International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 7 Issue 4, July-August 2023 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

NextJs File-Based Routing - A Review

Krutika Patil

Master of Science in Computer Science the University of Texas at Dallas, Tracy, CA, USA

ABSTRACT

Next Js is quickly gaining popularity as a full-stack React Js framework and a React framework for Production. Next Js helps us build efficient websites with solid Search Engine Optimization. Next.js, developed by Vercel, has become a popular framework for developing React applications. It provides useful features such as server-side rendering, static site generation, and an intuitive file-based routing system, revolutionizing how developers construct their applications. This paper delves into the intricacies of the file-based routing system in Next.js, discussing its principles, benefits, potential issues, and use cases bolstered by tangible coding examples.

KEYWORDS: *NextJs, file-based routing, full-stack, web development, react-router, reactJs, javascript*

How to cite this paper: Krutika Patil "NextJs File-Based Routing - A Review" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-7 | Issue-4, August 2023, pp.40-42, URL: www.ijtsrd.com/papers/ijtsrd58607.pdf

IJTSRD58607

INTRODUCTION

As per the NextJs official website, it is a full-stack React and React framework for Production. Let's dive into what each of these statements means. NextJs brings a few key features—file-based routing, server-side rendering, API integration, and static site generation. In a contemporary Js framework like React, we must add a dependency like 'react-router' to get routing capability. NextJs supports this out-of-the-box, eliminating the need for a routing package. NextJs also has a strong ability of "server-side rendering." The pages are rendered at the server and not the client, so the initial load is quick without needing a loading state before the render. The ability to generate content on the server also helps with better Search Engine Optimization since the Search engine crawlers see the content that the users see on the page. Since all the content gets pre-rendered on the server, it is beneficial to Optimize Search Engines (SSO). NextJs also offers API integration out-of-the-box, which helps integrate the backend services efficiently. Due to these critical features, NextJs is a full-stack React framework. Also, the attributes we discussed are crucial to building a Production-ready React application. Hence, NextJs is also a React framework for Production. Next.js introduces a file-based routing system that leverages the file system to create application routes. In contrast to traditional routing systems where routes need to be defined manually, Next.js automatically routes files under the pages directory, significantly simplifying the routing process.

The file-based routing system deviates from most client-side routing libraries like react-router, which need an explicit definition of each route. Instead, it uses a convention-over-configuration paradigm, automatically establishing routes based on the file structure. Let us now discuss the important key feature of NextJs (file-based routing).

A. Nested Routes

Consider a freshly created Next Js project. The project structure consists of a "**pages**" folder at the root, as shown in **Figure 1** below. The contents of this folder will decide the routing of the application. We will have an "**index.js**" file at the root of this folder. This file renders the contents at the "/" path.

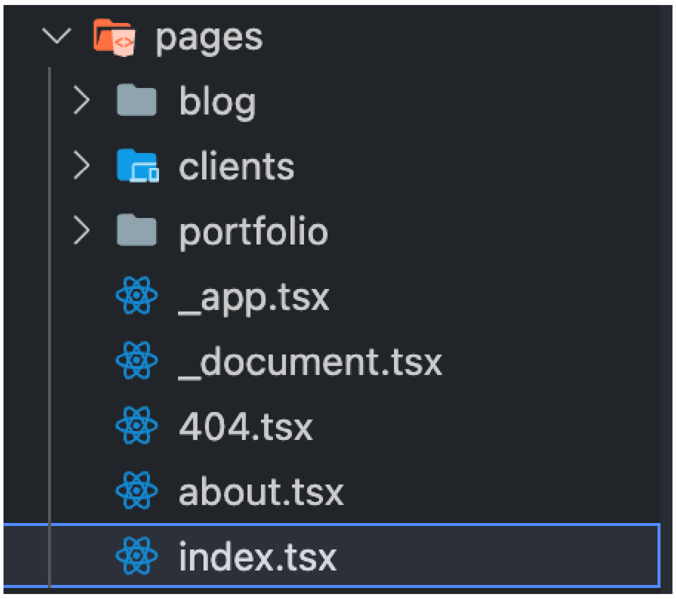


Figure 1: NextJs pages folder

To render the next route segment, we can in 2 ways:

1. Create a new folder with the name of the path segment and name the script inside "**index.js**." For example, if we would like to create a path "**/clients/**," we can create a folder named "**clients**" and then create a script inside this folder with the name "**index.js**."

