[개발환경] NextJS와 React의 파일 구조

구현 어려움 정도	개념
구현 완료	☑
기간	@November 28, 2024
정리 완료	☑

Next.js의 파일 구조

Next.js는 파일 기반 라우팅을 지원하므로 `/app` 또는 `/pages` 디렉토리 내 파일 이름에 따라 URL 경로가 자동 생성됩니다.

```
/pages
  ├── index.tsx          // 홈 페이지 ("/")
  ├── about.tsx          // "/about"
  ├── contact.tsx        // "/contact"
  ├── blog/
  |   ├── index.tsx      // "/blog"
  |   ├── [slug].tsx     // 동적 라우트 "/blog/:slug"
  └── api/
      └── hello.ts       // API 라우트 "/api/hello"

또는

/app (App Router 사용시)
  ├── layout.tsx         // 공통 레이아웃
  ├── page.tsx           // 홈 페이지 ("/")
  ├── about/
  |   ├── page.tsx       // "/about"
  |   └── layout.tsx     // About 전용 레이아웃
  ├── contact/
  |   └── page.tsx       // "/contact"
  ├── blog/
  |   ├── page.tsx       // "/blog"
  |   └── [slug]/
  |       └── page.tsx   // "/blog/:slug"
  └── api/
      └── hello/route.ts // API 라우트 "/api/hello"
```

특징

- **파일 기반 라우팅**: 디렉토리와 파일 이름이 URL로 자동 매핑됩니다.
- **API 라우트 지원**: `/pages/api` 또는 `/app/api`에 백엔드용 API를 작성 가능.
- **동적 라우팅**: 대괄호(`[]`)를 사용해 동적 경로를 정의합니다.
  - 예: `[slug].tsx` → `/blog/:slug`
- **중첩 레이아웃**: `layout.tsx`를 사용해 계층적 레이아웃 구성. (App Router)
- **페이지 단위 분리**: 모든 라우트는 `page.tsx` 파일로 정의됩니다. (App Router)

React의 파일 구조

React는 라우팅 라이브러리(예: React Router)를 추가로 사용해 개발자가 직접 구조를 설계해야 합니다.

```
/src
  ├── components/        // 재사용 가능한 컴포넌트
  |   ├── Header.tsx
```

```
│   ├── Footer.tsx
│   └── Navbar.tsx
├── pages/             // 페이지 컴포넌트
│   ├── Home.tsx       // 홈 페이지 ("/")
│   ├── About.tsx      // "/about"
│   └── Blog/
│       ├── BlogList.tsx // "/blog"
│       └── BlogPost.tsx // "/blog/:id"
├── App.tsx            // 전체 앱 구조와 라우팅 설정
├── index.tsx          // React 엔트리 포인트
└── styles/            // 스타일 파일
```

특징

- **라우팅 설정 필요**: React Router 같은 라이브러리를 사용해 직접 경로를 설정

```jsx
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Home from './pages/Home';
import About from './pages/About';
import BlogList from './pages/Blog/BlogList';
import BlogPost from './pages/Blog/BlogPost';

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/blog" element={<BlogList />} />
        <Route path="/blog/:id" element={<BlogPost />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

- **동적 라우팅**: React Router의 `useParams`를 사용해 구현.
- **자유로운 구조 설계**: 개발자가 폴더와 파일 구조를 마음대로 정할 수 있음.

Next.js와 React의 차이점

특징	Next.js	React
라우팅	파일 기반 라우팅 (자동)	개발자가 직접 설정 (React Router 필요)
동적 라우팅	`[slug].tsx` (자동 URL 매핑)	`useParams`와 Route 설정 필요
폴더 구조	규칙적 (파일 이름 = URL 경로)	자유로운 구조 설계 가능
API 라우트	내장 API 지원 (`/api`)	별도 서버 구현 필요
SEO 지원	내장된 `metadata`와 `head` 사용	외부 라이브러리 필요
SSR/SSG 지원	내장 기능 제공	별도 설정 필요

어떻게 선택해야 할까?

- **Next.js**: 파일 기반 라우팅, SEO, SSR/SSG 등 추가 기능이 필요할 때 적합.
- **React**: 프로젝트의 구조와 라우팅을 완전히 커스터마이징하려면 적합.

💬 이번 프로젝트에서는 NextJS의 App Router를 사용하고 학습해보고 싶어 NextJS를 선택하여 사용하도록 하겠습니다!