

Migrating from React Router

This guide will help you understand how to transition from [React Router](#) to [file-system based](#) routes with Next.js.

Using [next/link](#) and [next/router](#) will allow you to:

- Decrease bundle size by removing React Router as a dependency.
- Define your application routes through the file system.
- Utilize the latest improvements to the Next.js framework.

Basics

First, uninstall React Router. You'll be migrating to the built-in routing with Next.js.

```
npm uninstall react-router-dom
```

The `Link` component for performing client-side route transitions is slightly different from React Router.

```
// Before (React Router)
import { Link } from 'react-router-dom'

export default function App() {
  return <Link to="/about">About</Link>
}

// After (Next.js)
import Link from 'next/link'

export default function App() {
  return (
    <Link href="/about">
      <a>About</a>
    </Link>
  )
}
```

Most React applications that use React Router have a top-level navigation file, containing a list of routes. For example:

```
import { BrowserRouter as Router, Switch, Route } from 'react-router-dom'

export default function App() {
  return (
    <Router>
      <Switch>
        <Route path="/about">
          <h1>About</h1>
        </Route>
        <Route path="/blog">
          <h1>Blog</h1>
        </Route>
      </Switch>
    </Router>
  )
}
```

```

    <Route path="/">
      <h1>Home</h1>
    </Route>
  </Switch>
</Router>
)
}

```

With Next.js, you can express the same application structure in the file system. When a file is added to the `pages` directory it's automatically available as a route.

- `pages/about.js` → `/about`
- `pages/blog.js` → `/blog`
- `pages/index.js` → `/`

Nested Routes

In the example below, routes like `/blog/my-post` would render the `Post` component. If a slug was not provided, it would render the list of all blog posts.

```

import {
  BrowserRouter as Router,
  Switch,
  Route,
  useRouteMatch,
  useParams,
} from 'react-router-dom'

export default function Blog() {
  // Nested route under /blog
  const match = useRouteMatch()

  return (
    <Router>
      <Switch>
        <Route path={`/${match.path}/${slug}`}>
          <Post />
        </Route>
        <Route path={match.path}>
          <h1>All Blog Posts</h1>
        </Route>
      </Switch>
    </Router>
  )
}

function Post() {
  const { slug } = useParams()
  return <h1>Post Slug: {slug}</h1>
}

```

Rather than using the `:slug` syntax inside your `Route` component, Next.js uses the `[slug]` syntax in the file name for [Dynamic Routes](#). We can transform this to Next.js by creating two new files, `pages/blog/index.js` (showing all pages) and `pages/blog/[slug].js` (showing an individual post).

```
// pages/blog/index.js

export default function Blog() {
  return <h1>All Blog Posts</h1>
}

// pages/blog/[slug].js

import { useRouter } from 'next/router'

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

  return <h1>Post Slug: {slug}</h1>
}
```

Server Rendering

Next.js has built-in support for [Server-side Rendering](#). This means you can remove any instances of `StaticRouter` in your code.

Code Splitting

Next.js has built-in support for [Code Splitting](#). This means you can remove any instances of:

- `@loadable/server`, `@loadable/babel-plugin`, and `@loadable/webpack-plugin`
- Modifications to your `.babelrc` for `@loadable/babel-plugin`

Each file inside your `pages/` directory will be code split into its own JavaScript bundle during the build process. Next.js [also supports](#) ES2020 dynamic `import()` for JavaScript. With it you can import JavaScript modules dynamically and work with them. They also work with SSR.

For more information, read about [Dynamic Imports](#).

Scroll Restoration

Next.js has built-in support for [Scroll Restoration](#). This means you can remove any custom `ScrollToTop` components you have defined.

The default behavior of `next/link` and `next/router` is to scroll to the top of the page. You can also [disable this](#) if you prefer.

Learn More

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

Routing: [Learn more about routing in Next.js.](#)

Dynamic Routes: [Learn more about the built-in dynamic routes.](#)

Pages: [Enable client-side transitions with next/link.](#)