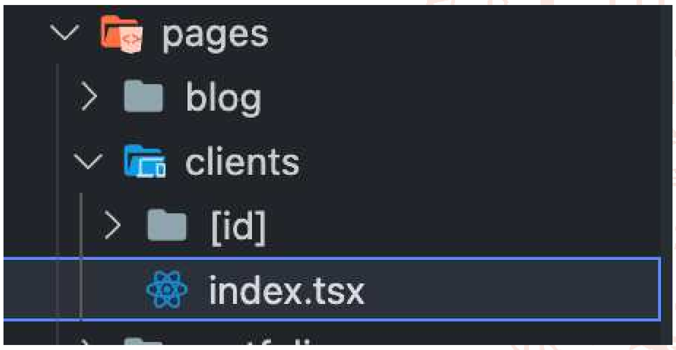


Figure 2: NextJs nested routes structure using folders

2. Create "**clients.js**" in the pages folder.

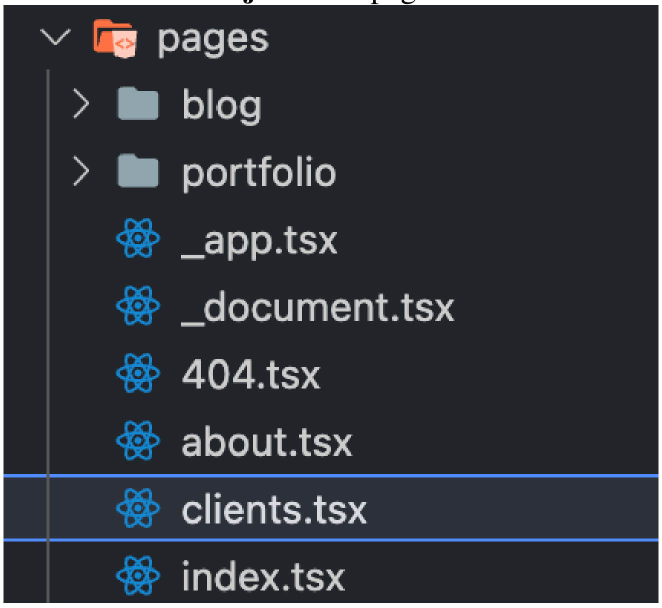


Figure 3: NextJs nested routes structure using file

Using both ways, we can render the content at "**/clients**." One advantage of the first method is that if we have nested routes for "**/clients**," we can create those. For example, if we need to render the content at "**/clients/client**," we can create a folder inside the "**clients**" folder with the name "**client**" and then have a "**index.js**" file inside the "**client**" folder.

Having nested routes is not possible with the second way.

B. Dynamic Routes

There are instances where we would like to render the routes dynamically. For example: suppose we have a structure like "**/clients/1**" where we display the client's details with id as 1; we wouldn't know beforehand all the possible values of the client to create those paths/ files. In such a case, NextJs offers a key feature in files-based routing known as "**dynamic routes**." Inside the "**clients**" folder, we can have a file named "**[id].js**." Please note the syntax. The keyword "**id**" is surrounded by square brackets, meaning the value of "**id**" will be determined dynamically. There is also another way to do this. We can create a new folder inside the "**clients**" folder with the name "**[id]**" and have an "**index.js**" file inside it.

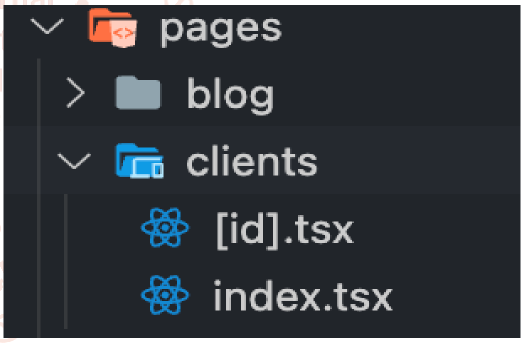
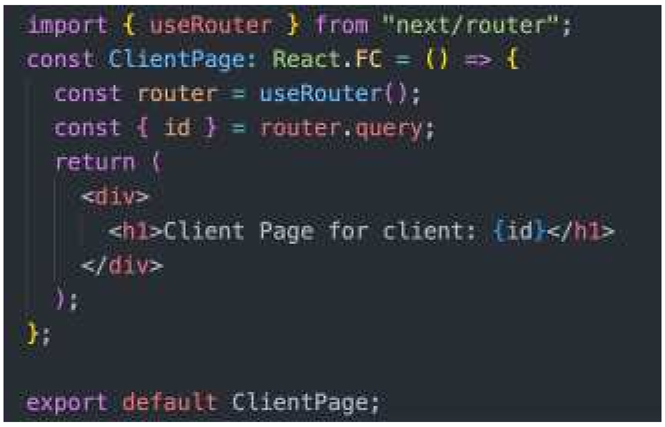


Figure 4: NextJs pages folder

The result of this method would also lead to us being able to render the content at "**clients/[id]**." On the **[id]** javascript page, we will be able to retrieve the value of "**id**" using the next js package "**next/router**." Using the code below in Figure 4:



```
import { useRouter } from "next/router";
const ClientPage: React.FC = () => {
  const router = useRouter();
  const { id } = router.query;
  return (
    <div>
      <h1>Client Page for client: {id}</h1>
    </div>
  );
};

export default ClientPage;
```

Figure 5: Using "next/router' to access the dynamic route query param

We import the “**use router**" library from the "**next/router**” package. Then, implementing the function "**userRouter**" gives us the router object. The "**query**" object inside the "**router**" object contains the keys to all the dynamic ids passed along the route. In this case, it will be the "id."

