The useRouter() Hook

17 min

We've learned how to use the <Link> component to navigate between URL segments, but we'd often like to programmatically navigate away from a URL segment. For example, if a user tries to access a URL segment without being logged in, we can auto-redirect them to a sign-up page.

Next.js provides the useRouter() hook, a part of the next/navigation package, which we can use to programmatically perform SPA-like navigation in *client components*.
Calling useRouter() returns a router object containing the following methods:

- push(path): Navigates to path by pushing path to the top of the browser history stack.
- back(): Navigates back one entry in the browser history stack.
- forward(): Navigates forward one entry in the browser history stack.
- replace(path): Navigate to path by replacing the top of the browser history stack with path.

We can use these methods like:

```
'use client'  // required client directive
// other imports
import { useRouter } from "next/navigation";  // import

export default function Authentication() {
  const router = useRouter();  // get router object

  useEffect(() => {
    if(!isAuthenticated()) {
      router.replace("/sign-up")  // redirect user by replacing current path and sending user to
/sign-up
    }
  }, [])
  return (
      // use router.back() as callback.
      <button onClick={router.back}>Return</button>
  )
}
```

To recap, if we need to navigate using our application structure (like static links), we use the <Link> component, but if we need programmatic navigation (like redirecting), we use the useRouter() hook.

Let's practice using programmatic navigation in our application.

Instructions

1. Checkpoint 1 Passed

1.

You'll notice our application has a footer with 3 non-functional buttons: "Back", "Home", and "Forward".

Begin in the <Footer> component in /components/Footer by importing the useRouter hook and replacing router with the router object.

Hint

Make sure you import useRouter from next/navigation.

2. Checkpoint 2 Passed

2.

Next, add the "Back" functionality so the user is taken back to their last visited path.

Run the code using npm run dev in the terminal. Once you see a console message indicating "Ready," click the Refresh Browser button in the workspace browser bar.

If successful, you should be able to click on "Back" and return to your last visited path.

Hint

Recall that the router object has a back() method, which can be used to programmatically navigate 1 entry back in the browser history stack.

Enter the command inside the terminal window.

The window can be refreshed when it displays "Ready."

```
> dev
> next dev -p 4001

  ▲ Next.js 14.0.1
  - Local:        http://localhost:4001

 ✓ Ready in 1156ms
```

3. Checkpoint 3 Passed

3.

Next, add the "Forward" functionality so the user is taken forward 1 entry in the browser history stack.

Run the code using npm run dev in the terminal. Once you see a console message indicating “Ready,” click the Refresh Browser button in the workspace browser bar.

If successful, you should be able to click on “Forward” and be taken forward to your most recently visited path.

Hint

Recall that the router object has a forward() method, which can be used to programmatically navigate 1 entry forward in the browser history stack.

Enter the command inside the terminal window.

The window can be refreshed when it displays “Ready.”

```
> dev
> next dev -p 4001

  ▲ Next.js 14.0.1
  - Local:        http://localhost:4001

 ✓ Ready in 1156ms
```

4. Checkpoint 4 Passed

4.

Finally, add the “Home” functionality so the user is taken to the /about URL segment.

Using the “Home” button should add a new entry to the top of the browser history stack.

Run the code using npm run dev in the terminal. Once you see a console message indicating “Ready,” click the Refresh Browser button in the workspace browser bar.

If successful, you should be able to click on “Home” and be taken to the /about page.

Hint

Recall that the router object has a push(path) method, which can be used to programmatically navigate to path by pushing a new entry to the top of the browser history stack.

Enter the command inside the terminal window.

The window can be refreshed when it displays “Ready.”

```
> dev
> next dev -p 4001

  ▲ Next.js 14.0.1
  - Local:        http://localhost:4001
```

✓ Ready in 1156ms

Footer.tsx

```
'use client'
import styles from './Footer.module.css'
import { useRouter } from 'next/navigation'
export default function Footer() {
  const router = useRouter();
  const callingPush = () => router.push("/about")
  return (
    <footer className={styles.footer}>
      <button onClick={router.back}>Back</button>
      <button onClick={callingPush}>Home</button>
      <button onClick={router.forward}>Forward</button>
    </footer>
  )
}
```