Next Js

(https://projects.100xdevs.com/tracks/nextjs-1/next-1)

Pre-requisities:

React

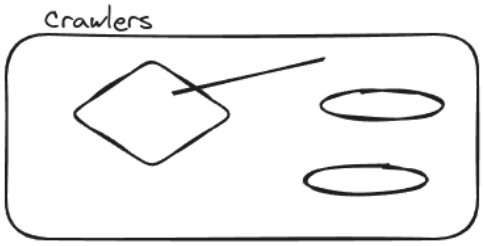
NextJs Intro

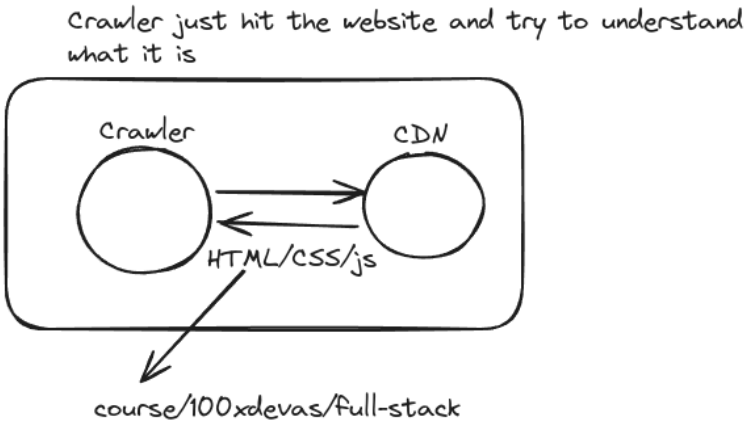
NextJS was a framework that was introduced because of some minor inconviniences in React

1. In a React project, you have to maintain a separate Backend project for your API routes
2. React does not provide out of the box routing (you have to use react-router-dom) . In next js we don't need external library to do the routing
3. **React is not SEO Optimised** . (Google wasn't unable to rank/crawl better)
    1. not exactly true today because of React Server components
    2. we'll discuss soon why
4. Waterfalling problem

Let's discuss some of these problems in the next slides

SEO Optimization

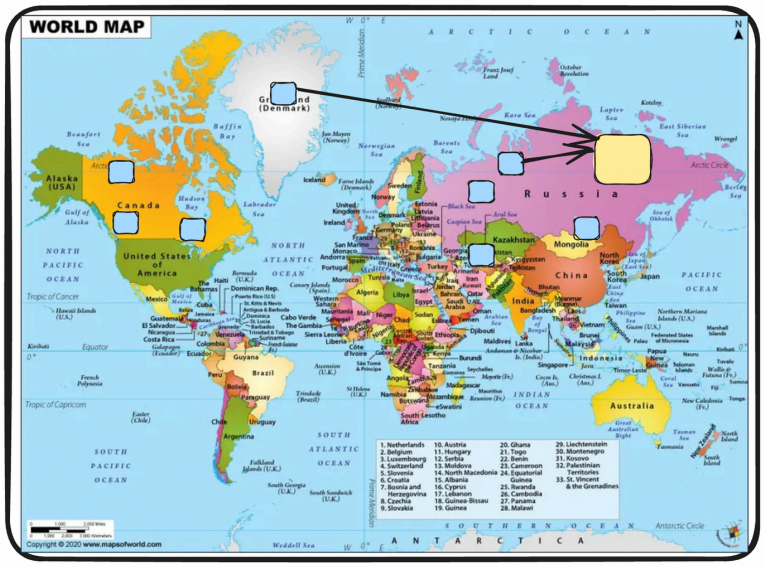




Google/Bing has a bunch of crawlers that hit websites and figure out what the website does.

It ranks it on Google based on the HTML it gets back

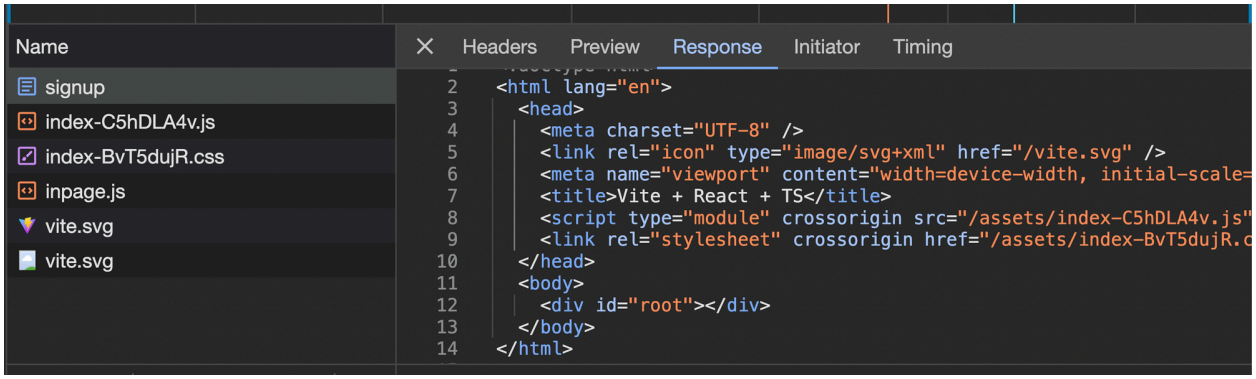
The crawlers DONT usually run your JS and render your page to see the final output.



While Googlebot can run JavaScript, `dynamically generated` content is harder for the scraper to index

Try visiting a react website

What does the Googlebot get back when they visit a website written in react?

