



Step 1 – NextJS Intro, Pre-requisites

Pre-requisites

You need to understand basic Frontend before proceeding to this track.

You need to know what **React** is and how you can create a simple application in it

NextJS Intro

NextJS was a framework that was introduced because of some **minor inconveniences** 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-

[Back to home](#)[Jump To ↗](#)[◀ Prev](#)[Next ▶](#)[Go to Top ↑](#)

1. not exactly true today because of React Server components



NextJS (Client side) 1 of 14 > on why

4. Waterfalling problem

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

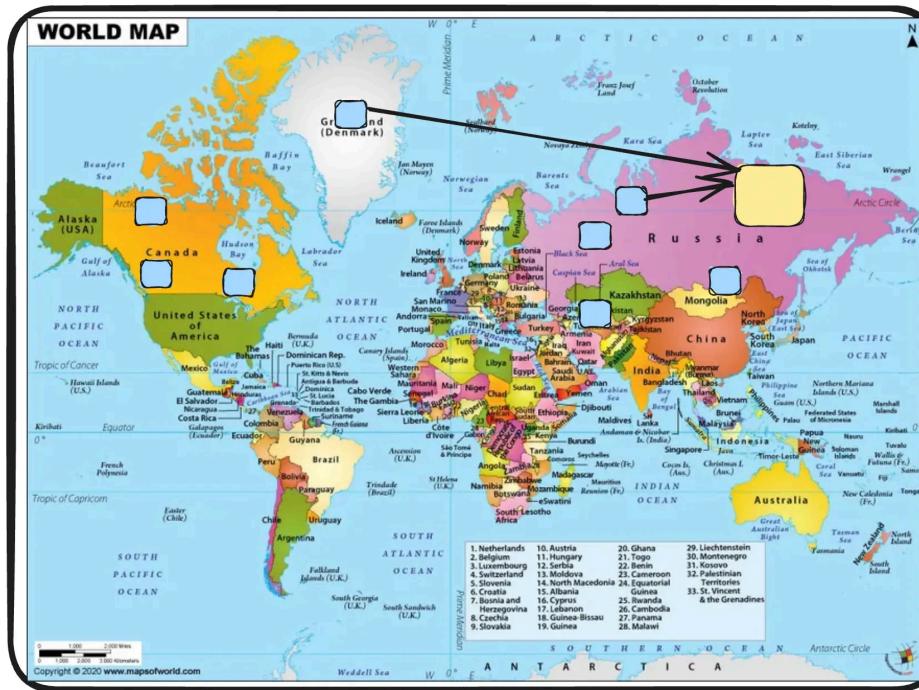
Step 2 – SEO Optimisation

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

NextJS (Client side) 1 of 14



Blue square: Crawler

Yellow square: Your react server

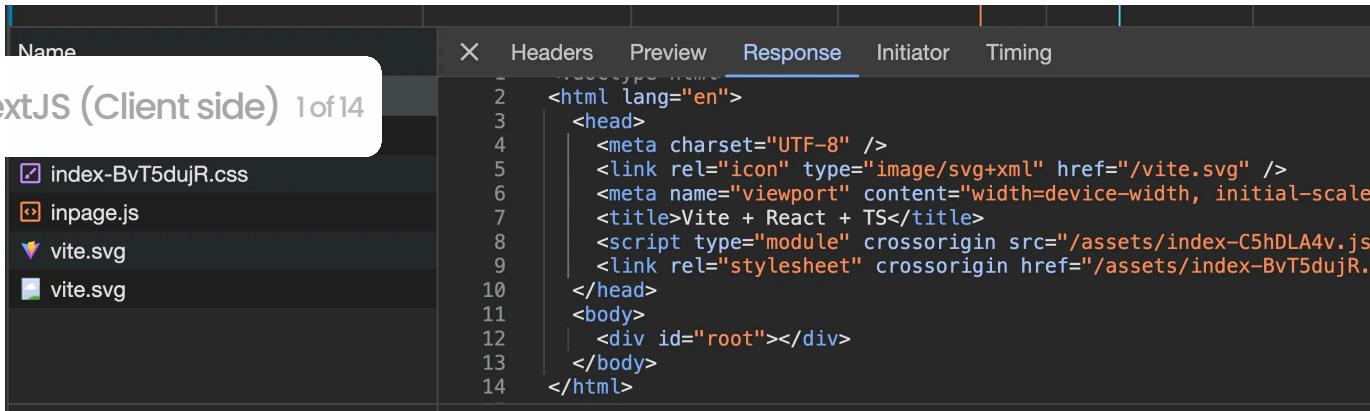


While Googlebot can run JavaScript, dynamically generated content is harder for the scraper to index

Try visiting a react website

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

...next.js



The screenshot shows the Network tab of a browser's developer tools. A request for 'NextJS (Client side) 1 of 14' is selected. The 'Response' tab is active, displaying the following HTML code:

```
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6     <meta name="viewport" content="width=device-width, initial-scale=1" />
7     <title>Vite + React + TS</title>
8     <script type="module" crossorigin src="/assets/index-C5hDLA4v.js" />
9     <link rel="stylesheet" crossorigin href="/assets/index-BvT5dujR.css" />
10    </head>
11    <body>
12      <div id="root"></div>
13    </body>
14  </html>
```

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.

Step 3 – Waterfalling problem

Let's say you built a blogging website in react, what steps do you think the



NextJS (Client side) 1 of 14

Medium

New

h

Anonymous • 2nd Feb 2024

my first blog

this is my first blog...

1 minute(s) read

Anonymous • 2nd Feb 2024

my first blog

this is my first blog...

1 minute(s) read

Anonymous • 2nd Feb 2024

my first blog

this is my first blog...

1 minute(s) read

Anonymous • 2nd Feb 2024

my new blog

my bnew blog...

