- 그럼 일단 mysql 부터 사용을 안하게 처리해 줍시다!



# 엇!?

- Mysql 코드를 전부 주석 처리 해주세요!
- 그리고 로컬에서 nodemon app.js 를 실행 시켜서 문제가 없는지 확인 합니다!

```
1 // const mysql = require('mysql');
2
3 // const connection = mysql.createConnection({
4 //   host: 'localhost',
5 //   user: 'root',
6 //   password: process.env.DB_PASSWORD,
7 //   port: '3306',
8 //   database: process.env.DB_DATABASE,
9 // });
10 // connection.connect();
11 // module.exports = connection;
12
```

```
$ nodemon app.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
서버는 4000번에서 실행 중입니다!
[]
```



# 엇!?

- 깃을 커밋하고 푸쉬해 줍니다!
- 그리고 다시 서버에서 git pull 실행!

```
[ec2-user@ip-172-31-34-194 4th_backend2]$ git pull --all
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), 497 bytes | 497.00 KiB/s, done.
From https://github.com/xenosign/4th_backend2
   3d26a1a..8f12d79  main       -> origin/main
Updating 3d26a1a..8f12d79
Fast-forward
 controllers/dbConnect.js | 22 ++++++++-----
 1 file changed, 11 insertions(+), 11 deletions(-)
[ec2-user@ip-172-31-34-194 4th_backend2]$ |
```



# 실행 테스트

- Node app.js

```
[ec2-user@ip-172-31-34-194 4th_backend2]$ node app.js  
서버는 4000번에서 실행 중입니다!
```



문지후로 #5353  
김경호

가즈아!!!!!!!



# PM2(Process Manager 2)

- Node.js 프로그램의 프로세스 관리자
- 여러분의 app.js 가 예상치 못하게 죽는다면? 서비스는 망하겠죠?
- PM2 는 여러분의 서비스가 예상치 못한 오류로 죽어도, 다시 살려 줍니다!
- 그럼 사용해 봅시다!
- PM2 설치 (글로벌로 설치하여 다른 곳에서도 사용이 가능하도록!)
  - `npm i pm2 -g`



