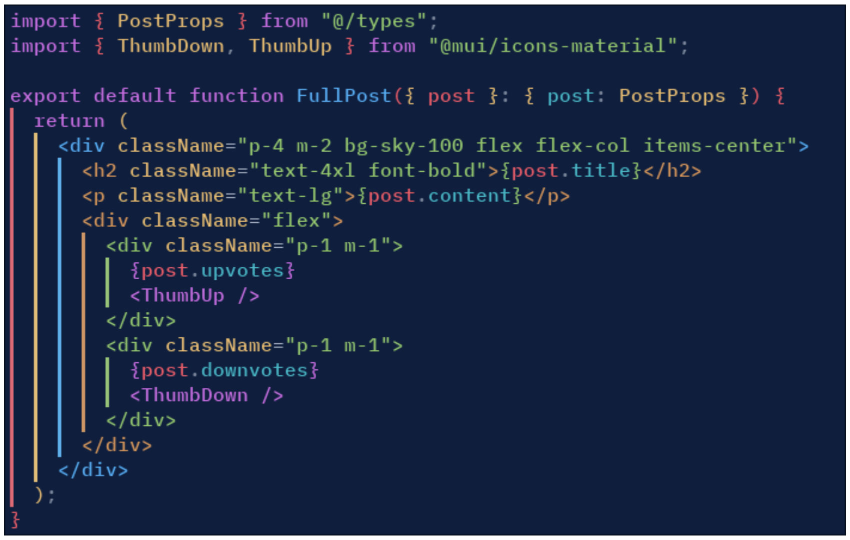
# Next.js: Dynamic Routing

To finish up our labs on Next.js we will be doing some dynamic routing, creating a unique page for each post.

**Task 1: Create a component to display a full post**

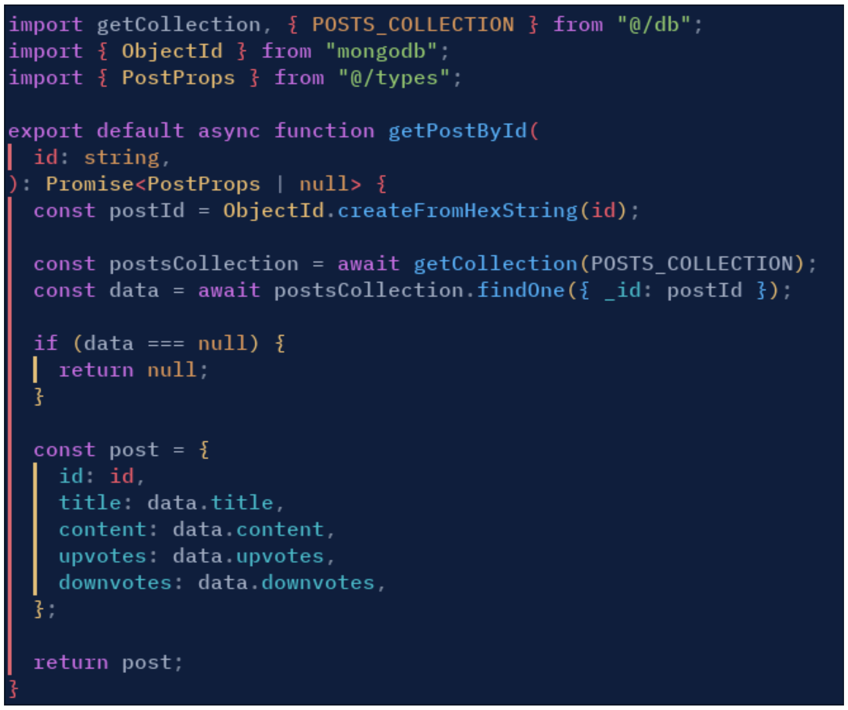
1. Create a component file called **full-post.tsx**



```tsx
import { PostProps } from "@/types";
import { ThumbDown, ThumbUp } from "@mui/icons-material";

export default function FullPost({ post }: { post: PostProps }) {
  return (
    <div className="p-4 m-2 bg-sky-100 flex flex-col items-center">
      <h2 className="text-4xl font-bold">{post.title}</h2>
      <p className="text-lg">{post.content}</p>
      <div className="flex">
        <div className="p-1 m-1">
          {post.upvotes}
          <ThumbUp />
        </div>
        <div className="p-1 m-1">
          {post.downvotes}
          <ThumbDown />
        </div>
      </div>
    </div>
  );
}
```

