Pages and Routing in NextJS

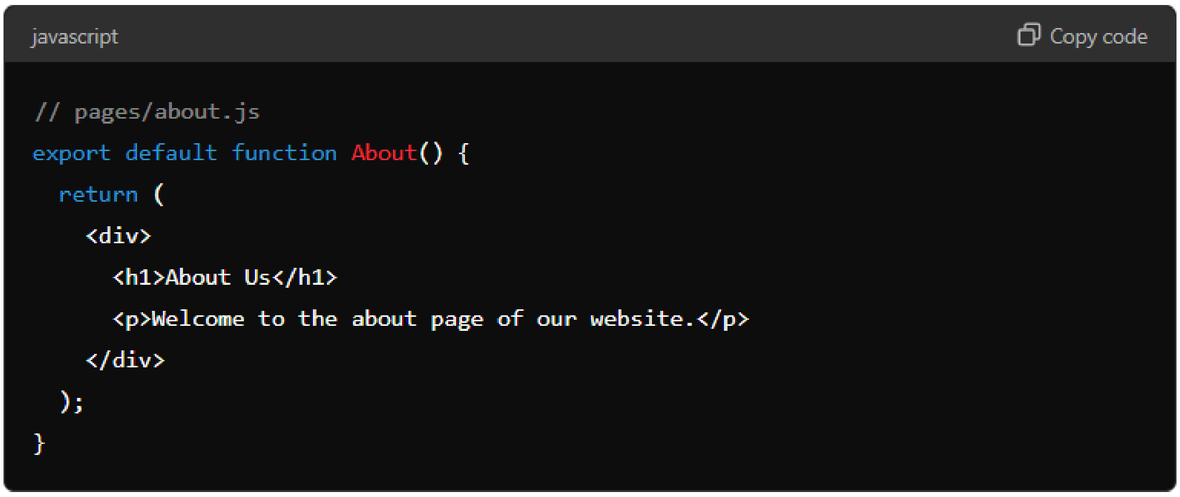
Next.js offers a straightforward and powerful file-based routing system that simplifies creating and managing pages in your application. The framework automatically generates routes based on the files and folders within the pages/ directory.

Creating Pages in Next.js

To create a page in Next.js, you simply add a new file to the pages/ directory. Each file corresponds to a route, and the name of the file (excluding the extension) becomes the URL path.

Basic Page Creation

1. **Create a New Page File:**
   - For example, to create an About page, add a file named about.js in the pages/ directory.
2. **Add Content to the Page:**
   - Inside about.js, define a React component that represents the content of this page.



```javascript
// pages/about.js
export default function About() {
  return (
    <div>
      <h1>About Us</h1>
      <p>Welcome to the about page of our website.</p>
    </div>
  );
}
```

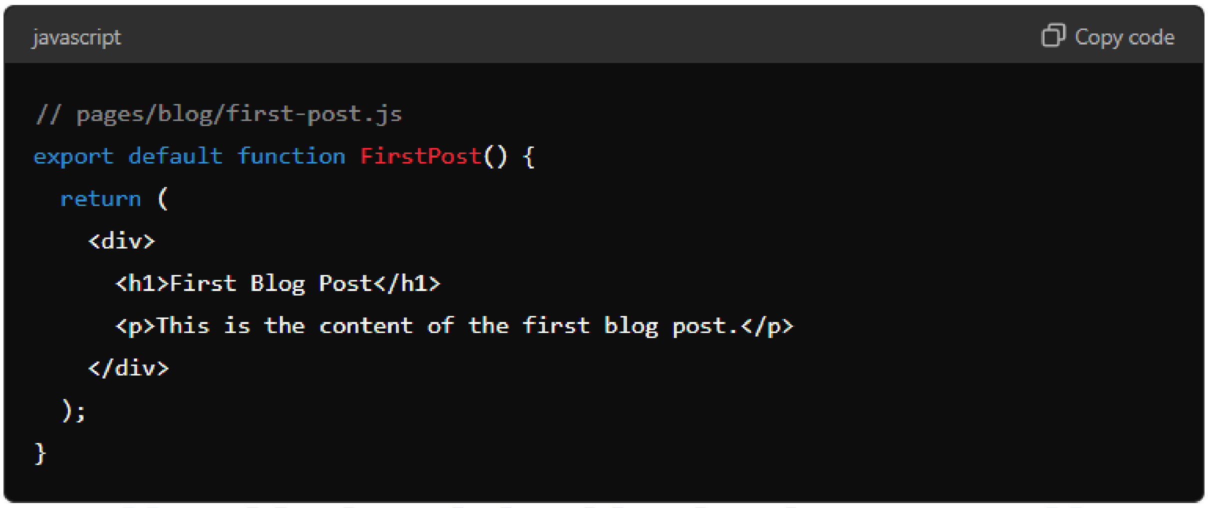
1. **Access the Page:**
   - Once the file is created, the page can be accessed by navigating to http://localhost:3000/about in your browser.

○

Nested Pages

To create nested routes, you can organize files within folders in the pages/ directory.

