웹어플리케이션보안
(캡스톤디자인)

2. Next.js

중부대학교 정보보호학과
이병천 교수
sultan@joongbu.ac.kr

전체 목차

1. 강의 개요
   - 캡스톤디자인 프로젝트 추진 방법
2. **Next.js**
   - **프런트엔드 프로그래밍**
   - **백엔드 프로그래밍**
   - **NextAuth**
3. 예제 웹서비스 : 쇼핑몰 구축
4. 예제 웹서비스 : 암호 활용 Forge
5. 캡스톤디자인 프로젝트 발표

2. Next.js

2.1 개발환경 구축

2.2 Next.js 기능 소개

2.1 개발환경 구축

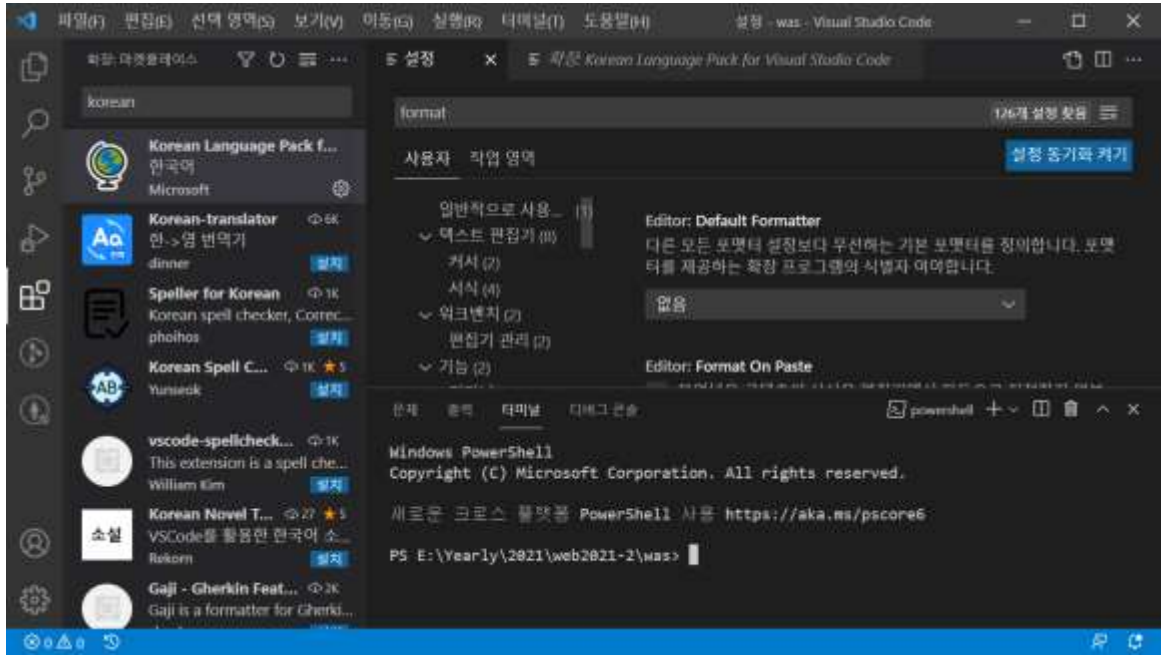
- VSCode 설치,
  - 환경 설정,
  - 유용한 확장 패키지 설치
- Node.js 설치
  - npm 사용법
- MongoDB
  - 서버 : Community server 설치 운영
  - GUI : Compass
  - 클라우드 서비스 : Atlas
  - 계정 등록, 로그인, 사용방법
- Tailwindcss 활용
- Git, github

VSCode 설치

- 웹에디터 VSCode 설치
  - 마이크로소프트에서 개발한 통합개발환경
  - https://code.visualstudio.com/
  - 현재 버전: 1.70.2
- 유용한 확장기능 설치 활용
  - Live Server: 개발용 로컬 웹서버
  - Prettier: 코드 포맷 관리
  - Bracket Pair Colorizer: 괄호 쌍을 색깔로 구별 표시
  - Auto close tag: HTML 태그 자동 완성
  - Indent-rainbow: 들여쓰기를 색깔로 구별 표시
  - Korean language pack: 한국어 UI 환경
  - 기타

VSCode 설치

- VSCode 각종 설정 기능
  - Font size, Tab size, Word wrap, Format on save 등
  - Prettier 설정 : single quote, semi colon
- 내장 명령창 활용
  - Ctrl+`
  - 여러 개의 명령창 열기
  - 디폴트 명령창 설정



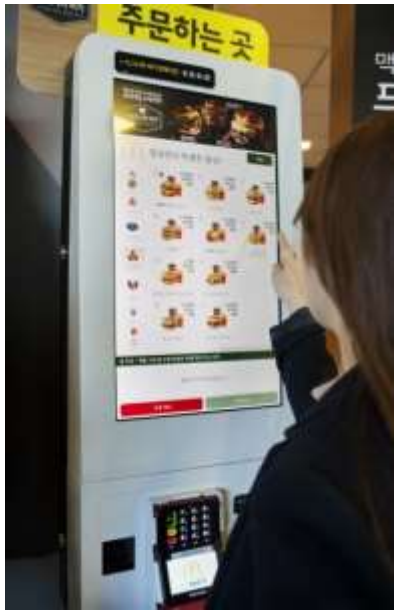
Node.js

- Node.js는 자바스크립트를 이용한 웹서비스 개발 언어로 최근 빠르게 확산되고 있음
  - 구글 크롬의 V8 자바스크립트 엔진에 기반한 서버측 플랫폼.
  - Apache와 같은 별도의 서버 프로그램을 사용하지 않고 서비스를 직접 제공
  - 이벤트기반, 비동기 I/O, 단일 스레드 루프를 통한 높은 처리 성능
- 대규모 회사들에서도 널리 사용 중
  - Microsoft, eBay
  - LinkedIn, Yahoo
  - WalMart, Uber
  - Oracle

https://nodejs.org/

Node.js

- Node.js는 비동기식 단일쓰레드로 모든 서비스 요구에 응답
- Node는 이벤트 기반 언어



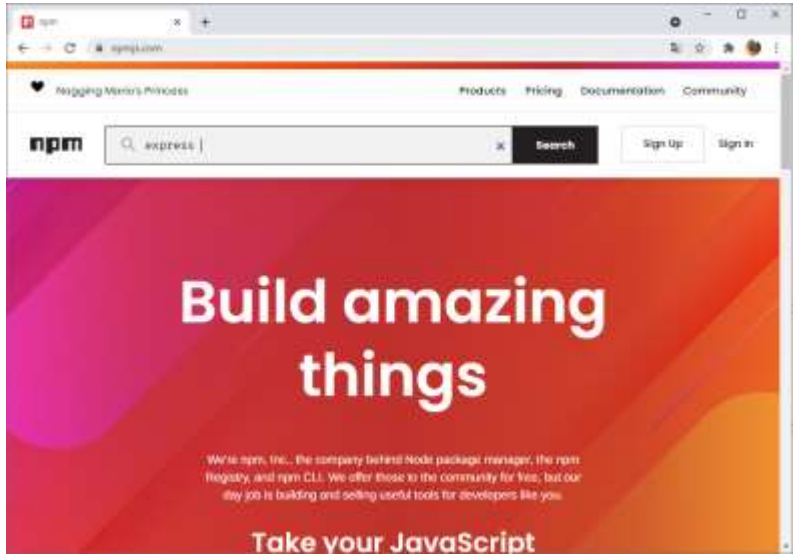
Node.js

- Node의 프로그래밍 방법론
  - 이벤트 기반 프로그래밍 (Event driven programming)
    - 마우스클릭, 키보드입력, 메시지입력 등 이벤트 발생에 따라 프로그램 진행. GUI 프로그래밍이 대표적인 사례. 웹서비스는 기본적으로 이벤트 기반 활동.
  - 비동기 프로그래밍 (Asynchronous programming)
    - 메인 프로그램은 이벤트를 기다림
    - 이벤트 발생시 이벤트핸들러에게 처리를 넘기고 메인 프로그램은 다른 이벤트를 기다림
    - 이벤트 처리가 끝나면 콜백(callback) 함수를 실행
  - 비동기 콜백 (Asynchronous callbacks, non-blocking callbacks)
    - 함수의 인자로 또 다른 함수를 지정하는 방식
    - 처리할 준비가 완료되면 실행되는 함수
    - 처리에 시간이 걸리는 경우에 유용
    - Asynchronous callback: non-blocking (작업을 시켜놓고 다른 일 수행 가능)

Node.js와 NPM

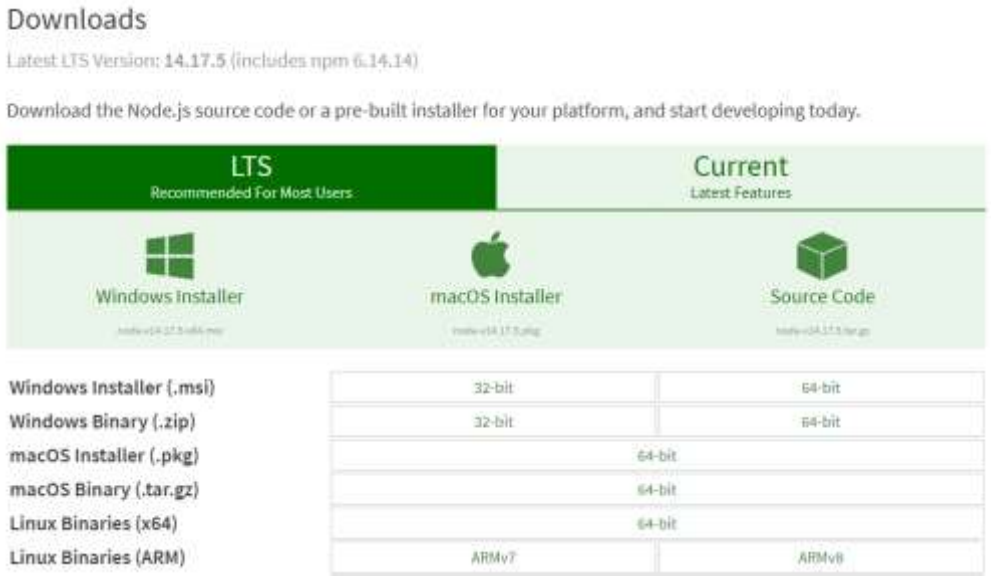
- NPM (Node Package Manager)
  - Node.js에서는 많은 확장 패키지들을 설치하여 사용할 수 있음.
  - NPM은 노드 패키지 관리자로서 패키지 관리가 매우 편리함. 노드를 유명하게 만든 강력한 기능.
  - Node.js 설치시 NPM도 기본 설치됨
  - 자신이 프로그래밍한 코드를 패키지 형태로 만들어 재활용할 수 있음

https://www.npmjs.com/



Node.js 설치

- Node.js
  - 자바스크립트 프로그램 실행 환경
  - https://nodejs.org/en/download/
  - NPM (노드 패키지 관리자)가 함께 설치됨
  - LTS (long-term stable) 버전으로 설치.
  - 현재 16.14.2 (npm 8.5.0)
  - 기본 설치 위치: C:₩Program Files₩nodejs
- 설치 후 버전 확인

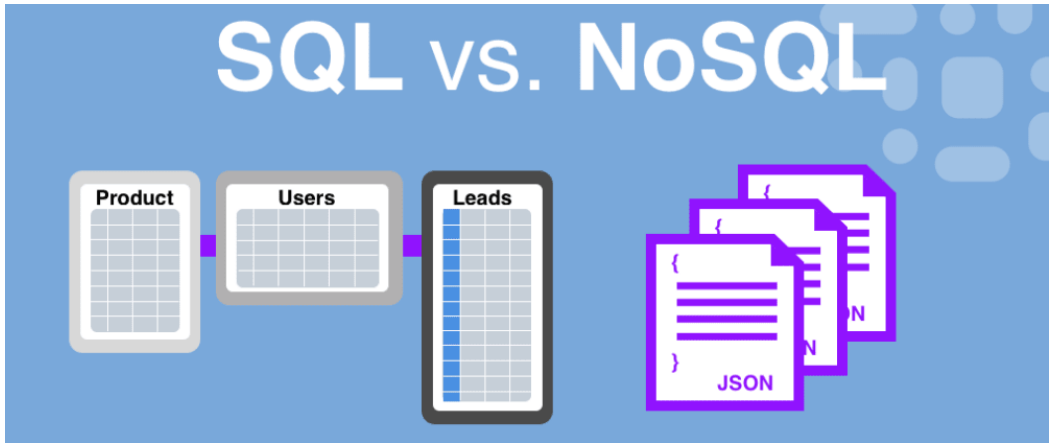



```
D:\WebDev\2022-2\webserver>node -v
v16.14.2

D:\WebDev\2022-2\webserver>npm -v
8.5.0
```

node - Node.js 프로그램 실행 명령어

npm - 노드 패키지 관리자

MongoDB



관계형 DB

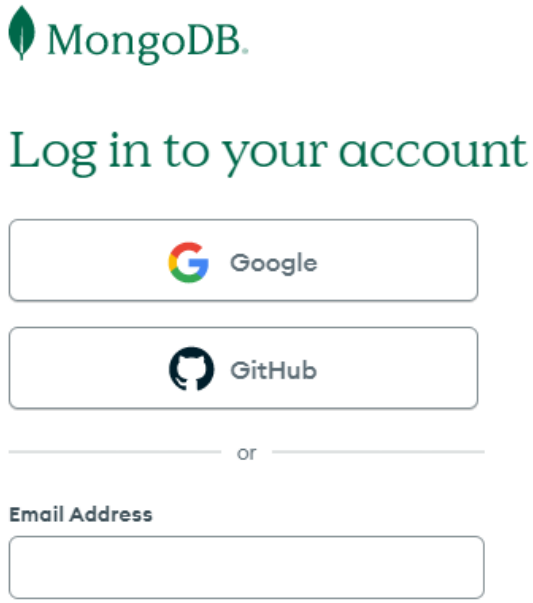
데이터를 테이블 형식으로 저장.
정보간의 관계를 이용.
MySQL

문서 기반 DB

비관계형 데이터베이스로 데이터를 문서 형태로 저장
JSON (JavaScript Object Notation)
복잡한 형태의 데이터를 JSON, XML, BSON(Binary JSON) 등의 포맷으로 데이터베이스에 넣을 수 있음