# PM2(Process Manager 2)

- PM2로 프로세스 실행하기
  - `pm2 start app.js`
- PM2로 프로세스 중단하기
  - `pm2 stop app.js`

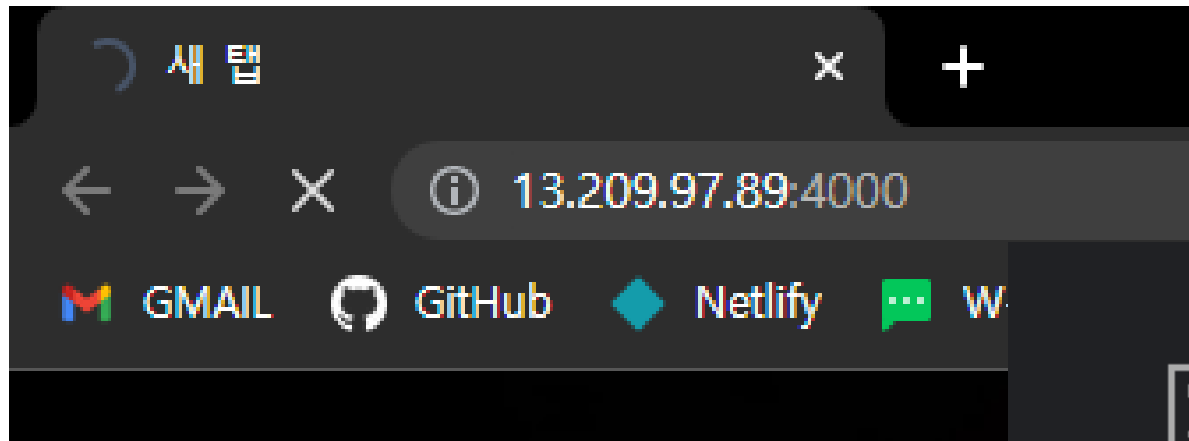
```
[PM2] Spawning PM2 daemon with pm2_home=/home/ec2-user/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ec2-user/app/backend/app.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	u	status	cpu	memory
0	app	fork	0	online	0%	32.2mb









## 사이트에 연결할 수 없음

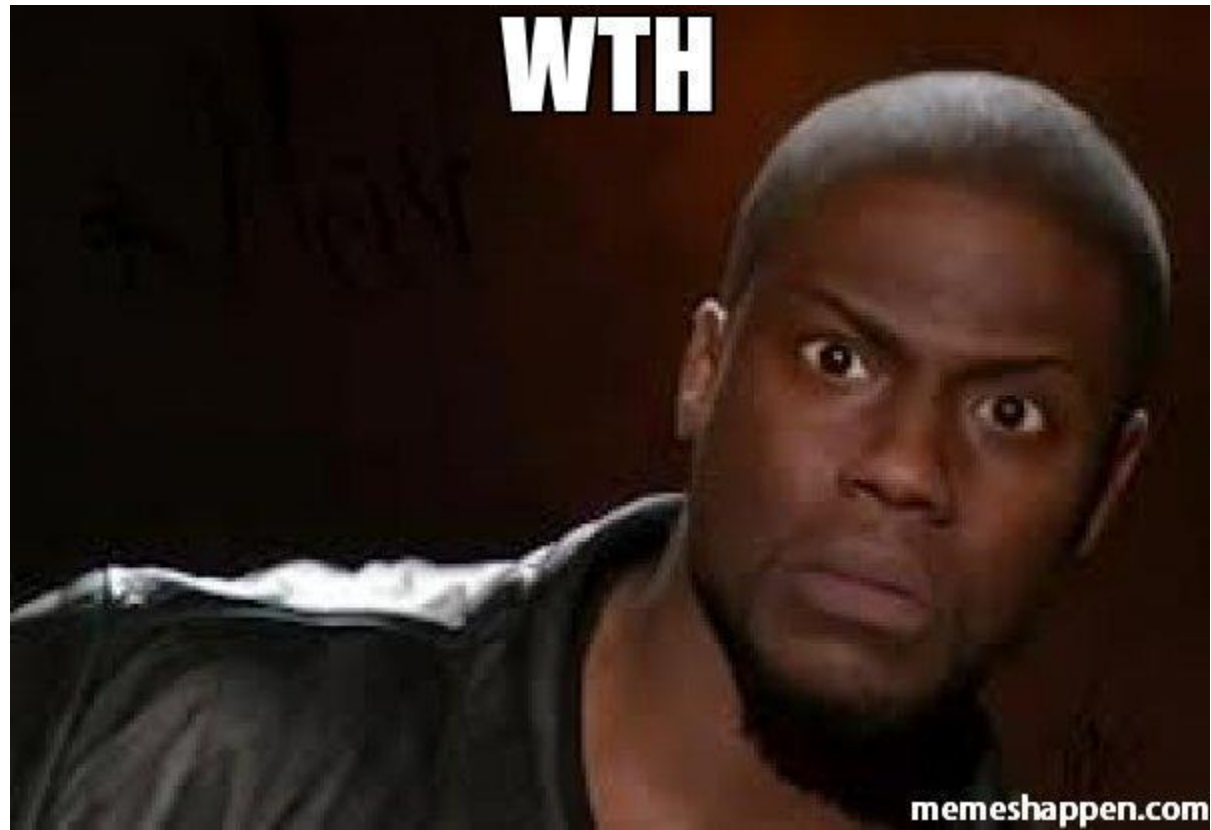
**13.209.97.89**에서 응답하는 데 시간이 너무 오래 걸립니다.

다음 방법을 시도해 보세요.

- 연결 확인
- 프록시 및 방화벽 확인
- [Windows 네트워크 진단 프로그램 실행](#)

ERR\_CONNECTION\_TIMED\_OUT

새로고침





# Port 설정이 필요합니다!

- 지금은 기본 설정이라서 port 번호 22번만 열려 있습니다!
- 직접 작성한 Port 번호를 보안 그룹에 추가하여 접속이 가능하게 만들어 봅시다!

세부 정보 | **보안** | 네트워킹 | 스토리지 | 상태 검사 | 모니터링 | 태그

▼ 보안 세부 정보

IAM 역할  
-

소유자 ID  
599697610402

보안 그룹  
sg-01b2877a9f373e986 (launch-wizard-1)

▼ 인바운드 규칙

Q 필터 규칙

보안 그룹 규칙 ID	포트 범위	프로토콜	원본	보안 그룹
sgr-04809f9db7715d339	22	TCP	0.0.0.0/0	launch-wizard-1



인바운드 규칙