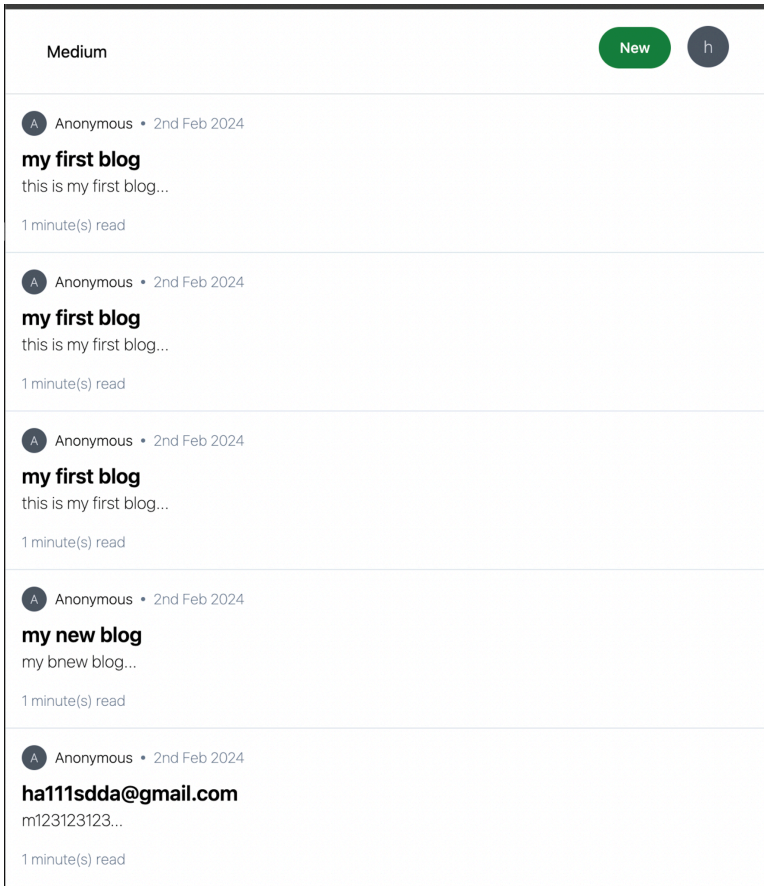


Googlebot has no idea on what the project is. It only sees Vite + React + TS in the original HTML response.

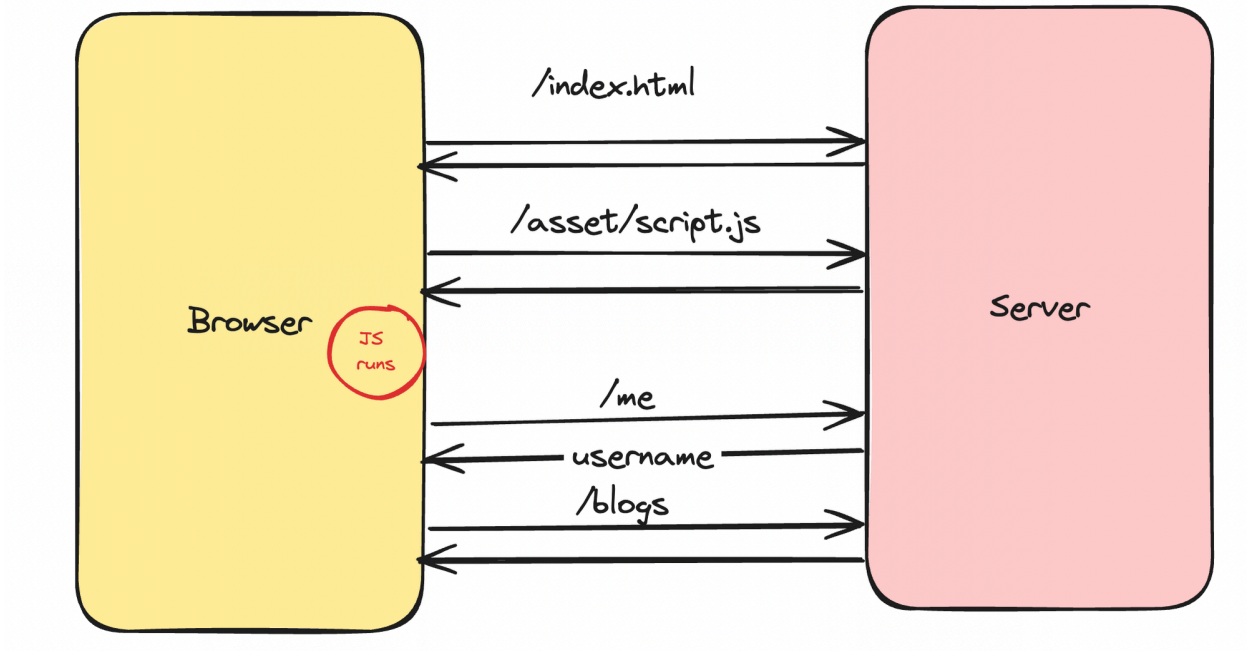
Ofcourse when the JS file loads eventually, things get rendered but the Googlebot doesn't discover this content very well.

Waterfalling problem

Let's take example of the blogging website in react, what steps do you think the request cycle takes?



Request cycle look like this



/me endpoints checks whether user is valid or not and if not then send them to sign/signup page

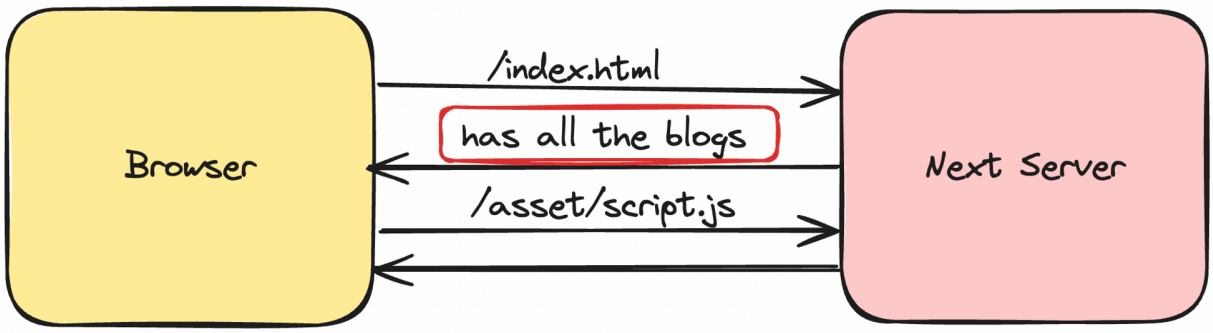
1. Fetching the index.html from the CDN
2. Fetching script.js from CDN
3. Checking if user is logged in (if not, redirect them to /login page)
4. Fetching the actual blogs

There are 4 round trips that happen one after the other (sequentially)

The "waterfalling problem" in React, and more broadly in web development, refers to a scenario where data fetching operations are chained or dependent on each other in a way that leads to inefficient loading behavior.

