

* **Index.js** is a special file name. It will act as the page to go to when users just write “/”
* For dynamic routes, we need to use [route Parameter].js format name.
* But Next.Js will check if we have a more concrete file for the exact value for dynamic routes match.

For example- we have [id].js inside the products folder. So, the route is /products/{id} (for java) and /products/:id (for node). Now. If we had another file inside the folder named list.js and we go to route /products/list it will not treat list as id, as there is more concrete file for this path.

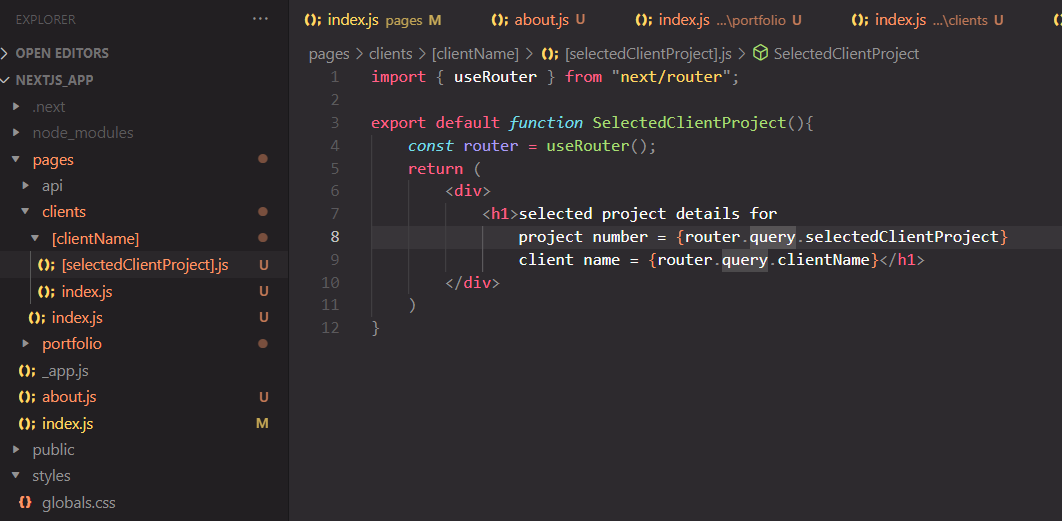
**Extracting the Dynamic Path data:**

* For functional component, we can use **useRouter** hook from next library. Here we get the router object returned from the useRouter() function.
* For class-based component there is a higher order component **withRouter**. Here we would get the router from the props.

This router object gives us access to various methods and data (like pathname, query etc) that we can use. For example- **router.query** gives us an object path data.

**Nested Dynamic Route and Paths:**

There can be scenario where we have multiple dynamic route and static path combination. For example- /clients/:id/:project . In such case, we can use dynamic folder name by wrapping it inside []. By using the router object, we can get data of all dynamic path data.



**Optional route parameters/Catch-All routes:**

Sometimes we might want optional paths, for example – **/posts /:year/:month /:title**. Here we might want different formats. Some examples might be

* /posts/2021
* /posts/2021/5
* /posts/2021/5/a-new-hope

To achieve this, there is a special syntax. It’s like the spread operator in JavaScript. For example, the file name can be […slug].js. This will catch all routes in a single array. How to use this might a be a good exercise for brainstorming.

**Link component:**

* Similar to react, if we use <a> for links, it will have a page refresh and the whole purpose of single page application is lost. So here we use Link component from next.

Important here is that Link is the default export of that file, so we don’t need curly braces.

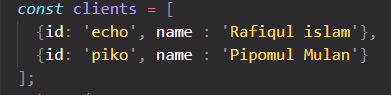
import Link from 'next/link'

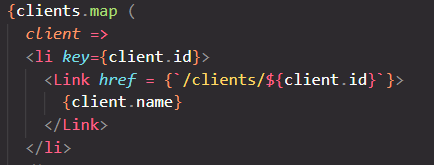
* It has some advantages, like it **pre fetch** the component data as soon as we hover over it.
* It has **replace** that makes it not being able to go back once component is loaded.
* One difference from react Link is that it has **href** attribute, instead of the to attribute.

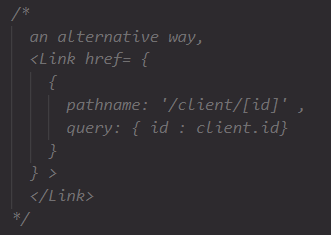
**Navigating to dynamic Routes:**

To navigate to **/clients/[id]**

* First get the list of the client, usually fetching from an api
* Then we can map the client array to a list either using the string literal or using an object







**Navigate Programmatically**

Let’s imagine, we want a button which will lead to some other link. For example, in cases of form or auth etc. Router object has some useful methods to help out with this.

* **router.push()** this method will take to the link passed in as argument. We can also pass an object (like the example in the dynamic route)
* **router.replace()** this is similar, just replace the current page in the history object. So, we can’t go back.