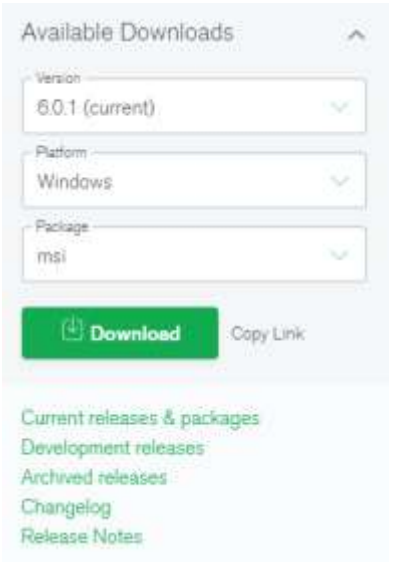
MongoDB 로컬 서버 설치

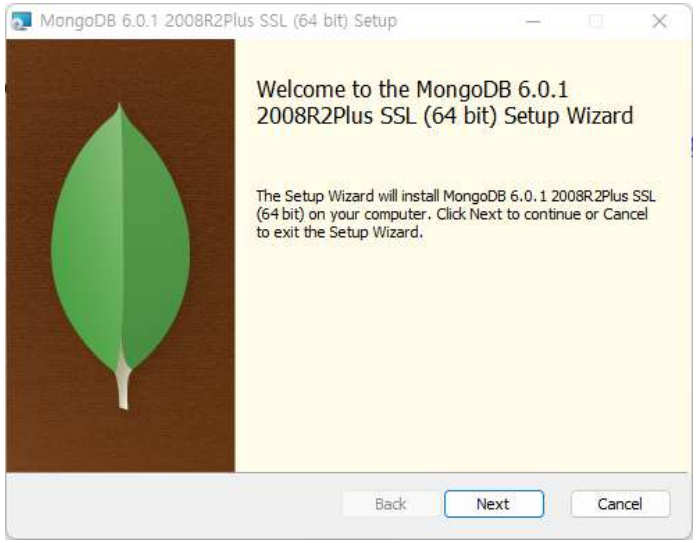
- DB 서버 운영
  - MongoDB community server 설치
  - https://www.mongodb.com/try/download/community
- GUI 클라이언트 프로그램 활용
  - MongoDB Compass 설치
  - https://www.mongodb.com/products/compass
  - MongoDB 활용을 위한 GUI 프로그램
- 클라우드 DB 서비스 활용
  - Atlas: MongoDB as a Service
  - https://www.mongodb.com/
  - 회원가입 및 활용 방법 안내

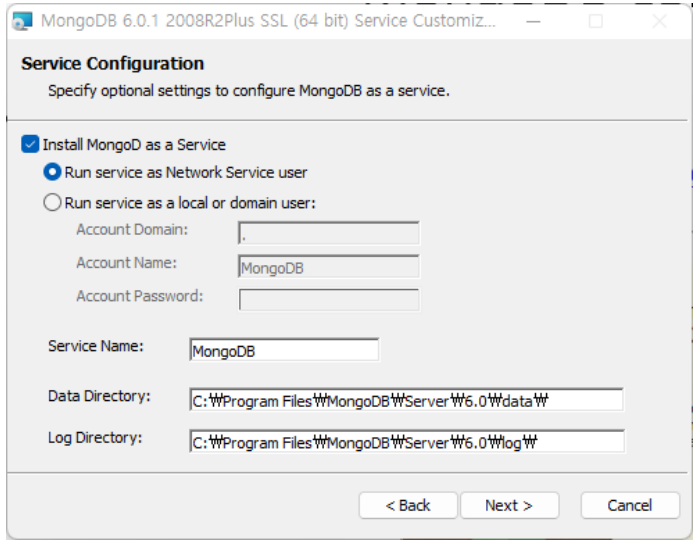


MongoDB community server

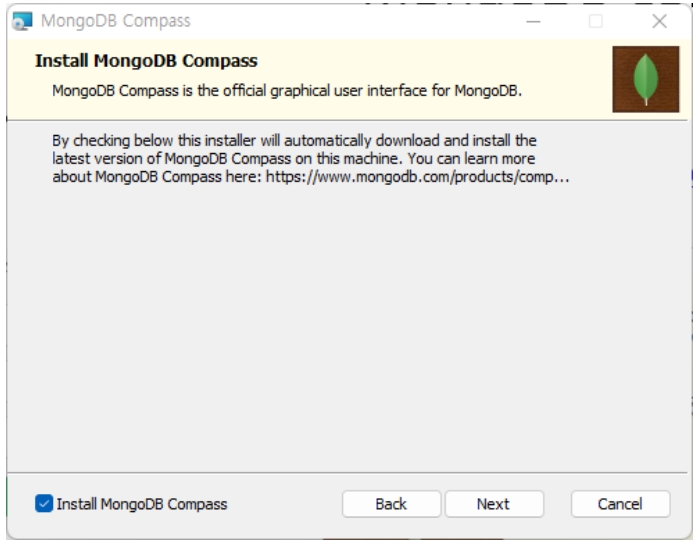
- 서버 다운로드 설치
  - https://www.mongodb.com/try/download/community





