

Next.js는 React를 기반으로 한 프레임워크로, 서버 사이드 렌더링(SSR)과 정적 사이트 생성(SSG) 기능을 제공합니다. Next.js의 가장 큰 장점 중 하나는 내장된 파일 기반 라우팅 시스템입니다. 이 튜토리얼에서는 Next.js 프로젝트를 생성하고, 라우팅을 설정하여 페이지를 이동하는 방법을 알아보겠습니다.

## 1. Next.js 프로젝트 생성

먼저, Next.js 프로젝트를 생성합니다. Next.js는 `create-next-app` 명령어를 통해 쉽게 생성할 수 있습니다.

```
npx create-next-app@latest my-nextjs-app
```

위 명령을 실행하면 Next.js 프로젝트가 생성됩니다. 프로젝트 디렉토리로 이동합니다.

```
cd my-nextjs-app
```

## 2. 프로젝트 구조 이해

Next.js 프로젝트를 생성하면 기본적으로 아래와 같은 디렉토리 구조가 만들어집니다:

```
my-nextjs-app/  
├── pages/  
│   ├── _app.js  
│   ├── index.js  
│   └── api/  
│       └── hello.js  
├── public/  
├── styles/  
├── .gitignore  
├── package.json  
└── README.md
```

- `pages/` : Next.js의 파일 기반 라우팅 시스템을 관리하는 디렉토리입니다. 각 파일은 고유한 경로를 가집니다.
- `public/` : 정적 파일(이미지, 폰트 등)을 저장하는 디렉토리입니다.
- `styles/` : CSS 파일을 저장하는 디렉토리입니다.
- `index.js` : 프로젝트의 루트 페이지입니다. `/` 경로에 해당합니다.
- `_app.js` : 모든 페이지에서 공통으로 사용되는 레이아웃이나 초기 설정을 관리합니다.

### 3. 기본 라우팅 설정

Next.js에서는 `pages/` 디렉토리 안에 파일을 생성하면 자동으로 해당 경로로 라우팅됩니다. 기본적으로 생성된 `index.js` 파일은 `/` 경로에 매핑됩니다.

## 3.1 새로운 페이지 추가

이제 몇 가지 새로운 페이지를 추가하여 라우팅을 설정해 보겠습니다.

- `pages/about.js` : `/about` 경로에 매핑될 페이지
- `pages/contact.js` : `/contact` 경로에 매핑될 페이지

먼저 `pages/about.js` 파일을 생성하고 아래와 같이 작성합니다:

```
// pages/about.js
import Link from 'next/link';

export default function About() {
  return (
    <div>
      <h1>About Page</h1>
      <p>This is the about page.</p>
      <Link href="/">Go back to Home</Link>
    </div>
  );
}
```

다음으로 `pages/contact.js` 파일을 생성하고 아래와 같이 작성합니다:

```
// pages/contact.js
import Link from 'next/link';

export default function Contact() {
  return (
    <div>
      <h1>Contact Page</h1>
      <p>This is the contact page.</p>
      <Link href="/">Go back to Home</Link>
    </div>
  );
}
```



### 3.2 기본 페이지 수정

pages/index.js 파일을 열고 다음과 같이 수정합니다:

```
// pages/index.js
import Link from 'next/link';

export default function Home() {
  return (
    <div>
      <h1>Home Page</h1>
      <p>Welcome to the home page!</p>
      <nav>
        <ul>
          <li>
            <Link href="/about">Go to About Page</Link>
          </li>
          <li>
            <Link href="/contact">Go to Contact Page</Link>
          </li>
        </ul>
      </nav>
    </div>
  );
}
```

## 4. 동적 라우팅 설정

Next.js에서는 동적 라우팅도 매우 쉽게 설정할 수 있습니다. 예를 들어, `/blog/[id]` 경로에서 특정 블로그 포스트를 보여주고 싶다면, `pages/blog/[id].js` 파일을 생성하면 됩니다.

```
// pages/blog/[id].js
import { useRouter } from 'next/router';
import Link from 'next/link'; // Link 컴포넌트 импорт

export default function BlogPost() {
  const router = useRouter();
  const { id } = router.query;

  return (
    <div>
      <h1>Blog Post {id}</h1>
      <p>This is the blog post with ID: {id}</p>
      <Link href="/">Go back to Home</Link>
    </div>
  );
}
```

`[id]` 파일명은 동적 경로를 나타내며, 해당 경로의 값은 `useRouter` 훅을 통해 접근할 수 있습니다

## 5. 프로젝트 실행

이제 프로젝트를 실행하여 라우팅이 올바르게 작동하는지 확인할 수 있습니다.

```
npm run dev
```

이 명령어는 개발 서버를 시작하고, 브라우저에서 `http://localhost:3000` 을 열면 페이지를 확인할 수 있습니다.

- `http://localhost:3000/` : 홈 페이지
- `http://localhost:3000/about` : "About" 페이지
- `http://localhost:3000/contact` : "Contact" 페이지
- `http://localhost:3000/blog/1` : 동적 라우팅을 사용한 "Blog Post 1" 페이지
- `http://localhost:3000/blog/2` : 동적 라우팅을 사용한 "Blog Post 2" 페이지



localhost:3000

# Home Page

Welcome to the home page!

- [Go to About Page](#)
- [Go to Contact Page](#)



localhost:3000/about

# About Page

This is the about page.

[Go back to Home](#)



localhost:3000/contact

# Contact Page

This is the contact page.

[Go back to Home](#)



localhost:3000/blog/1

# Blog Post 1

This is the blog post with ID: 1

[Go back to Home](#)

## 6. 결론

이 튜토리얼에서는 Next.js의 기본적인 라우팅 기능을 살펴보고, 파일 기반 및 동적 라우팅을 설정하는 방법을 알아보았습니다. Next.js의 라우팅 시스템은 매우 직관적이며, 페이지 추가 및 관리가 쉽습니다. 이를 바탕으로 다양한 페이지를 구현하고, 복잡한 애플리케이션을 효율적으로 개발할 수 있습니다.