Using The <Link> Component

22 min

When exploring web applications, we often use links to navigate.

Next.js provides a few ways to give us **Single Page Application** (SPA) like browser navigation. Remember that SPA navigation refers to the idea of changing the browser path without needing to make a new request to the server. In this exercise, we will be exploring the <Link> component.

A <Link> component can be used like:

```
const selectedUser = "25"  // user selected user id
<section>
  <Link href="/users">Users</Link>
  <Link href=`/users/${selectedUser}`>User: {selectedUser}</Link>  {/* dynamic path*/}
  <Link href="/settings" replace>My Settings</Link> {/* replaces current path in browser
history*/}
  <Link href="/info">Info</Link>
</section>
```

Where:

- The href prop determines the path we want to navigate to.
- The href prop can be used to link to a dynamic segment by creating the dynamic path (in the above example, we use a template literal).
- The text content ("Users", "My Settings", "Info") is the text displayed in the UI.
- The replace prop is used to replace the current URL path with the href path.

The <Link> component is an extension to the <a> element that adds **prefetching** functionality. With prefetching, Next.js will prefetch route segments automatically so when a user navigates to those segments, the browser does not need to *reload*.

<Link> components can be used in conjunction with the usePathname() hook from the next/navigation package to apply some "active" styles to the <Link>. usePathname() returns the current path in the URL as a string and can be used to apply styles to a<Link> like:

```
'use client'
import Link from 'next/link'
import { usePathname } from "next/navigation";

const pathname = usePathname()  // current path: /users
<section>
  <Link href="/users" className={pathname === "/users" ? styles.active : ""}>Users</Link> {/*
currently active*/}
  <Link href="/info" className={pathname === "/info" ? styles.active : ""}>Info</Link> {/* not
active */}
</section>
```

Note that usePathname() can only be used within client components so we use the 'use client' directive.

Let’s practice adding navigation links to our header so users have an easier time navigating between our different URL segments.

Instructions

1. Checkpoint 1 Passed

1.

Currently, the <Navigation> component in /components/Navigation uses <a> element for navigation. This is causing an unpleasant user experience as the browser is reloading.

Start by importing the <Link> component and usePathname() hook.

Hint

The <Link> component is part of the next/link package. The usePathname() hook is part of the next/navigation package.

2. Checkpoint 2 Passed

2.

Next:

  - Call usePathname() and store it in a constant called pathname.
  - Replace the <a> elements with the <Link> component.

Run npm run dev. Once “Ready” appears, click the refresh button in the workspace browser.

If successful, you should be able to click the navigation links without the browser refreshing.

Hint

Remember that the Next.js <Link> component is an extension of the <a> element.

Enter the command npm run dev 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 an indication that the current path matches a link. Apply the "active" class to the <Link> component when the current path matches the href prop value.

Hint

Make sure to use pathname to check if it matches the <Link> href value.

4. Checkpoint 4 Passed

4.

Finally, if you navigate to /authors/ben you'll notice clicking the links don't accurately modify the path. Fix this in the page.tsx in /authors/[name] by:

- Replacing the <a> element with a <Link> component.
- Using slug to create a dynamic path in the href value.

Run npm run dev. Once "Ready" appears, click the refresh button in the workspace browser.

If completed successfully, clicking the links should update the path to reflect the clicked article.

Note that you will still get a not found (404) error; we will fix this in a future exercise.

Hint

Make sure you modify the href value where slug is appended to /articles.

Enter the command npm run dev 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
```

Navigation.tsx

```
'use client'


import styles from './Navigation.module.css'

import Link from 'next/link'

import { usePathname } from "next/navigation";


export default function Navigation() {

   const pathname = usePathname();

   return (

   <nav className={styles.nav}>

      <Link href="/about" className={pathname === "/about" ? styles.active : ""}>About</Link>

      <Link href="/articles" className={pathname === "/articles" ? styles.active : ""}>Articles</Link>

      <Link href="/categories" className={pathname === "/categories" ? styles.active : ""}>Categories</Link>

   </nav>

  )

}
```

page.tsx

```
import { AUTHORS } from "../../../store/data"

import Link from 'next/link'


export default function AuthorsPage({ params }: { params: { name: string } }) {

  const author = AUTHORS[params.name]


  return (

   <section>

     <h1>Articles by {params.name}</h1>
```

```
      <ul>
        {author && author.articles.map((slug) => (
          <li key={slug}>
            <Link href={"/articles/replace-me"}>{slug}</Link>
          </li>
        ))}
      </ul>
    </section>
  )
}
```