Wouldn't it be better that instead of sequentially in a single request we can get everything (html one)

What does NextJs provide



NextJs Offering

Next.js provides you the following upsides over React

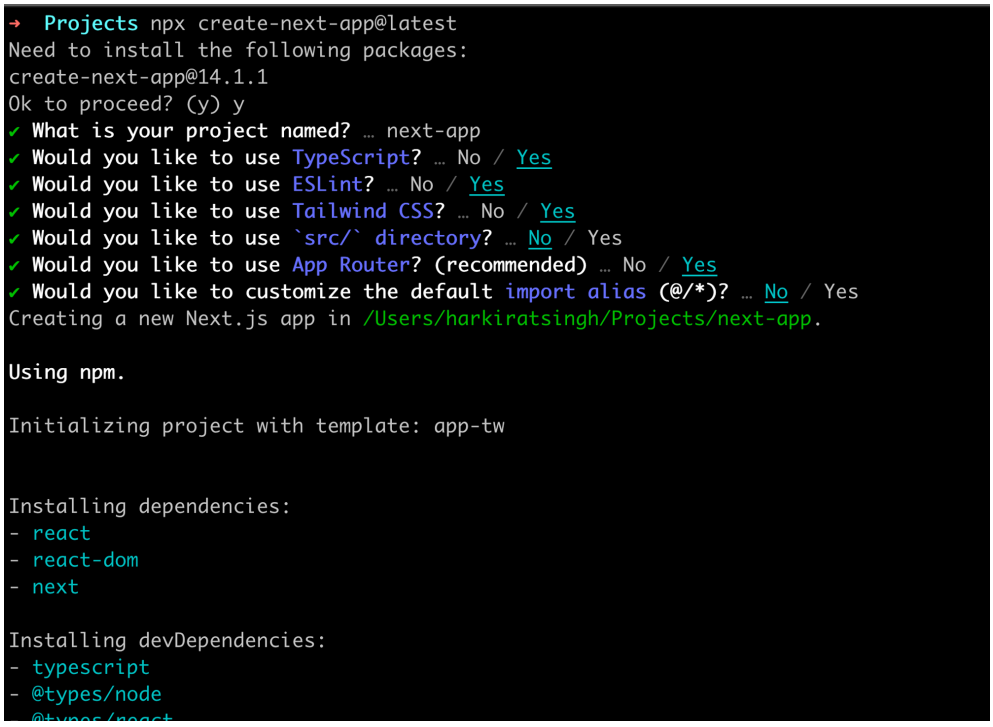
1. Server side rendering(**SSR**) - Get's rid of SEO problems (Server was able to render some jsx and put it in html)
2. API routes - Single codebase with frontend and backend (all code can be directly stored in single codebase)
3. File based routing (no need for react-router-dom)
4. Bundle size optimisations (know how to make html/css/js very small), Static site generation(**SSG**)
5. Maintained by the Vercel team

Downsides

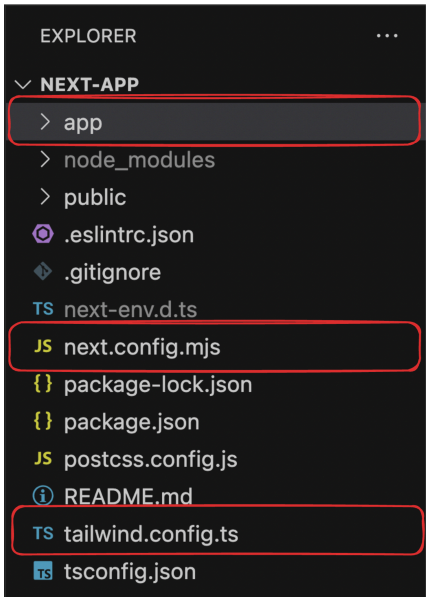
1. Can't be distributed via a CDN, always needs a server running that does server side rendering and hence is expensive
2. Very opinionated, very hard to move out of it

Bootstrap a simple NextJs App

npx create-next-app@latest



File structure



1. next.config.mjs - Nextjs configuration file
2. tailwind.config.js - Tailwind configuration file
3. app - Contains all your code/components/layouts/routes/apis

```
/** @type {import('next').NextConfig} */
const nextConfig = {};

export default nextConfig;
```

This file if will be important when we see transpiling

Bootstrap the project

- Remove everything from app/page.tsx and return an empty div
- Remove the css bits (not the tailwind headers) from the global.css file

Page.tsx

```
import Image from "next/image";

export default function Home() {
 return (
```

```
        <div>
          hi there
        </div>
  );
}
```

Understanding routing in Next

Routing in react

https://blog-six-tan-47.vercel.app/signup

```
function App() {

  return (
    <>
      <BrowserRouter>
        <Routes>
          <Route path="/signup" element={<Signup />} />
          <Route path="/signin" element={<Signin />} />
          <Route path="/blog/:id" element={<Blog />} />
        </Routes>
      </BrowserRouter>
    </>
  )
}
```

Routing in NextJs

Next.js has a file based router (https://nextjs.org/docs/app/building-your-application/routing/defining-routes)

This means that the way you create your files, describes what renders on a route (what will be render on signup will depend on what app did you define)

1. Let's add a new folder in app called signup
2. Let's add a file called page.tsx inside app/signup

page.tsx

```
export default function Signup() {
   return (
     <div>
       hi from the signup page
     </div>
   );
 }
```

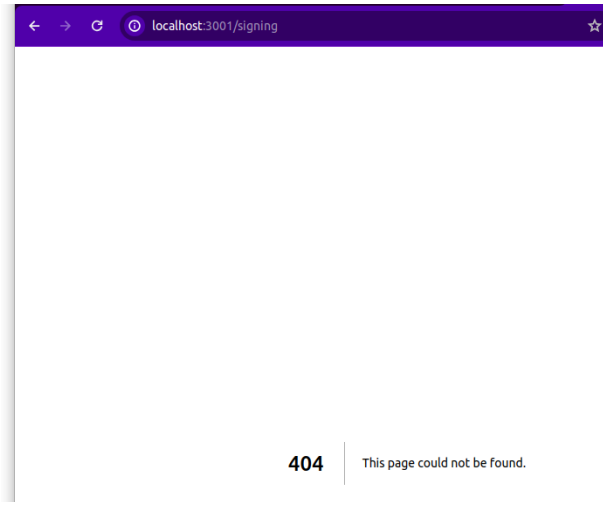
3. npm run dev

hi from the signup page

Final folder structure

Adding **signin** route

Let's see what it shows for route not defined



Prettify the signin page

New signin page

```
export default function Signin() {
   return <div className="h-screen flex justify-center flex-col">
       <div className="flex justify-center">
       <a href="#" className="block max-w-sm p-6 bg-white border
border-gray-200 rounded-lg shadow hover:bg-gray-100 ">
               <div>
                   <div className="px-10">
                       <div className="text-3xl font-extrabold">
                           Sign in
                       </div>
                   </div>
                   <div className="pt-2">
                       <LabelledInput label="Username"
placeholder="harkirat@gmail.com" />
                       <LabelledInput label="Password" type={"password"}
placeholder="123456" />
                       <button type="button" className="mt-8 w-full
text-white bg-gray-800 focus:ring-4 focus:ring-gray-300 font-medium
rounded-lg text-sm px-5 py-2.5 me-2 mb-2">Sign in</button>
                   </div>
               </div>
           </a>
       </div>
```