1 minute(s) read

Anonymous • 2nd Feb 2024

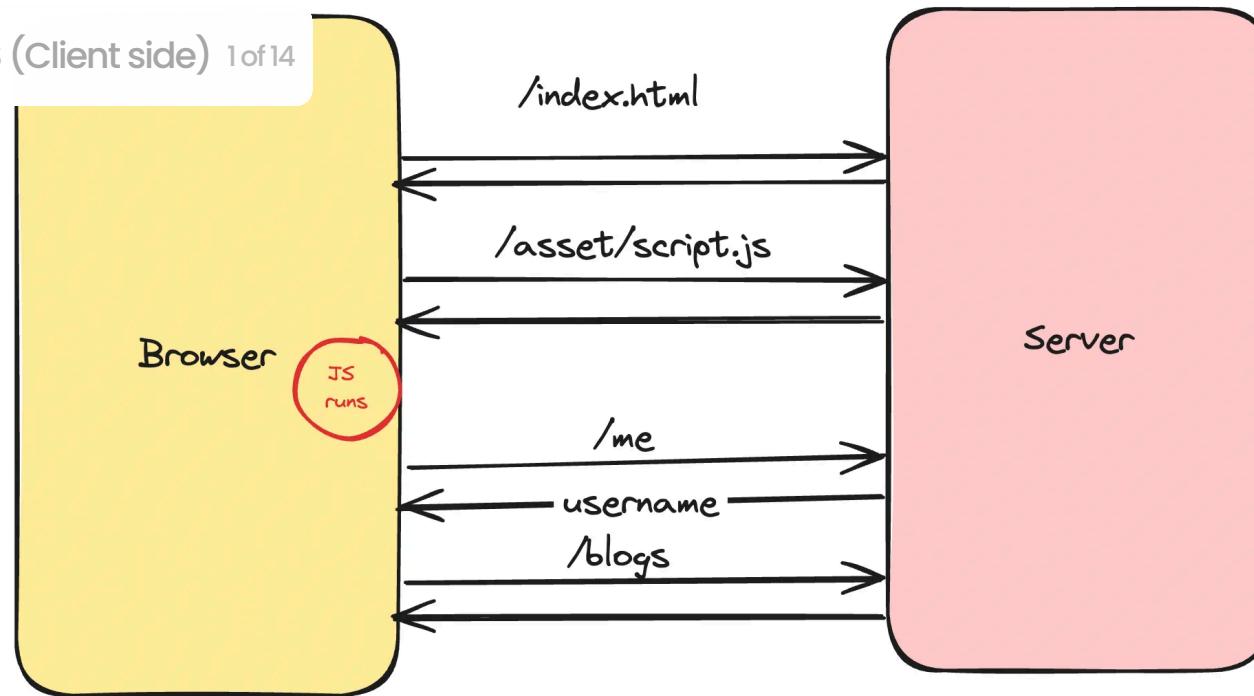
ha111sdda@gmail.com

m123123123...

1 minute(s) read

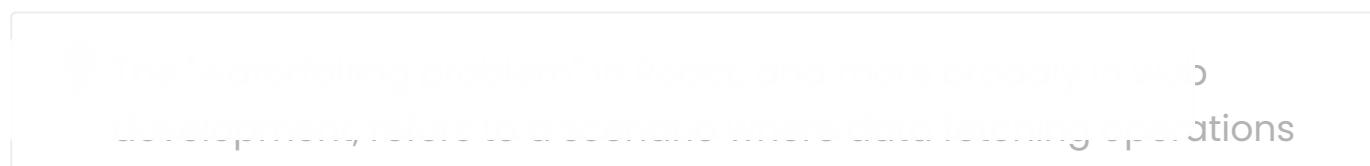


NextJS (Client side) 1 of 14



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)



are chained or dependent on each other in a way that leads to
NextJS (Client side) 1 of 14



What does nextjs provide you?



Step 4 – Next.js offerings

Next.js provides you the following **upsides** over React

1. Server side rendering – Gets rid of SEO problems
2. API routes – Single codebase with frontend and backend
3. File based routing (no need for react-router-dom)
4. Bundle size optimisations, Static site generation
5. Maintained by the Vercel team

Downsides -



NextJS (Client side) 1 of 14
ed 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

