## 4.1 디렉터리 구조 구성

Next.js 에서는 특정 파일과 디렉터리가 지정된 위치에 있어야 한다.

예)

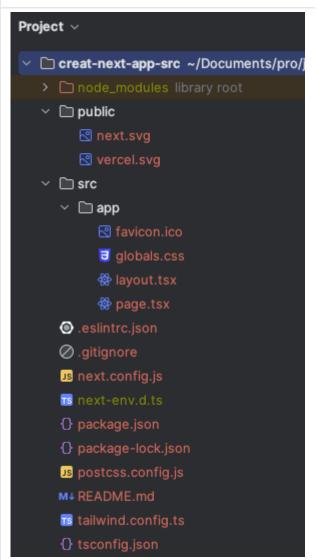
파일 : \_app.js , \_document.js 디렉토리 : pages/ , public/

default	src사용
	Project ∨
	✓ □ create-next-app-src ~/Documents/property.
	> 🗀 node_modules library root
next-js-app - node_modules/ - package.json	∨ □ public
	favicon.ico
	next.svg
	vercel.svg     vercel.svg     vercel.svg
	∨ 🗀 src
	> 🗀 pages
	> 🗀 styles
- pages/	<ul><li>eslintrc.json</li></ul>
- public/ - styles/	
	Js next.config.js
	s next-env.d.ts
	Opackage.json
	Opackage-lock.json
	Js postcss.config.js
	M↓ README.md
	s tailwind.config.ts
	{} tsconfig.json
app router	app router + src

#### default

# Project ~ app globals.css layout.tsx page.tsx ∨ □ public next.svg eslintrc.json .gitignore Js next.config.js next-env.d.ts {} package.json {} package-lock.json Js postcss.config.js

## src사용



## next-js-app

M↓ README.md

{} tsconfig.json

tailwind.config.ts

- node modules/
- package.json
- pages/
- public/
- styles/
- node\_modules/: Next.js 프로젝트의 의존성 패키지를 설치하는 디렉터리
- pages/: 웹 애플리케이션의 페이지 파일을 저장하고 라우팅 시스템을 만드는 디렉터리
- public/: 컴파일된 CSS 및 자바스크립트 파일, 이미지, 아이콘 등의 정적 자원을 저장 하고 제공하는 디렉터리
- styles/: 스타일링 포맷(CSS,SASS,LESS 등)과 관계없이 스타일링 모듈을 저장하는 디렉터리

Next.js에서는 pages/ 디렉터리를 src/ 디렉터리 안으로 옮길 수 있다. 이렇게 옮기면 프로젝트 최상위 디렉터리가 좀 더 간결해진다. 프로젝트 내에 pages/ 와 src/pages 디렉터리가 둘 다 있는 경우 Next.js가 src/pages/ 디렉터리를 무시한다. 최상위에 있는 pages/ 디렉터리의 우선순위가 더 높다.

## 컴포넌트 구성

## 아토믹 디자인 원칙

create-next-app ~	/Documents/pro/ja
Components	
atoms 🗀	
molecules	
organisms 🗀	
🗀 templates	
> 🗀 node_modules	library root
> 🗀 pages	
> 🗀 public	
> 🗀 styles	

atoms

코드에서 사용되는 가장 기본적인 컴포넌트들

molecules

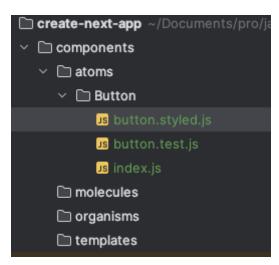
atoms에 속한 컴포넌트 여러 개를 조합하여 좀 더 복잡한 구조를 만드는 컴포넌트들 organisms

molecules 와 atoms를 섞어서 더 복잡한 구조의 컴포넌트를 만든다.

templates

일종의 페이지 스켈레톤으로, 어디에 orfanisms, atoms, molecules를 배치할지 결정해서 사용자가 접근할 수 있는 페이지를 만든다.

Button 컴포넌트 예시

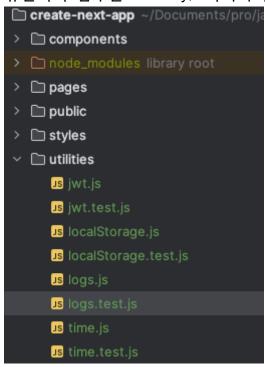


이런식으로 컴포넌트를 구성하면 필요할 때 컴포넌트를 찾고 수정하기 쉽다.