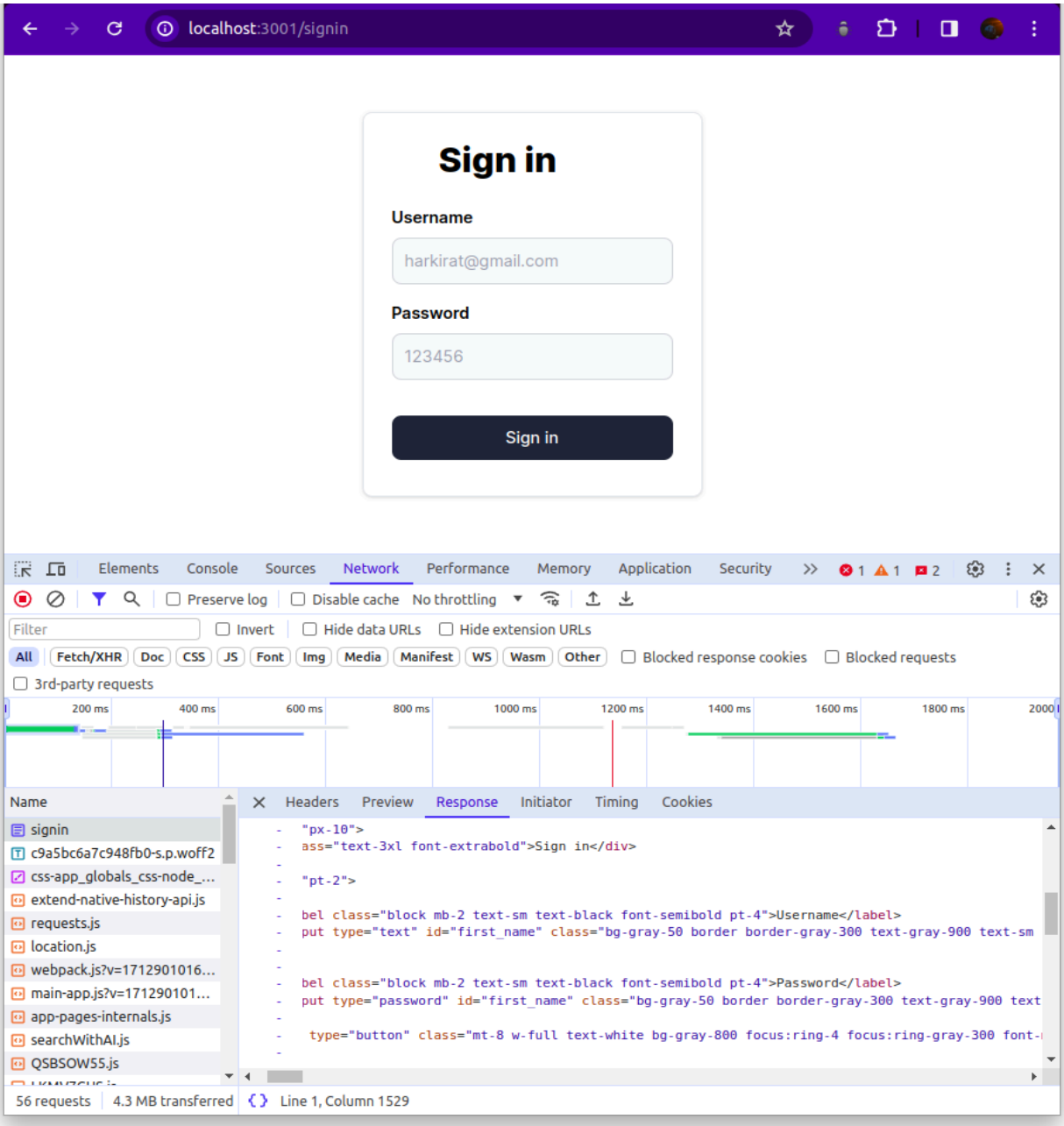
```
    </div>
}

interface LabelledInputType {
    label: string;
    placeholder: string;
    type?: string;
}

function LabelledInput({ label, placeholder, type }: LabelledInputType) {
    return <div>
        <label className="block mb-2 text-sm text-black font-semibold
pt-4">{label}</label>
        <input type={type || "text"} id="first_name" className="bg-gray-50
border border-gray-300 text-gray-900 text-sm rounded-lg
focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5"
placeholder={placeholder} required />
    </div>
}
```

Server Side Rendering (SSR)

Let's see what response we get when we get back HTML file.

Scrapper can see password, signup, etc in the initial html passed .
Now if GoogleBot tries to scrape your page, it'll understand that this is a signup page without running any Javascript.
The first index.html file it get's back will have context about the page since it was server side rendered
Well populated html page
Hence bot can rank it correctly

Rendering means putting something in DOM . SSR means created Html on the server itself. Not generating DOM element in client side