Step 5 – Let's bootstrap a simple Next app

npx create-next-app@latest



```
→ Projects npx create-next-app@latest
          Following packages:
NextJS (Client side) 1 of 14

✓ What is your project named? ... next-app
✓ Would you like to use TypeScript? ... No / Yes
✓ Would you like to use ESLint? ... No / Yes
✓ Would you like to use Tailwind CSS? ... No / Yes
✓ Would you like to use `src/` directory? ... No / Yes
✓ Would you like to use App Router? (recommended) ... No / Yes
✓ Would you like to customize the default import alias (@/*)? ... No / Yes
Creating a new Next.js app in /Users/harkiratsingh/Projects/next-app.
```

Using npm.

Initializing project with template: app-tw

Installing dependencies:

- react
- react-dom
- next

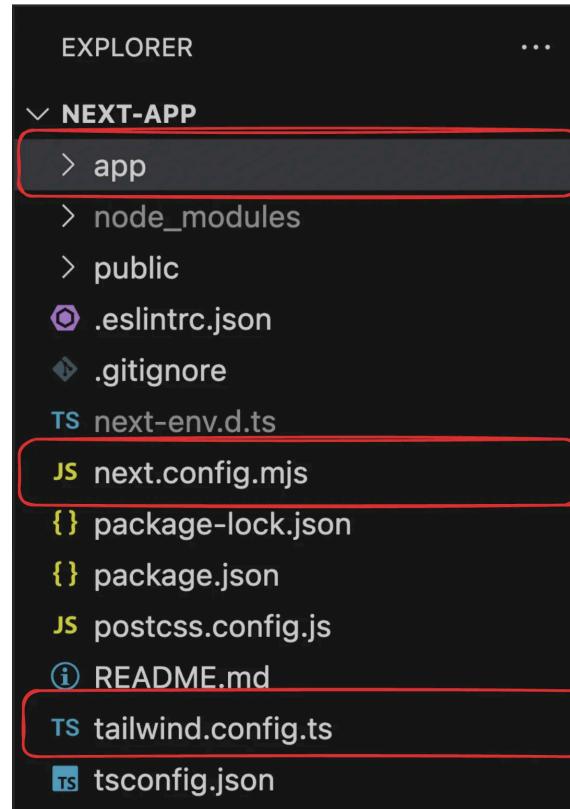
Installing devDependencies:

