15. 레이아웃과 페이지 만들기

지금까지

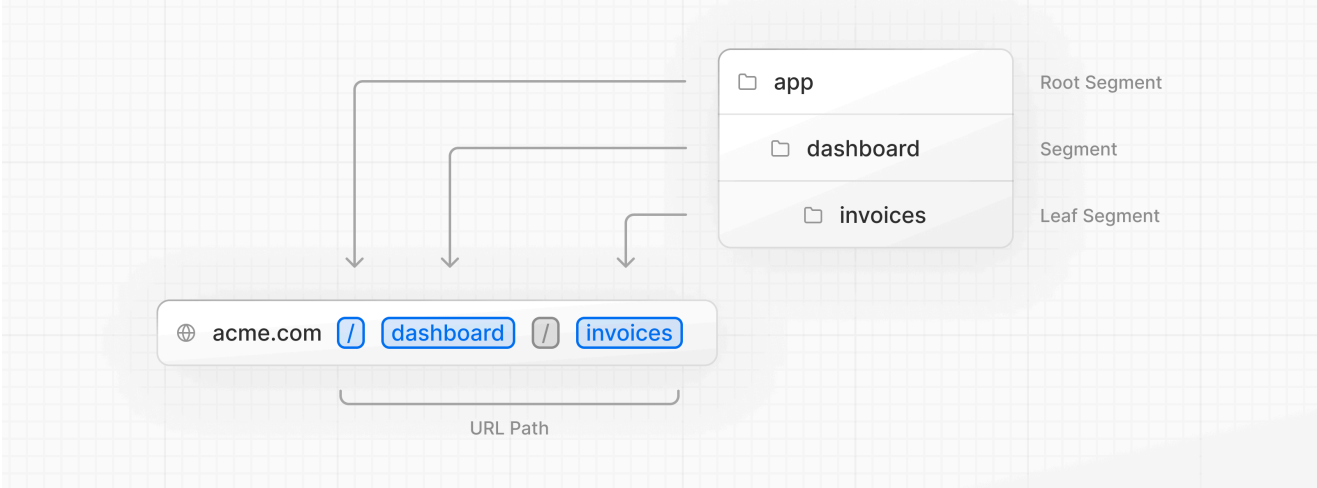
현재 애플리케이션에는 홈 페이지 하나만 존재합니다.
이번 챕터에서는 **레이아웃(layout)** 과 **페이지(page)** 를 추가해
새로운 URL 경로(route) 를 만드는 방법을 배웁니다.

이 챕터에서 배우는 것

- 파일 기반 라우팅(file-system routing)으로 `dashboard` 경로 만들기
- 폴더와 파일이 URL 세그먼트와 어떻게 연결되는지 이해하기
- 여러 대시보드 페이지에 공유되는 중첩 레이아웃(nested layout) 만들기
- Colocation(코드 근접 배치), Partial Rendering(부분 렌더링), Root Layout 개념 이해하기

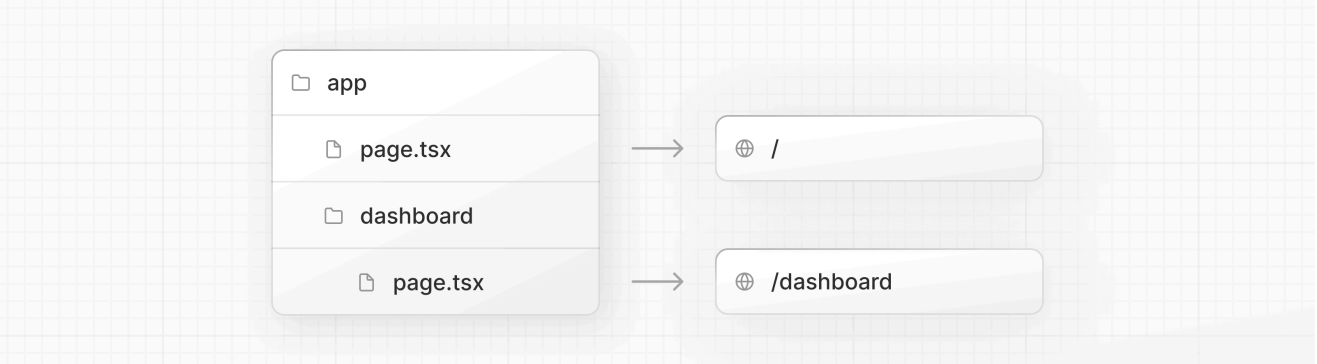
중첩 라우팅 (Nested Routing)

Next.js는 **파일 시스템 기반 라우팅**을 사용합니다.
폴더 구조가 URL 경로를 만듭니다.



