

Next.js 토크아보기

The React Framework for Web

1.



1. Vercel

NEXT.JS



SWR

React Hooks for Data Fetching



NextAuth.js

Authentication for Next.js

2. Next.js의 특징

1) file-system based routing



Using App Router

Features available in /app



Using Pages Router

Features available in /pages

1) file-system based routing

pages router: 파일명을 기준으로 라우팅 (p.301 참고)

`pages/index.js` → `/`

`pages/blog/index.js` → `/blog`

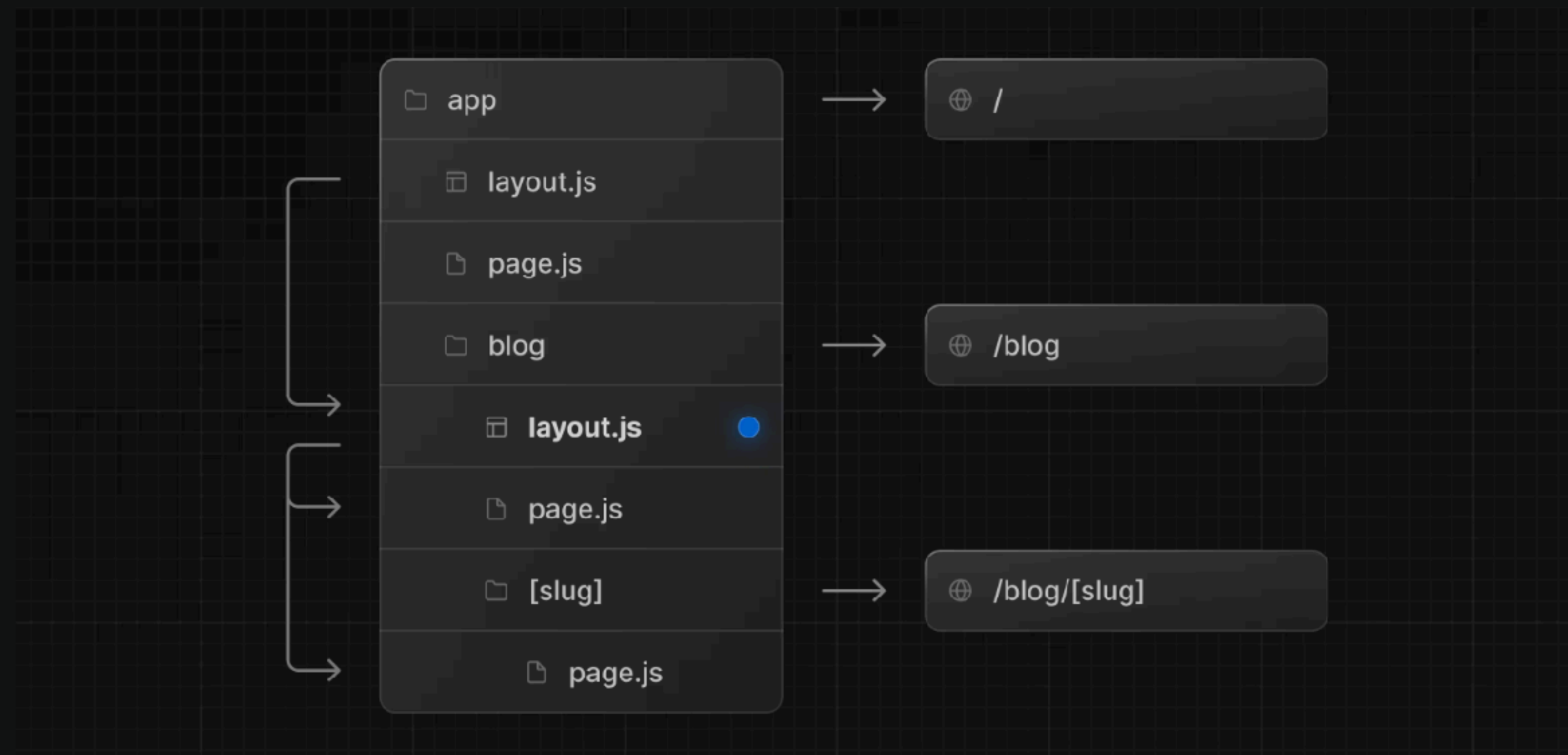
`pages/blog/first-post.js` → `/blog/first-post`

`pages/dashboard/settings/username.js` → `/dashboard/settings/username`

`pages/posts/[id].js`, then it will be accessible at `posts/1`, `posts/2`, etc.

