Creating a local environment

In this article

You can run the GitHub Docs application locally on your computer.

Articles in the "Contributing to GitHub Docs" section refer to the documentation itself and are a resource for GitHub staff and open source contributors.

About GitHub Docs site structure

The GitHub Docs site was originally a Ruby on Rails web application. Some time later it was converted into a static site powered by Jekyll. A few years after that it was migrated to Nanoc, another Ruby static site generator.

Today it's a dynamic Node.js webserver powered by Express, using middleware to support proper HTTP redirects, language header detection, and dynamic content generation to support the various flavors of GitHub's product documentation, like GitHub.com and GitHub Enterprise Server.

The tooling for this site has changed over the years, but many of the tried-and-true authoring conventions of the original Jekyll site have been preserved.

- Content is written in Markdown files, which live in the `content` directory.
- Files in the `data` directory are available to templates via the `{% data %}` tag.
- Markdown files can contain frontmatter.
- The `redirect_from` Jekyll plugin behavior is supported.

Setting up your local environment

You can clone the GitHub Docs repository and run the application locally on your computer, after some initial setup.

Installing Node.js

The GitHub Docs site is powered by Node.js. It runs on macOS, Windows, and Linux environments.

To run the site, you'll need Node.js. To install Node.js, download the "LTS" installer from nodejs.org. To check which Node version you need, you can see the `package.json` file in the GitHub Docs repository. The Node version is listed in the `engine` field, similar to the following example, which indicates you can use Node major version 16 or Node major version 18.

```
"engines": {
    "node": "^16 || ^18"
}
```

If you're using `nodenv`, see the `nodenv` docs for instructions on switching Node.js versions.

Starting a local GitHub Docs server

Once you've installed Node.js (which includes the popular `npm` package manager), open your terminal and run the following commands.

```
git clone https://github.com/github/docs
cd docs
npm ci
npm run build
npm start
```

You should now have a running server. To access your local preview environment, visit localhost:4000 in your browser.

When you're ready to stop your local server, type `Ctrl`+`C` in your terminal window.

> **Note:** You should typically only need to run `npm ci` and `npm run build` each time you pull the latest version of a branch.
>
> - `npm ci` does a clean install of dependencies, without updating the `package-lock.json` file.
> - `npm run build` creates static assets, such as JavaScript and CSS files.

If you would like to read more about debugging and troubleshooting the GitHub Docs application, see "Troubleshooting your environment" in the github/docs repository.

Using browser shortcuts

The `script/bookmarklets` directory in the GitHub Docs repository contains browser shortcuts that can help with reviewing GitHub documentation. For more information, see the directory's `README`.

Enabling different languages

By default, the local server does not run with all supported languages enabled. If you need to run a local server with a particular language, you can temporarily edit the `start` script in `package.json` and update the `ENABLED_LANGUAGES` variable.

For example, to enable Japanese and Portuguese in addition to English, you can edit `package.json` and set `ENABLED_LANGUAGES='en,ja,pt'` in the `start` script. Then restart the server for the change to take effect.

> **Note:** Before you commit your changes, you should revert the `package.json` file to its original state.

The supported language codes are defined in `src/languages/lib/languages.js`.

Using GitHub Codespaces

As an alternative to running GitHub Docs locally, you can use GitHub Codespaces. GitHub

Codespaces enable you to edit, preview, and test your changes directly from your browser.

For more information about using a codespace for working on GitHub documentation, see "Working on GitHub Docs in a codespace."

Further reading

- Creating reusable content
- Components
- Data
- Middleware
- Scripts
- Tests