- 각 폴더는 URL의 한 세그먼트를 의미합니다.
- `/app/page.tsx` → `/` (홈)
- `/app/dashboard/page.tsx` → `/dashboard`

`page.tsx` 파일은 **필수**입니다. 해당 폴더를 URL로 접근할 수 있게 만듭니다.



대시보드 페이지 만들기

1. `/app` 폴더 안에 `dashboard` 폴더를 새로 만듭니다.
2. 그 안에 `page.tsx` 파일을 만들고 다음 코드를 추가합니다:

/app/dashboard/page.tsx

```
export default function Page() {
  return <p>Dashboard Page</p>;
}
```

3. 개발 서버가 실행 중이라면
   http://localhost:3000/dashboard 에 접속해보세요.

→ **"Dashboard Page"** 가 표시됩니다.

Practice: 대시보드 하위 페이지 만들기

연습 문제
다음 두 개의 페이지를 만들어봅시다:

- `/dashboard/customers`
  `<p>Customers Page</p>` 출력
- `/dashboard/invoices`
  `<p>Invoices Page</p>` 출력

✨ Reveal the solution

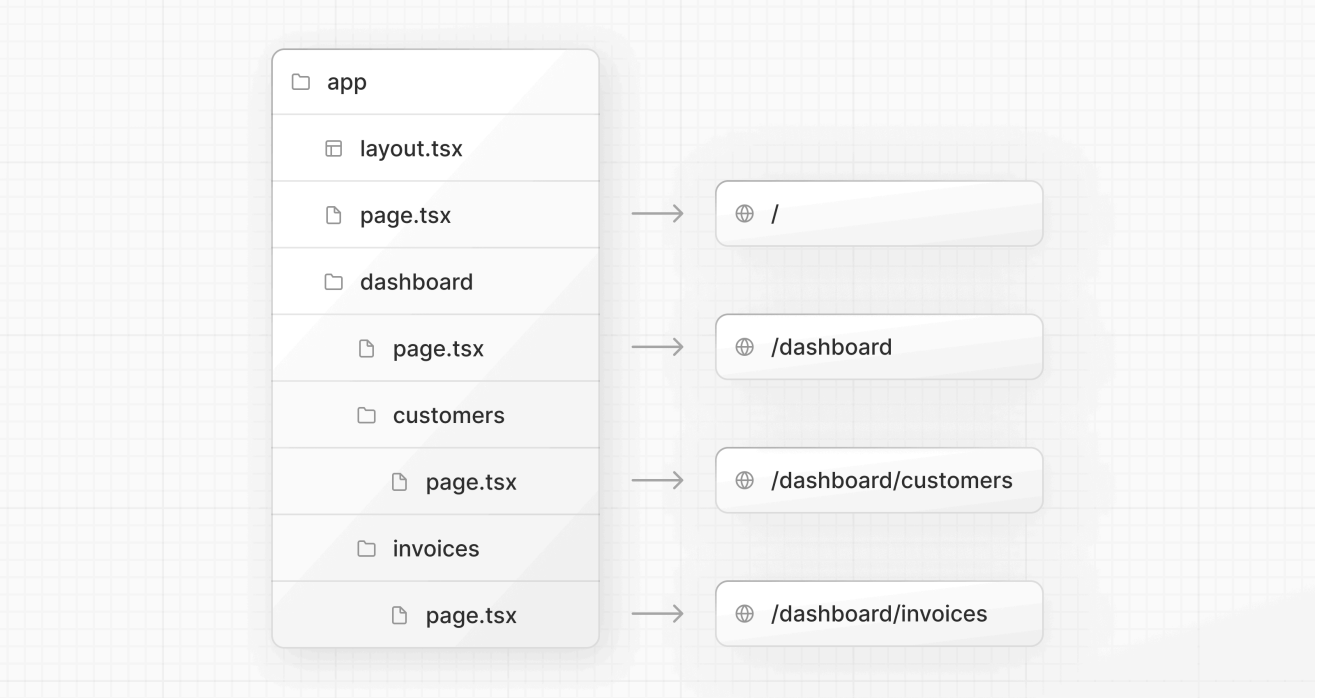
/app/dashboard/customers/page.tsx

```tsx
export default function Page() {
  return <p>Customers Page</p>;
}
```