아웃바운드 규칙

태그

이제 Reachability Analyzer를 사용하여 네트워크 연결을 확인할 수 있습니다.

Reachability Analyzer 실행



인바운드 규칙 (1/1)

보안 그룹 규칙 필터



태그 관리

인바운드 규칙 편집



1



<input checked="" type="checkbox"/>	Name ▾	보안 그룹 규칙 ID ▾	IP 버전 ▾	유형 ▾	프로토콜 ▾	포트 범위 ▾	소스 ▾	설명
<input checked="" type="checkbox"/>	-	sgr-04809f9db7715d3...	IPv4	SSH	TCP	22	0.0.0.0/0	-



# 인바운드 규칙 편집 정보

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

## 인바운드 규칙 정보

보안 그룹 규칙 ID	유형 <small>정보</small>	프로토콜 <small>정보</small>	포트 범위 <small>정보</small>	소스 <small>정보</small>	설명 - 선택 사항 <small>정보</small>	
sgr-04809f9db7715d339	SSH ▼	TCP	22	사용자 지정 ▼	<input type="text" value="Q"/>	<input type="text"/>
					<input type="text" value="0.0.0.0/0 X"/>	<input type="button" value="삭제"/>



## 인바운드 규칙 정보

보안 그룹 규칙 ID      유형 정보      프로토콜 정보      포트 범위 정보      소스 정보      설명 - 선택 사항 정보

sgr-04809f9db7715d339

SSH ▼

TCP

22

사용자 지정 ▼

Q

삭제

0.0.0.0/0 X

-

사용자 지정 TCP ▼

TCP

4000

사용자 지정 ▼

Q |

삭제

CIDR 블록

0.0.0.0/0

0.0.0.0/8