1. **Create a Folder and File:**
    - For example, to create a blog post route, you can create a folder blog inside pages and add a file first-post.js inside it.



```javascript
// pages/blog/first-post.js
export default function FirstPost() {
  return (
    <div>
      <h1>First Blog Post</h1>
      <p>This is the content of the first blog post.</p>
    </div>
  );
}
```

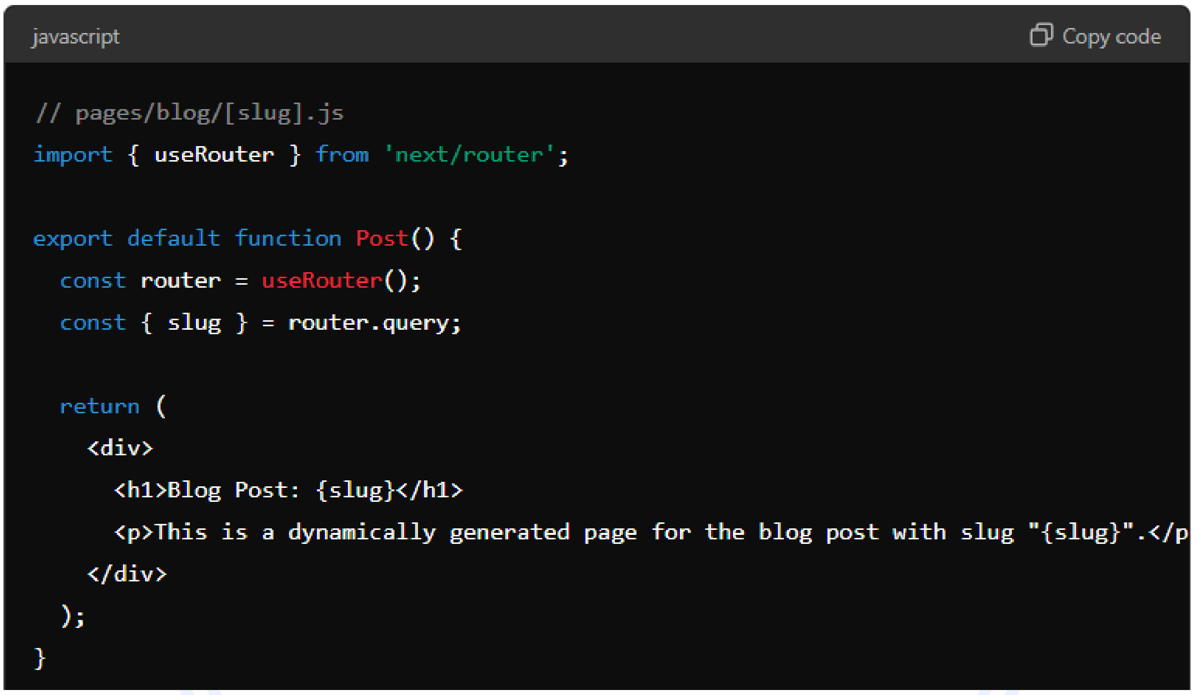
1. **Access the Nested Page:**
    - This page will be available at http://localhost:3000/blog/first-post.

Dynamic Routing

Dynamic routing allows you to create pages that can handle dynamic parameters in the URL, such as user IDs or slugs.

Creating Dynamic Routes

1. **Create a Dynamic Route File:**
    - To create a dynamic route, use square brackets in the filename. For instance, to create a dynamic route for blog posts, create a file named [slug].js inside the pages/blog directory.



```javascript
// pages/blog/[slug].js
import { useRouter } from 'next/router';

export default function Post() {
  const router = useRouter();
  const { slug } = router.query;

  return (
    <div>
      <h1>Blog Post: {slug}</h1>
      <p>This is a dynamically generated page for the blog post with slug "{slug}".</p
    </div>
  );
}
```

Access Dynamic Routes:

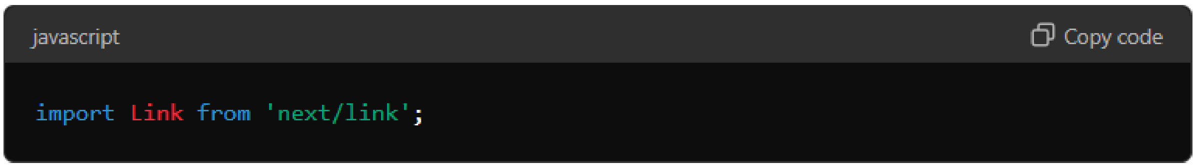
- This setup will allow you to access routes like http://localhost:3000/blog/my-first-post, where my-first-post is the dynamic segment (slug) of the URL.

Link Component for Navigation

Next.js provides a built-in Link component that enables client-side navigation between pages. It improves performance by preloading linked pages when they appear in the viewport, resulting in faster navigation.

