Many applications are hosted at something other than the root (`/`) of their domain.

For example, a Gatsby blog could live at `example.com/blog/` , or a site could be hosted on GitHub Pages at `example.github.io/my-gatsby-site/` .

Each of these sites needs a prefix added to all paths on the site. So a link to `/my-sweet-blog-post/` should be rewritten as `/blog/my-sweet-blog-post` .

In addition, links to various resources (JavaScript, CSS, images, and other static content) need the same prefix, so that the site continues to function correctly when served with the path prefix in place.

Adding the path prefix is a two step process, as follows:

Add to `gatsby-config.js`

Firstly, add a `pathPrefix` value to your `gatsby-config.js` .

```
module.exports = {
  pathPrefix: `/blog`,
}
```

Build

The final step is to build your application with either the `--prefix-paths` flag or `PREFIX_PATHS` environment variable, like so:

```
gatsby build --prefix-paths
```

```
PREFIX_PATHS=true gatsby build
```

If this flag is not passed, Gatsby will ignore your `pathPrefix` and build the site as if hosted from the root domain.

Serve

Serve your application with the `--prefix-paths` flag, like so:

```
gatsby serve --prefix-paths
```

If this flag is not passed, Gatsby will ignore your `pathPrefix` .

In-app linking

Gatsby provides APIs and libraries to make using this feature seamless. Specifically, the [`Link`] component has built-in functionality to handle path prefixing.

For example, if you want to link to the location `/page-2` , but the actual link will be prefixed (e.g. `/blog/page-2`); you don't need to hard code the prefix into your links. By using the Gatsby `Link` component, paths will automatically be prefixed with the `pathPrefix` value assigned in your `gatsby-config.js` file. If you later migrate away from using a path prefix, your links will *still* work seamlessly.

For example, when navigating to the `page-2` location in the `Link` component below; the location will be automatically prefixed with your assigned `pathPrefix` value.

```
import React from "react"
import { Link } from "gatsby"
import Layout from "../components/layout"

function Index() {
  return (
    <Layout>
      {/* highlight-next-line */}
      <Link to="page-2">Page 2</Link>
    </Layout>
  )
}
```

If you want to do programmatic/dynamic navigation, this is also possible! Expose the Gatsby `navigate` helper, and this too automatically handles path prefixing.

```
import React from "react"
import { navigate } from "gatsby"
import Layout from "../components/layout"

export default function Index() {
  return (
    <Layout>
      {/* Note: this is an intentionally contrived example, but you get the idea!
*/}
      {/* highlight-next-line */}
      <button onClick={() => navigate("/page-2")}>
        Go to page 2, dynamically
      </button>
    </Layout>
  )
}
```

Add the path prefix to paths using `withPrefix`

For pathnames you construct manually, there's a helper function, [`withPrefix`] that prepends your path prefix in production (but doesn't during development where paths don't need to be prefixed).

Additional considerations

The [`assetPrefix`] feature can be thought of as semi-related to this feature. That feature allows your assets (non-HTML files, e.g. images, JavaScript, etc.) to be hosted on a separate domain, for example a CDN.

This feature works seamlessly with `assetPrefix`. Build out your application with the `--prefix-paths` flag and you'll be well on your way to hosting an application with its assets hosted on a CDN, and its core functionality available behind a path prefix. If you use `assetPrefix`, your `pathPrefix` will be changed to `<assetPrefix>/<pathPrefix>`. If you need to access the same `pathPrefix` as in your `gatsby-config.js`, consider using `basePath`.