Contents

Next.js

Folder Structure

To create a page, add a page file inside the app directory and default export a React component. Folders are used to define the route segments that map to URL segments. Files (like page and layout) are used to create UI that is shown for a segment.

Root

- `lib` and `ui` have **no framework meaning** for Next (no `page.tsx`)

path	features
`./app`	root dir for react
`./app/page.tsx`	starting point (`/` route)
`./app/layout.tsx`	root layout
`./app/not-found.tsx`	404 page
`./app/lib/`	type definitions, REST server client code, server side code
`./app/ui/`	React components for the user interface
`./app/public/`	static resources, here: images (png files)
`./app/user/`	`/user` page
`./app/global-error.tsx`	error handling

User Page

path	features
`./app/user/layout.tsx`	user page layout

path	features
`./app/user/loading.tsx`	fallback UI (loading screen) upon navigation
`./app/user/not-found.tsx`	custom 404 page for user
`./app/user/error.tsx`	error handling
`./app/user/[id]`	dynamic subpages

Advantages

- SEO improvements through **SSR** (Server Side Rendering) and **SSG** (Static Site Generation)
- Automatic **Code Splitting** (Chunking)
- Simplified *file-based* routing
- Easy full-stack development through API routes
- **Scoped CSS** and **SASS** support
- **TypeScript** support
- Image optimization
- **HMR** (Hot Module Replacement)
- Page pre-fetch
- Minimal configuration on Vercel

Structure

Each route is reflected in the directories and files under "app" and must have at least a "page.tsx" file which defines a Page component creating the content of the (sub)page.

Layout

`layout.tsx` defines an optional **Layout component** creating the layout of page. The result of the Page component is passed to the Layout component through a children prop.

- **Partial rendering**: when the user navigates to a certain path, only the Page components are rerendered, not the layouts
- A layout is UI that is shared between multiple pages.
- On navigation, layouts preserve state, remain interactive, and do not rerender
- You can define a layout by default exporting a React component from a layout file.
- The component should accept a children prop which can be a page or another layout
- Nesting layouts is possible

```
import '@/app/global.css'; // : import global css file to top level component
```

```
export default function RootLayout({
    children,
  }: {
    children: React.ReactNode;
  }) {
    return (
      <html lang="en">
        <body className={`${inter.className} antialiased`}>{children}</body>
      </html>
      // The layout above is called a root layout because it's defined at the root of the
      // app directory.
      // The root layout is required and must contain html and body tags.
    );
}
```

Page

- A nested route is a route composed of multiple URL segments.
- For example, the `/details/[id]/edit` route is composed of four segments:
  - `/` - the root segment
  - `/details` - the details segment
  - `/[id]` - the dynamic id segment (slug)
  - `/edit` - the edit segment

```
// 'use server';

interface EditListProps {
    params: {
        id: string;
    };
}

// async component
export default async function EditDetails({ params }: { params: { id: string } }) {
    console.log('invoice id = ' + params.id);
    const id = params.id;
    const friend = JSON.parse(await getFriend(Number(id)));
    console.log('friend = ' + JSON.stringify(friend));

    return (
        <></>
    );
};

// alternative props with interface
function EditDetailsWithInterface({ params }: EditListProps) {
};
```

Loading

Loading generates code to display while component Page ist not finished generating output.

This approach is called **streaming**. We display a **skeleton** of the final layout while data is being loaded. See files `loading.tsx` and `ui/skeletons.ts`

`loading.tsx` defines component Loading:

```
import DashboardSkeleton from "../../ui/skeletons";

export default function Loading() {
  // simple version:
  //return <div>Loading...</div>;

  // more complex version: show dashboard skeleton
  return <DashboardSkeleton />;
}
```

- Has to be in the same folder as the `page.tsx` it is masking