- typescript
- @types/node
- @types/react
- @types/react-dom
- autoprefixer
- postcss
- tailwindcss
- eslint
- eslint-config-next

File structure



NextJS (Client side) 1 of 14



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

Bootstrap the project



NextJS (Client side) 1 of 14

Understanding routing in Next

NextJS (Client side) 1 of 14 

<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 Next.js

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

This means that the way you create your files, describes what renders on a route

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>
  );
}
```

1. Start the application locally

```
npm run dev
```

A screenshot of a browser window showing a Next.js client-side project structure. The address bar indicates the URL is `localhost:3000/signup`. The page title is "NextJS (Client side) 1 of 14". The main content area displays a file tree under the "app" directory, specifically the "signup" folder. Inside "signup", there are files: "page.tsx" (highlighted with a red box), "favicon.ico", "# globals.css", "layout.tsx", and another "page.tsx" (also highlighted with a red box). Red arrows point from both highlighted "page.tsx" files to the URL "localhost:3000/signup" and "localhost:3000/", respectively. The file tree also shows a "..." entry at the top right.

Final folder structure

A screenshot of a terminal or code editor showing the final folder structure of a Next.js app. The structure is as follows:

```
app
  └── signup
      ├── favicon.ico
      ├── # globals.css
      ├── layout.tsx
      └── page.tsx
```

The "page.tsx" file in the root "signup" folder is highlighted with a red box and has a red arrow pointing to the URL "localhost:3000/signup". Another "page.tsx" file inside "layout.tsx" is also highlighted with a red box and has a red arrow pointing to the URL "localhost:3000/". The file "globals.css" is preceded by a yellow star icon. The file "layout.tsx" is preceded by a yellow hash symbol icon. The "favicon.ico" file is preceded by a yellow starburst icon.

Step 7 – Prettify the signin page



NextJS (Client side) 1 of 14

Let's replace the signup page with a prettier one

```
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-
        <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="password" />
          <button type="button" className="mt-8 w-full text-white bg-gray-900 py-2 px-4 rounded">Sign in</button>
        </div>
      </a>
    </div>
```