C. Catch-all Routes

For catch-all routes, Next.js allows using **[[...param]]**. It will match **/blob/a**, **/blog/a/b**, **/blog/a/b/c,** etc.

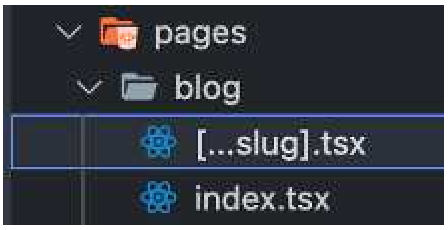
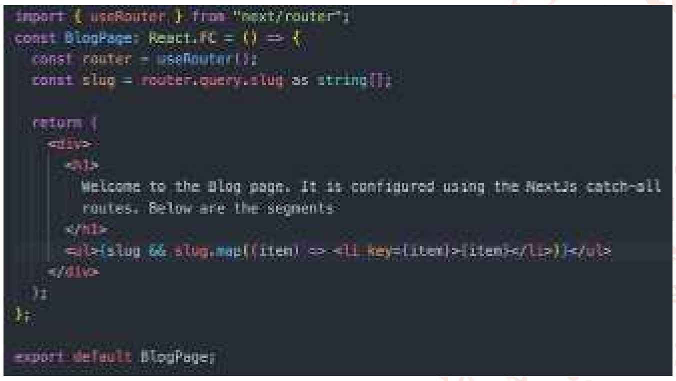


Figure 6: Catch-all route

We will be able to access the route segments using the '**next/router**' package again, as depicted in the below code:



```
import { useRouter } from "next/router";
const BlogPage: React.FC = () => {
  const router = useRouter();
  const slug = router.query.slug as string[];

  return (
    <div>
      <h1>
        Welcome to the Blog page. It is configured using the NextJs catch-all
        routes. Below are the segments
      </h1>
      <ul>{slug && slug.map((item) => <li key={item}>{item}</li>)}</ul>
    </div>
  );
};

export default BlogPage;
```

Figure 7: Retrieving all path segments of a catch-all route using ‘next/router’

D. Benefits

NextJs file-based routing offers several benefits:

Simplicity and Predictability: The automatic routing system significantly simplifies the routing process, making it predictable and intuitive. Developers can easily predict the URL structure of their application based on the file structure.

Code Splitting and Prefetching: Next.js automatically splits your code into separate bundles, so each page only loads what's necessary. It also prefetches link data on hover, enabling instant page transitions.

Dynamic and Catch-All Routes: Next.js supports dynamic routes, generating routes based on data. Catch-all routes enable matching paths that can have an arbitrary number of segments.

E. USE CASES

1. **Blog or Documentation Site**: With dynamic routes, you can easily set up a blog or documentation site where each post or document is a separate page.
2. **E-commerce Site**: You can use file-based routing to create product pages with dynamic routes, where the URL contains the product id or name.

CONCLUSION

Next.js's file-based routing is a game-changer, introducing a simple, intuitive routing system that offers developers significant flexibility in structuring their applications. Its ability to automatically route, split code, and prefetch data lead to performance benefits and faster development times. Whether you're creating a blog, an e-commerce site, or any other web application, Next.js's file-based routing can be a powerful tool to enhance your development process.

While this paper aimed to explore the core concepts of file-based routing in Next.js, we recommended further digging into the Next.js documentation to explore progressive ideas such as API routes, custom servers, and middleware, as they offer further opportunities for optimization and control in your applications.

Despite its many benefits, it's important to note that Next.js's file-based routing might only fit some use cases, particularly applications that require a more flexible or complex routing logic. As with any technology decision, it's crucial to carefully consider the trade-offs and ensure it aligns well with your project's requirements.

REFERENCES

[1] Krutika Patil, Sanath Dhananjayamurty Javagal, "React state management and side-effects – A Review of Hooks," IRJET Journal, volume 9, 2022, https://www.irjet.net/archives/V9/i12/IRJET-V9I1225.pdf.

[2] Krutika Patil "Redux State Management System - A Comprehensive Review" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-6 | Issue-7, December 2022, pp.1021-1027, URL: https://www.ijtsrd.com/papers/ijtsrd52530.pdf.

[3] https://www.udemy.com/course/react-the-complete-guide-incl-redux/learn/lecture/25599228?start=405#overview

[4] https://www.udemy.com/course/nextjs-react-the-complete-guide/learn/lecture/25145398#overview