Layouts

Layout.tsx

```
import type { Metadata } from "next";
import { Inter } from "next/font/google";
import "./globals.css";
```

```tsx
const inter = Inter({ subsets: ["latin"] });

export const metadata: Metadata = {
 title: "Create Next App",
 description: "Generated by create next app",
};

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

What this does

https://nextjs.org/docs/app/building-your-application/routing/pages-and-layouts

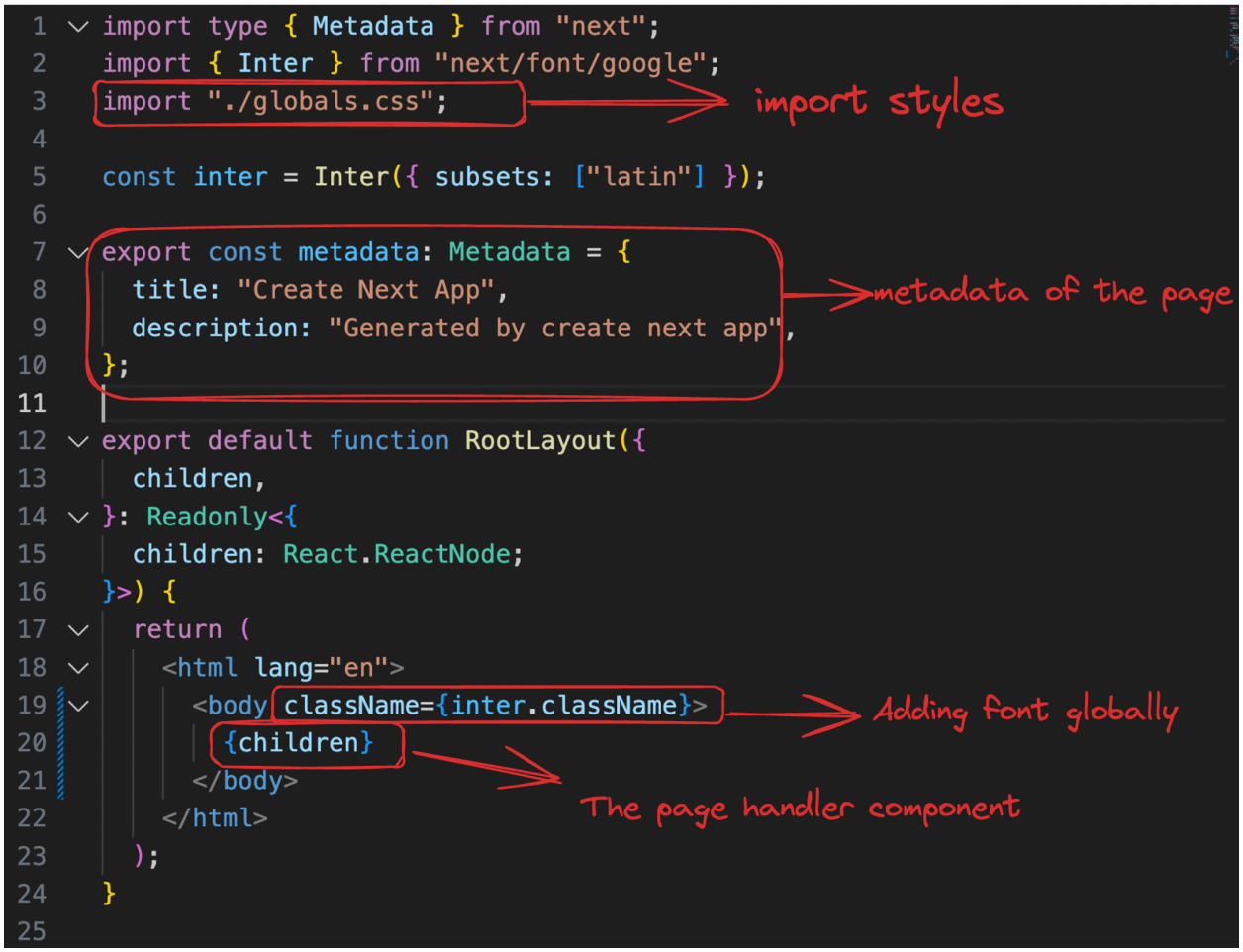
What are layouts

Layouts let you wrap all child pages inside some logic

Exports a metadata object , needs to be put somewhere , we can see the title.

Write now we have single root layout.tsx

Why are we rendering children?

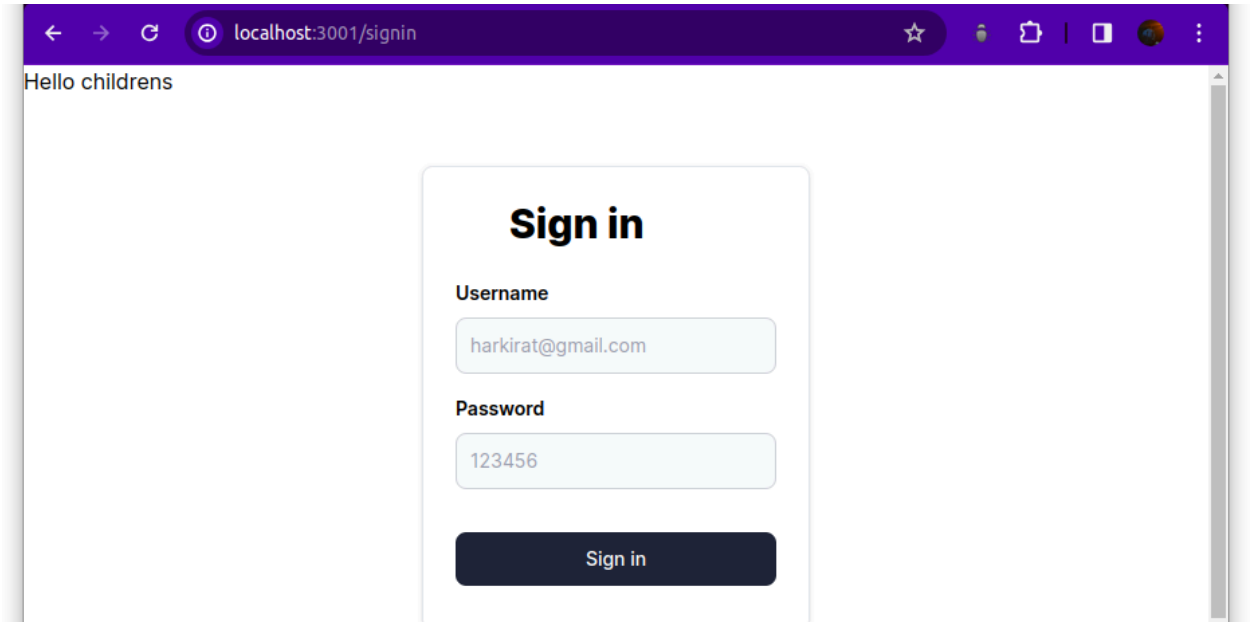


Lets explore layout.tsx

