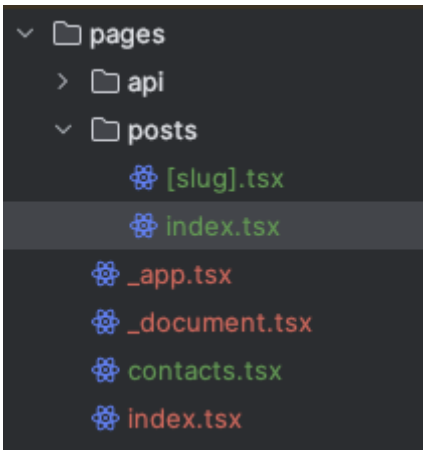
### 3.1 라우팅 시스템

기존 리액트의 React Router, Reach Router, Wouter 와 같은 라이브러리는 클라이언트에서의 라우팅만 구현할 수 있다.

Next.js는 다른 방법을 사용하는데 파일 시스템 기반 페이지와 라우팅 이다.

Next.js 프로젝트는 기본적으로 pages/ 디렉토리를 가지고 있다.(13버전 이전 기준 이후 버전에서는 app/ 디렉토리를 추천하고 있다.)

- 동적 라우팅 규칙
- 경로 매개변수




#### 페이지에서 경로 매개변수 사용하기

getServerSideProps 나 getStaticProps 함수는 객체를 반환해야 한다.

#### 컴포넌트에서 경로 매개변수 사용하기

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

function Greet(){
    const { query } = useRouter();
    return <h1>Hello {query.name}</h1>
}

export default Greet;
```

useRouter 훅을 사용해서 query 매개변수를 가져온다. 이 매개변수에는 경로 매개변수와 분석된 쿼리 매개변수 문자열값이 있다.

#### 클라이언트에서의 내비게이션

Next.js는 기본적으로 현재 화면에 표시되는 페이지의 모든 Link에 대해 연결된 부분 또는 페이지를 미리 읽어온다.
미리 불러오는 기능은 Link 컴포넌트에 preload={false}라는 속성을 전달해서 비활성화할 수 있다.

```
import Link from "next/link";

1 usage
function Homepage() : JSX.Element {
    return (
      <>
        <div>This is the homepage</div>
        <Link href='/blog/2023-01-01/happy-new-year'>Read post</Link><p />
        <Link href='/blog/2023-03-05/match-update'>Read post</Link><p />
        <Link href='/blog/2023-04-23/i-love-nextjs'>Read post</Link><p/>
          <Link
              href={{
                  pathname: '/blog/[date]/[slug]',
                  query: {
                      date: '2024-01-01',
                      slug: 'happy-new-year',
                      foo: 'bar'
                  }
              }}
          >
              Read post
          </Link>
      </>
      );
}

no usages
export default Homepage;
```

router.push 메서드

## 3.2 정적 자원 제공

정적자원은 이미지, 폰트, 아이콘, 컴파일한 CSS 또는 JS 파일과 같은 동적으로 변하지 않는 모든 종류의 파일을 의미한다.
정적자원은 /public 디렉터리 안에 저장한다.
Next.js는 /public 디렉터리 안에 있는 모든 파일을 정적 지원으로 간주하고 제공한다.

**자동 이미지 최적화**

자동 이미지 최적화의 장점은 클라이언트가 이미지를 요구할 때 최적화한다는 점이다.
? 만일 최적화된 이미지를 next.js가 캐싱하고 있다면 이미지가 엄청 많은 시스템에서는 문제가 되지 않을까?

**외비 서비스를 통한 자동 이미지 최적화**

## 3.3 메타데이터

**공통 메타 태그 그룹**

## 3.4 _app.js와 _document.js 페이지 커스터마이징

**_app.js 페이지**

Next.js는 프로젝트를 생성하면 기본으로 다음과 같은 pages/_app.js 파일을 만든다.

```
import '@/styles/globals.css'
import type { AppProps } from 'next/app'

no usages
export default function App({ Component, pageProps }: AppProps) {
  return <Component {...pageProps} />
}
```

```
import App from 'next/app';

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

MyApp.getInitialProps = async (appContext) => {
  const appProps = await App.getInitialProps(appContext);
  const additionalProps = await fetch(...);
  return {
    ..appProps,
    ...additionalProps,
  };
}

export default MyApp;
```

**_documents.js 페이지**

_document.js 페이지를 수정할 때 이 Html, Head, Main, NextScript 이 네 가지를 반드시 불러와야 한다. 이 중에서 하나라도 빠지면 Next.js 애플리케이션은 제대로 작동하지 않습니다.