아토믹 디자인 원칙을 따르면 프로젝트 구조를 간결하게 유지하고 유지 보수하기 쉽다는 점에서 추천

#### 유틸리티 구성

유틸리티 스크립트: 컴포넌트를 만들지 않는 코드 파일 유틸리티 함수는 utility/ 디렉터리 아래 저장

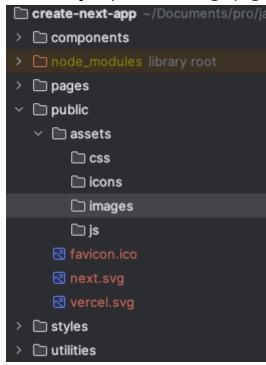


#### 정적 자원 구성

정적 파일은 public/ 디렉터리 아래에 두면 프레임워크가 알아서 해준다. 일반적인 웹 사이트에서는 다음과 같은 정적 자원을 사용한다.

- 이미지
- 컴파일한 자바스크립트 파일
- 컴파일한 CSS 파일

- 아이콘(favicon 및 웹 앱 아이콘)
- mainfest.json, robot.txt 등의 정적 파일



이미지는 내장 Image 컴포넌트를 사용해서 처리하는 것이 좋다.

icons/ 디렉터리는 주로 <mark>웹 앱 매니페스트</mark> 아이콘을 제공할 용도로 사용된다. 웹 앱 매니페스트: JSON파일로, 앱의 이름이나 모바일 기기에 앱을 설치할 때 표시할 아이 콘과 같이 프로그레시브 웹 앱에 관한 유용한 정보를 가지고 있다. 매니페스트 파일은 public/ 디렉터리 아래에 manifest.json 파일을 만들어 추가할 수 있 다.

```
> in components
                               {
                                 "name": "My Next.js App",
                                 "short_name": "Next.js App",
> 🗀 pages
                                 "description": "A test app made with next.js",

∨ □ public

                                 "background_color": "#a600ff",
  > 🗀 assets
                                 "display": "standalone",
   favicon.ico
                                 "theme_color": "#a600ff",
                                 "icons": [
                                  {

    ▼ vercel.svg

                                     "src": "/assets/icons/icon-192.png",
> 🗀 styles
                                     "type": "image/png",
> 🗎 utilities
                                     "sizes": "192x192"
  eslintrc.json
                                   },
  .gitignore
                                     "src": "/assets/icons/icon-512.png",
  Js next.config.js
                                     "type": "image/png",
  next-env.d.ts
                                     "sizes": "512x512"
  {} package.json
  {} package-lock.json
  Js postcss.config.js
                               }
  M↓ README.md
```

## 스타일 파일 구성

CSS-in-JS 프레임워크 : Emotion, styled-components, JSS 등 별도로 스카일 파일을 만든다.

공통 스타일이나 유틸리티 기능을 관리할 필요가 있을 시 Next.js가 기본으로 만들어주는 styles/ 디렉터리를 사용하는 것이 좋다.

[CHAPTER 6 CSS와 내장 스타일링 메서드] 와 [CHAPTER 7 UI 프레임워크] 에서 더 자세히소개

## lib 파일 구성

lib 파일은 서드파티 라이브러리를 감싸는 스크립트를 지칭한다.

## next-js-app

- lib/
  - graphql/
    - index.js
    - queries/
      - query1.js
      - query2.js
    - mutations/
      - mutation1.js
      - mutation2.js

## 4.2 데이터 불러오기

- 정적 페이지를 만들 때 getStaticProps함수를 사용해서 빌드시점에 데이터를 불러오는 경우
- 서버가 페이지를 렌더링할 때 getServerSideProps를 통해 실행 도중 데이터를 불러오
   는 경우

Next.js가 Node.js의 MySQL 클라이언트 라이브러리 등을 이요해서 데이터를 가져오는 것은 안정하지 않기 때문에 ANTI 패턴이다.

데이터베이스에 대한 접근 및 질의는 스프링 등의 백엔드 프레임워크에서 처리하는 것이좋다.

### 서버가 데이터 불러오기

#### 두가지 방법

- 1. Node.js의 내장 HTTP 라이브러리를 사용
  - 1. 별도의 의존성 라이브러리를 설치할 필요 없이 바로 불러와 쓸수 있다.
  - 2. 서드 파티 HTTP 클라이언트와 비교해보았을때 설정하고 처리해할 작업이 더 많으 편이다.
- 2. HTTP 클라이언트 라이브러리를 사용
  - 1. 서버에서 HTTP 요청을 더 쉽게 만들 수 있는 편이다.
  - 2. 유명한 HTTP 클라이언트인 Axios

## 서버에서 REST API 사용하기

가장 많이 사용하는 인증방식: OAuth 2.0, JWT, API 키. 코딩 참조 클라이언트가 데이터 불러오기