Dynamic Routes

When you don't know the exact segment names ahead of time and want to create routes from dynamic data, you can use Dynamic Segments that are filled in at request time or prerendered at build time.

Convention

A Dynamic Segment can be created by wrapping a folder's name in square brackets: `[folderName]`. For example, `[id]` or `[slug]`.

Dynamic Segments can be access from `useRouter`.

Example

For example, a blog could include the following route `pages/blog/[slug].js` where `[slug]` is the Dynamic Segment for blog posts.

```
import { useRouter } from 'next/router';

export default function Page() {
  const router = useRouter();
  return <p>Post: {router.query.slug}</p>;
}
```

Route	Example URL	`params`
`pages/blog/[slug].js`	`/blog/a`	`{ slug: 'a' }`
`pages/blog/[slug].js`	`/blog/b`	`{ slug: 'b' }`

Route	Example URL	`params`
`pages/blog/[slug].js`	`/blog/c`	`{ slug: 'c' }`

Catch-all Segments

Dynamic Segments can be extended to **catch-all** subsequent segments by adding an ellipsis inside the brackets `[...folderName]`.

For example, `pages/shop/[...slug].js` will match `/shop/clothes`, but also `/shop/clothes/tops`, `/shop/clothes/tops/t-shirts`, and so on.

Route	Example URL	`params`
`pages/shop/[...slug].js`	`/shop/a`	`{ slug: ['a'] }`
`pages/shop/[...slug].js`	`/shop/a/b`	`{ slug: ['a', 'b'] }`
`pages/shop/[...slug].js`	`/shop/a/b/c`	`{ slug: ['a', 'b', 'c'] }`

Optional Catch-all Segments

Catch-all Segments can be made **optional** by including the parameter in double square brackets: `[[...folderName]]`.

For example, `pages/shop/[[...slug]].js` will **also** match `/shop`, in addition to `/shop/clothes`, `/shop/clothes/tops`, `/shop/clothes/tops/t-shirts`.

The difference between **catch-all** and **optional catch-all** segments is that with optional, the route without the parameter is also matched (`/shop` in the example above).

Route	Example URL	`params`
`pages/shop/[[...slug]].js`	`/shop`	`{}`
`pages/shop/[[...slug]].js`	`/shop/a`	`{ slug: ['a'] }`
`pages/shop/[[...slug]].js`	`/shop/a/b`	`{ slug: ['a', 'b'] }`
`pages/shop/[[...slug]].js`	`/shop/a/b/c`	`{ slug: ['a', 'b', 'c'] }`

Next Steps

For more information on what to do next, we recommend the following sections

Pages Router > ... > Routing

Linking and Navigating

Learn how navigation works in Next.js, and how to use the Link Component and `useRouter` hook.

Pages Router > ... > Functions

useRouter

Learn more about the API of the Next.js Router, and access the router instance in your page with the useRouter hook.