```
interface LabelledInputType {
```

NextJS (Client side) 1 of 14

```
    type?: string;
}
```

```
function LabelledInput({ label, placeholder, type }: LabelledInputType) {
  return <div>
    <label className="block mb-2 text-sm text-black font-semibold pt-4">
      <input type={type || "text"} id="first_name" className="bg-gray-50 bor
    </div>
}
```



The image shows a 'Sign in' form. At the top center is the text 'Sign in'. Below it is a 'Username' label followed by an input field containing the email 'harkirat@gmail.com'. Below the input field is a 'Password' label.

Step 8 – Server side rendering

Let's try exploring the response from the server on the `/signup` route

1. Run `npm run dev`
2. Visit `http://localhost:3000/signup`
3. Notice the response you get back in your HTML file

The screenshot shows a browser developer tools Network tab for a Next.js application. The request 'hi from the signup page' is selected. The response body contains the HTML code for a signup page, including the text 'hi from the signup page'. A red arrow points to this specific text in the response body.

Name	Headers	Preview	Response	Initiator	Timing	Cookies
webpack-hmr			1 <!DOCTYPE html>			
signup			- <html lang="en">			
c9a5bc6a7c948f...			- <head>			
layout.css?v=17...			- <meta charSet="utf-8"/>			
webpack.js?v=1...			- <meta name="viewport" content="width=device-width, initial-scale=1"/>			
main-app.js?v=1...			- <link rel="preload" href="/_next/static/media/c9a5bc6a7c948fb0-s.p.wof...			
app-pages-inter...			- <link rel="stylesheet" href="/_next/static/css/app/layout.css?v=1709368...			
favicon.ico			- <link rel="preload" as="script" fetchPriority="low" href="/_next/static/chunks/main-app.js?v=1709368676760" async="...">			
			- <script src="/_next/static/chunks/main-app.js?v=1709368676760" async="..."></script>			
			- <title>Create Next App</title>			
			- <meta name="description" content="Generated by create next app"/>			
			- <link rel="icon" href="/favicon.ico" type="image/x-icon" sizes="16x16" />			
			- <meta name="next-size-adjust"/>			
			- <script src="/_next/static/chunks/polyfills.js" noModule=""></script>			
			- </head>			
			- <body class="className_aeaf075u">			
			- <div>hi from the signup page</div>			
			- <script src="/_next/static/chunks/webpack.js?v=1709368676760" async="...">			
			- <script>			
			- (self._next_f = self._next_f []).push([0]);			
			- self._next_f.push([2, null])			
			- </script>			
			- <script>			

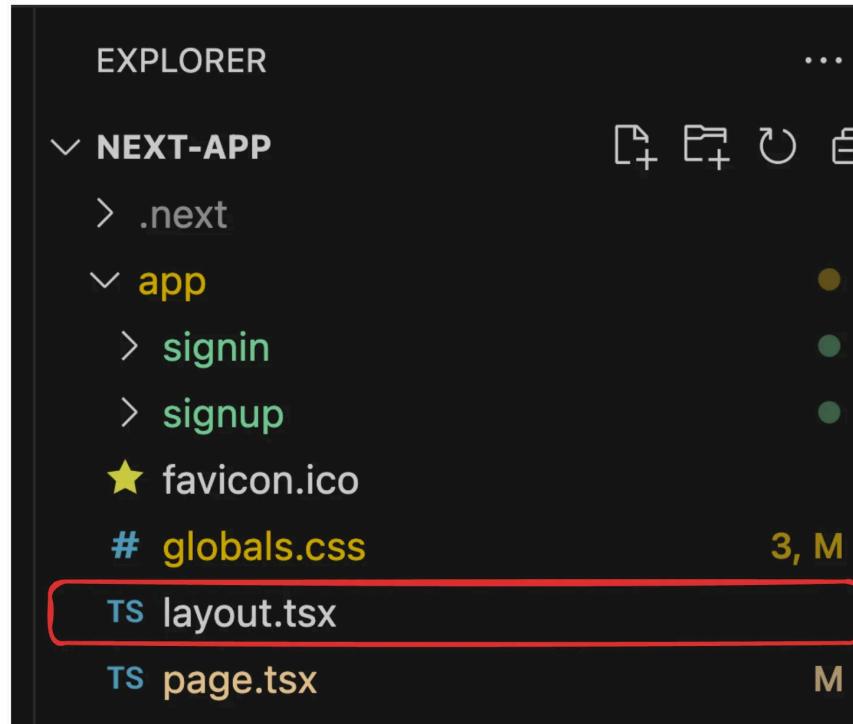
Now if **GoogleBot** tries to scrape your page, it'll understand that this is a **signup page** without running any Javascript.

The first **index.html** file it gets back will have context about the page since it was **server side rendered**

You'll notice a file in your `app` folder called `layout.tsx`



NextJS (Client side) 1of14 yes (Ref <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.

Let's explore `layout.tsx`

NextJS (Client side) 1 of 14