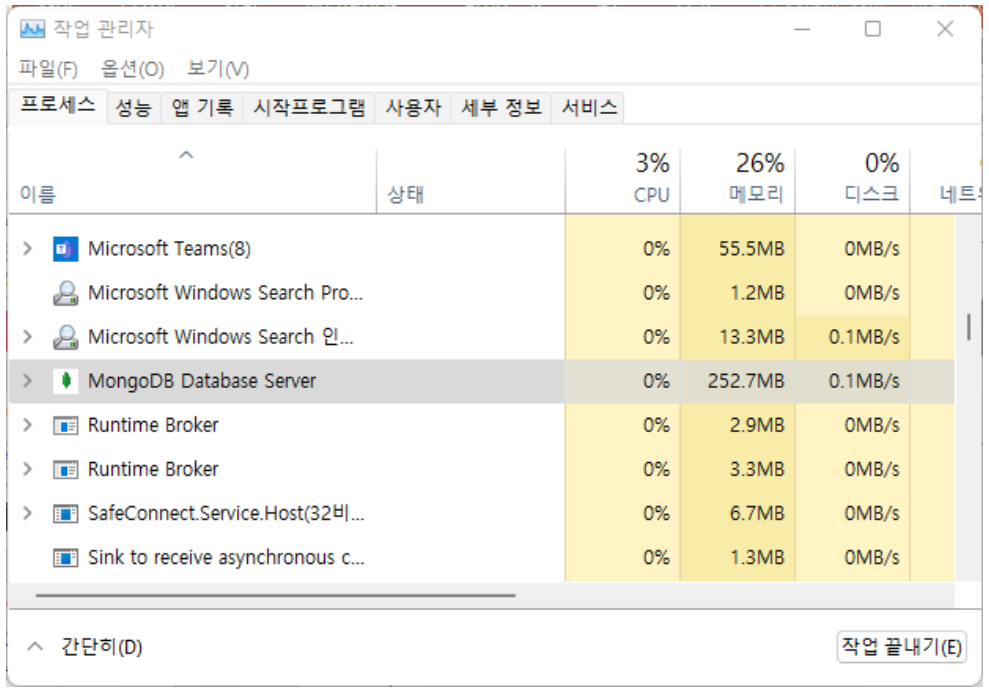


서버 설치 위치

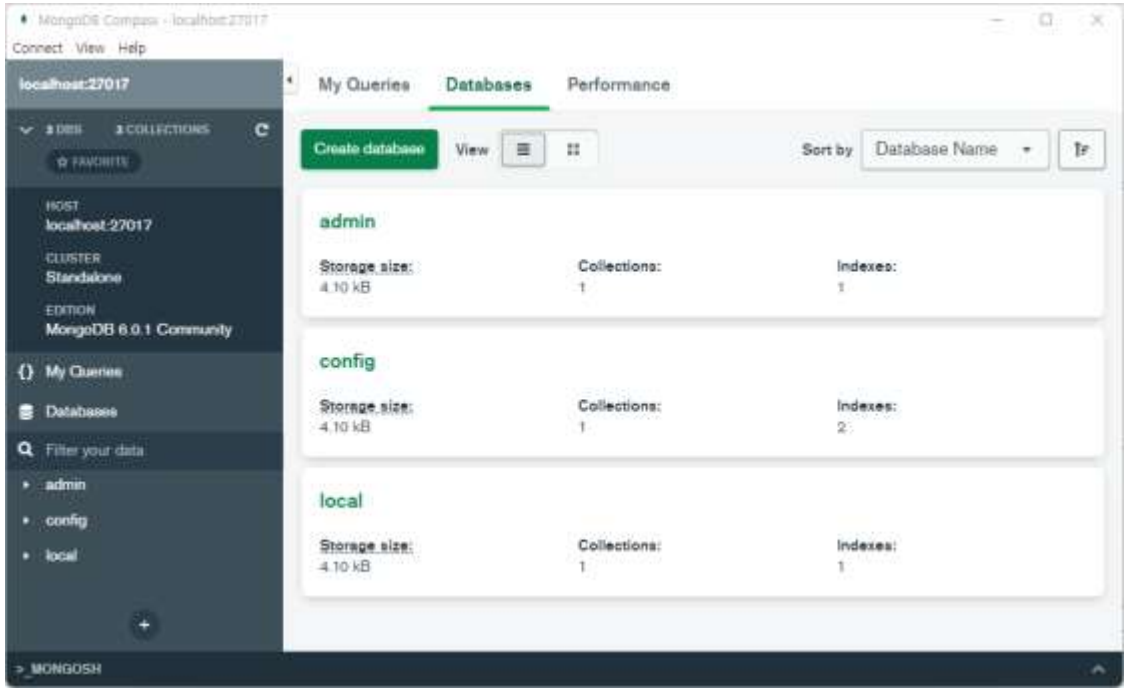


Compass 설치 옵션

Compass 로 서버 연결



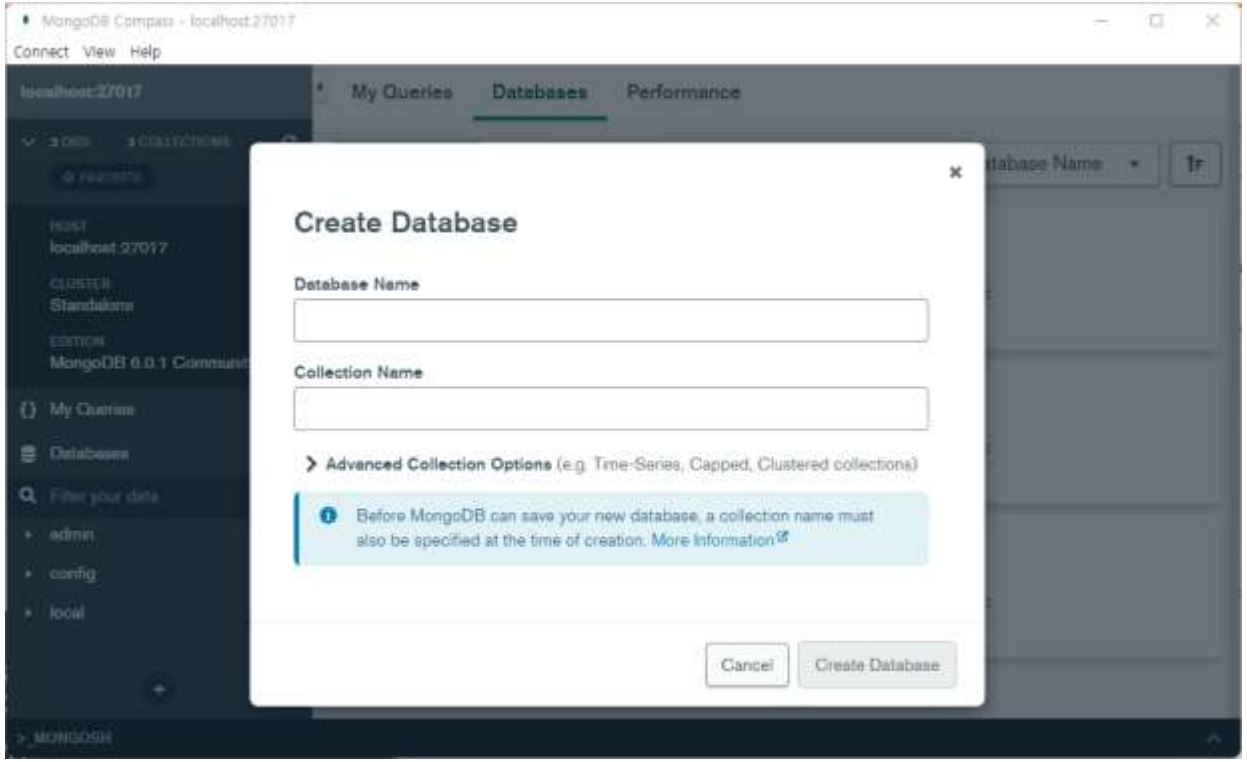
백그라운드로 서버 동작중
부팅시 서버 자동 실행

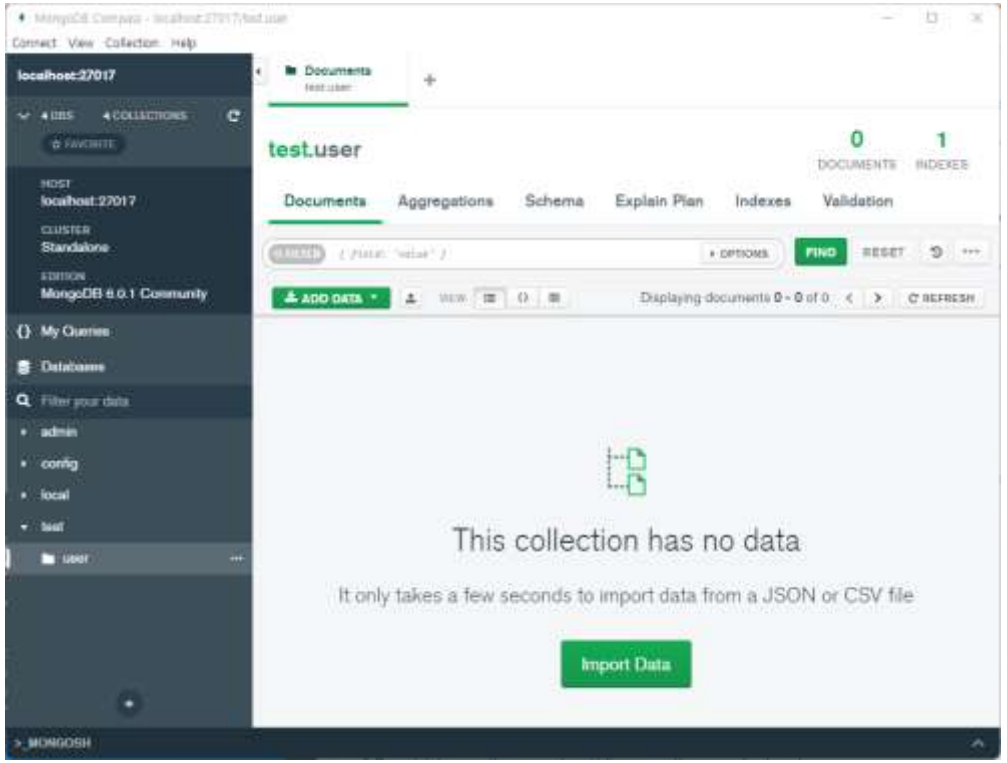


Compass로 서버 연결

로컬서버 연결 주소
mongodb://localhost:27017

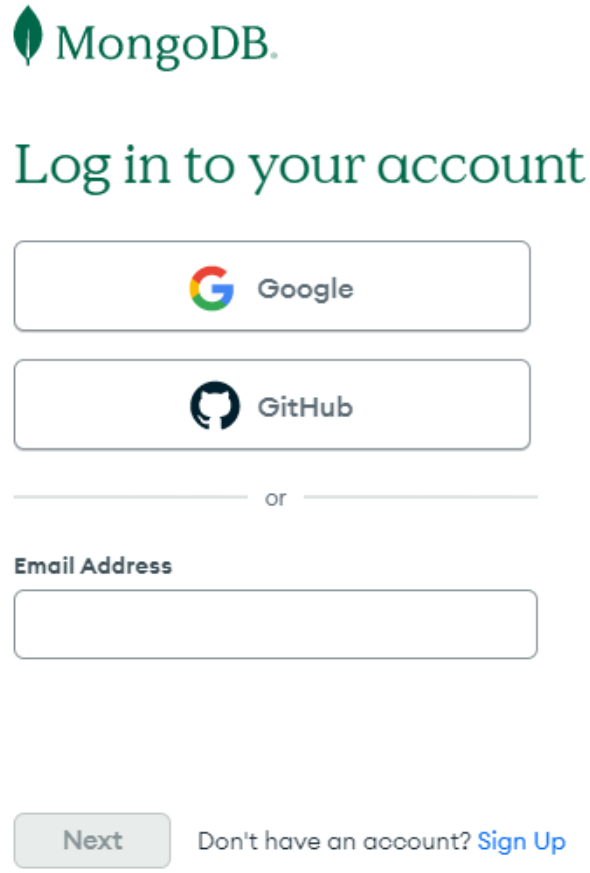
Database, collection 만들기





MongoDB 클라우드 서비스 활용

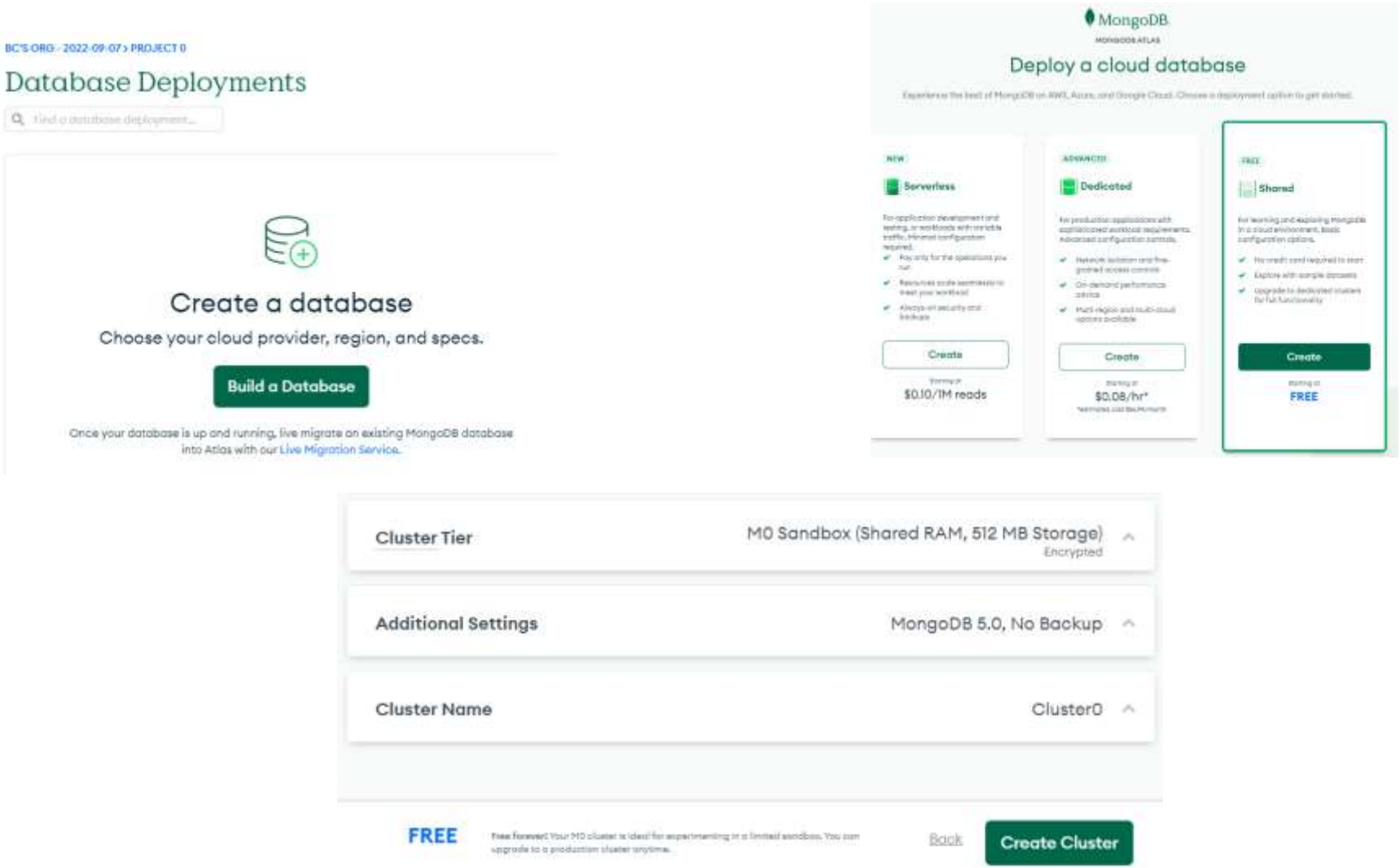
- MongoDB Atlas
  - https://www.mongodb.com/atlas
  - Sign Up (회원가입)
    - Google, Github, Email
  - Sign In (로그인)



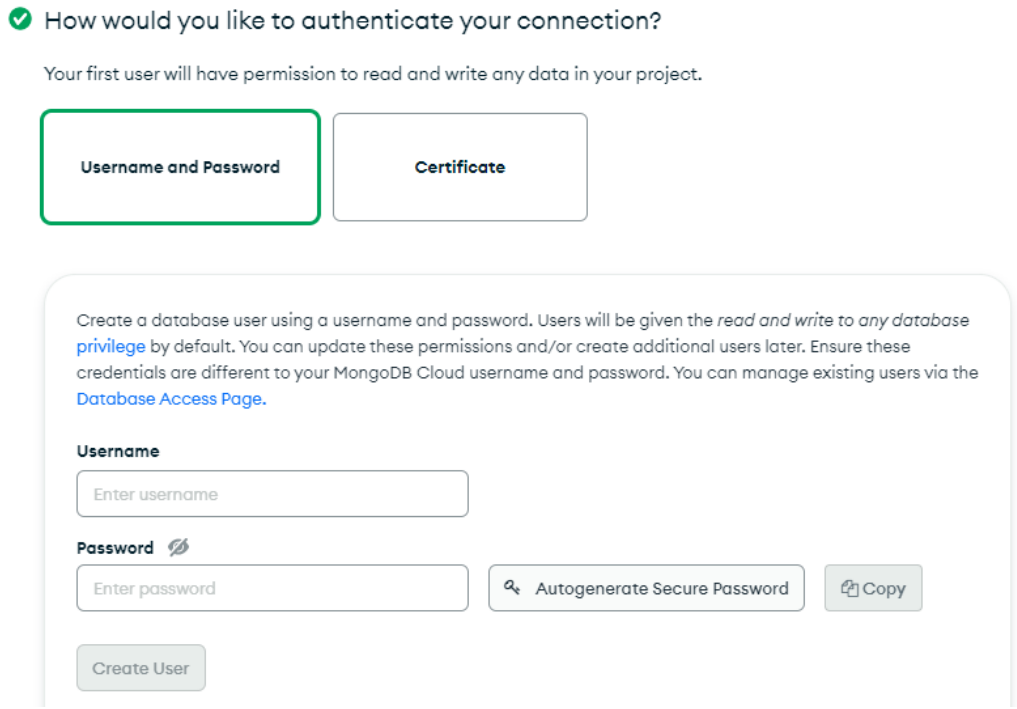
MongoDB 클라우드 서비스 활용

- 초기 설정
  - Organization명, project명 자동 생성됨
  - Cluster명 선택: 클라우드 서비스, 지역 등, Free tier 선택
    - Free tier는 cluster명을 하나만 선택 가능
  - 관리자 계정 생성 필요
  - 초기 클러스터 생성 및 설정에 시간이 걸림. 약간 기다린 후 대시보드가 로드됨.
  - 최초의 DB, collection 생성
  - 완성된 대시보드 로드

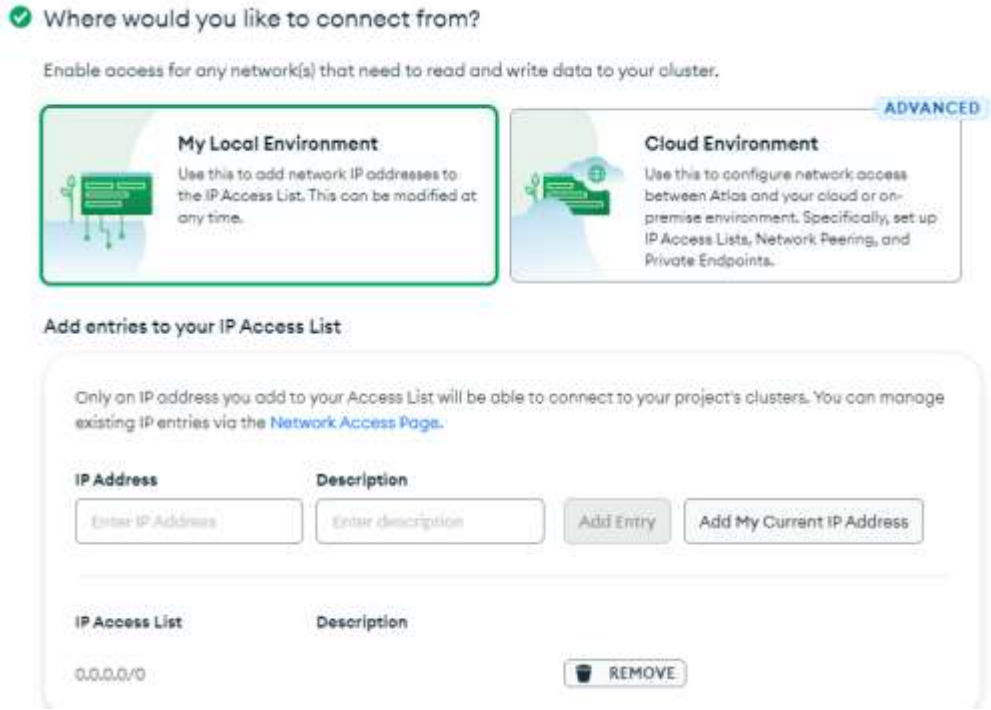
초기 클러스터 생성


BC'S ORG - 2022-09-07 > PROJECT 0
Database Deployments
Create a database
Choose your cloud provider, region, and specs.
Build a Database
Once your database is up and running, live migrate an existing MongoDB database into Atlas with our Live Migration Service.
MongoDB
Deploy a cloud database
Serverless
Create
$0.10/1M reads
Dedicated
Create
$0.08/hr*
Shared
Create
FREE
Cluster Tier
M0 Sandbox (Shared RAM, 512 MB Storage)
Encrypted
Additional Settings
MongoDB 5.0, No Backup
Cluster Name
Cluster0
FREE
Back
Create Cluster

관리자 계정 생성, IP 접근제어 설정

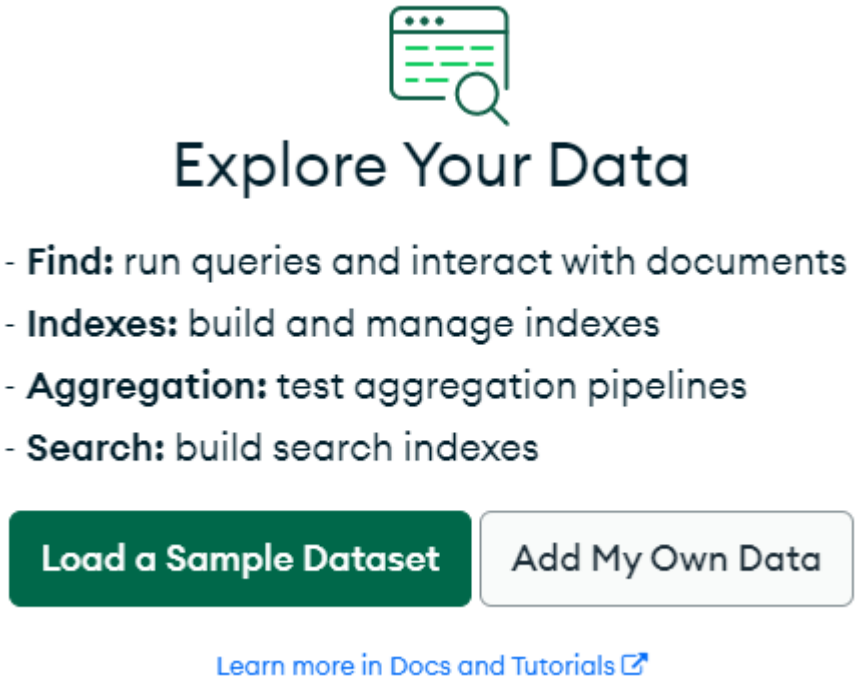


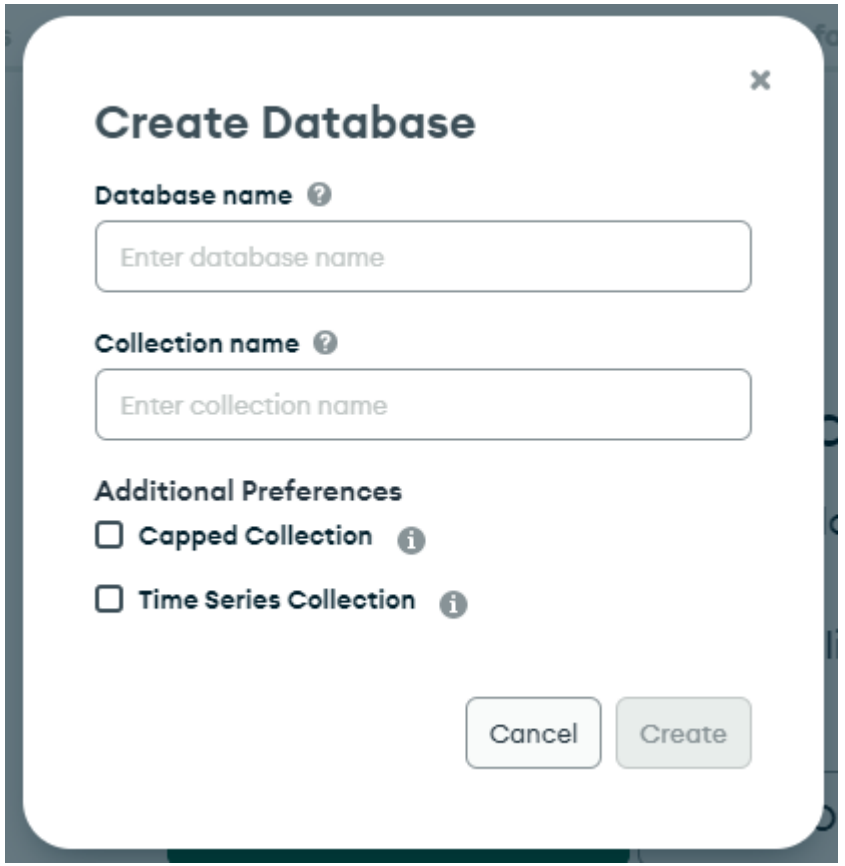
관리자 계정 생성



IP 접근제어 설정

최초 데이터베이스 생성


Explore Your Data
- Find: run queries and interact with documents
- Indexes: build and manage indexes
- Aggregation: test aggregation pipelines
- Search: build search indexes
Load a Sample Dataset
Add My Own Data
Learn more in Docs and Tutorials


