next/link

▶ **Examples**

> *Before moving forward, we recommend you to read Routing Introduction first.*

Client-side transitions between routes can be enabled via the `Link` component exported by `next/link`.

For an example, consider a `pages` directory with the following files:

- `pages/index.js`
- `pages/about.js`
- `pages/blog/[slug].js`

We can have a link to each of these pages like so:

```
import Link from 'next/link'

function Home() {
  return (
    <ul>
      <li>
        <Link href="/">
          <a>Home</a>
        </Link>
      </li>
      <li>
        <Link href="/about">
          <a>About Us</a>
        </Link>
      </li>
      <li>
        <Link href="/blog/hello-world">
          <a>Blog Post</a>
        </Link>
      </li>
    </ul>
  )
}

export default Home
```

`Link` accepts the following props:

- `href` - The path or URL to navigate to. This is the only required prop
- `as` - Optional decorator for the path that will be shown in the browser URL bar. Before Next.js 9.5.3 this was used for dynamic routes, check our previous docs to see how it worked. Note: when this path differs from the one provided in `href` the previous `href` / `as` behavior is used as shown in the previous docs.
- `passHref` - Forces `Link` to send the `href` property to its child. Defaults to `false`
- `prefetch` - Prefetch the page in the background. Defaults to `true`. Any `<Link />` that is in the viewport (initially or through scroll) will be preloaded. Prefetch can be disabled by passing `prefetch={false}`. When `prefetch` is set to `false`, prefetching will still occur on hover. Pages using Static

Generation will preload `JSON` files with the data for faster page transitions. Prefetching is only enabled in production.

- `replace` - Replace the current `history` state instead of adding a new url into the stack. Defaults to `false`
- `scroll` - Scroll to the top of the page after a navigation. Defaults to `true`
- `shallow` - Update the path of the current page without rerunning `getStaticProps`, `getServerSideProps` or `getInitialProps`. Defaults to `false`
- `locale` - The active locale is automatically prepended. `locale` allows for providing a different locale. When `false` `href` has to include the locale as the default behavior is disabled.

If the route has dynamic segments

There is nothing to do when linking to a dynamic route, including catch all routes, since Next.js 9.5.3 (for older versions check our previous docs). However, it can become quite common and handy to use interpolation or an URL Object to generate the link.

For example, the dynamic route `pages/blog/[slug].js` will match the following link:

```
import Link from 'next/link'

function Posts({ posts }) {
  return (
    <ul>
      {posts.map((post) => (
        <li key={post.id}>
          <Link href={`/blog/${encodeURIComponent(post.slug)}`}>
            <a>{post.title}</a>
          </Link>
        </li>
      ))}
    </ul>
  )
}

export default Posts
```

If the child is a custom component that wraps an `<a>` tag

If the child of `Link` is a custom component that wraps an `<a>` tag, you must add `passHref` to `Link`. This is necessary if you're using libraries like styled-components. Without this, the `<a>` tag will not have the `href` attribute, which hurts your site's accessibility and might affect SEO. If you're using ESLint, there is a built-in rule `next/link-passhref` to ensure correct usage of `passHref`.

```
import Link from 'next/link'
import styled from 'styled-components'

// This creates a custom component that wraps an <a> tag
const RedLink = styled.a`
  color: red;
```

```
`

function NavLink({ href, name }) {
  // Must add passHref to Link
  return (
    <Link href={href} passHref>
      <RedLink>{name}</RedLink>
    </Link>
  )
}

export default NavLink
```

- If you're using emotion's JSX pragma feature (`@jsx jsx`), you must use `passHref` even if you use an `<a>` tag directly.
- The component should support `onClick` property to trigger navigation correctly

If the child is a functional component

If the child of `Link` is a functional component, in addition to using `passHref` , you must wrap the component in `React.forwardRef` :

```
import Link from 'next/link'

// `onClick`, `href`, and `ref` need to be passed to the DOM element
// for proper handling
const MyButton = React.forwardRef(({ onClick, href }, ref) => {
  return (
    <a href={href} onClick={onClick} ref={ref}>
      Click Me
    </a>
  )
})

function Home() {
  return (
    <Link href="/about" passHref>
      <MyButton />
    </Link>
  )
}

export default Home
```

With URL Object

`Link` can also receive a URL object and it will automatically format it to create the URL string. Here's how to do it:

```
import Link from 'next/link'
```

```
function Home() {
  return (
    <ul>
      <li>
        <Link
          href={{
            pathname: '/about',
            query: { name: 'test' },
          }}
        >
          <a>About us</a>
        </Link>
      </li>
      <li>
        <Link
          href={{
            pathname: '/blog/[slug]',
            query: { slug: 'my-post' },
          }}
        >
          <a>Blog Post</a>
        </Link>
      </li>
    </ul>
  )
}

export default Home
```

The above example has a link to:

- A predefined route: `/about?name=test`
- A dynamic route: `/blog/my-post`

You can use every property as defined in the Node.js URL module documentation.

Replace the URL instead of push

The default behavior of the `Link` component is to `push` a new URL into the `history` stack. You can use the `replace` prop to prevent adding a new entry, as in the following example:

```
<Link href="/about" replace>
  <a>About us</a>
</Link>
```

Disable scrolling to the top of the page

The default behavior of `Link` is to scroll to the top of the page. When there is a hash defined it will scroll to the specific id, like a normal `<a>` tag. To prevent scrolling to the top / hash `scroll={false}` can be added to `Link`:

```
<Link href="/#hashid" scroll={false}>
  <a>Disables scrolling to the top</a>
</Link>
```