20/05/24

useParams()

useParams was introduced in Next.js 13 and is used in conjunction with the new App Router (which uses the app directory) to access dynamic route parameters in Client Components.

Here's a recap of how you can use useParams in a Next.js 13 application with the App Router:

Step-by-Step Guide
Project Structure:
Ensure your project is using the new App Router by placing your files in the app directory. Here's a basic structure:

```
app/
  └── blogs/
      └── [id]/
          └── page.js
components/
  └── SinglePost.js
utility/

  └── request.js
```

2.Dynamic Route File:
In the app/blogs/[id]/page.js file, you can use useParams to access the dynamic id parameter.

```
"use client"; // Indicates that this is a Client Component

import { useEffect, useState } from "react";
import { useParams } from "next/navigation";
import { fetchBlog } from "@/utility/request";
import ArticleList from "@/components/SinglePost";

const Blogpage = () => {
  const { id } = useParams(); // Retrieve the dynamic id
  const [blog, setBlog] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchBlogData = async () => {
      if (!id) return;
      try {
        const blog = await fetchBlog(id);
        setBlog(blog);
      } catch (error) {
        console.error("Error fetching blog:", error);
      } finally {
        setLoading(false);
      }
    };

    fetchBlogData();
  }, [id]);

  if (!blog && !loading) {
    return <h1 className="text-center text-2xl font-bold">Blog Not found</h1>;
  }

  return <>{!loading && blog && <ArticleList blog={blog} />}</>;
};

export default Blogpage;
```

3. fetchBlog Utility Function:
   Ensure your fetchBlog function correctly fetches the blog data.

```
// utility/request.js
export const fetchBlog = async (id) => {
  const response = await fetch(`/api/blogs/${id}`);
  if (!response.ok) {
    throw new Error("Network response was not ok");
  }
  return await response.json();
};
```

4.ArticleList Component:
Create a component to display the blog details.

```
// components/SinglePost.js
const ArticleList = ({ blog }) => {
  return (
    <div>
      <h1>{blog.title}</h1>
      <p>{blog.content}</p>
    </div>
  );
};

export default ArticleList;
```

nsure Your API Route:
Ensure that your API route is set up to handle requests for individual blog posts

```
// pages/api/blogs/[id].js
export default async (req, res) => {
  const { id } = req.query;
  const blog = await getBlogById(id); // Replace with your data fetching logic
  if (!blog) {
    return res.status(404).json({ error: "Blog not found" });
  }
  res.status(200).json(blog);
};
```

Summary

- useParams: Used in the app directory to access dynamic route parameters in Next.js 13.
- Client Component: Indicate that your component is a Client Component with "use client" at the top.
- Data Fetching: Use the useEffect hook to fetch data based on the dynamic id.
- Utility Functions and Components: Organize your code into utility functions and reusable components for clean and maintainable code.