0.0.0.0/16

0.0.0.0/24

규칙 추가

취소

변경 사항 미리 보기

규칙 저장





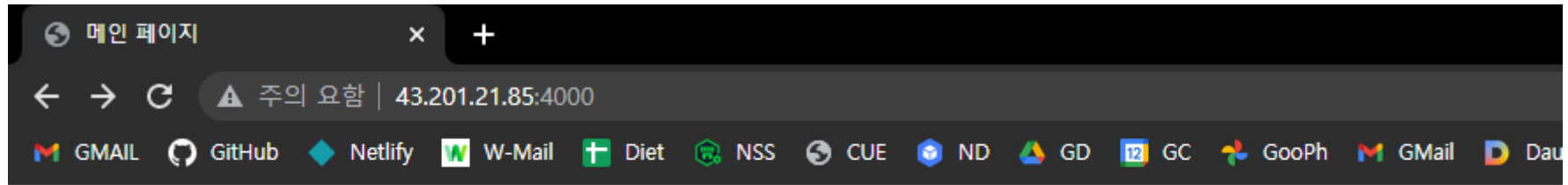
인바운드 규칙 (2)

태그 관리

인바운드 규칙 편집

보안 그룹 규칙 필터

<input type="checkbox"/>	Name ▾	보안 그룹 규칙 ID ▾	IP 버전 ▾	유형 ▾	프로토콜 ▾	포트 범위 ▾	소스 ▾	설명
<input type="checkbox"/>	-	sgr-02c5b04a937f12212	IPv4	사용자 지정 TCP	TCP	4000	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-04809f9db7715d3...	IPv4	SSH	TCP	22	0.0.0.0/0	-



# Welcome to Tetz Express Service!

[로그인 바로가기](#)

[회원가입 바로가기](#)

[게시판 서비스](#)

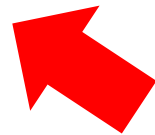


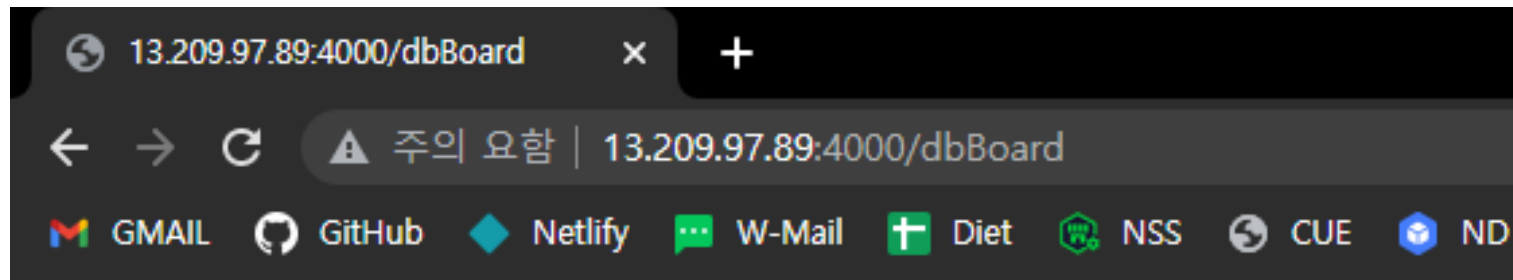
# Welcome to Tetz Express Service!

[로그인 바로가기](#)

[회원가입 바로가기](#)

[게시판 서비스](#)





로그인이 필요한 서비스 입니다.

[로그인 페이지로 이동](#)

## 로그인

아이디

비밀번호

로그인



## Tetz Board

현재 등록 글 : 2

글쓰기

로그아웃

작성자 : 11

11

11

수정

삭제

작성자 : 11

11

11

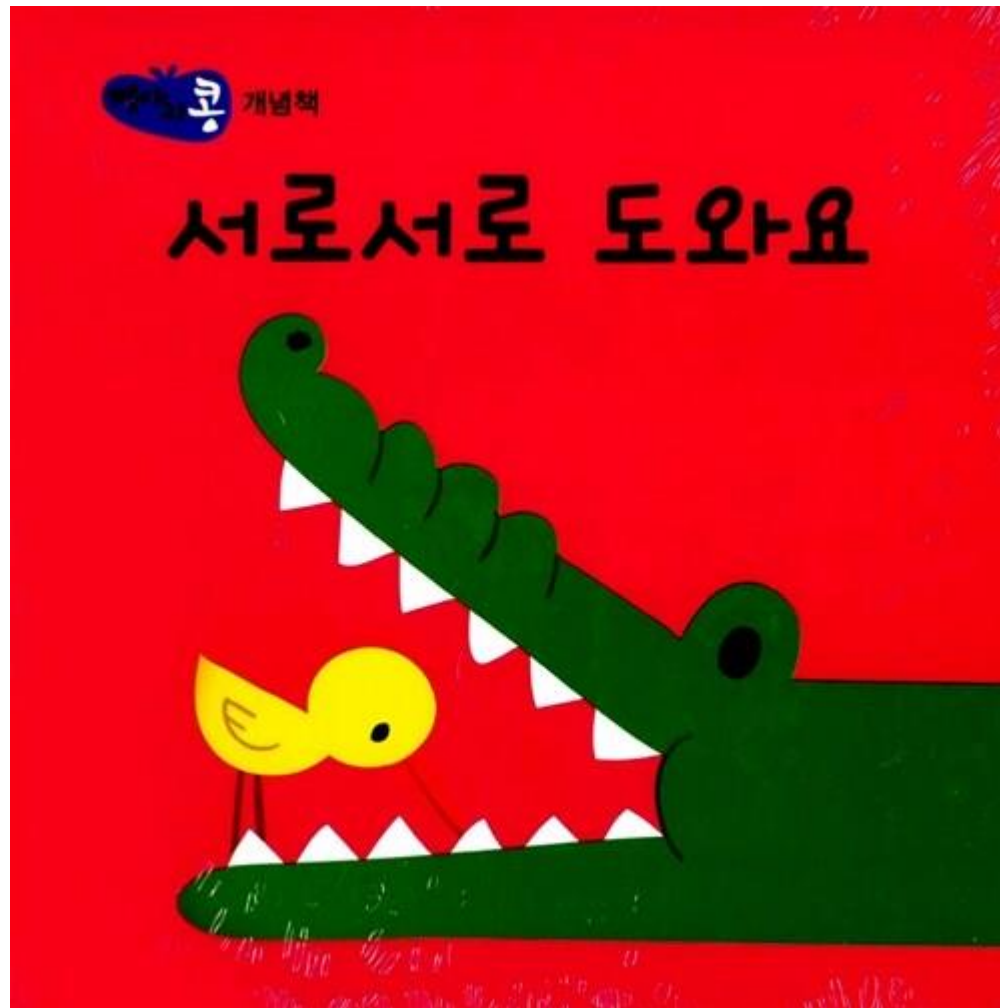
수정

삭제



# SLACK 댓글로 각자 AWS 주소를 올려 주세요!

- 서로 서비스를 방문해서 회원 가입하고 로그인하고 글도 남겨 봅시다!
- 웹 서비스를 개발하는 건 이런 재미가 중요하니까요!









BACK END