1) file-system based routing

app router: 폴더명을 기준으로 라우팅 (p.719 참고)



1) file-system based routing

file convention: 해당 이름으로 설정하면 Next.js가 알아서 렌더링

pages router

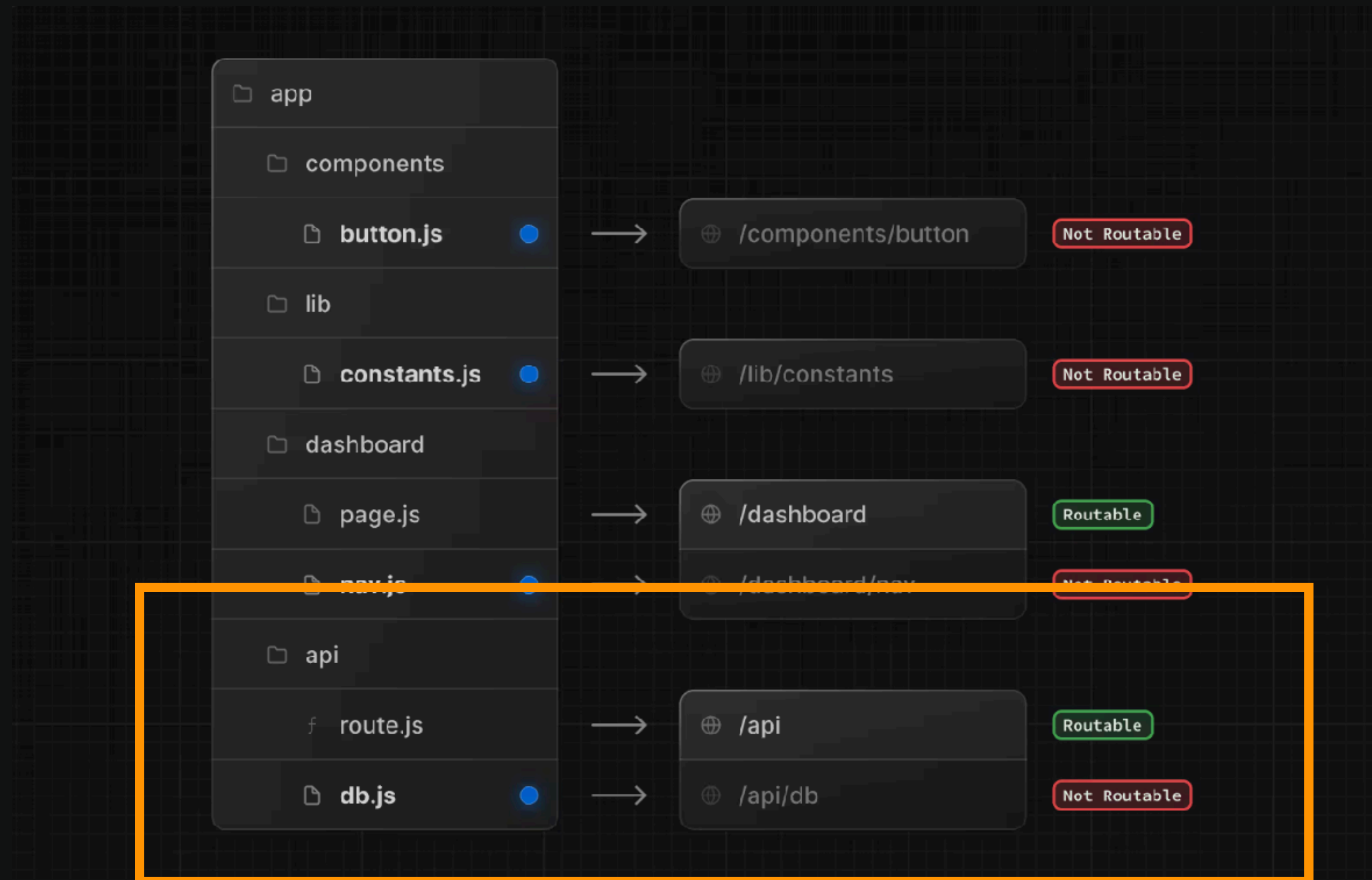
<code>_app</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Custom App
<code>_document</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Custom Document
<code>_error</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Custom Error Page
<code>404</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	404 Error Page
<code>500</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	500 Error Page

app router

<code>layout</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Layout
<code>page</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Page
<code>loading</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Loading UI
<code>not-found</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Not found UI
<code>error</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Error UI
<code>global-error</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Global error UI
<code>route</code>	<code>.js</code> <code>.ts</code>	API endpoint
<code>template</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Re-rendered layout
<code>default</code>	<code>.js</code> <code>.jsx</code> <code>.tsx</code>	Parallel route fallback page