```
1 ✓ import type { Metadata } from "next";
  from "next/font/google";
  s.css"; import styles
2
3
4
5   const inter = Inter({ subsets: ["latin"] });
6
7 ✓ export const metadata: Metadata = {
8   title: "Create Next App",
9   description: "Generated by create next app",
10};
11
12 ✓ export default function RootLayout({
13   children,
14 }: Readonly<{
15   children: React.ReactNode;
16 }>) {
17 ✓   return (
18 ✓     <html lang="en">
19 ✓       <body className={inter.className}>
20 ✓         {children}
21       </body>
22     </html>
23   );
24 }
25
```

import styles

metadata of the page

Adding font globally

The page handler component

Assignment

Try adding a simple Appbar

return ()

NextJS (Client side) 1 of 14 g="en">

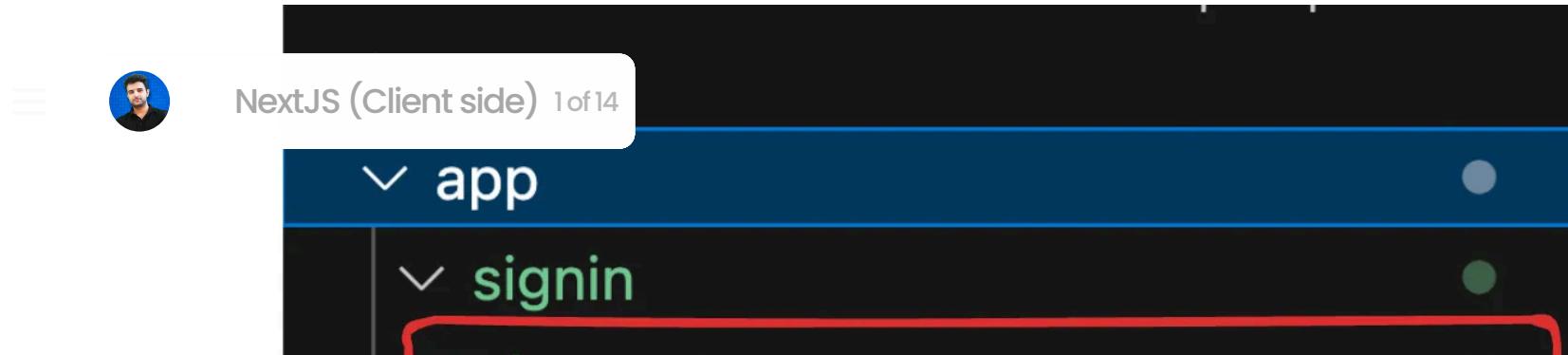
```
<body className={inter.className}>
  <div className="p-4 border-b">
    Medium
  </div>
  {children}
</body>
</html>
);
```

Medium

hi there

Step 10 ~ 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`



Step 11 – Merging routes

What if you 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



You can access the routes at

<http://localhost:3000/auth/signup> and <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.



NextJS (Client side) 1 of 14



Step 12 – 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

1. Create a new folder called `components` in the root of the project
2. Add a new component there called `Signin.tsx`
3. Move the signin logic there
4. Render the `Signin` component in `app/(auth)signin/page.tsx`

Solution



```
        Signin() {  
NextJS (Client side) 1of14 <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-md transition duration-300 ease-in-out hover:bg-gray-100 focus:outline-none focus:ring focus:ring-gray-300 focus:ring-opacity-40">  
            <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" type="text" value="harkirat@gmail.com"/>  
                    <LabelledInput label="Password" type={ "password" } placeholder="password" value="password"/>  
                    <button type="button" className="mt-8 w-full text-white bg-gray-900 py-2 px-4 rounded-lg transition duration-300 ease-in-out hover:bg-gray-800 focus:outline-none focus:ring focus:ring-gray-300 focus:ring-opacity-40">  
                        Sign in  
                    </button>  
                </div>  
            </div>  
        </a>  
    </div>  
</div>  
        }
```

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

```
    type) {  
return <div>
```



NextJS (Client side) 1 of 14

```
<label className="block mb-2 text-sm text-black font-semibold pt-1 pb-1" htmlFor={id} style={{ border: `1px solid ${color}` }}>{label}</label>
```