BACK END

***over***



# React JS





# React JS



가즈아



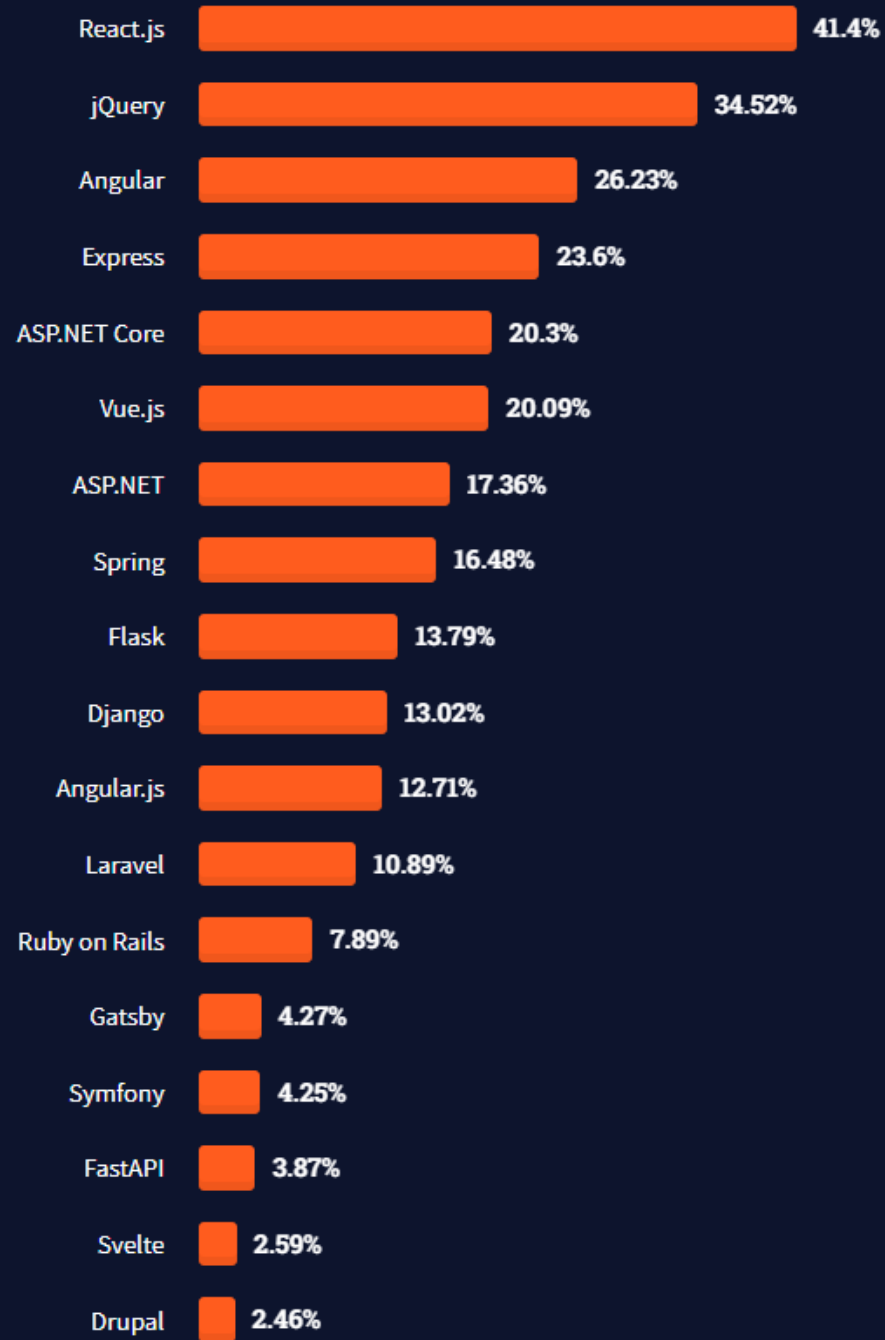
# Why?

**React JS**





# 생태계





site: stackoverflow.com react

전체 뉴스 동영상 이미지 쇼핑

검색결과 약 34,900,000개 (0.48초)

[https://stackoverflow.com > questions > tagged > reactjs](https://stackoverflow.com/questions/tagged/reactjs)

### Newest 'reactjs' Questions - Stack Overflow

React is a JavaScript library for building user interfaces. ... I'm getting 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)





**JOBKOREA**  × |

경력 ▾

학력 ▾

기업형태 ▾

고용형태 ▾

연봉 ▾

조건추가 ▾

채용정보

기업정보

총 2,142건

**JOBKOREA**  × |

경력 ▾

학력 ▾

기업형태 ▾

고용형태 ▾

연봉 ▾

조건추가 ▾

채용정보

기업정보

총 4,346건

**JOBKOREA**  ×

경력 ▾

학력 ▾

기업형태 ▾

고용형태 ▾

연봉 ▾

채용정보

기업정보

총 412건

**JOBKOREA**

경력 ▾

학력 ▾

기업형태 ▾

고용형태 ▾

연봉 ▾

채용정보

기업정보

총 1,006건

**JOBKOREA**  × |

경력 ▾

학력 ▾

기업형태 ▾

고용형태 ▾

연봉 ▾

채용정보

기업정보

총 1,797건



# Virtual DOM

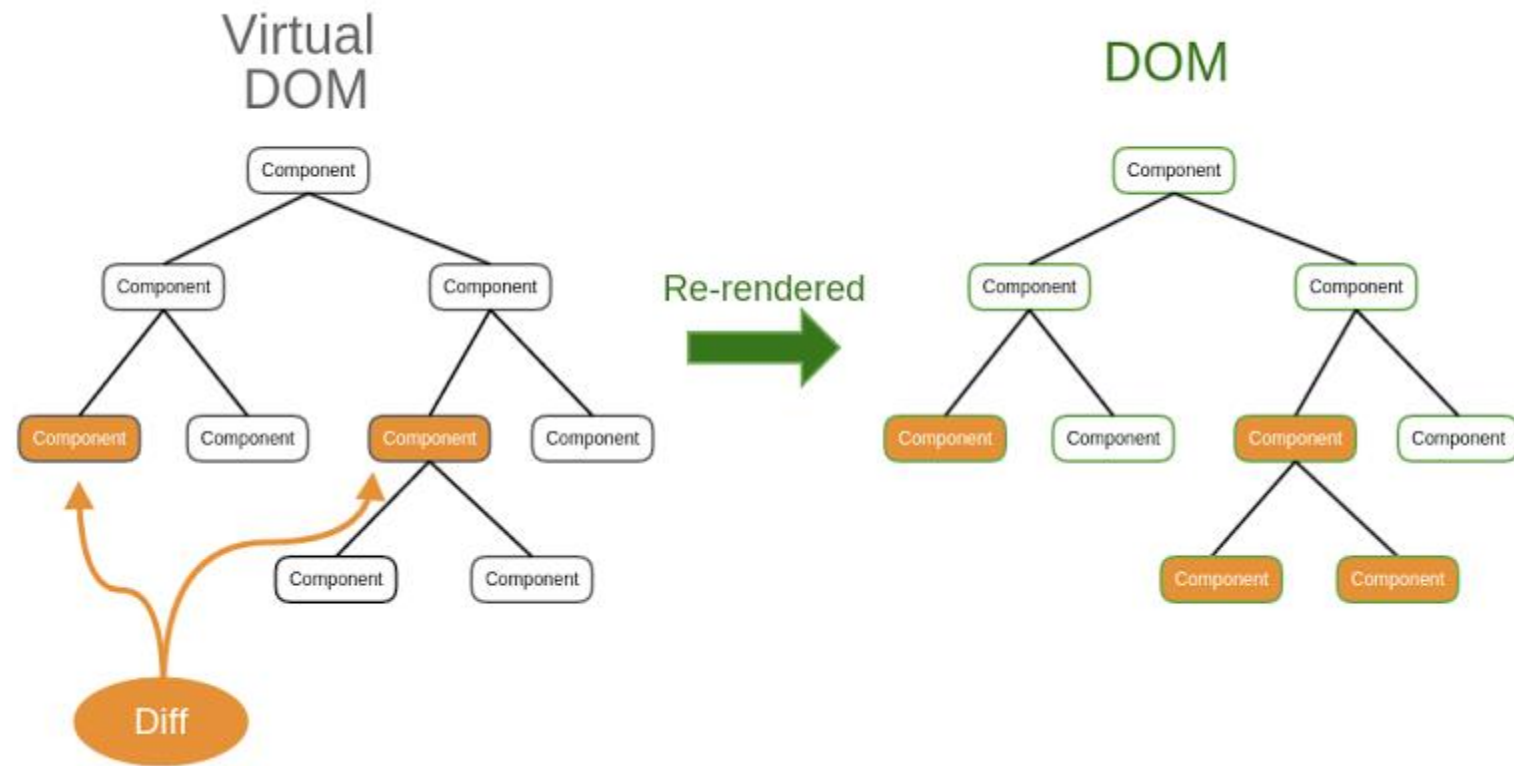
