Create Database
Database name
Enter database name
Collection name
Enter collection name
Additional Preferences
Capped Collection
Time Series Collection
Cancel
Create

로그인 후 대시보드

- Cluster
- Organization
- Project

- Database
- Collection

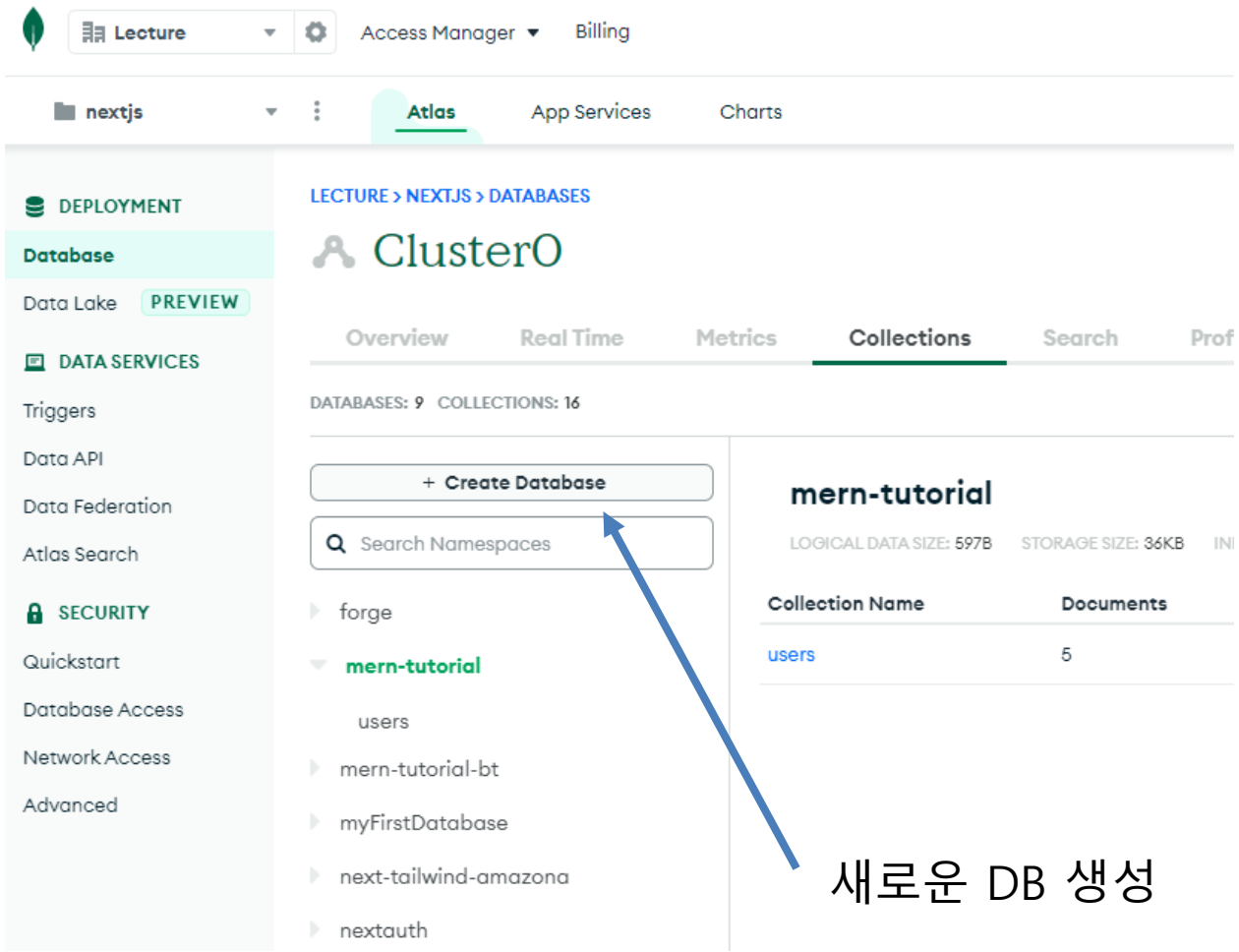
organization

project

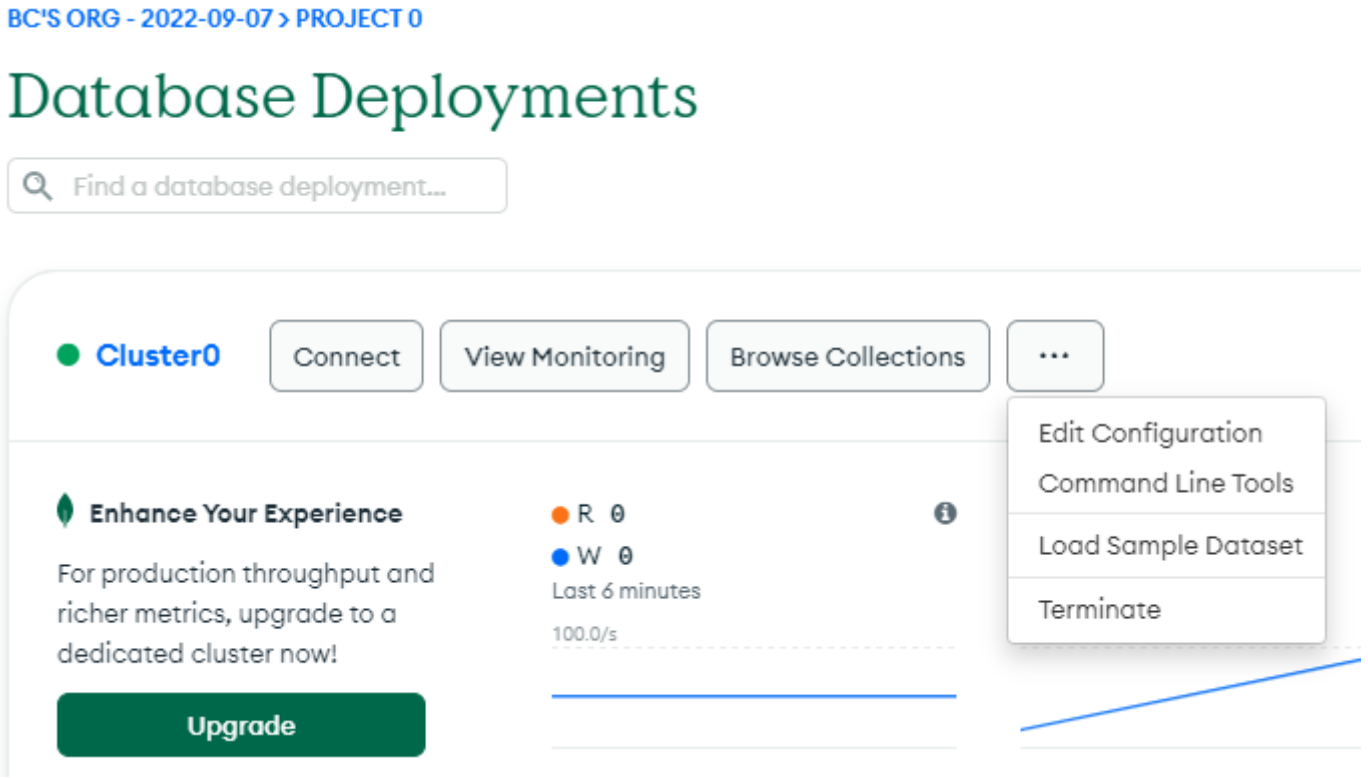
database

계정정보 관리

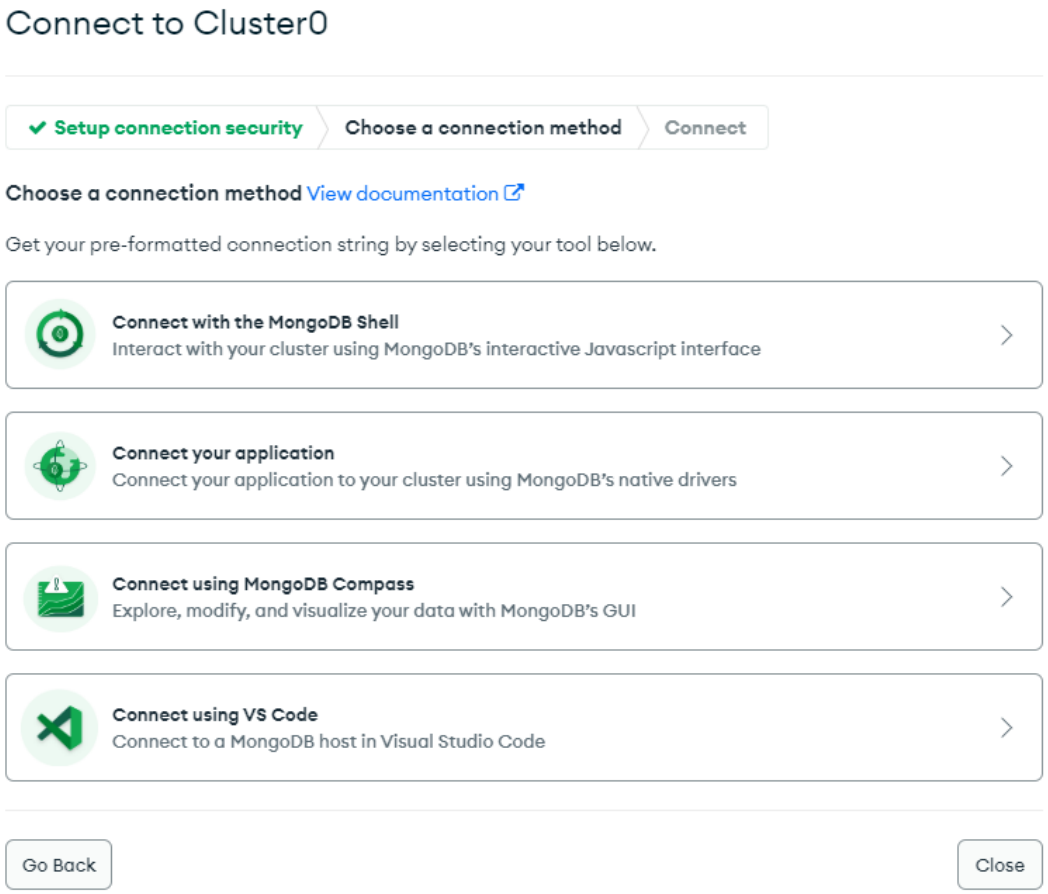
접근제어

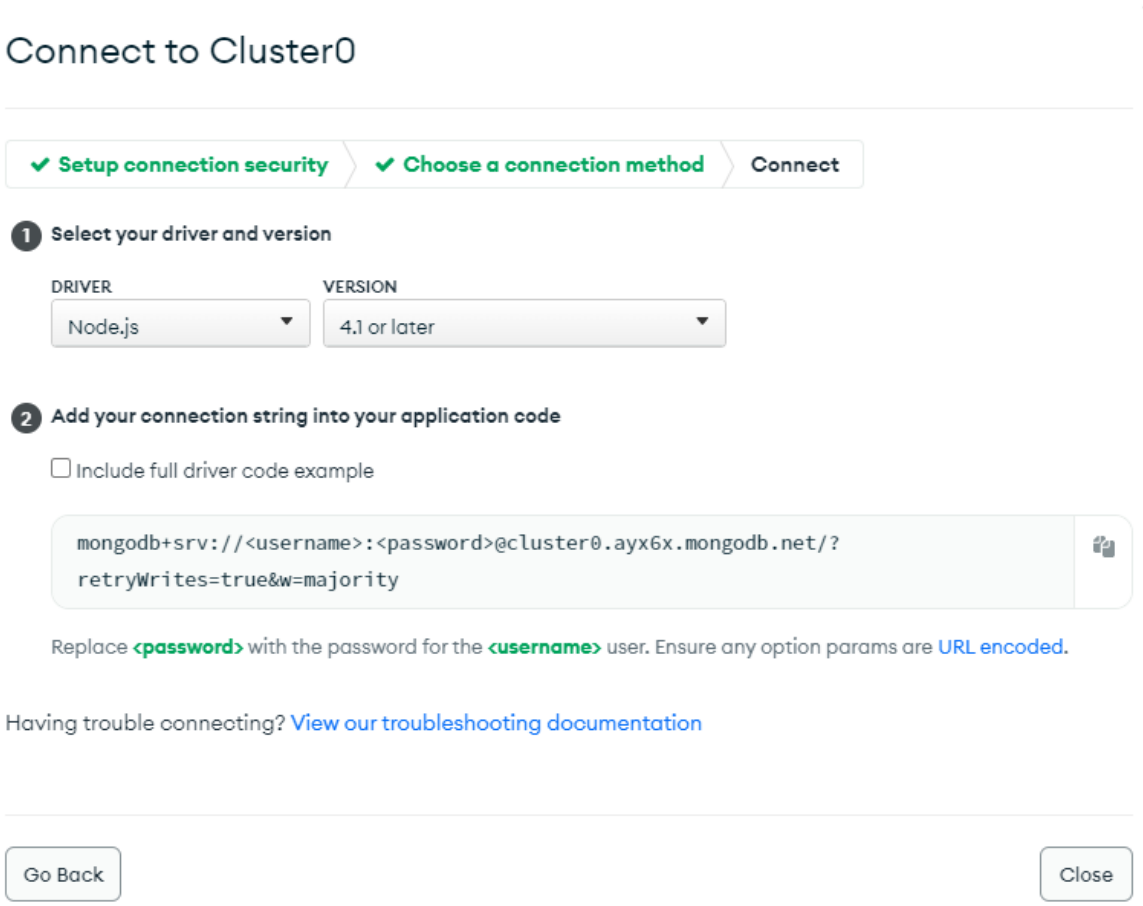


클러스터 환경 설정, 클러스터 삭제



MongoDB와 Application의 연결





자세한 활용법은 예제 프로젝트 개발 진행시 설명...

