Serving users content in a way that is adapted to their language & culture is part of a great user experience. When you make an effort to adapt web content to a user's location, that practice is called internationalization (i18n).

There are two hard parts of internationalization:

- **Content storage and workflow.** Coordinating with internal or external translators to translate both existing and new content into required languages can be time-consuming. In addition, adding another approver into the content publishing process can slow things down without careful workflow design.
- **Display, templating, and routing.** At minimum, internationalization means users must be redirected, either to a subdomain (eg `fr.example.com/blog`) or path prefix (eg `example.com/fr/blog`). In addition, internationalization efforts also come with logic around sections or pages that should be present in some languages but not others.

This guide is a brief look at the options that exist for enhancing your Gatsby project for internationalization.

Choosing a package

There are a few React i18n packages out there. Several options include react-intl, the community Gatsby plugin and react-i18next. There are several factors to consider when choosing a package: Do you already use a similar package in another project? How well does the package meet the needs of your users? Are you or your team already familiar with a certain package? Is the package well documented and maintained?

gatsby-plugin-i18n

This plugin helps you use `react-intl`, `i18next` or any other i18n library with Gatsby. This plugin does not translate or format your content, but rather it creates routes for each language, allowing Google to more easily find the correct version of your site, and if you need to, designate alternative UI layouts.

The naming format follows .**languageKey**.js for files and /**languageKey**/path/fileName for URLs.

Example:

File - src/pages/about.**en**.js

URL - /**en**/about

gatsby-plugin-i18n on GitHub

react-intl

React-intl is a part of the FormatJS set of i18n libraries and provides support for over 150+ languages. It builds on JavaScript's Internationalization API providing enhanced APIs and components. React-intl uses React context and HOCs (Higher Order Components) to provide translations allowing you to dynamically load language modules as you need them. There are also polyfill options available for older browsers that do not support the base JavaScript i18n API.

More detailed information about react-intl's APIs and components, including demos, are available in the documentation.

react-i18next

React-i18next is an internationalization library built on the i18next framework. It uses components to make sure translations render correctly or to re-render your content when the user language changes.

React-i18next is more extensible than other options with a variety of plugins, utilities, and configurations. Common plugins allow for detecting a user's language or adding an additional layer of local caching. Other options include caching, a backend plugin to load translations from your server, or bundling translations with webpack.

This framework also has experimental support for the React suspense API and it supports a stable version of React hooks.

Other resources

- Building i18n with Gatsby
- Building Eviction Free NYC with GatsbyJS + Contentful
- Gatsby i18n packages
- Gatsby i18n articles
- W3C's i18n resources