# Virtual DOM

부드럽고 빠르다!

아키텍처에 대한 고민 ↓



# 거인의 어깨를 빌려라

성공 공식을 읽다

배연국 지음

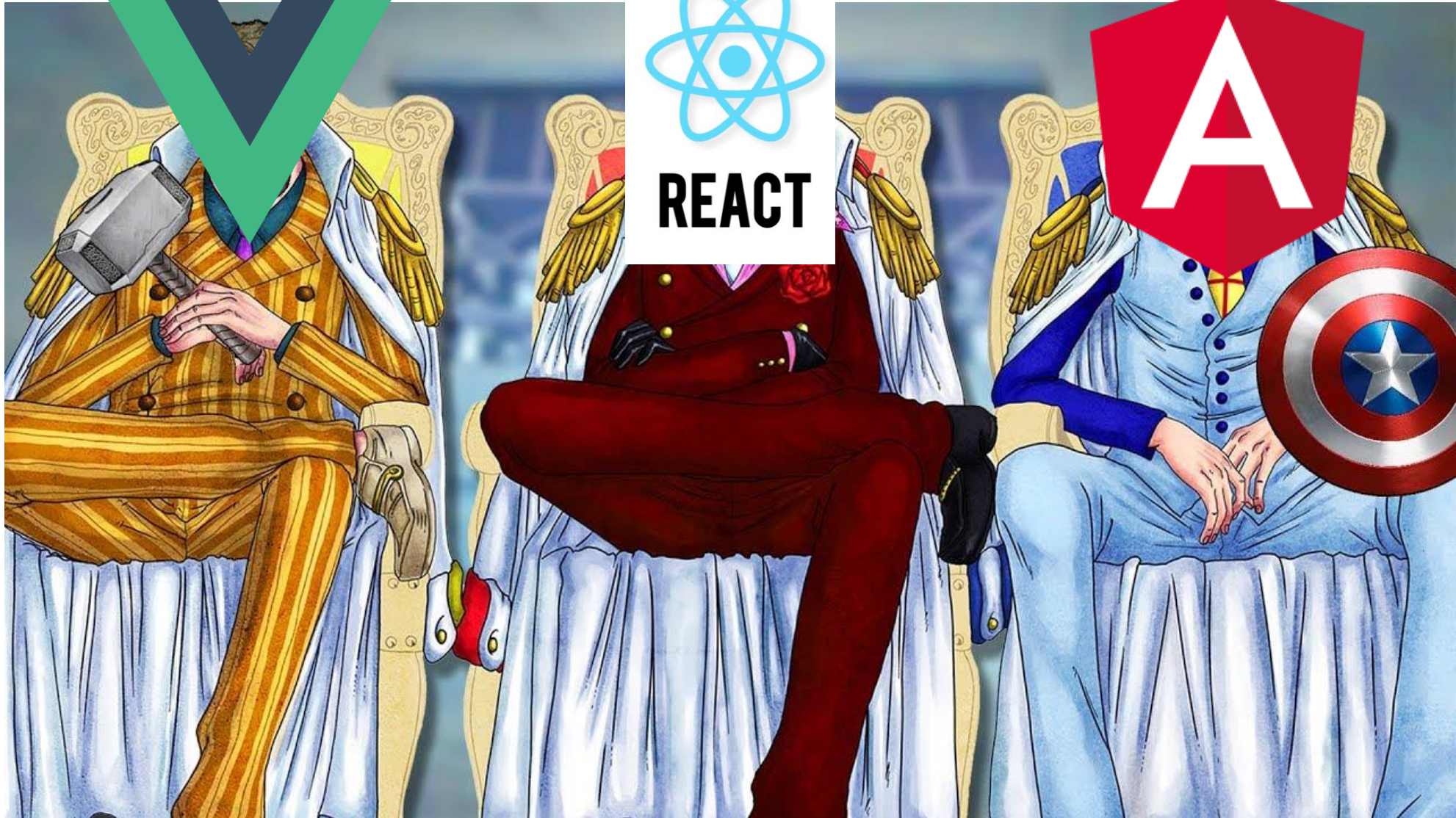
당신을 성공으로 안내할  
영웅들의 감동 스토리!



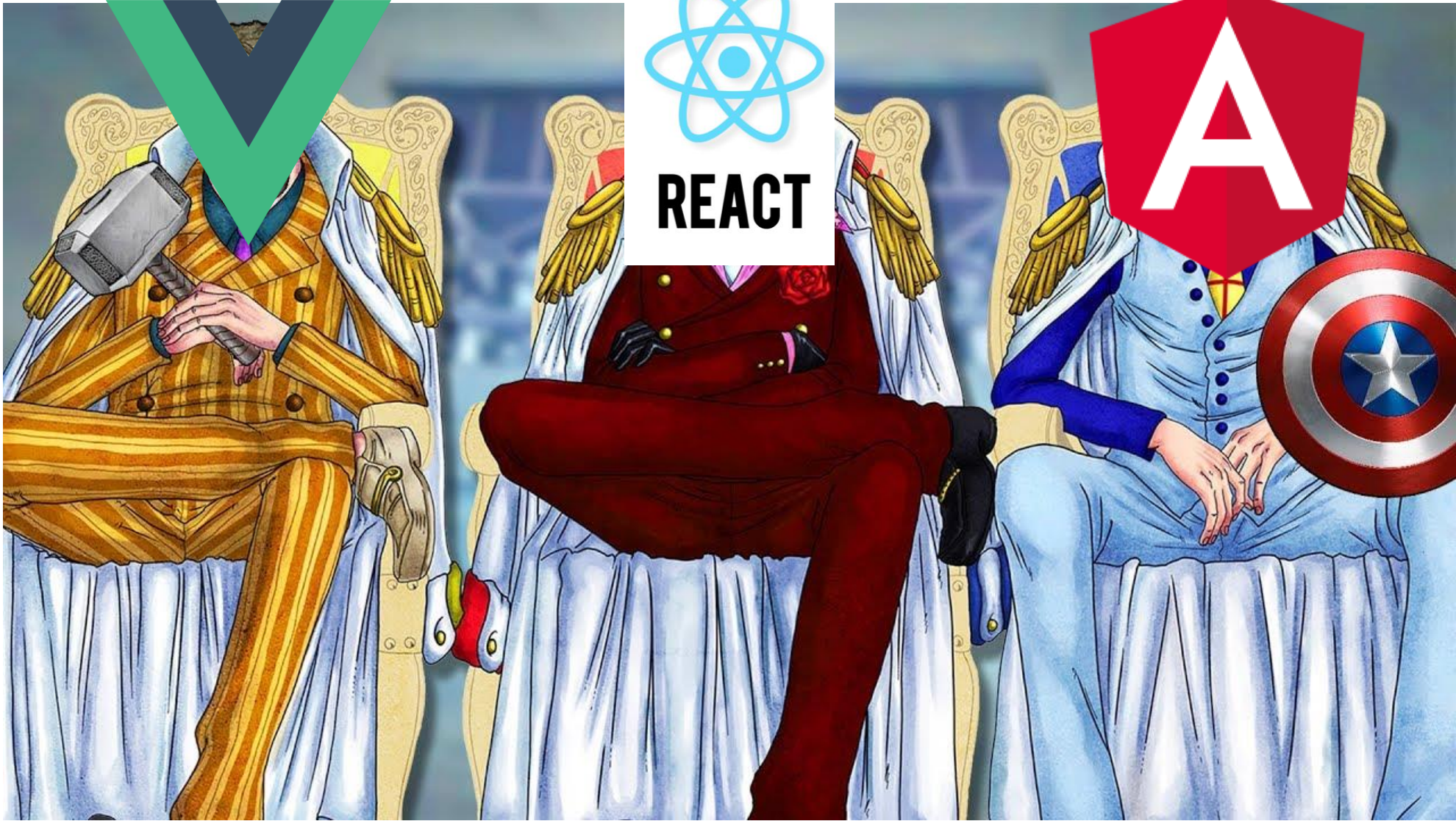
지상사

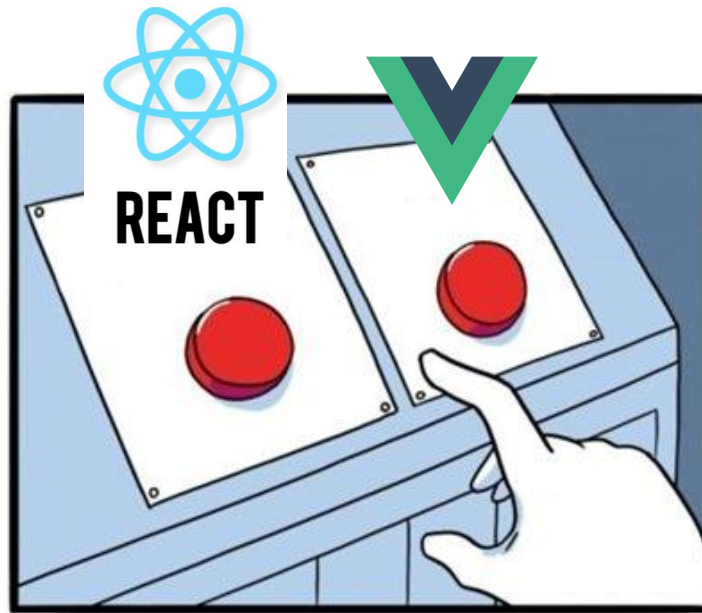


**facebook**











# React JS







일단  
시작하기!





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

# React

사용자 인터페이스를 만들기 위한 JavaScript 라이브러리

시작하기

자습서 읽어보기 >



React

문서

자습서

블로그

커뮤니티

🔍 검색

v18.2.0



Languages

GitHub

여러분이 사용하고 있는 코드 편집기를 사용하길 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 페이지](#)를 만드는 것이 최고의 방법일 것입니다. 설치하는 데 1분밖에 걸리지 않습니다!

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

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!

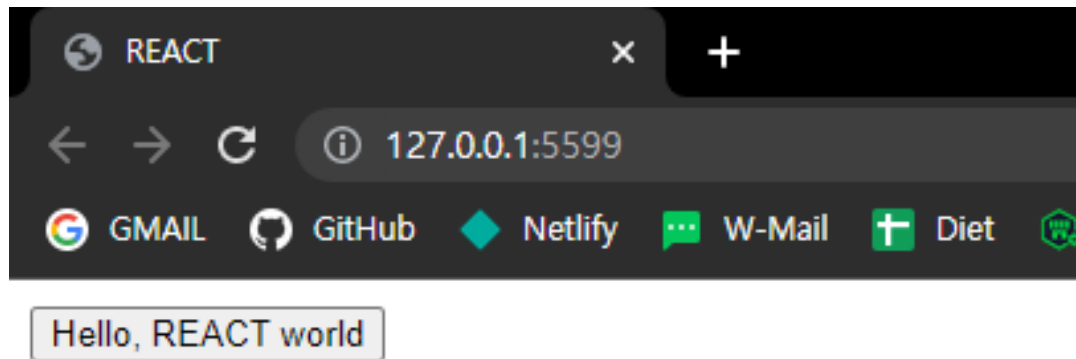
```
// import React from 'react';
// import ReactDOM from 'react-dom';
// 함수형 컴포넌트
function HelloWorldButton() {
  return React.createElement(
    "button",
    { onClick: () => {} },
    "Hello, REACT world"
  );
}