/app/dashboard/invoices/page.tsx

```tsx
export default function Page() {
  return <p>Invoices Page</p>;
}
```

✅ 각각의 폴더 안에 `page.tsx` 파일을 추가해 경로를 만들었습니다.



대시보드 레이아웃 만들기

대시보드 페이지들 간에는 **공통 네비게이션(사이드 메뉴)** 이 필요합니다.
Next.js에서는 **`layout.tsx` 파일**로 여러 페이지에 걸쳐 UI를 공유할 수 있습니다.

레이아웃 생성

1. `/app/dashboard` 폴더 안에 `layout.tsx` 파일을 만듭니다.
2. 다음 코드를 붙여넣습니다:

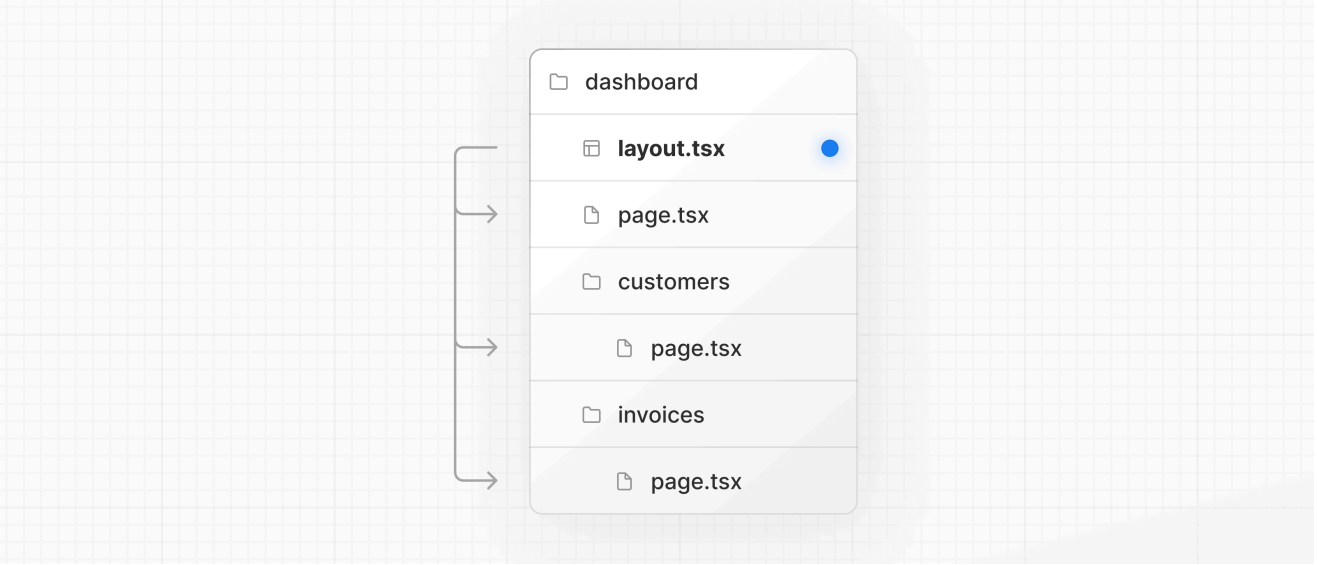
/app/dashboard/layout.tsx

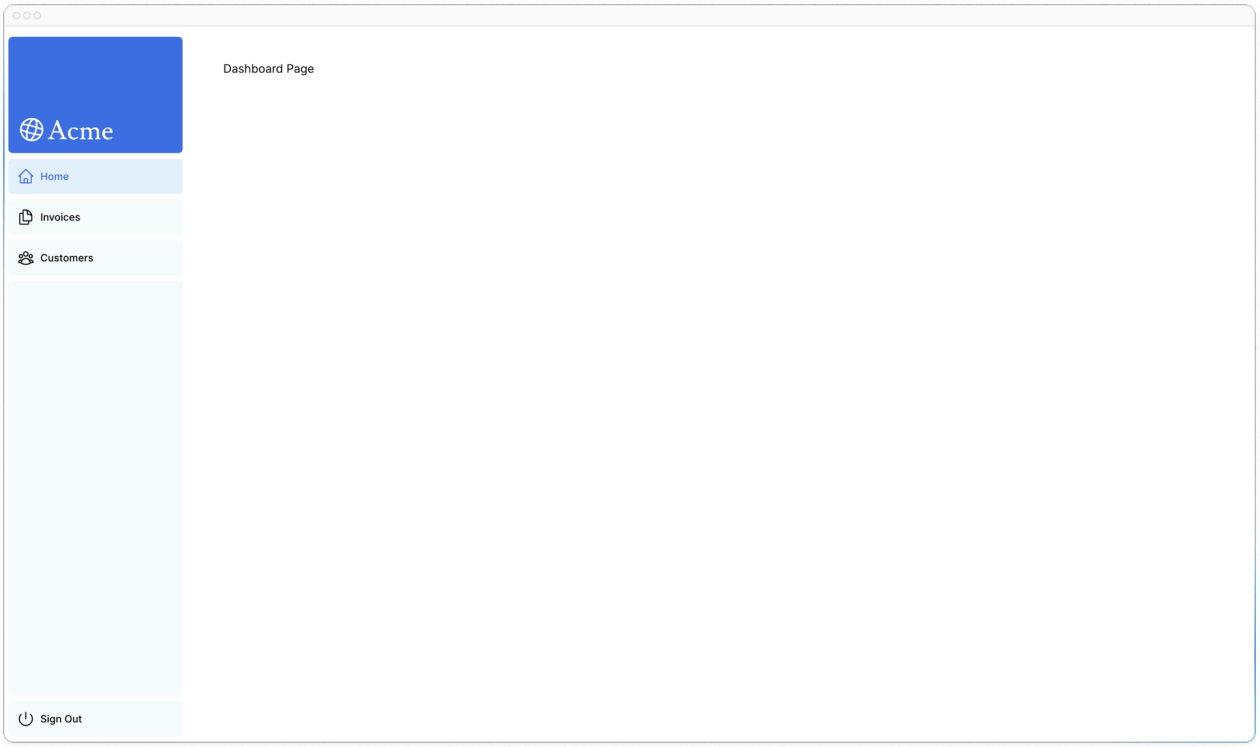
```tsx
import SideNav from '@/app/ui/dashboard/sidenav';

export default function Layout({ children }: { children: React.ReactNode
```

```
}) {
  return (
    <div className="flex h-screen flex-col md:flex-row md:overflow-
hidden">
      <div className="w-full flex-none md:w-64">
        <SideNav />
      </div>
      <div className="flex-grow p-6 md:overflow-y-auto md:p-12">{children}
</div>
    </div>
  );
}
```

- `SideNav` 컴포넌트를 사이드바로 렌더링합니다.
- `children` 안에 각 페이지(`page.tsx`)의 내용이 들어갑니다.





레이아웃 동작 방식

- `/dashboard/page.tsx`
- `/dashboard/customers/page.tsx`
- `/dashboard/invoices/page.tsx`

이 페이지들은 모두 `/dashboard/layout.tsx` 안에 **자동으로 중첩**되어 렌더링됩니다.

Partial Rendering (부분 렌더링)