```
<input type="text" id="first_name" value={value} onChange={handleInput} style={{ width: '100%', height: '1.5em', padding: '0.5em' }} />
```

```
<div style={{ display: 'flex', gap: '10px', margin: '10px 0' }}>
```

```
<button style={{ border: '1px solid #ccc', padding: '5px 10px', cursor: 'pointer' }} onClick={handleClick}>Sign In</button>
```

```
<button style={{ border: '1px solid #ccc', padding: '5px 10px', cursor: 'pointer' }} onClick={handleClick}>Sign Up</button>
```

```
</div>
```

▼ app/(auth)/signin.tsx

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

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

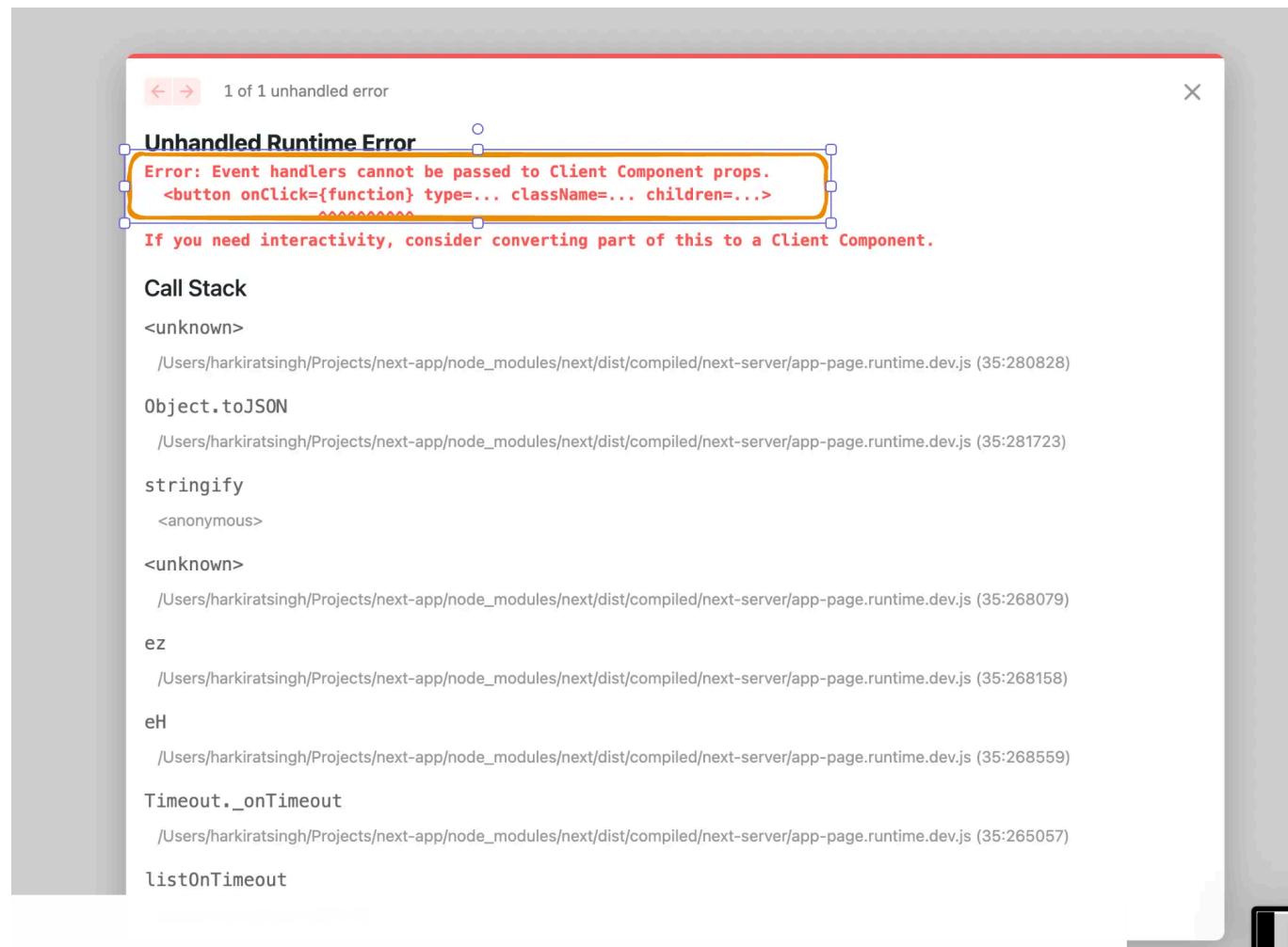
Step 13 – Add a button onclick handler

Now try adding a `onClick` handler to the `button` on the sign in page

NextJS (Client side) 1 of 14 "er clicked on signin")