```
return (
   <html lang="en">
     <body className={inter.className}>
       Hello childrens
       {children}
       </body>
   </html>
 );
```

We can see “Hello childrens” text all over the places

Sign.tsx



Signup page

This file is creating layout for every page , every page we create in wrap around layout.tsx

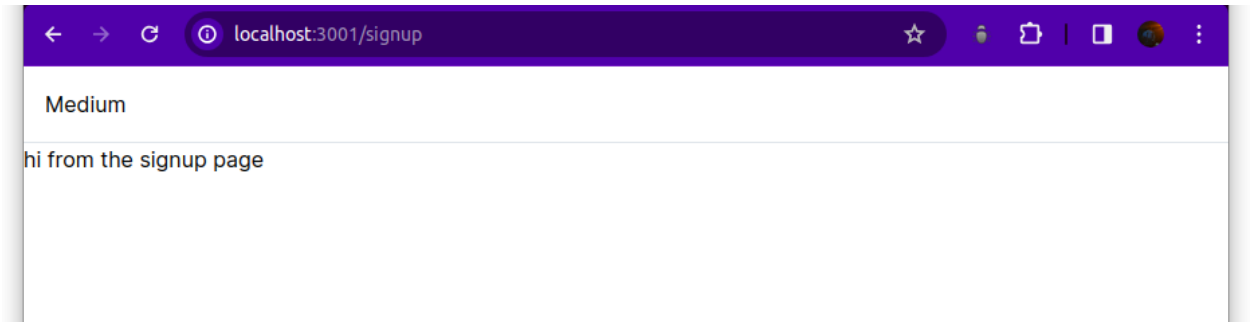
Html -> body -> “Hello children” rendered -> {Children} rendered

Why do we do ?

We have observe that the top appbar which is common in all the suppose ecommerce website let define appbar in the layout.tsx

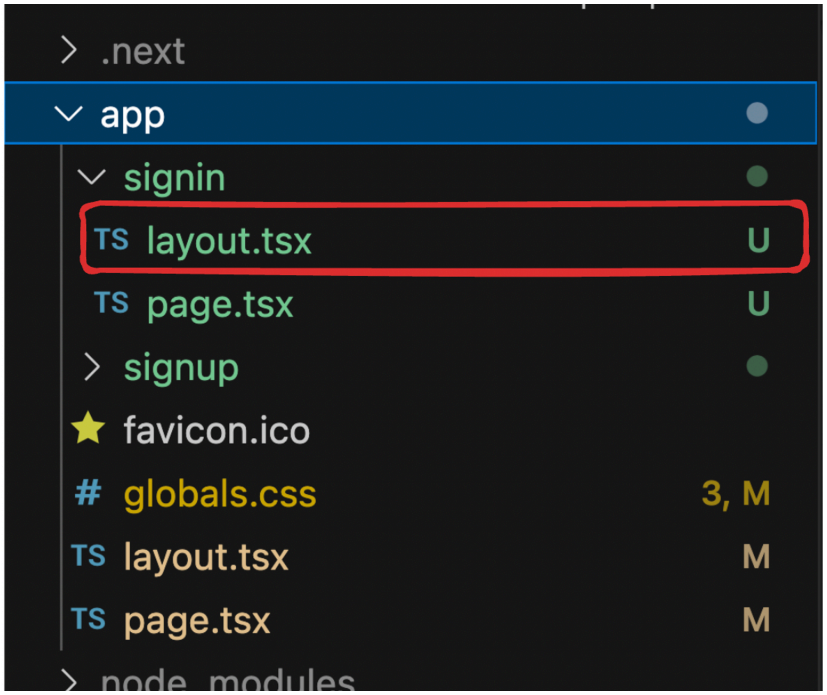
```
return (
   <html lang="en">
```

```
    <body className={inter.className}>
      <div className="p-4 border-b">
        Medium
      </div>
      {children}
      </body>
  </html>
);
```



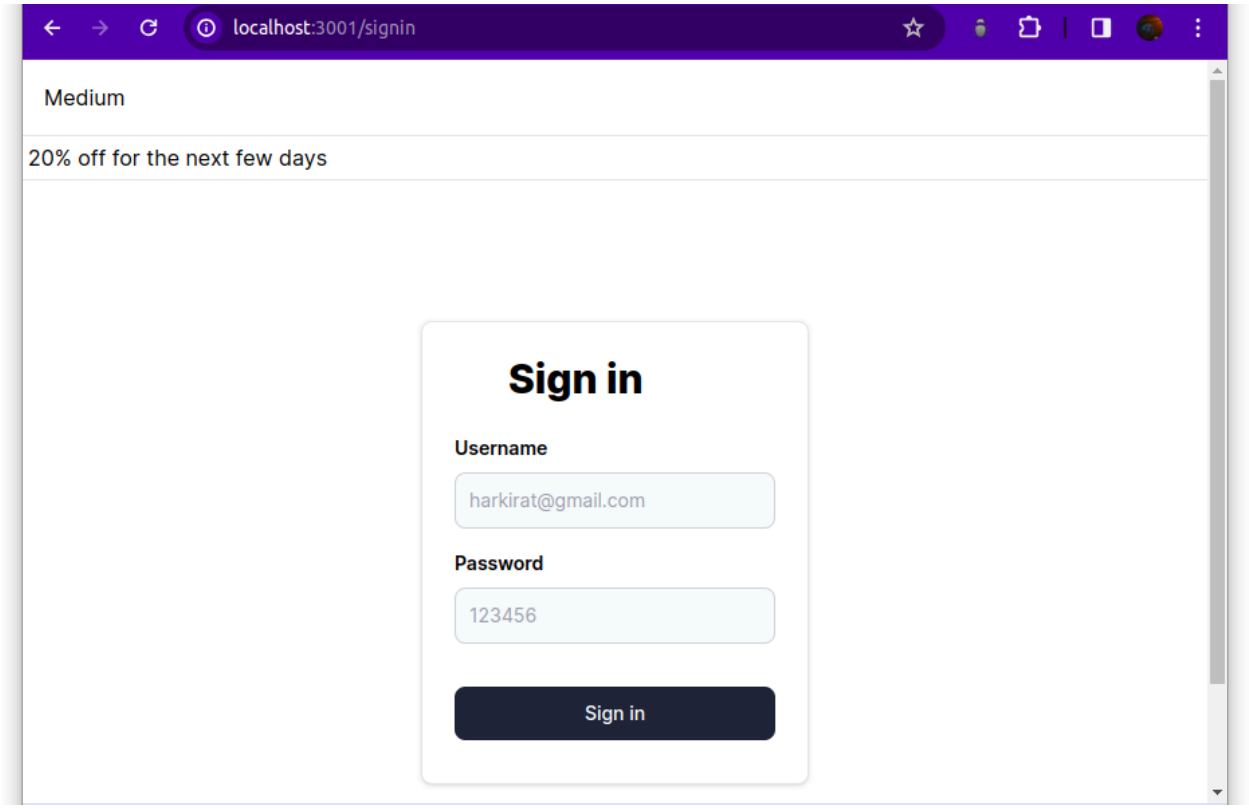
Layouts in sub routes

What if you wan't all routes that start with /signin to have a banner that says Login now to get 20% off



But we want banner for only the pages which start with the /signin