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 때문 입니다
- 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 함수로 HTML 요소를 만들어 내고 관리를 하는듯 하네요?
- 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 content 로 text 를 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>
```

```
▼ <div id="app"> == $0
  ▼ <button>
    ▼ <div>
      <span>Hello, React world</span>
    </div>
  </button>
</div>
```

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









# 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 />);
```



편안



# 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 친구 입니다!
- 그래서 예전 이름이 Six to Five 였었죠 😊



# Babel



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



# Webpack



# Webpack

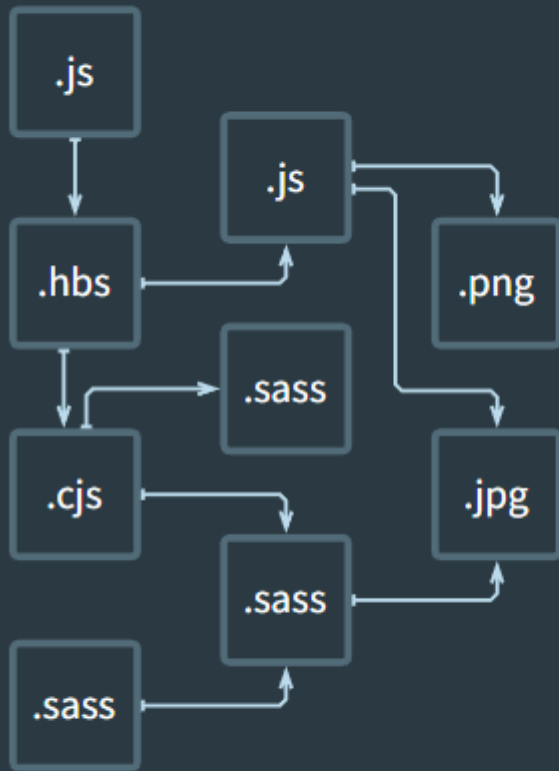
- <https://webpack.js.org/>
- JS 는 사실 별 기능이 없던 언어 입니다!
- 그래서 브랜던 아이크가 자바를 배껴서 단 10일만에 만들 수 있었습니다
- 그런데 웹이 커지고, 점점 더 요구하는게 많아지다보니 기능이 추가되기 시작 했습니다
- 그런데, 이건 웹이죠? 즉 통신을 기반으로 한 서비스 입니다!



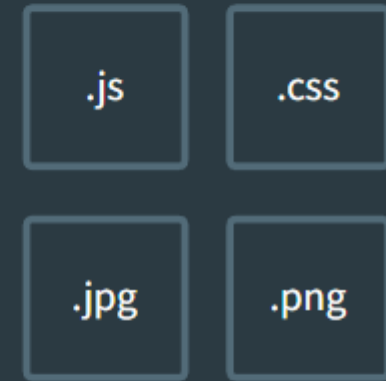
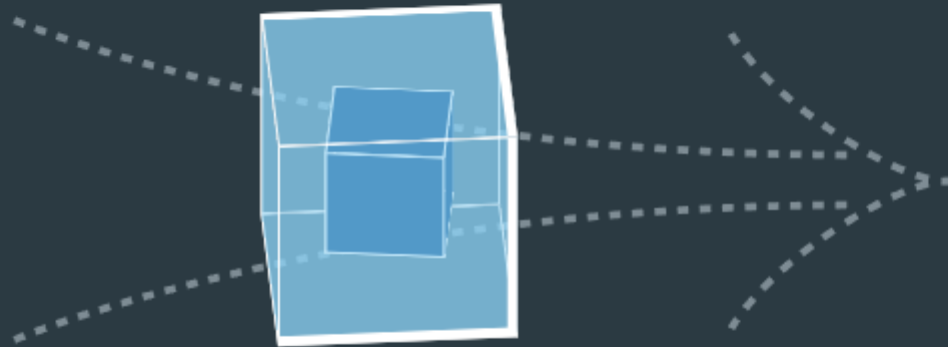
# Webpack

- 그 모든 기능을 다 때려넣고 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



MODULES WITH DEPENDENCIES



STATIC ASSETS



# React JS





# REACT?!

- 그럼 React 하려면 항상 바벨 추가하고? 빌드할 때에는 Webpack 설정 해줘야 해요?
- 우리에게엔 페이스북(요즘은 Meta) 계입니다!





# 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**





# 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 klona@^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](#)

Elements Console Sources <b>Network</b> Performance Memory Appli					
<div>● ● 🔍 <input type="checkbox"/> Preserve log <input checked="" type="checkbox"/> Disable cache No throttling ▼ 📶 ⬆️ ⬇️</div> <div>Filter <input type="checkbox"/> Invert <input type="checkbox"/> Hide data URLs</div> <div>All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other <input type="checkbox"/> Has blocke</div> <div><input type="checkbox"/> 3rd-party requests</div> <div>1000 ms 2000 ms 3000 ms 4000 ms</div>					
Name	Status	Type	Initiator	Size	Time
localhost	200	docum...	Other	1.3 kB	2 m
bundle.js	200	script	(index)	360 kB	47 m
logo.6ce24c58023cc2f8fd88fe9...	200	svg+xml	react-dom.deve...	1.7 kB	3 m
manifest.json	200	manifest	Other	936 B	3 m
favicon.ico	200	x-icon	Other	3.9 kB	3 m
logo192.png	200	png	Other	5.7 kB	3 m
ws	101	websoc...	WebSocketClie...	0 B	Pendin
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					



ws	101	websoc...	WebSocketClie...	0 B	Pendin
js.js	200	script	content.js:32	1.4 kB	3 n
dom.js	200	script	content.js:32	2.0 kB	2 n
js.js	200	script	content.js:32	1.4 kB	2 n
dom.js	200	script	content.js:32	2.0 kB	1 n
js.js	200	script	content.js:32	1.4 kB	2 n
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 썩이나 하는 이유는 React 에서 필요한 필수 모듈 등이 전부다 들어와 있기 때문입니다!
- 즉, 아까 배웠던 Webpack 이 압축하기 전 이라서 그렇습니다!!