Using the Link Component

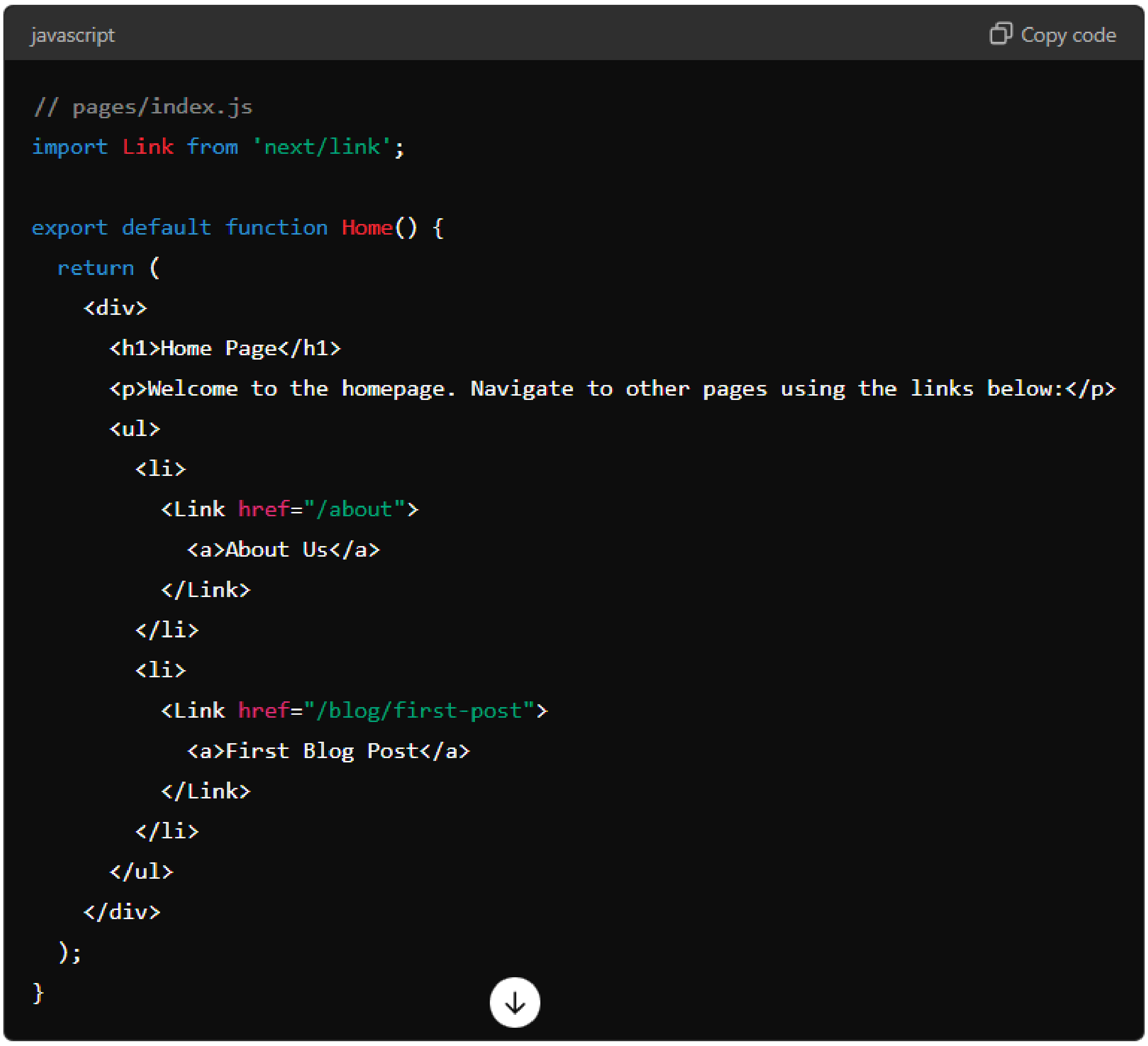
1. **Import the Link Component:**



```javascript
import Link from 'next/link';
```

2. **Create Navigation Links:**

- Use the Link component to link to different pages within your application.



```javascript
// pages/index.js
import Link from 'next/link';

export default function Home() {
  return (
    <div>
      <h1>Home Page</h1>
      <p>Welcome to the homepage. Navigate to other pages using the links below:</p>
      <ul>
        <li>
          <Link href="/about">
            <a>About Us</a>
          </Link>
        </li>
        <li>
          <Link href="/blog/first-post">
            <a>First Blog Post</a>
          </Link>
        </li>
      </ul>
    </div>
  );
}
```

3. **Client-Side Navigation:**

- Clicking on these links will navigate to the respective pages without a full page reload, providing a smoother user experience.

Summary

- **Pages and Routing:** Next.js uses a file-based routing system, where each file in the pages/ directory corresponds to a route.
- **Dynamic Routing:** Dynamic routes are created by using square brackets ([]) in filenames, allowing the page to capture dynamic parameters from the URL.
- **Link Component:** The Link component enables fast, client-side navigation between pages in your Next.js application.

These features make Next.js an efficient framework for building complex and highly dynamic web applications with ease.