```
import React from "react";

export default function
SigninLayout({children}:{children:React.ReactNode}){
   return <div className="border-b">
      <div className="border-b p-1">
          20% off for the next few days
      </div>
      {children}
   </div>
}
```

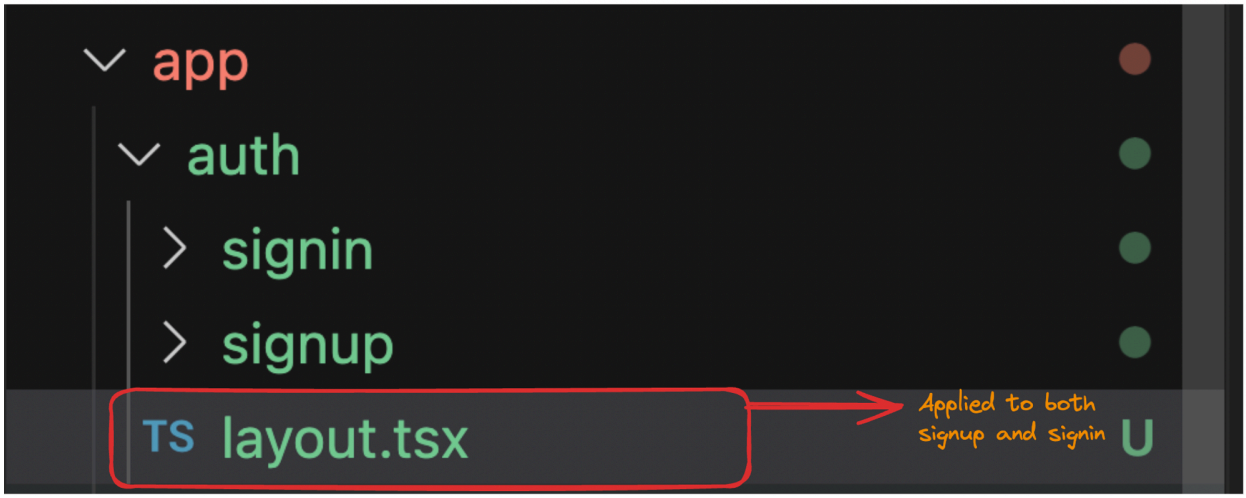


Merging routes

What if we wan't to get the banner in both signup and signin ?

Approach 1:

Move both the signin and signup folder inside a auth folder where we have the layout.



We can access routes at

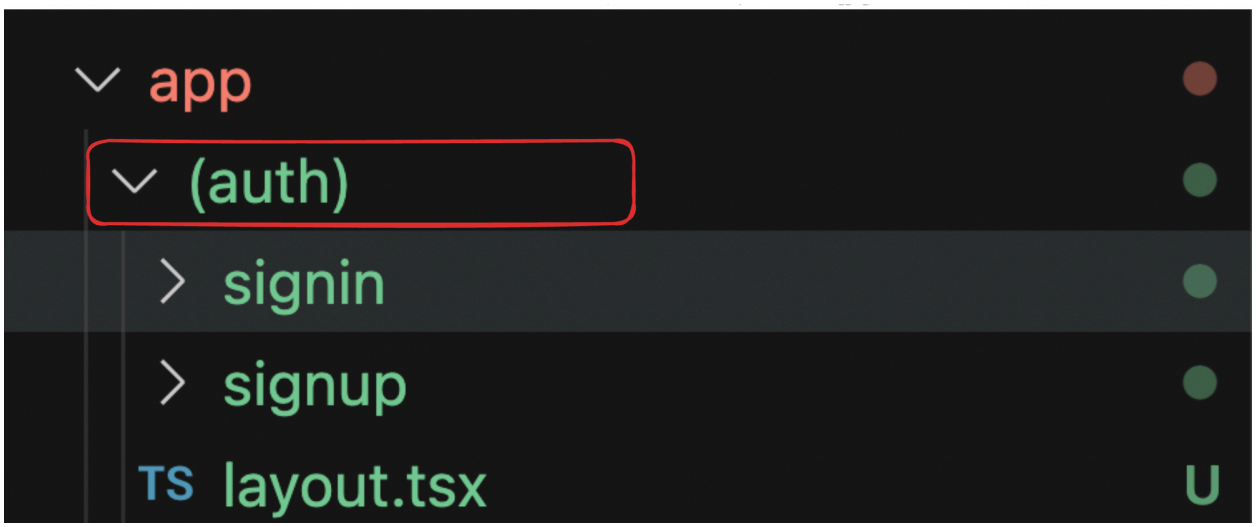
http://localhost:3000/auth/signup

http://localhost:3000/auth/signin

Approach 2:

You can use create a new folder with () around the name.

This folder is ignored by the router.



We can access routes at

http://localhost:3000/auth/signup

http://localhost:3000/auth/signin

components directory

You should put all your components in a components directory and use them in the app routes rather than shoving everything in the route handler

We can name the folder anything we want but have to make sure that we have added it in the tailwind.config.ts , since it need to know where we have write down our tailwind code

Lets do it for page.tsx inside signin folder

component -> SigninComponent.tsx

```
export const SigninComponent = function(){
   return <div className="h-screen flex justify-center flex-col">
       <div className="flex justify-center">
       <a href="#" className="block max-w-sm p-6 bg-white border
border-gray-200 rounded-lg shadow hover:bg-gray-100 ">
               <div>
                   <div className="px-10">
                       <div className="text-3xl font-extrabold">
                           Sign in
                       </div>
                   </div>
                   <div className="pt-2">
                       <LabelledInput label="Username"
placeholder="harkirat@gmail.com" />
                       <LabelledInput label="Password" type={"password"}
placeholder="123456" />
                       <button type="button" className="mt-8 w-full
text-white bg-gray-800 focus:ring-4 focus:ring-gray-300 font-medium
rounded-lg text-sm px-5 py-2.5 me-2 mb-2">Sign in</button>
                   </div>
               </div>
           </a>
       </div>
   </div>
}
```