Tailwind CSS 활용

- Tailwind CSS
  - https://tailwindcss.com/
  - A utility-first CSS framework
  - 클래스 이름을 지정하기만 하면 디자인이 바로 적용됨
  - 사용 매뉴얼 https://tailwindcss.com/docs/installation

Tailwind CSS

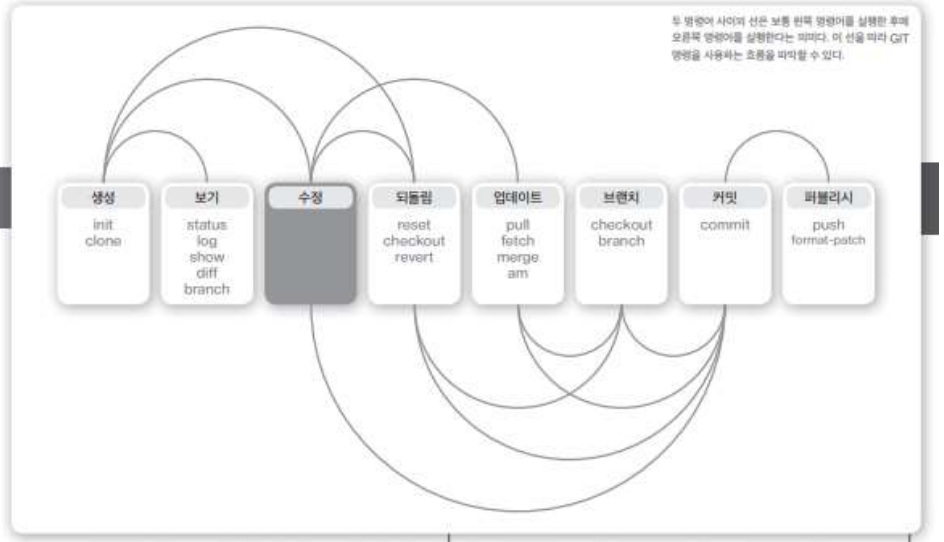
- **Install Tailwind CSS with Next.js**
  - https://tailwindcss.com/docs/guides/nextjs
- **Install Tailwind CSS with Create React App**
  - https://tailwindcss.com/docs/guides/create-react-app

Git and Github

- Git
  - source 관리를 위한 분산 버전 관리 시스템
  - 최초로 리눅스 토발즈가 리눅스 커널 개발에 이용하려고 개발
  - 프로그램 다운로드 및 설치 https://git-scm.com/downloads
  - 현재 버전: 2.37.3 (2022.8.30)
  - Git GUI clients

GIT 작업 흐름과 명령어

도움말: git "명령어" --help
Git의 글로벌 설정은 $HOME/.gitconfig에 저장 (git config --help)



생성

새 저장소 생성하기
```
cd ~/projects/myproject
git init
git add .
```

기존 저장소 Clone하기
```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone you@host.org/project.git
```

보기

워킹 디렉터리의 파일 상태 보기
```
git status
```

파일의 변경사항 보기
```
git diff
```

$ID1과 $ID2 사이의 변경사항 보기
```
git diff $id1 $id2
```

커밋 히스토리 보기
```
git log
```

특정 파일의 커밋 히스토리별 변경사항 보기
```
git log -p $file $dir/ec/tory/
```

특정 파일을 누가 언제 고쳤는지 보기
```
git blame $file
```

$ID 커밋 보기
```
git show id
```

$ID 버전의 파일 보기
```
git show $id:$file
```

로컬 브랜치들 보기
```
git branch
```
(*' 표시는 현재 브랜치를 나타냄)

범례

$id - 커밋 ID, 브랜치 이름, 태그 이름을 나타냄
$file - 파일 이름
$branch - 브랜치 이름

개념

Git 기본

master : 기본 브랜치
origin : 기본 리모트 저장소
HEAD : 현재 브랜치
HEAD^ : HEAD의 부모
HEAD~4 : HEAD의 부모의 부모의 부모의 부모

되돌림

마지막 커밋 시점으로 되돌리기
```
git reset --hard
```
▲Hard Reset은 되돌릴 수 없음

마지막 커밋 내용을 되돌리고 커밋하기
```
git revert HEAD
```
새로운 커밋 생성

특정 커밋 내용을 되돌리고 커밋하기
```
git revert $id
```
새로운 커밋 생성

마지막 커밋 수정하기
```
git commit -a --amend
```
(잘못 커밋해서 수정하고 싶을 때)

$id 시점의 파일을 꺼내기
```
git checkout $id $file
```

브랜치

브랜치를 Checkout하기
```
git checkout $id
```

$branch1을 $branch2에 Merge하기
```
git checkout $branch2
git merge $branch1
```

새 브랜치 만들기
```
git branch $branch
```

$other와 같은 커밋을 가리키는 브랜치를 새로 만들고 바로 Checkout하기
```
git checkout -b $new_branch $other
```

$branch를 삭제하기
```
git branch -d $branch
```

업데이트

origin에서 최신 데이터를 가져오기
```
git fetch
```
(Merge하지는 않음)

origin에서 최신 데이터를 가져와 Merge하기
```
git pull
```
(Fetch하고 Merge까지 함)

누군가 보낸 패치를 Merge하기
```
git am -3 patch-mbox
```
(충돌이 발생하면 해결 후 git am --resolved)

퍼블리시

현재 모든 수정사항을 커밋하기
```
git commit -a
```

다른 개발자에게 보낼 패치를 작성하기
```
git format-patch origin
```

origin으로 업데이트를 Push하기
```
git push
```

버전이나 마일스톤을 생성하기
```
git tag v1.0
```

유용한 명령어

문제가 발생한 커밋 이진탐색하기
```
git bisect start   (이진탐색 시작)
git bisect good $id   ($id를 문제 없는 상태로 표시)
git bisect bad $id   ($id를 문제 있는 상태로 표시)

git bisect bad/good   (현재 상태가 문제가 있는지 없는지 설정)
git bisect visualize   (gitk를 실행하여 확인함)
git bisect reset   (시작했던 상태로 Checkout 함)
```

저장소의 무결성 검사 및 저장소 청소하기
```
git fsck
git gc --prune
```

워킹 디렉터리에서 'foo()'라는 문자열 검색하기
```
git grep "foo()"
```

Merge 충돌 해결

Merge 충돌 내용 보기
```
git diff   (충돌 내용 전체 보기)
git diff --base $file   (Merge Base를 기준으로)
git diff --ours $file   (현 브랜치를 기준으로)
git diff --theirs $file   (Merge할 브랜치를 기준으로)
```

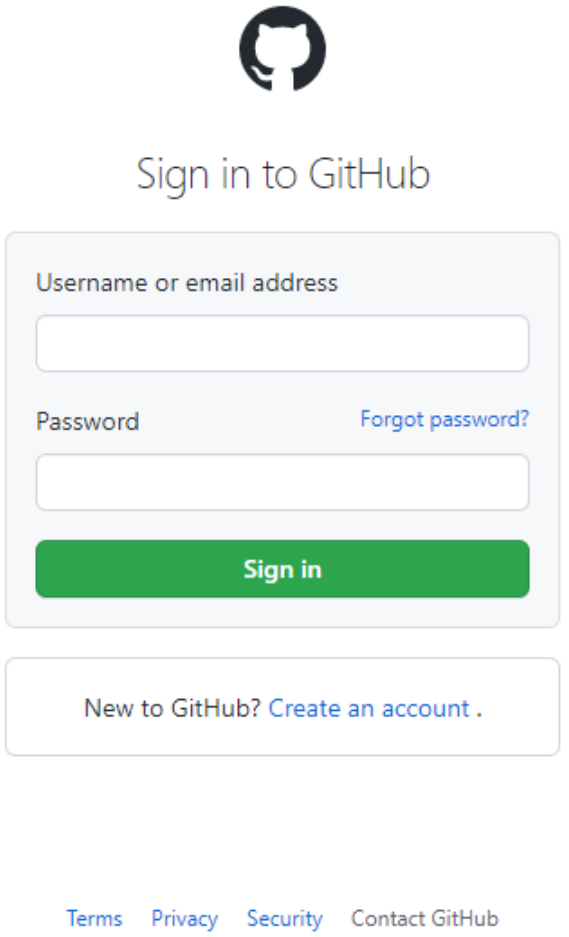
충돌이 생기는 패치 버리기
```
git reset --hard
git rebase --skip
```

충돌 해결 후 Merge 진행하기
```
git add $conflicting_file   (충돌 해결한 파일)
git rebase --continue
```

Git 명령어 익히기

Github

- Github
  - 소스 관리 포털 서비스
  - https://github.com/
  - 회원가입 (sign up)
  - 로그인 (sign in)



Git 명령어 흐름

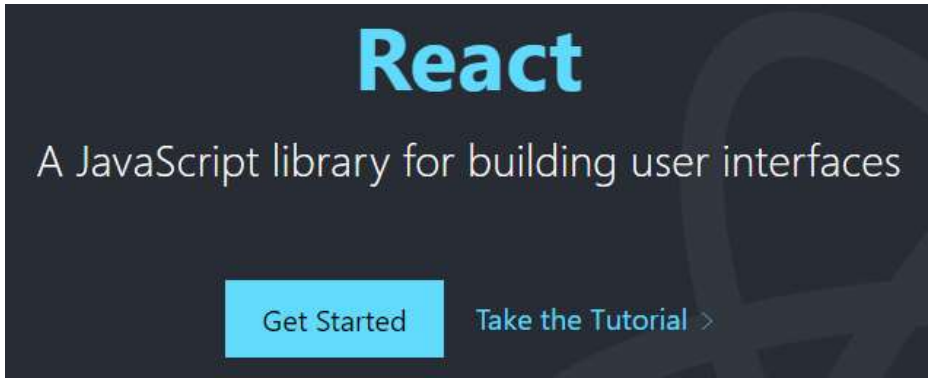
- 0. 프로젝트 폴더로 이동
- 1. 저장소 초기화
  - > git init
- 2. 현재 폴더의 모든 파일들을 git 저장소에 추가
  - > git add .
  - .gitignore에 선언된 폴더, 파일들은 제외
- 3. 커밋, 전송 준비
  - >git commit -am ‘first’
  - 구별할 수 있는 메시지 추가
- 4. 원격 저장소 지정
  - > git remote
- 5. 푸시, 전송
  - > git push, 실제 코드를 github로 업로드

2.2 Next.js 소개

- Angular vs. React vs. Vue

React vs. Next.js

React



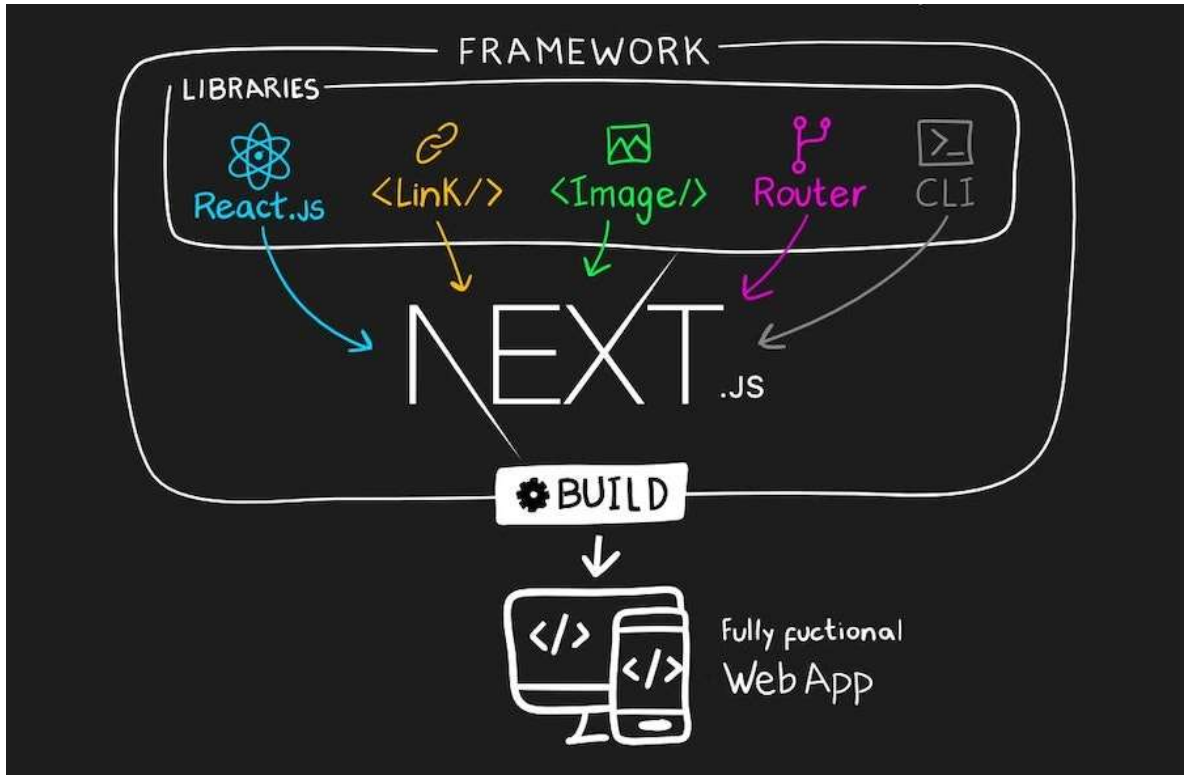
프런트엔드 UI library. 뷰 기능만 제공.
많은 기술들과 결합하여 사용해야 함.
React 개발은 복잡!!

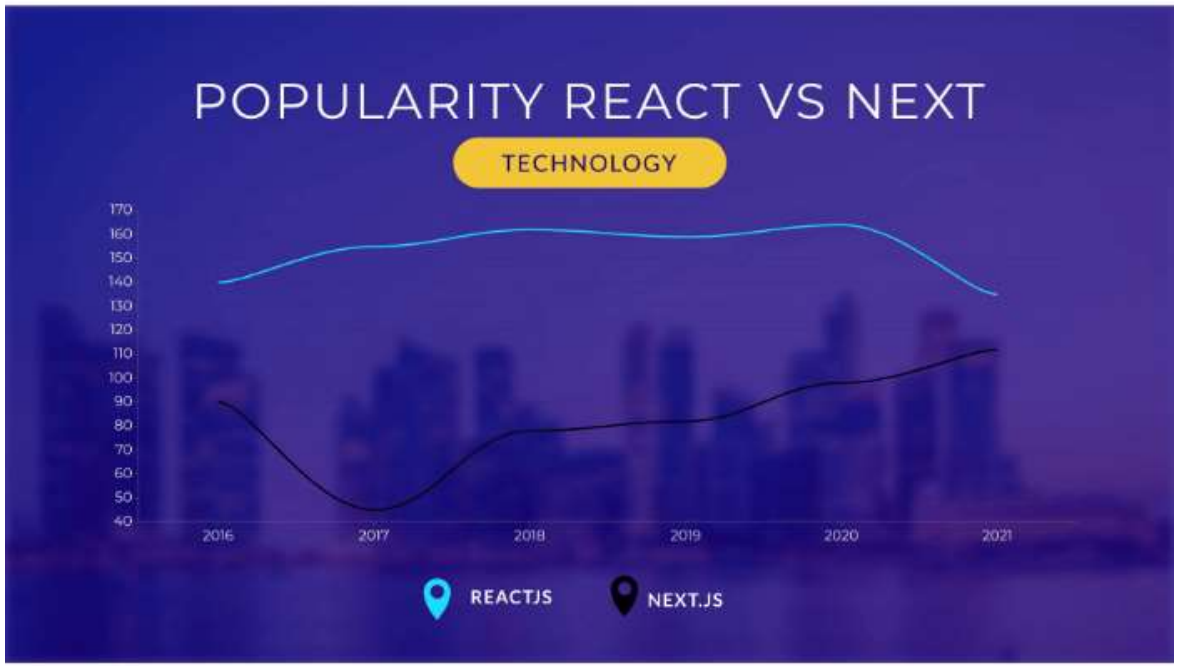
Next.js

React 기술에 기반한
풀스택 웹개발 프레임워크

Next.js

- React vs. Next.js





Next.js의 다양한 기능들

The Web SDK

Next.js has all the tools you need to make the Web. Faster.