# 빌드 해보기



# 빌드 하기

- 프로젝트 폴더에 가서 빌드를 를 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.0215976b.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=/^
```

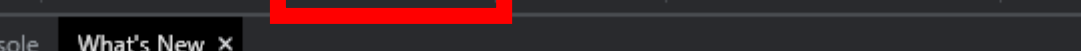
```
[ :A-Z_a-z\u00C0-\u00D6\u00D8-\u00F6\u00F8-\u02FF\u0370-\u037D\u037F-\u1FFF\u200C-\u200D\u2070-\u218F\u2C00-\u2FEF\u3001-\uD7FF\uF900-\uFDCF\uFDF0-\uFFFFD] [ :A-Z_a-z\u00C0-\u00D6\u00D8-\u00F6\u00F8-\u02FF\u0370-\u037D\u037F-\u1FFF\u200C-\u200D\u2070-\u218F\u2C00-\u2FEF\u3001-\uD7FF\uF900-\uFDCF\uFDF0-\uFFFFD]-. 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 autoPlay 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=/[\-:]([a-z])/g;function y(e){return e[1].toUpperCase()}functio hasOwnProperty(n)?v[n]:null;(null!==l?0!==l.type:r||!(2<n.length)||“o“!==n[0]&&“O“!==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?!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 댔네요!
- 그럼 한번 실제로 빌드 결과물을 개발자 도구로 용량이 어찌 되는지 봅시다!
- 빌드 된 프로젝트 결과물의 경우 그냥 html 파일을 켜다고 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!
```



The screenshot shows the Chrome DevTools Performance tab. A red box highlights the text "166 kB resources" in the bottom status bar. Other visible text includes "12 requests", "69.1 kB transferred", "Finish: 6.20 s", "DOMContentLoaded: 47 ms", and "Load: 58 ms".



**webpack**



# 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 View 를 변화 합니다
- 즉, 쇼핑몰에서 상품 목록 페이지를 하나하나 다 그려줄 필요 없이 하나의 컴포넌트만 만들고 변화되는 데이터만 props 로 전달하여 재사용 하면 됩니다!



어디든지

언제든 일주일

게스트 추가



호스트 되기



1

기상천외한 숙소

국립공원

통나무집

풍차

상징적 도시

섬

해변 근처

초소형 주택

디자인

캠핑카

A자형 주택

호숫가

북극

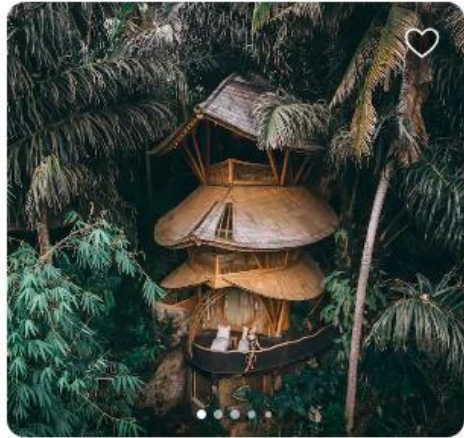
멋진 수영장

등글

서핑



필터



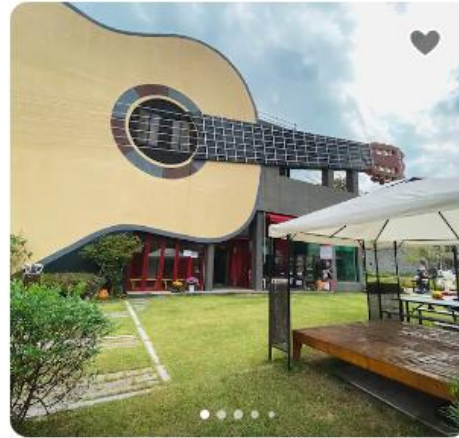
Abiansemal, 인도네시아  
5,275km  
4월 13일~18일  
₩483,768 /박

★ 4.88



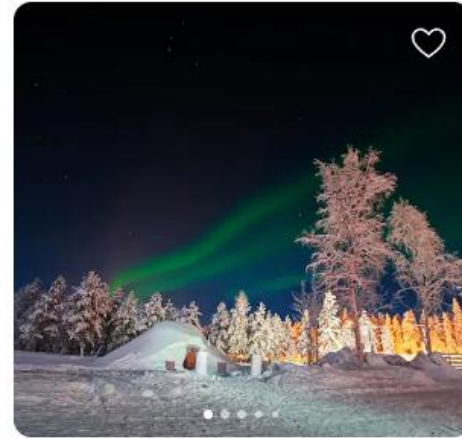
Abiansemal, 인도네시아  
5,275km  
12월 13일~19일  
₩1,550,904 /박

★ 5.0



Sindun-myeon, Icheon-si, 한국  
46km  
10월 3일~8일  
₩107,673 /박

★ 4.8



Pelkosenniemi, 핀란드  
6,604km  
12월 28일 ~ 1월 2일  
₩202,046 /박

★ 4.81



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

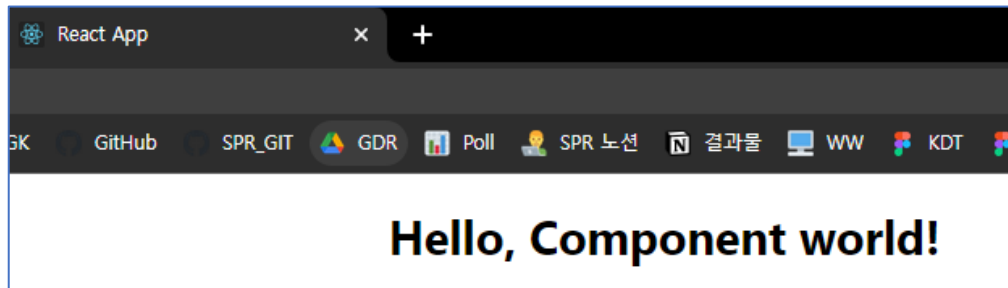
✖ Warning: The tag <mainHeader> is unrecognized in this react-dom.development.js:86
browser. If you meant to render a React component, start its name with an uppercase
letter.
    at mainHeader
    at div
    at App
```

- PascalCase 기억 나시나요?
- 컴포넌트는 JS에서 클래스 또는 생성자 함수와 비슷한 역할(정의 → 재사용)을 하다보니 PascalCase 를 사용합니다!
- 그리고 애초에 컴포넌트는 클래스로 만들어 졌었기 때문입니다!



# PascalCase 로 변경

- 파일명, 컴포넌트명을 전부 PascalCase 로 변경하여 적용해 봅시다!



- 페이지를 새로 고침 않아도 적용이 되어 있죠?
- 이것이 바로 리액트의 장점인 Virtual DOM 의 효과 입니다!
- 컴포넌트 레벨에서의 변화가 있으면 페이지가 자동으로 해당 부분만을 리렌더링 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 컴포넌트 입니다!
- 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 것 입니다!
- 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 Hooks 라는 기능의 도입으로 같은 역할 수행 가능!
- 최근에는 대부분 이거만 씁니다!



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



# 함수형 컴포넌트를 클래스형 컴포넌트로!

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



```
function MainHeader() {  
  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:  'BtnToNavr' 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 렌더링이 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 초창기에는 리액트의 핵심 기능인 State 기능을 쓸 수 없었습니다!
- 하지만 16.8 버전 이후 부터는 useState 라는 메소드(Hooks)를 제공하여 함수형 컴포넌트에서도 State 사용이 가능해 졌습니다
- useState 를 이용해서 state 의 값을 초기화 하고 state 를 변경 할 수 있는 함수를 지정 → 지정한 함수를 이용해서 state 를 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!