1) file-system based routing

api route



2. Next.js의 특징

2) Data Fetching



2) Data Fetching

pages router: `getStaticPaths`와 `getStaticProps` 그리고 `getStaticProps` 사용

- `getStaticPaths`: 동적 경로를 정의하여 정적으로 생성할 페이지 결정
- `getStaticProps`: 정적 페이지에 필요한 데이터를 가져오기

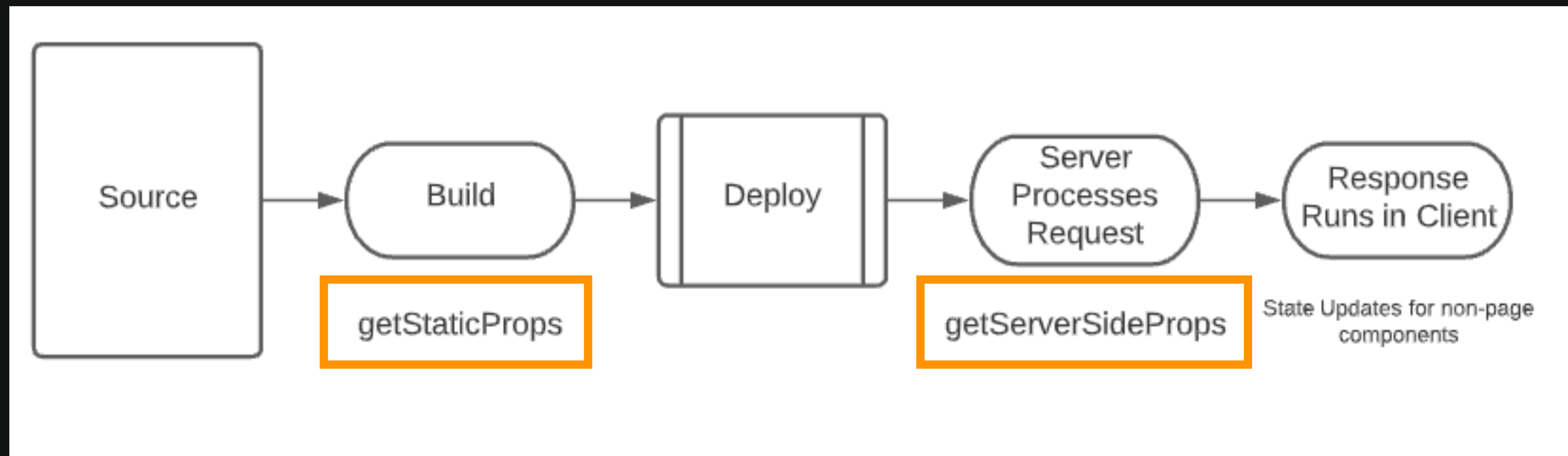
=> 두 함수를 통해 빌드 시점에 이미 페이지에 fetching

- `getServerSideProps`: 정적 페이지에 필요한 데이터를 가져오기

=> 클라이언트가 요청할 때 마다 페이지에 fetching


2) Data Fetching

getStaticProps vs getServerSideProps



2) Data Fetching

* `getInitialProps`: 클라이언트와 서버 모두에서 실행 가능 (legacy)



```
1 const App = () => <h1>This is a NextJS app.</h1>;  
2  
3 App.getInitialProps = async (ctx) => ({  
4   isServer: ctx.hasOwnProperty('req'),  
5 });
```

2) Data Fetching

app router: `generateStaticParams`와 `fetch` 사용

- `generateStaticParams`: `getStaticPaths`와 같은 역할
- `fetch`: 정적 페이지에 필요한 데이터를 가져오기

=> 두 함수를 조합해서, 페이지에 데이터를 전달 (서버 사이드 렌더링)