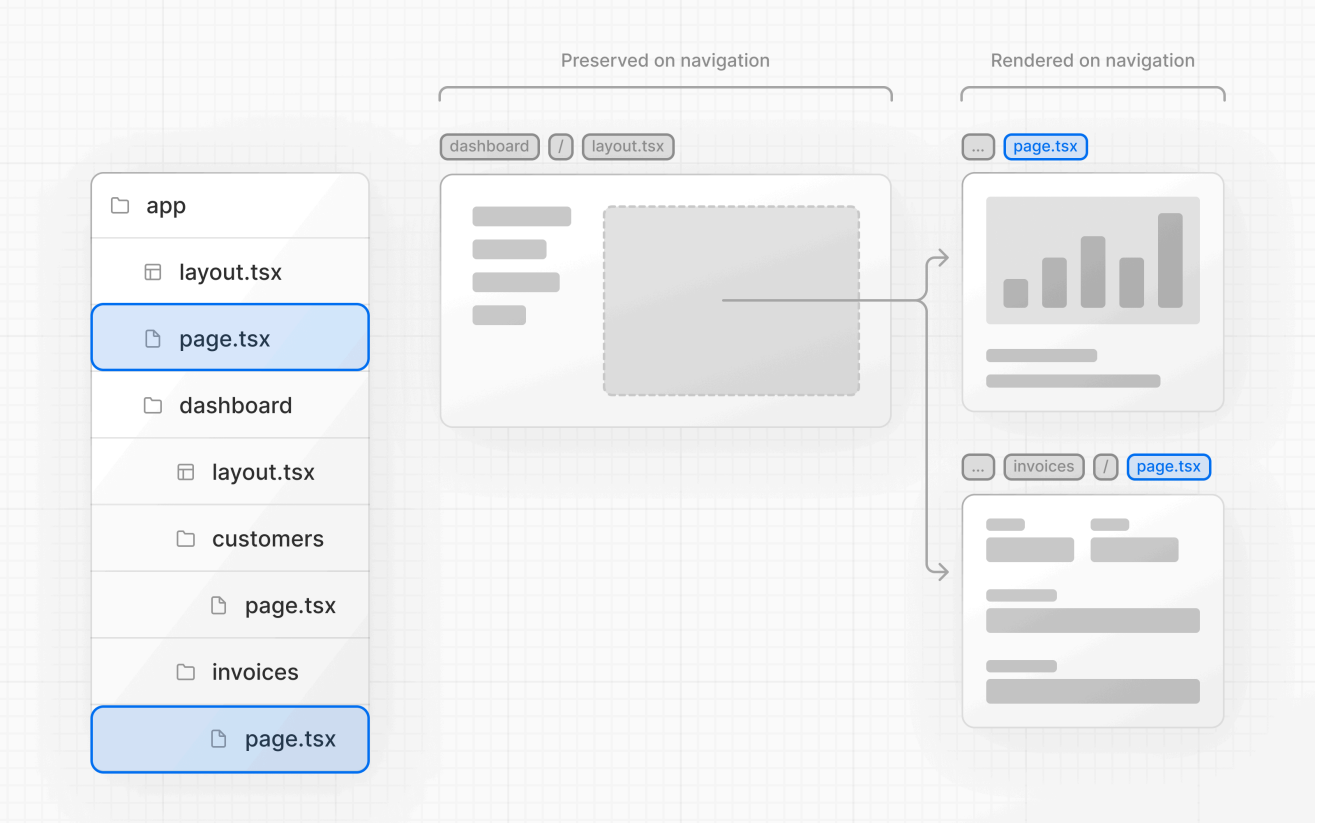
Next.js에서는 페이지 이동 시:

- 레이아웃(layout)은 **변경되지 않고 유지**되고
- 하위 페이지(page)만 **부분적으로 갱신**됩니다.

이를 **Partial Rendering**이라고 부릅니다.

장점:

- 레이아웃이 리렌더링되지 않기 때문에
- 사이드 메뉴나 상단 바 같은 요소들의 **상태를 유지**할 수 있습니다.



루트 레이아웃 (Root Layout)

Chapter 3에서 `/app/layout.tsx` 파일을 만들었습니다.

/app/layout.tsx

```
import '@/app/ui/global.css';
import { inter } from '@/app/ui/fonts';

export default function RootLayout({ children }: { children:
React.ReactNode }) {
  return (
    <html lang="en">
      <body className={`${inter.className} antialiased`}>{children}</body>
    </html>
  );
}
```

- 이 파일은 **모든 페이지의 최상위 구조**를 담당합니다.
- 예를 들어 `<html>`, `<body>`, 글로벌 스타일, 메타데이터를 설정합니다.

> 주의:
> 대시보드 전용 레이아웃(`dashboard/layout.tsx`)은 **루트 레이아웃과 별도로** 작동합니다.