Image Optimization

<Image> and Automatic Image Optimization with instant builds.

Documentation →

Internationalization

Built-in Domain & Subdomain Routing and Automatic Language detection.

Documentation →

Next.js Analytics

A true lighthouse score based on real visitor data & page-by-page insights

Documentation →

Zero Config

Automatic compilation and bundling. Optimized for production from the start.

Documentation →

Hybrid: SSG and SSR

Pre-render pages at build time (SSG) or request time (SSR) in a single project.

Documentation →

Incremental Static Regeneration

Add and update statically pre-rendered pages incrementally after build time.

Documentation →

TypeScript Support

Automatic TypeScript configuration and compilation.

Documentation →

Fast Refresh

Fast, reliable live-editing experience, as proven at Facebook scale.

Documentation →

File-system Routing

Every component in the `pages` directory becomes a route.

Documentation →

API Routes

Optionally create API endpoints to provide backend functionality.

Documentation →

Built-in CSS Support

Create component-level styles with CSS modules. Built-in Sass support.

Documentation →

Middleware

Dynamic routing defined by code instead of configuration.

Documentation →

주요 기능

- File-based routing
  - Pages 폴더에 파일을 작성하면 자동 라우팅
- Pre-rendering
  - 서버가 HTML 파일로 만들어서 제공, SEO(검색엔진 최적화)에 장점
- API routes
  - 백엔드 API를 제공할 수 있음. 풀스택 웹서비스 가능.
- Support for CSS modules
  - 여러가지 CSS 모듈 사용 가능
- Authentication
  - 인증 기능 기본 제공
- Development, Production build system
  - Build시 HTML 작성하여 이것으로 서비스

프로젝트 시작

- 프로젝트명 결정
  - 예: was
- 새로운 프로젝트 시작
  - > npx create-next-app@latest or
  - > yarn create next-app or
  - > pnpm create next-app
- 개발 서버 시작
  - > npm run dev
  - > yarn dev
  - > pnpm dev

```
npx create-next-app@latest
# or
yarn create next-app
# or
pnpm create next-app
```

Package.json

```
{
  "name": "was",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "next": "12.2.5",
    "react": "18.2.0",
    "react-dom": "18.2.0"
  },
  "devDependencies": {
    "eslint": "8.22.0",
    "eslint-config-next": "12.2.5"
  }
}
```

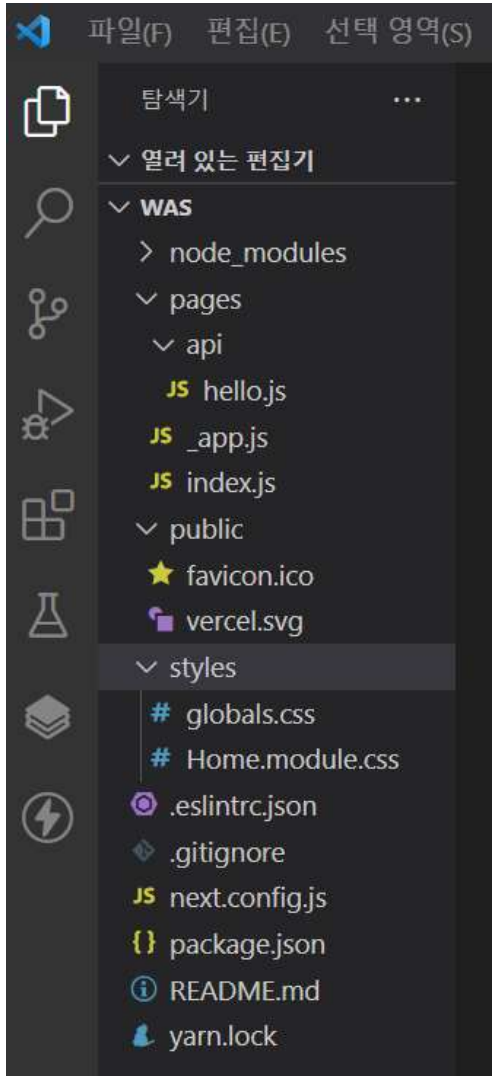
> npm run dev
> npm run build
> npm run start

설치된 패키지

개발시에만 사용되는 패키지

프로젝트 폴더 구성

- Pages
  - api 폴더는 백엔드 폴더
  - 나머지는 frontend
- Public
  - 외부에서 접근, 사용 가능한 폴더
  - 이미지 등 정적인 컨텐츠 관리에 사용
- Styles
  - CSS 스타일 파일
    - Global style
    - Style module



File-based routing

- File-based routing
  - Next.js는 파일 기반 라우팅 기능을 기본 제공함
  - pages 폴더 내에 *.js 파일을 작성하면 파일 라우팅 기능이 자동 적용됨 (react에서는 라우팅을 설정해야 적용됨)
  - pages는 약속된 폴더명이므로 변경하면 안됨
  - 새로운 페이지 생성: about.js, profile.js, …

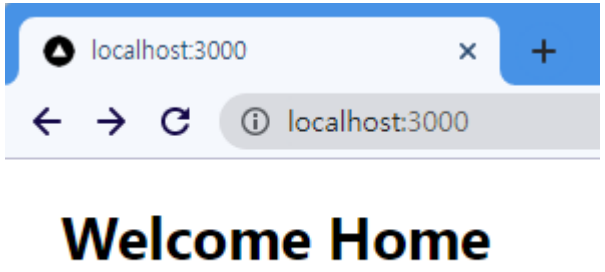
File-based routing

- 라우팅 방식
  - Route with pages: 페이지.js
  - Nested routes: 폴더/폴더/폴더/페이지.js
  - Dynamic routes: [id].js
  - Nested dynamic route: [id]/index.js
  - Catch-all routes: [[…params]].js

File-based routing

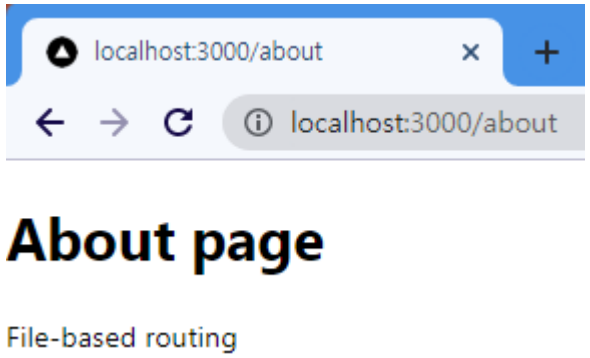
pages/index.js

```
import styles from '../styles/Home.module.css'

export default function Home() {
  return (
    <div className={styles.container}>
      <h1>Welcome Home</h1>
    </div>
  )
}
```



pages/about.js

```
export default function About() {
  return (
    <div>
      <h1>About page </h1>
      <p>File-based routing</p>
    </div>
  )
}
```



React Component의 형식

```
// rfc = reactFunctionalComponent
export default function about() {
  return (
    <div>about</div>
  )
}
```

```
// rafc = reactArrowFunctionalComponent
export const about = () => {
  return (
    <div>about</div>
  )
}
```

```
// rfce
function about() {
  return (
    <div>about</div>
  )
}
export default about
```

```
// rafce
const about = () => {
  return (
    <div>about</div>
  )
}
export default about
```

export default 를 해야 외부에서 Component 사용할 수 있음

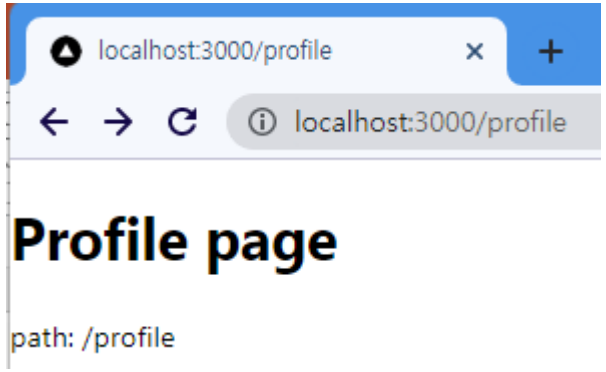
Dynamic Routing

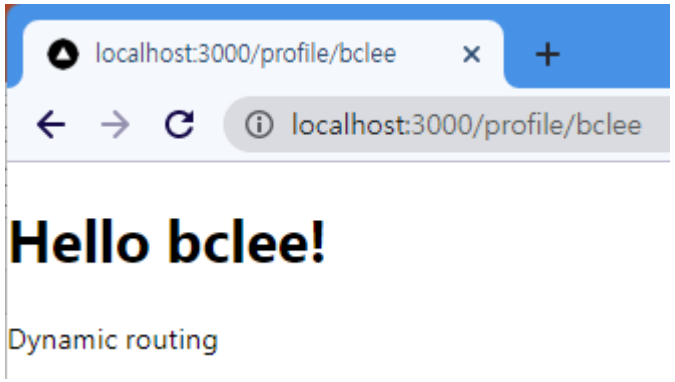
pages/profile/index.js

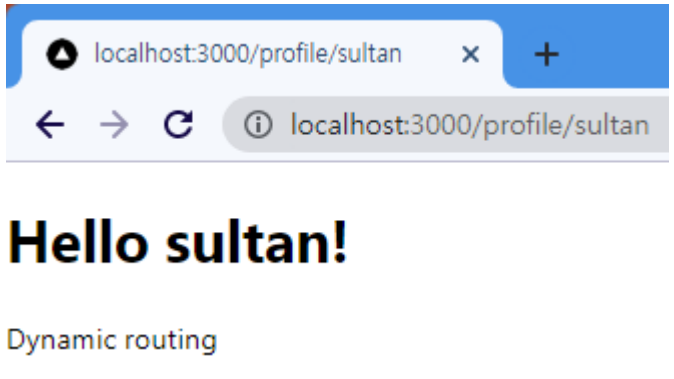
```
export default function ProfileDefault() {
  return (
    <div>
      <h1>Profile page</h1>
      <p> path: /profile </p>
    </div>
  )
}
```

pages/profile/[username].js

```
import { useRouter } from 'next/router'
export default function Profile() {
  const router = useRouter()
  const { username } = router.query
  return (
    <div>
      <h1>Hello {username}!</h1>
      <p> Dynamic routing </p>
    </div>
  )
}
```



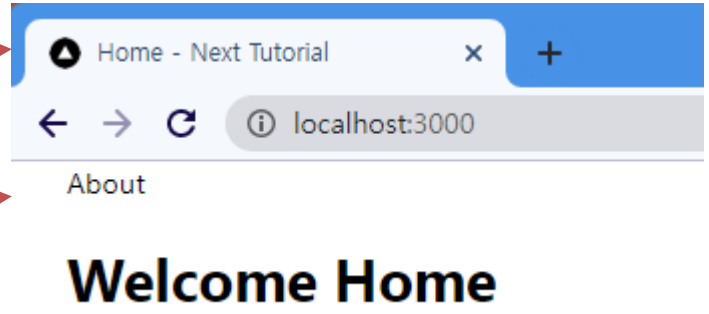




Head, Link

pages/index.js

```
import Head from 'next/head'
import Link from 'next/link'
import styles from '../styles/Home.module.css'

export default function Home() {
  return (
    <div className={styles.container}>
      <Head>
        <title> Home - Next Tutorial</title>
      </Head>
      <Link href="about">About</Link>
      <h1>Welcome Home</h1>
    </div>
  )
}
```



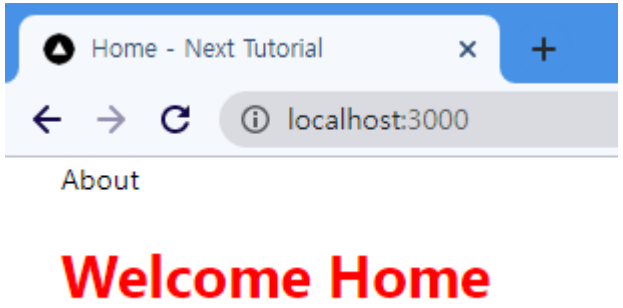
Style

pages/index.js

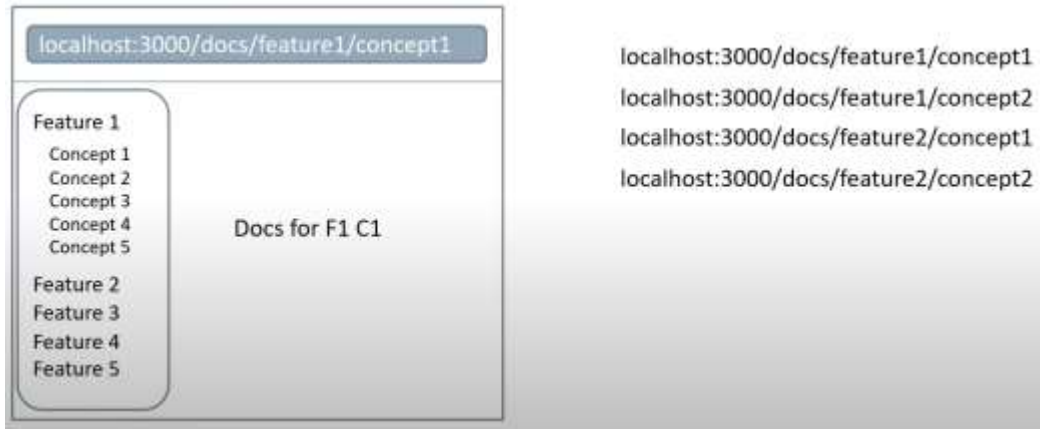
```
import Head from 'next/head'
import Link from 'next/link'
import styles from '../styles/Home.module.css'

export default function Home() {
  return (
    <div className={styles.container}>
      <Head>
        <title> Home - Next Tutorial</title>
      </Head>
      <Link href="about">About</Link>
      <h1 className={styles.homeTitle}>Welcome Home</h1>
    </div>
  )
}
```

styles/Home.module.css

```
.container {
  padding: 0 2rem;
}

.homeTitle {
  color: red;
}
```



Catch-all routing



Catch-all routes: [[...params]].js

pages/docs/[[...params]].js

```
import { useRouter } from 'next/router'

export default function Doc() {
  const router = useRouter()
  const { params = [] } = router.query
  console.log(params)

  if (params.length === 2) {
    return (
      <h1>
        Viewing docs for feature {params[0]} and concept {params[1]}
      </h1>
    )
  } else if (params.length === 1) {
    return <h1>Viewing docs for feature {params[0]}</h1>
  }

  return <h1>Docs Home Page</h1>
}
```

Menu 만들기

components/Navbar.js

```
import Link from 'next/link'
import styles from '../styles/Navbar.module.css'

export default function Navbar() {
  return (
    <div className={styles.navbar}>
      <div className={styles.links}>
        <Link href="/">
          <a>Home</a>
        </Link>
        <Link href="/about">
          <a>About</a>
        </Link>
        <Link href="/profile">
          <a>Profile</a>
        </Link>
      </div>
    </div>
  )
}
```

