On this page

i18n - Introduction

It is **easy to translate a Docusaurus website** with its internationalization (i18n) support.

Goals

It is important to understand the **design decisions** behind the Docusaurus i18n support.

For more context, you can read the initial RFC and PR.

i18n goals

The goals of the Docusaurus i18n system are:

- **Simple**: just put the translated files in the correct filesystem location
- **Flexible translation workflows**: use Git (monorepo, forks, or submodules), SaaS software, FTP
- **Flexible deployment options**: single, multiple domains, or hybrid
- **Modular**: allow plugin authors to provide i18n support
- **Low-overhead runtime**: documentation is mostly static and does not require heavy JS libraries or polyfills
- **Scalable build-times**: allow building and deploying localized sites independently
- **Localize assets**: an image of your site might contain text that should be translated
- **No coupling**: not forced to use any SaaS, yet integrations are possible
- **Easy to use with Crowdin**: a lot of Docusaurus v1 sites use Crowdin and should be able to migrate to v2
- **Good SEO defaults**: we set useful SEO headers like `hreflang` for you
- **RTL support**: locales reading right-to-left (Arabic, Hebrew, etc.) are supported and easy to implement
- **Default translations**: classic theme labels are translated for you in many languages

i18n non-goals

We don't provide support for:

- **Automatic locale detection**: opinionated, and best done on the server (your hosting provider)
- **Translation SaaS software**: you are responsible to understand the external tools of your choice
- **Translation of slugs**: technically complicated, little SEO value

Translation workflow

Overview

Overview of the workflow to create a translated Docusaurus website:

1. **Configure**: declare the default locale and alternative locales in `docusaurus.config.js`
2. **Translate**: put the translation files at the correct filesystem location
3. **Deploy**: build and deploy your site using a single or multi-domain strategy

Translation files

You will work with three kinds of translation files.

Markdown files

This is the main content of your Docusaurus website.

Markdown and MDX documents are translated as a whole, to fully preserve the translation context, instead of splitting each sentence as a separate string.

JSON files

JSON is used to translate:

- Your React code: standalone React pages in `src/pages`, or other components
- Layout labels provided through `themeConfig`: navbar, footer
- Layout labels provided through plugin options: docs sidebar category labels, blog sidebar title...

The JSON format used is called **Chrome i18n**:

```
{
  "myTranslationKey1": {
    "message": "Translated message 1",
    "description": "myTranslationKey1 is used on the homepage"
  },
  "myTranslationKey2": {
    "message": "Translated message 2",
    "description": "myTranslationKey2 is used on the FAQ page"
  }
}
```

The choice was made for 2 reasons:

- **Description attribute**: to help translators with additional context
- **Widely supported**: Chrome extensions, Crowdin, Transifex, Phrase, Applanga, etc.

Data files

Some plugins may read from external data files that are localized as a whole. For example, the blog plugin uses an `authors.yml` file that can be translated by creating a copy under `i18n/[locale]/docusaurus-plugin-content-blog/authors.yml`.

Translation files location

The translation files should be created at the correct filesystem location.

Each locale and plugin has its own `i18n` subfolder:

```
website/i18n/[locale]/[pluginName]/...
```

NOTE

For multi-instance plugins, the path is `website/i18n/[locale]/[pluginName]-[pluginId]/...`.

Translating a very simple Docusaurus site in French would lead to the following tree:

```
website/i18n
└── fr
    ├── code.json  # Any text label present in the React code
    │              # Includes text labels from the themes' code
    ├── docusaurus-plugin-content-blog # translation data the blog plugin needs
    │   └── 2020-01-01-hello.md
    │
    ├── docusaurus-plugin-content-docs # translation data the docs plugin needs
    │   ├── current
    │   │   ├── doc1.md
    │   │   └── doc2.mdx
    │   └── current.json
    │
    └── docusaurus-theme-classic # translation data the classic theme needs
        ├── footer.json   # Text labels in your footer theme config
        └── navbar.json   # Text labels in your navbar theme config
```

The JSON files are initialized with the `docusaurus write-translations` CLI command. Each plugin sources its own translated content under the corresponding folder, while the `code.json` file defines all text labels used in the React code.

Each content plugin or theme is different, and **defines its own translation files location**:

- Docs i18n
- Blog i18n
- Pages i18n

- Themes i18n

Edit this page

*Last updated on **Mar 23, 2023** by **Sébastien Lorber***

Previous
« Deployment

Next
i18n - Tutorial »