```
interface LabelledInputType {
    label: string;
    placeholder: string;
    type?: string;
}

function LabelledInput({ label, placeholder, type }: LabelledInputType) {
    return <div>
        <label className="block mb-2 text-sm text-black font-semibold
pt-4">{label}</label>
        <input type={type || "text"} id="first_name" className="bg-gray-50
border border-gray-300 text-gray-900 text-sm rounded-lg
focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5"
placeholder={placeholder} required />
    </div>
}
```

Now app-> (auth) -> Signin -> page.tsx

```
import { SigninComponent } from "@/components/Signin"

export default function Signin() {
    return <div>
        <SigninComponent />
    </div>
}
```

Add a button onclick handler

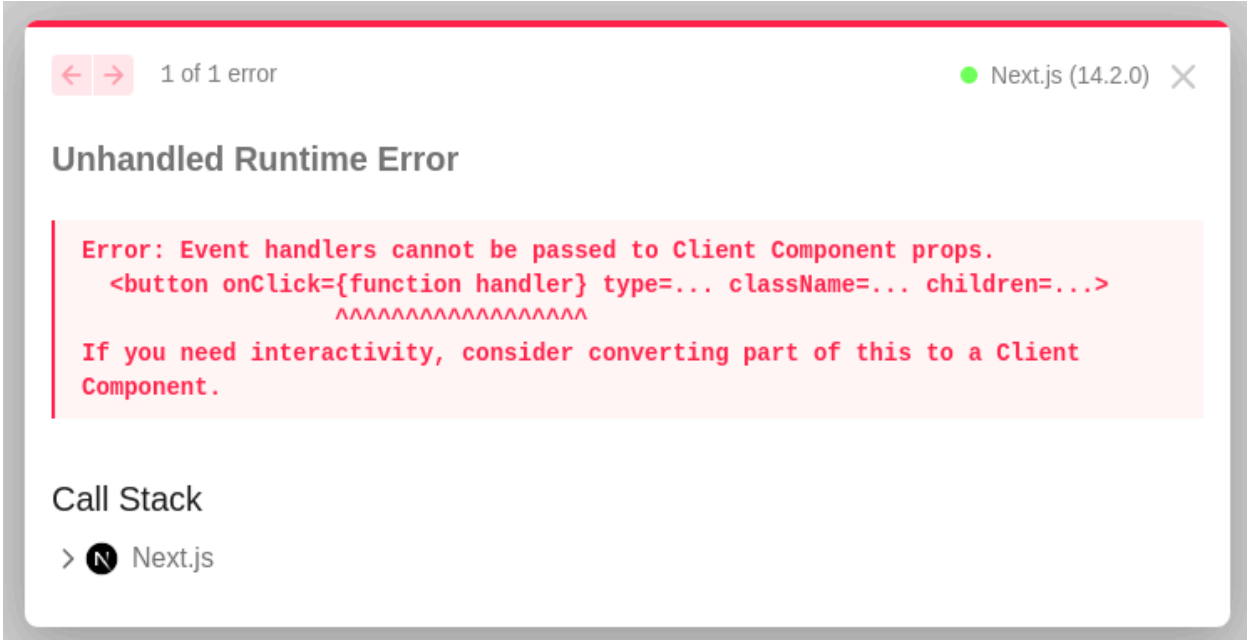
```
<button type="button" className="mt-8 w-full text-white bg-gray-800
focus:ring-4 focus:ring-gray-300 font-medium rounded-lg text-sm px-5
py-2.5 me-2 mb-2">Sign in</button>
```

Added

```
function handler(){
        console.log("hi there")
    }
```

```
<button onClick={handler} type="button" className="mt-8 w-full text-white bg-gray-800 focus:ring-4 focus:ring-gray-300 font-medium rounded-lg text-sm px-5 py-2.5 me-2 mb-2">Sign in</button>
```

We get an error message



Something is present in onclick which result in the error message.

Client and server components

Reference: https://nextjs.org/learn/react-foundations/server-and-client-components

NextJS expects you to identify all your components as either client or server

As the name suggests

1. Server components are rendered on the server
2. Client components are pushed to the client to be rendered

By default, all components are **server** components.

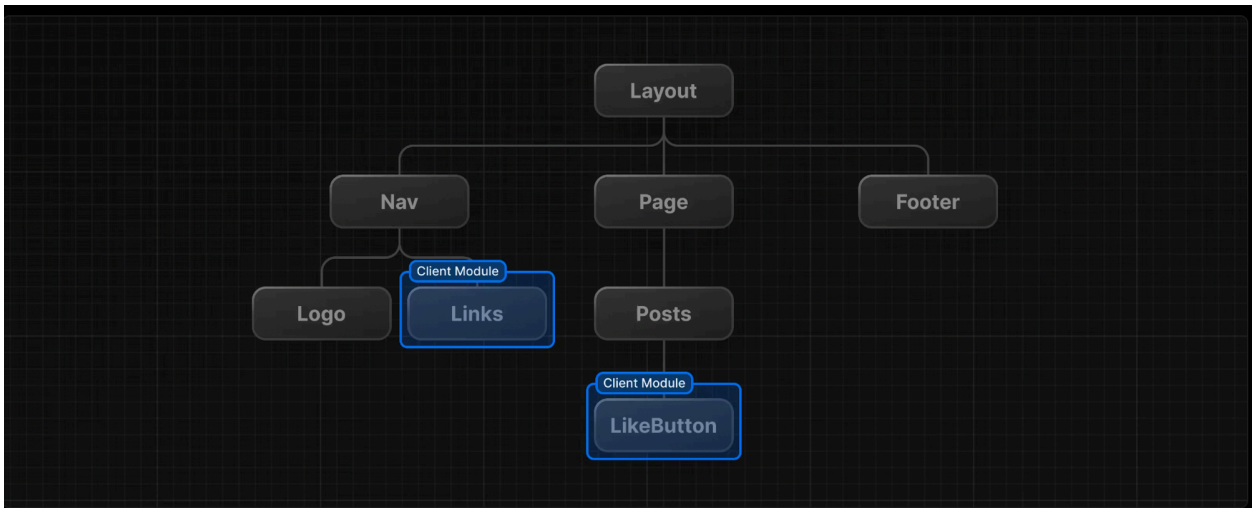
If you wan't to mark a component as a client component, you need to add the following to the top of the component -

"use client"

When should you create client components?

1. Whenever you get an error that tells you that you need to create a client component
2. Whenever you're using something that the server doesn't understand (useEffect, useState, onClick…)

Rule of thumb is to defer the client as much as possible



Some more ref: https://github.com/vercel/next.js/discussions/43153

We will use keyword "use client" on the top of SigninComponent , we can further separate button into separate container