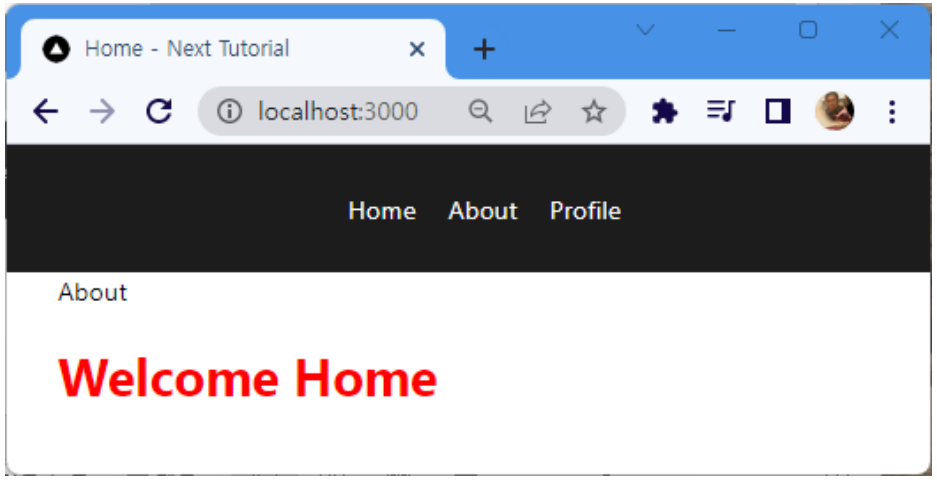
styles/Navbar.module.css

```
.navbar {
  width: 100%;
  height: 80px;
  background-color: rgb(28, 28, 28);
  display: flex;
  justify-content: center;
}

.links {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 100%;
  color: white;
}

.links a {
  margin-left: 20px;
}
```

Menu 만들기

components/Layout.js

```
import Navbar from './Navbar'

export default function Layout({ children }) {
  return (
    <>
      <Navbar />
      <div>{children}</div>
    </>
  )
}
```



pages/_app.js

```
import Layout from '../components/Layout'
import '../styles/globals.css'

function MyApp({ Component, pageProps }) {
  return (
    <Layout>
      <Component {...pageProps} />
    </Layout>
  )
}

export default MyApp
```

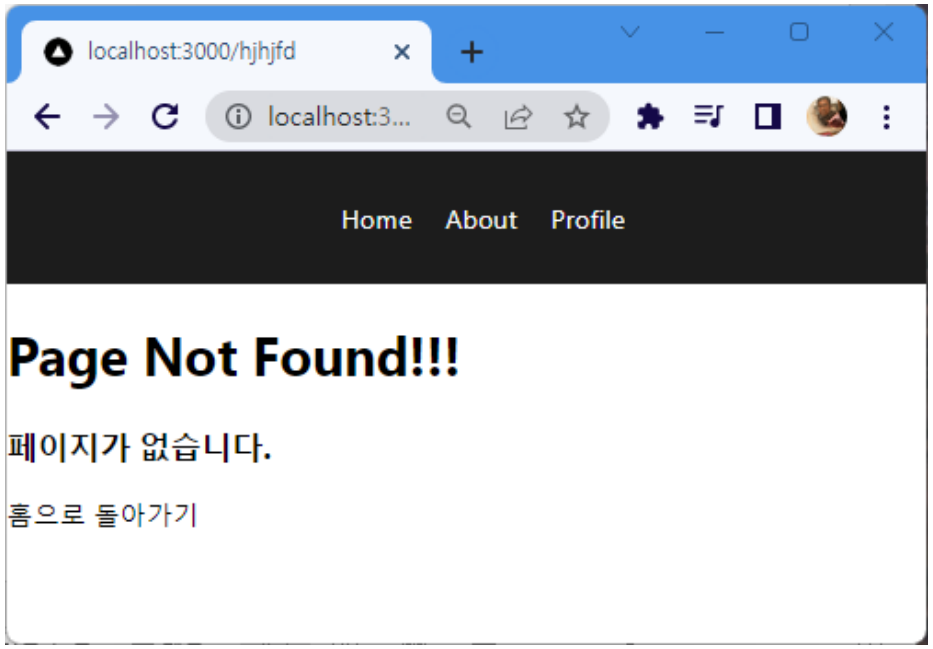
Custom 404 page

pages/404.js

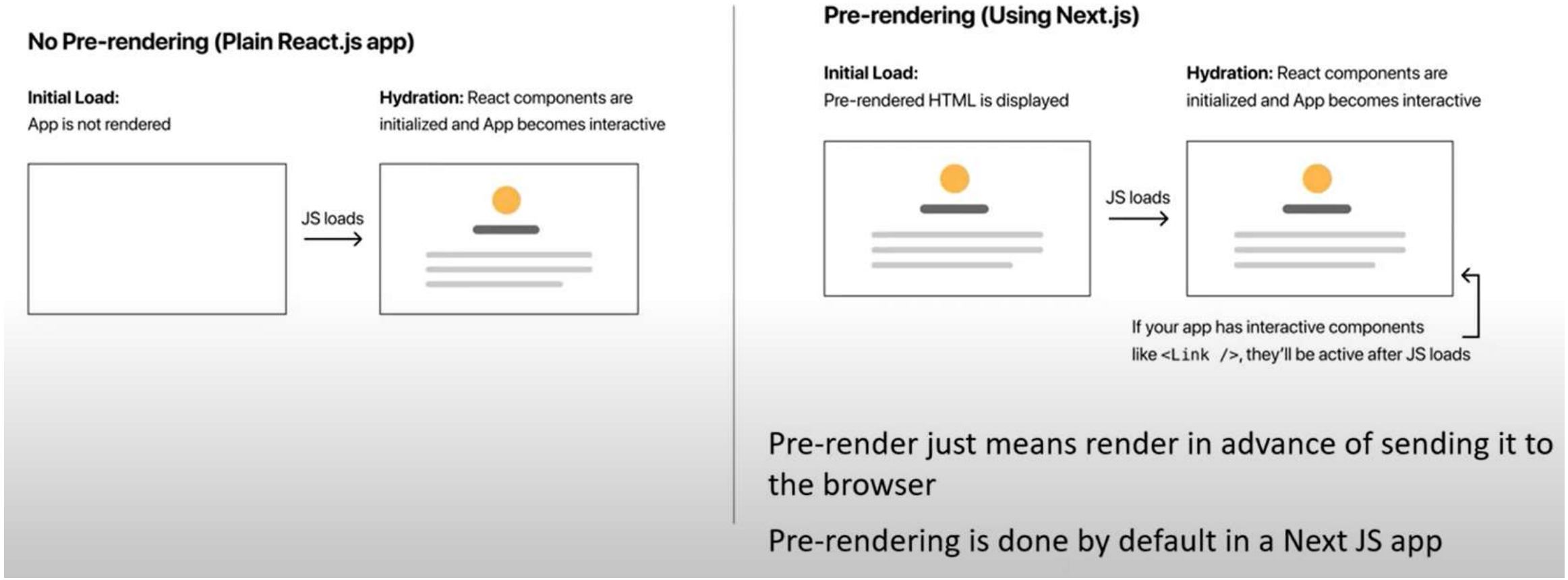
```
import Link from 'next/link'

export default function PageNotFound() {
  return (
    <div>
      <h1> Page Not Found!!!</h1>
      <h3> 페이지가 없습니다. </h3>
      <Link href="/">홈으로 돌아가기 </Link>
    </div>
  )
}
```

디폴트 에러 페이지

404 | This page could not be found.