**Task 2: Write a server-side function to retrieve information on a single post**

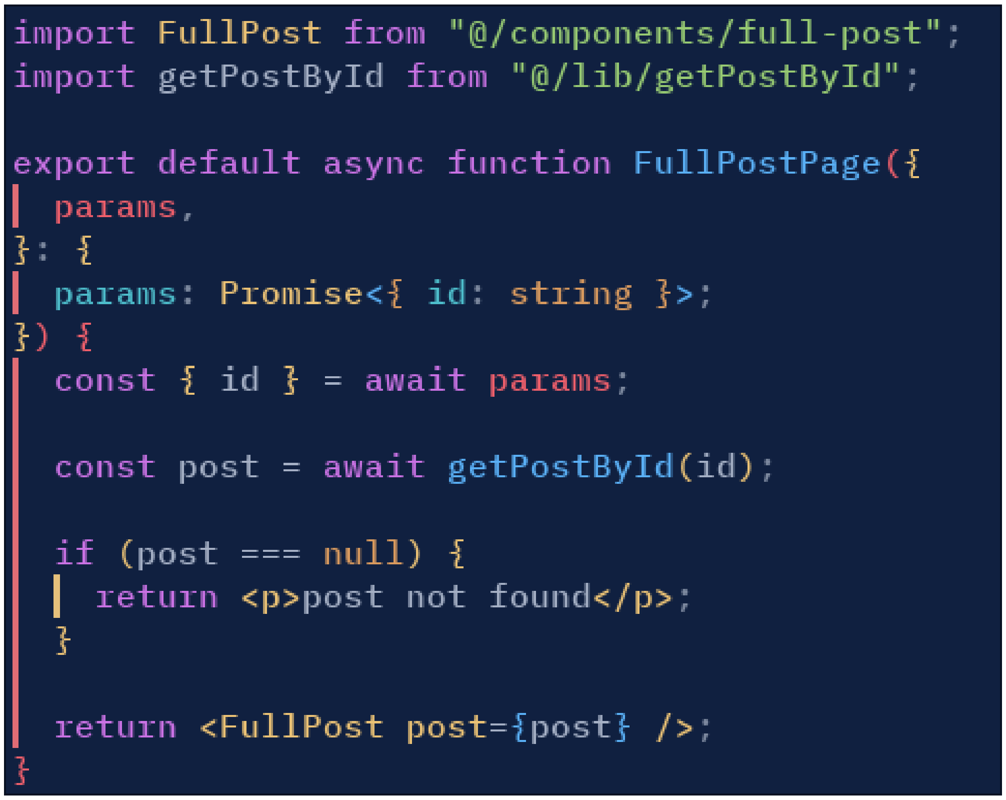
1. Create another **/lib** file called **getPostById.ts**
    a. Here we define a function that takes in a hex string of a post's id and returns the corresponding post



```ts
import getCollection, { POSTS_COLLECTION } from "@/db";
import { ObjectId } from "mongodb";
import { PostProps } from "@/types";

export default async function getPostById(
  id: string,
): Promise<PostProps | null> {
  const postId = ObjectId.createFromHexString(id);

  const postsCollection = await getCollection(POSTS_COLLECTION);
  const data = await postsCollection.findOne({ _id: postId });

  if (data === null) {
    return null;
  }

  const post = {
    id: id,
    title: data.title,
    content: data.content,
    upvotes: data.upvotes,
    downvotes: data.downvotes,
  };

  return post;
}
```