2. Next.js의 특징

3) Styling



2. Next.js의 특징

4) Optimizing Images

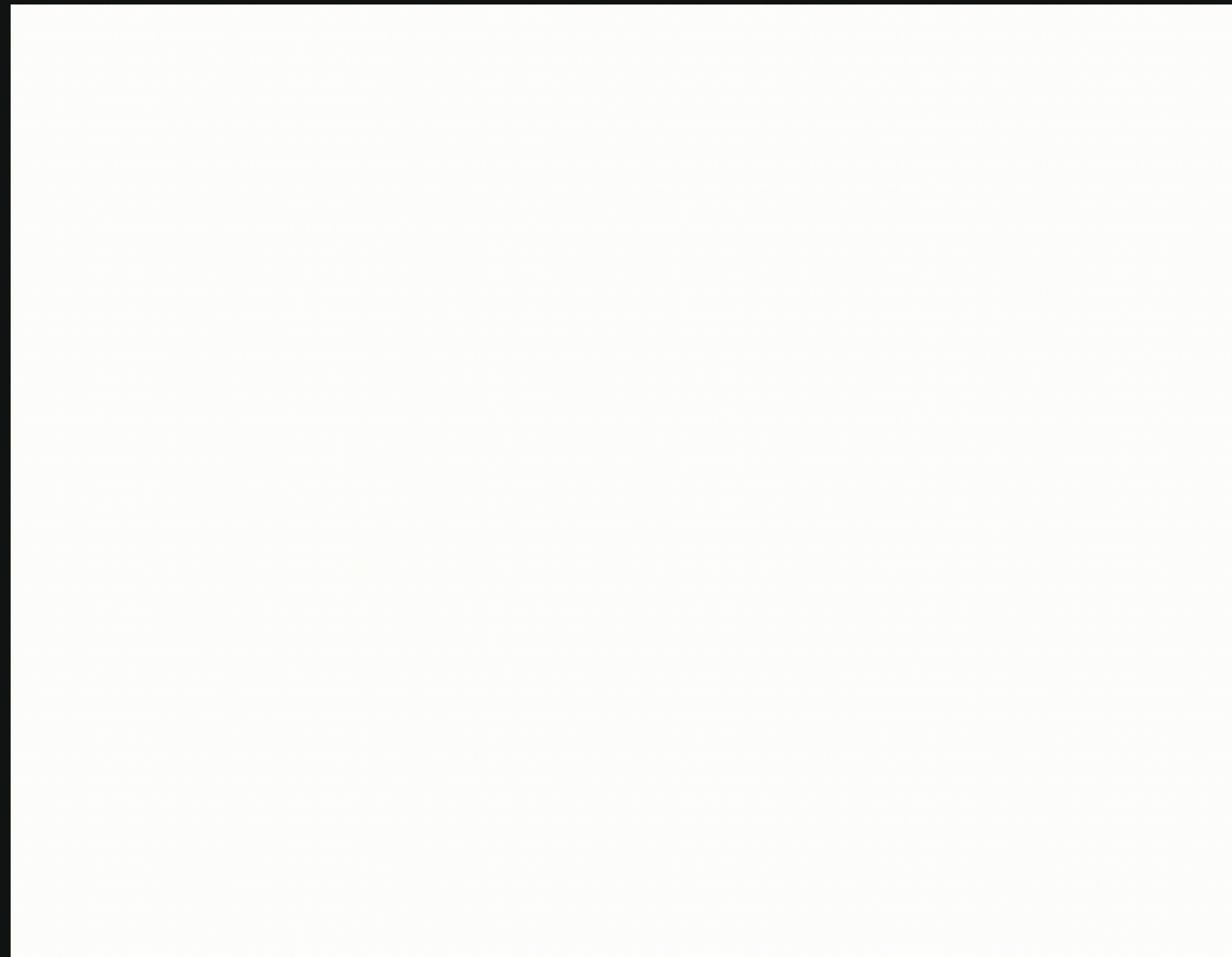
JS app/page.js



```
1  import Image from 'next/image'
2
3  export default function Page() {
4    return (
5      <Image
6        src="https://s3.amazonaws.com/my-bucket/profile.png"
7        alt="Picture of the author"
8        width={500}
9        height={500}
10     />
11    )
12  }
```

4) Optimizing Images

Cumulative Layout Shift (CLS)



4) Optimizing Images

priority

JS app/page.js

```
1  import Image from 'next/image'
2
3  export default function Home() {
4    return (
5      <>
6        <h1>My Homepage</h1>
7        <Image
8          src="/me.png"
9          alt="Picture of the author"
10         width={500}
11         height={500}
12         priority
13       />
14        <p>Welcome to my homepage!</p>
15      </>
16    )
17  }
```

4) Optimizing Images

반응형

```
1  import Image from 'next/image'
2  import mountains from '../public/mountains.jpg'
3
4  export default function Fill() {
5    return (
6      <div
7        style={{
8          display: 'grid',
9          gridGap: '8px',
10         gridTemplateColumns: 'repeat(auto-fit, minmax(400px, auto))',
11       }}
12     >
13       <div style={{ position: 'relative', height: '400px' }}>
14         <Image
15           alt="Mountains"
16           src={mountains}
17           fill
18           sizes="(min-width: 808px) 50vw, 100vw"
19           style={{
20             objectFit: 'cover', // cover, contain, none
21           }}
22         />
23       </div>
24       {/* And more images in the grid... */}
25     </div>
26   )
27 }
```