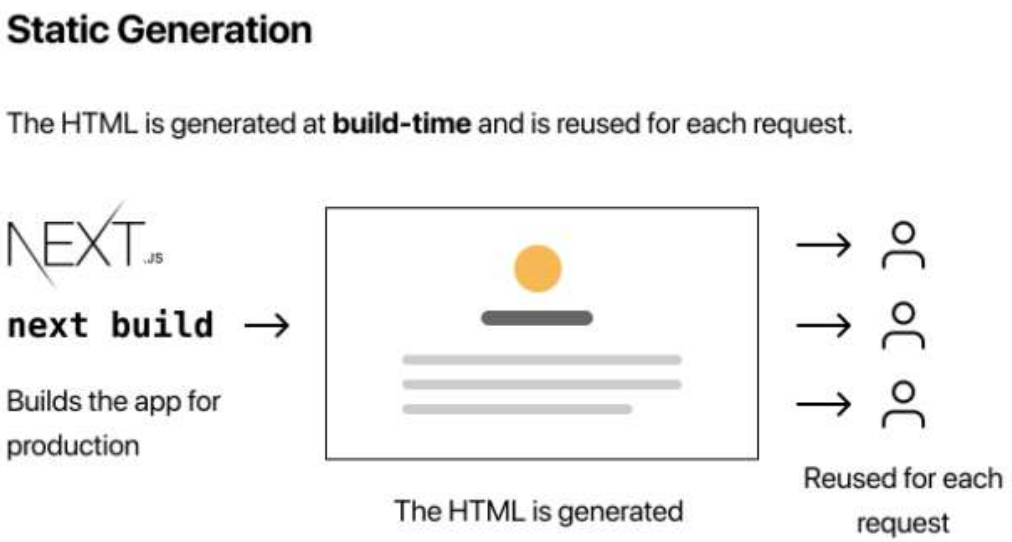


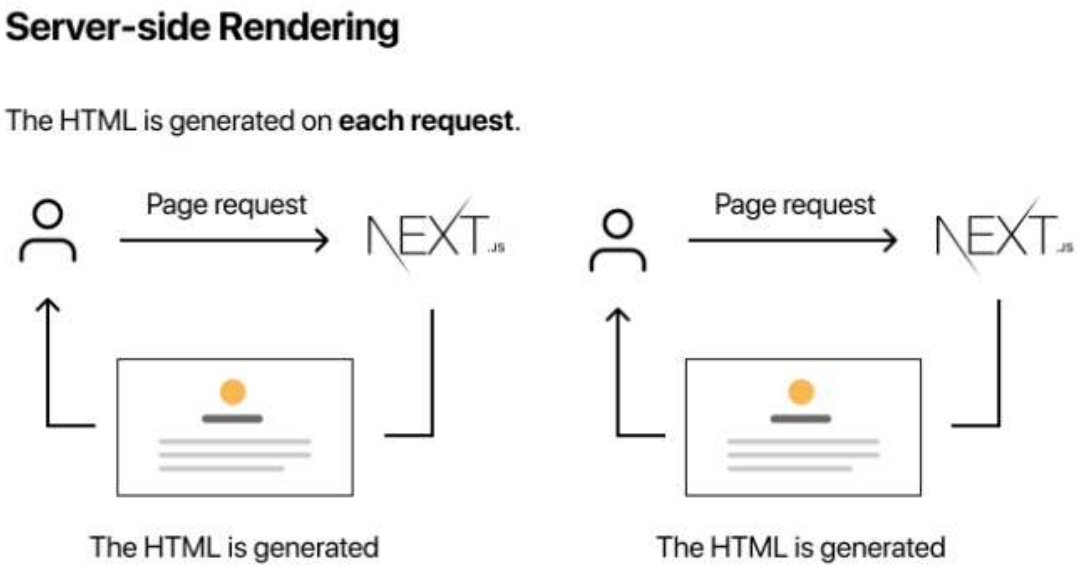
Pre-rendering in Next.js



Next.js는 pre-rendering 방식의 웹서비스 제공
페이지 요청시 서버에서 html을 생성하여 클라이언트에게 제공

SSG vs. SSR





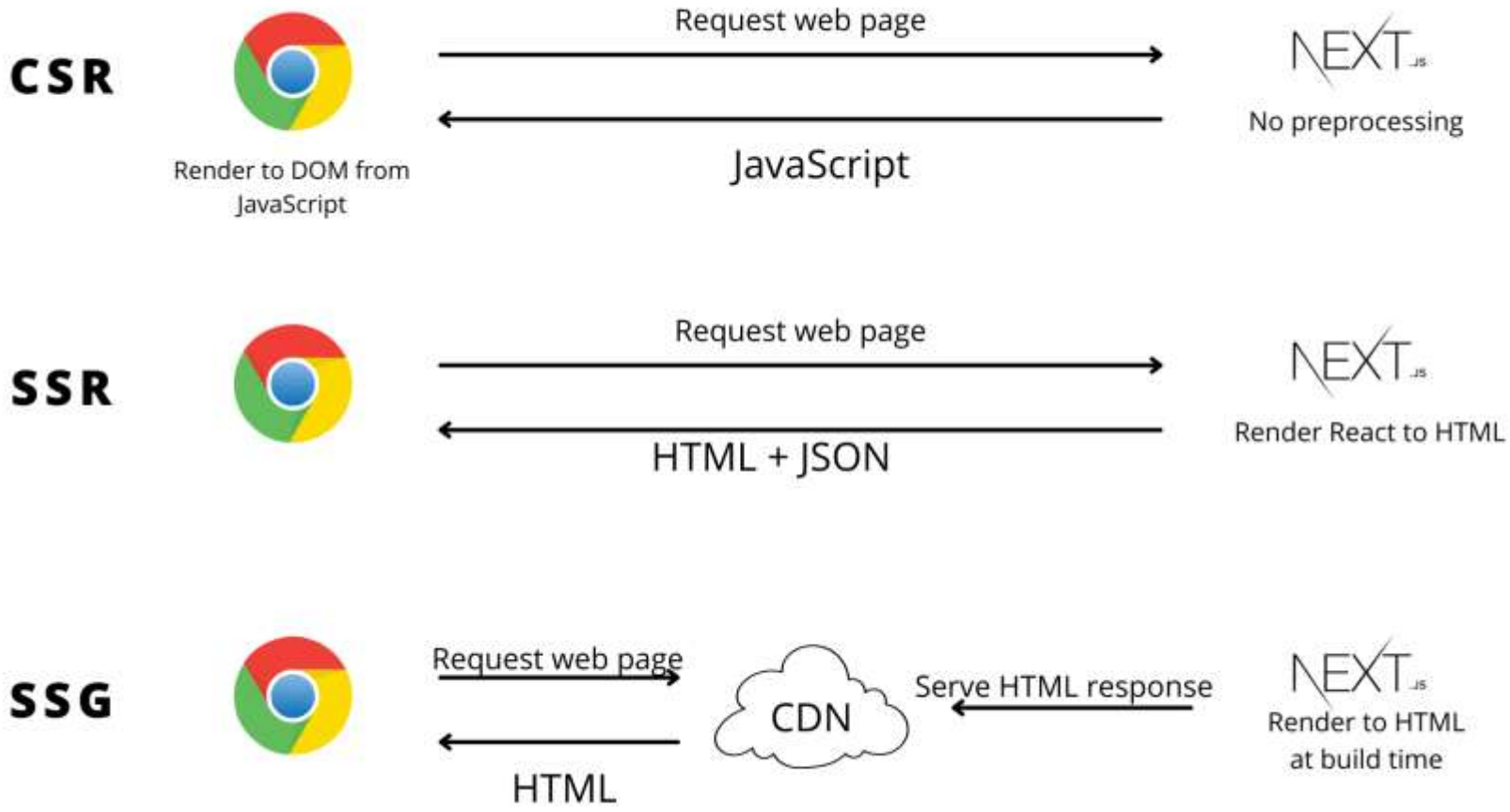
Static Site Generation

서비스 빌드시 HTML 파일을 생성하여
이것으로 서비스 제공
고정된 컨텐츠 서비스에 적합

Server-Side Rendering

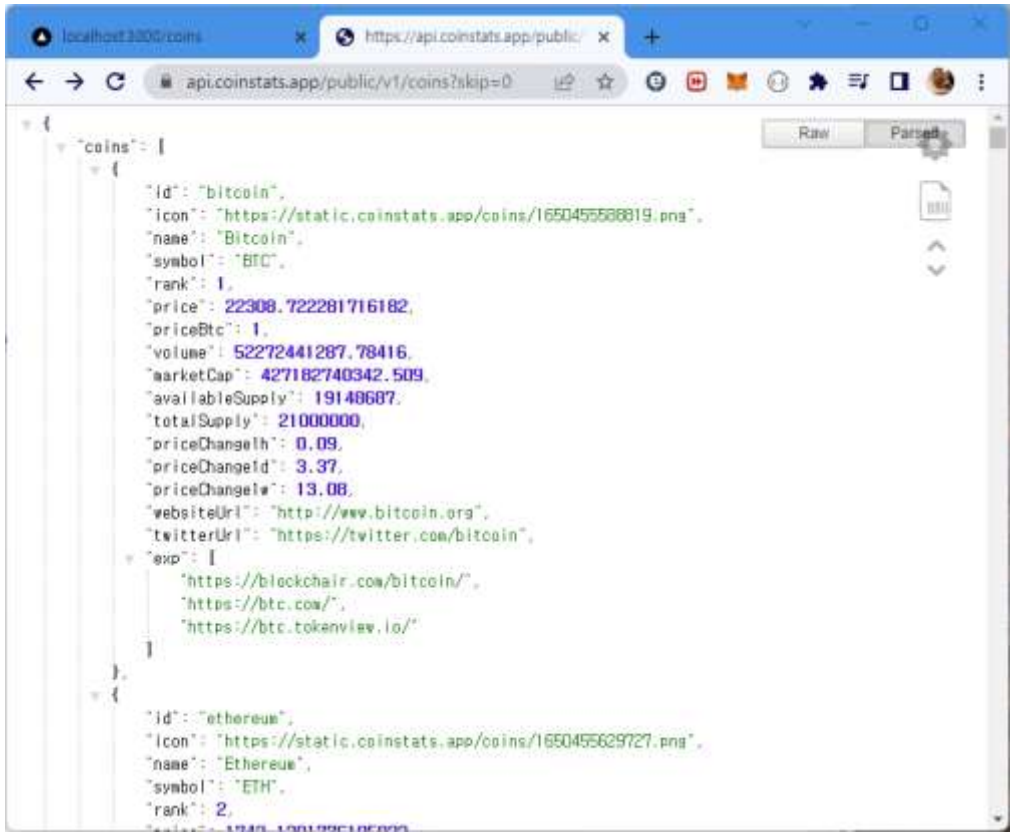
서비스 요청시마다 서버가 HTML을 새로 생성하여
서비스 제공. 시간이 소요됨.
데이터가 자주 바뀌는 경우에 이를 반영 가능

CSR vs. SSR vs. SSG



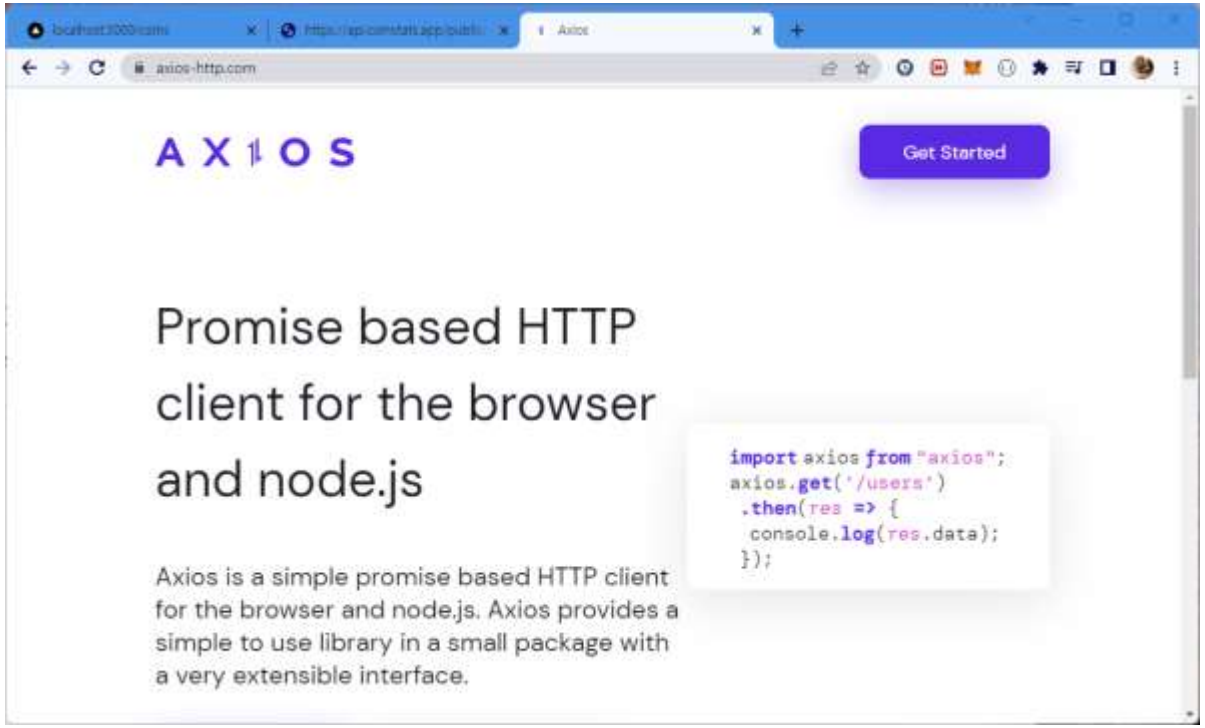
Data Fetching

- 예제: 암호화폐 가격 표시하기
  - https://api.coinstats.app/public/v1/coins?skip=0
  - 공개 API로부터 데이터 읽어와서 표시하기



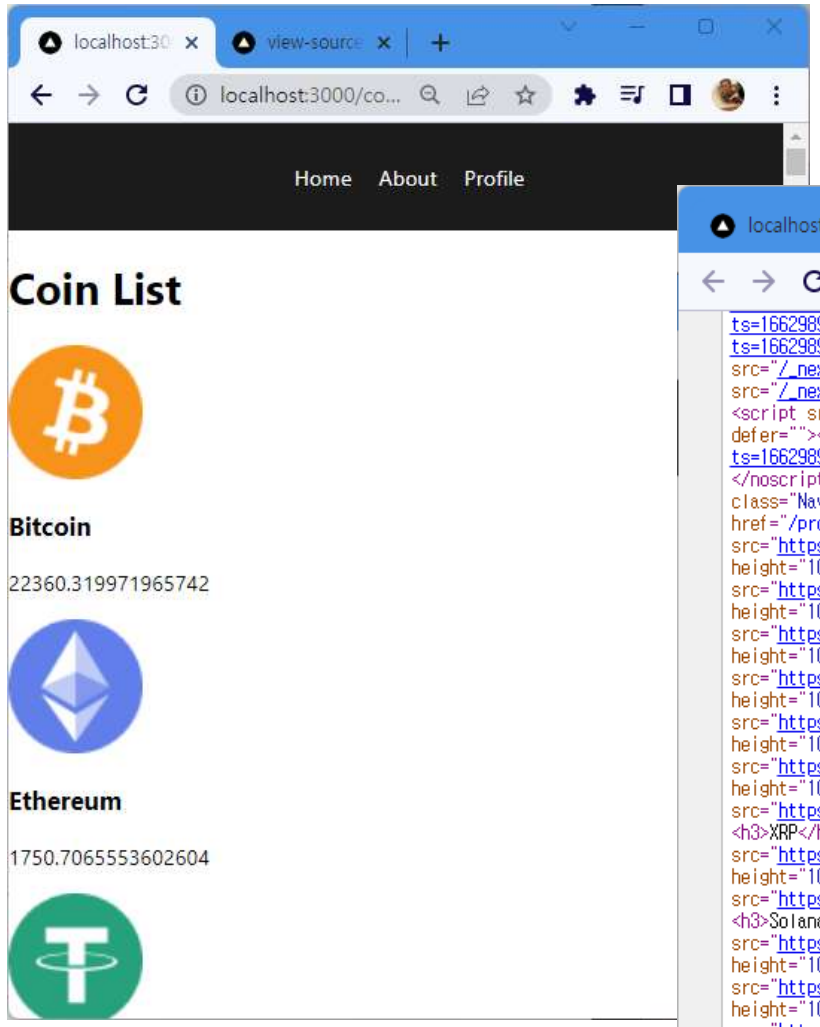
Axios 설치

- Axios 설치하기
  - **Promise based HTTP client for the browser and node.js**
  - > npm install axios



pages/coins/index.js

```
import axios from 'axios'

export default function CoinList({ coinData }) {
  console.log(coinData)
  const coins = coinData.coins
  return (
    <div>
      <h1>Coin List </h1>
      {coins.map((coin) => {
        return (
          // eslint-disable-next-line react/jsx-key
          <div key={coin.id}>
            <img src={coin.icon} width={100} height={100} />
            <h3>{coin.name}</h3>
            <p>{coin.price}</p>
          </div>
        )
      })}
    </div>
  )
}

export const getStaticProps = async () => {
  const result = await axios.get(
    'https://api.coinstats.app/public/v1/coins?skip=0'
  )

  return {
    props: {
      coinData: result.data,
    },
    revalidate: 10,
  }
}
```



getStaticProps 을 이용한 SSG
revalidate 옵션을 이용한 SSG 재생성

컴포넌트 활용

pages/coins/index.js

```
import axios from 'axios'
import Coin from '../../components/Coin'
import styles from '../../styles/Coin.module.css'

export default function CoinList({ coinData }) {
  const coins = coinData.coins
  return (
    <div className={styles.container}>
      <h1>Coin List </h1>
      <div className={styles.coinContainer}>
        {coins.map((coin) => {
          return (
            <div key={coin.id} className={styles.coinItem}>
              <Coin coin={coin} />
            </div>
          )
        })}
      </div>
    </div>
  )
}
...중략...
```

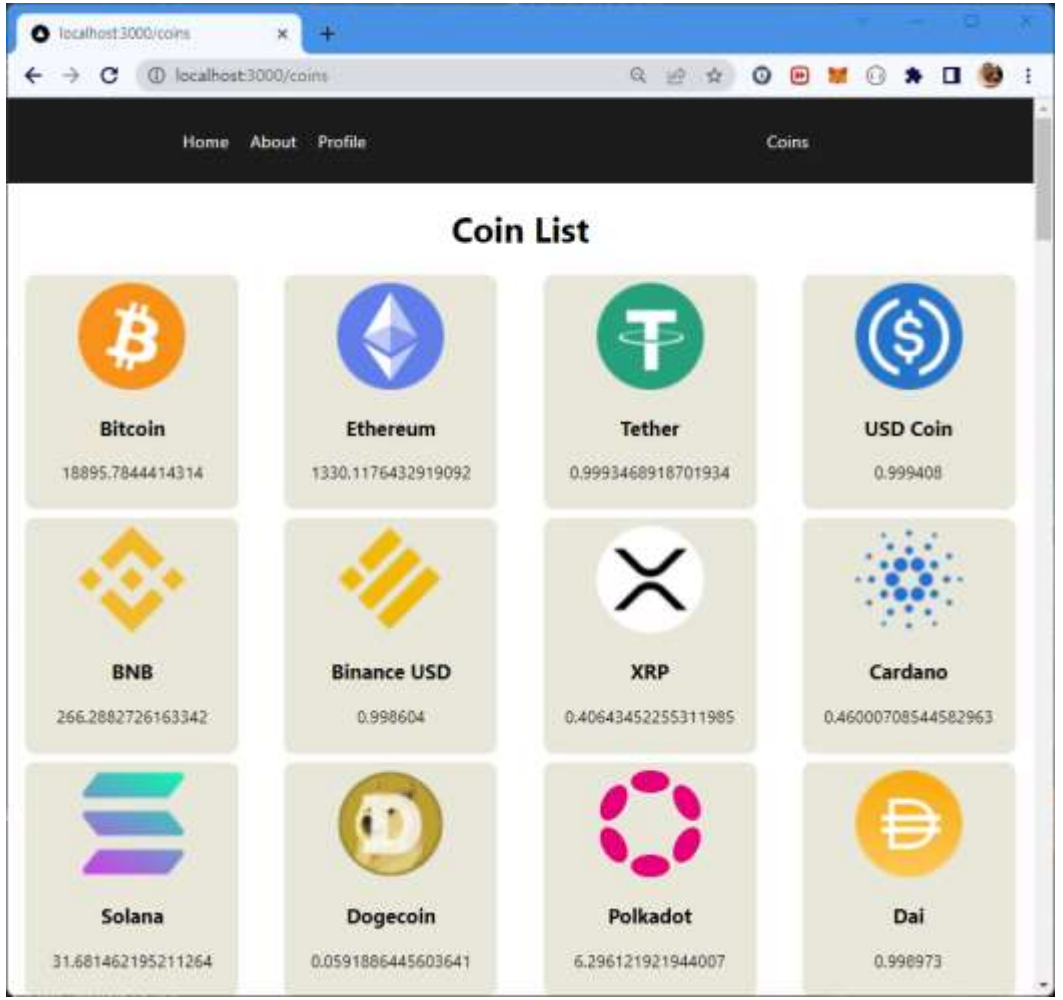
components/Coin.js

```
export default function Coin({ coin }) {
  return (
    <div>
      <img src={coin.icon} width={100} height={100} />
      <h3>{coin.name}</h3>
      <p>{coin.price}</p>
    </div>
  )
}
```

Grid 레이아웃

styles/Coin.module.css

```
.container {
  text-align: center;
}

.coinContainer {
  display: grid;
  grid-template-columns: auto auto auto auto auto;
  justify-content: space-around;
  grid-gap: 0.5rem;
}

.coinItem {
  text-align: center;
  border-radius: 10px;
  width: 200px;
  padding: 0.5rem 1rem;
  background-color: rgb(232, 231, 217);
}
```



요약

- 개발환경 구축
  - VSCode, node.js, mongoDB, Tailwind CSS, Git, Github
- Next.js 기능 소개
  - 프로젝트 생성: npx create-next-app
  - 라우팅: 파일 기반 라우팅, 다이나믹 라우팅, catch-all-routing
  - 스타일 활용
  - 데이터 가져오기: SSG, getStaticProps
  - 공개 API 활용
  - Axios 패키지 사용
  - 컴포넌트: Layout, Navbar