JJ type="button" className="mt-8 w-full text-white bg-gray-800 focus:ring



You will notice an error when you open the page





NextJS (Client side) 1 of 14 happening here? Let's explore in the next slide

Step 14 – Client and server components

Ref - <https://nextjs.org/learn/react-foundations/server-and-client->



NextJS (Client side) 1 of 14

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 want to mark a component as a **client** component, you need to add the following to the top of the component -

"use client"

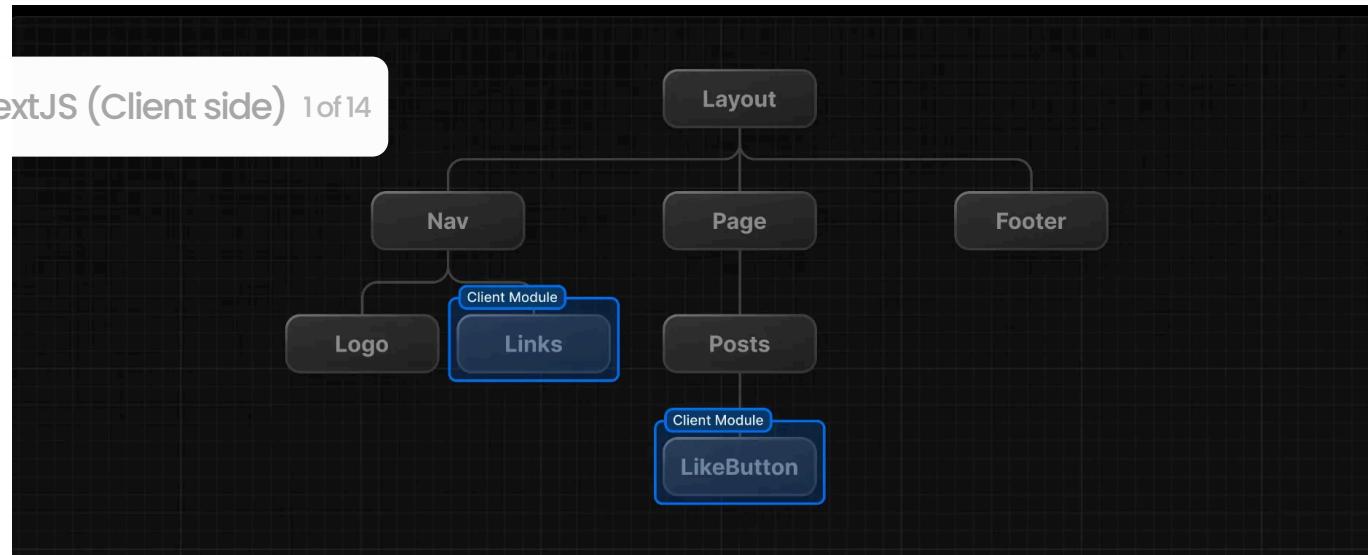


When should you create **client components** ?

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...)



NextJS (Client side) 1 of 14



Assignment

Try updating `components/Signin.tsx` to make it a client component

You will notice that the error goes away

Some nice readings -

<https://github.com/vercel/next.js/discussions/43153>