**Task 3: Create a dynamic route**

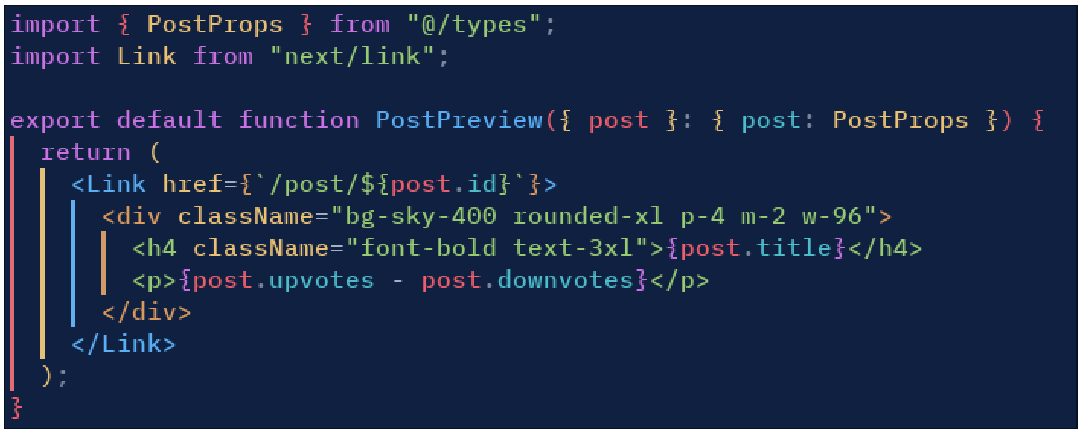
1. To create a dynamic route in Next.js, create a subdirectory of the **app** directory. The dynamic route directory should be named the appropriate variable surrounded by brackets
2. We will first create a subdirectory of **app** called **post**
    a. This can help prevent namespace/routing conflicts
3. Inside the **post** directory, create a **[id]** directory. This is the dynamic path/routing
    a. Note: dynamic routing may change with new versions of Next.js [https://nextjs.org/docs/app/building-your-application/upgrading/version-15](https://nextjs.org/docs/app/building-your-application/upgrading/version-15)
    b. This is the same as **/post/:id** in **react-router**
4. Inside the **app/post/[id]** directory, create a **page.tsx** file
    a. Here we await the dynamic input id from the path and try to display the corresponding post. If no post with that id exists, we display an appropriate message
        i. Another way to handle a post that does not exist could be to direct the user to another page, such as the page displaying the post previews. If you would like to implement this, look into the [redirect()](redirect()) function



```
import FullPost from "@/components/full-post";
import getPostById from "@/lib/getPostById";

export default async function FullPostPage({
  params,
}: {
  params: Promise<{ id: string }>;
}) {
  const { id } = await params;

  const post = await getPostById(id);

  if (post === null) {
    return <p>post not found</p>;
  }

  return <FullPost post={post} />;
}
```

**Task 4: Link to each post’s page**

1. Edit the **PostPreview** component so that it links to the full page of its post when clicked



```tsx
import { PostProps } from "@/types";
import Link from "next/link";

export default function PostPreview({ post }: { post: PostProps }) {
  return (
    <Link href={`/post/${post.id}`}>
      <div className="bg-sky-400 rounded-xl p-4 m-2 w-96">
        <h4 className="font-bold text-3xl">{post.title}</h4>
        <p>{post.upvotes - post.downvotes}</p>
      </div>
    </Link>
  );
}
```

**Optional Additional Work**

1. You may want to implement upvoting and downvoting. You may find the following documentation helpful in implementing this functionality
   a. updateOne()
   b. update operators
   c. $inc
2. You may want to implement replying. To do this, you will likely want to adjust the **PostProps** type. You will likely find the previously mentioned documentation helpful as well as the following
   a. $push
   b. $sort