About GitHub Pages and Jekyll

In this article

Jekyll is a static site generator with built-in support for GitHub Pages.

GitHub Pages is available in public repositories with GitHub Free and GitHub Free for organizations, and in public and private repositories with GitHub Pro, GitHub Team, GitHub Enterprise Cloud, and GitHub Enterprise Server. For more information, see "GitHub's plans."

About Jekyll

Jekyll is a static site generator with built-in support for GitHub Pages and a simplified build process. Jekyll takes Markdown and HTML files and creates a complete static website based on your choice of layouts. Jekyll supports Markdown and Liquid, a templating language that loads dynamic content on your site. For more information, see Jekyll.

Jekyll is not officially supported for Windows. For more information, see "Jekyll on Windows" in the Jekyll documentation.

We recommend using Jekyll with GitHub Pages. If you prefer, you can use other static site generators or customize your own build process locally or on another server. For more information, see "About GitHub Pages."

Configuring Jekyll in your GitHub Pages site

You can configure most Jekyll settings, such as your site's theme and plugins, by editing your *_config.yml* file. For more information, see "Configuration" in the Jekyll documentation.

Some configuration settings cannot be changed for GitHub Pages sites.

```
lsi: false
safe: true
source: [your repo's top level directory]
incremental: false
highlighter: rouge
gist:
  noscript: false
```

```
kramdown:
  math_engine: mathjax
  syntax_highlighter: rouge
```

By default, Jekyll doesn't build files or folders that:

- are located in a folder called `/node_modules` or `/vendor`
- start with `_`, `.`, or `#`
- end with `~`
- are excluded by the `exclude` setting in your configuration file

If you want Jekyll to process any of these files, you can use the `include` setting in your configuration file.

Front matter

To set variables and metadata, such as a title and layout, for a page or post on your site, you can add YAML front matter to the top of any Markdown or HTML file. For more information, see "Front Matter" in the Jekyll documentation.

You can add `site.github` to a post or page to add any repository references metadata to your site. For more information, see "Using `site.github`" in the Jekyll Metadata documentation.

Themes

You can add a Jekyll theme to your GitHub Pages site to customize the look and feel of your site. For more information, see "Themes" in the Jekyll documentation.

You can add a supported theme to your site on GitHub. For more information, see "Supported themes" on the GitHub Pages site and Adding a theme to your GitHub Pages site using Jekyll".

To use any other open source Jekyll theme hosted on GitHub, you can add the theme manually. For more information, see themes hosted on GitHub and "Adding a theme to your GitHub Pages site using Jekyll".

You can override any of your theme's defaults by editing the theme's files. For more information, see your theme's documentation and "Overriding your theme's defaults" in the Jekyll documentation.

Plugins

You can download or create Jekyll plugins to extend the functionality of Jekyll for your site. For example, the jemoji plugin lets you use GitHub-flavored emoji in any page on your site the same way you would on GitHub. For more information, see "Plugins" in the Jekyll documentation.

GitHub Pages uses plugins that are enabled by default and cannot be disabled:

- `jekyll-coffeescript`
- `jekyll-default-layout`
- `jekyll-gist`
- `jekyll-github-metadata`
- `jekyll-optional-front-matter`
- `jekyll-paginate`
- `jekyll-readme-index`

- `jekyll-titles-from-headings`
- `jekyll-relative-links`

You can enable additional plugins by adding the plugin's gem to the `plugins` setting in your *_config.yml* file. For more information, see "Configuration" in the Jekyll documentation.

For a list of supported plugins, see "Dependency versions" on the GitHub Pages site. For usage information for a specific plugin, see the plugin's documentation.

> **Tip:** You can make sure you're using the latest version of all plugins by keeping the GitHub Pages gem updated. For more information, see "Testing your GitHub Pages site locally with Jekyll" and "Dependency versions" on the GitHub Pages site.

GitHub Pages cannot build sites using unsupported plugins. If you want to use unsupported plugins, generate your site locally and then push your site's static files to GitHub Enterprise Cloud.

Syntax highlighting

To make your site easier to read, code snippets are highlighted on GitHub Pages sites the same way they're highlighted on GitHub Enterprise Cloud. For more information about syntax highlighting on GitHub Enterprise Cloud, see "Creating and highlighting code blocks."

By default, code blocks on your site will be highlighted by Jekyll. Jekyll uses the Rouge highlighter, which is compatible with Pygments. Pygments has been deprecated and not supported in Jekyll 4. If you specify Pygments in your *_config.yml* file, Rouge will be used as the fallback instead. Jekyll cannot use any other syntax highlighter, and you'll get a page build warning if you specify another syntax highlighter in your *_config.yml* file. For more information, see "About Jekyll build errors for GitHub Pages sites."

If you want to use another highlighter, such as `highlight.js`, you must disable Jekyll's syntax highlighting by updating your project's *_config.yml* file.

```
kramdown:
  syntax_highlighter_opts:
    disable : true
```

If your theme doesn't include CSS for syntax highlighting, you can generate GitHub's syntax highlighting CSS and add it to your project's `style.css` file.

```
rougify style github > style.css
```

Building your site locally

If you are publishing from a branch, changes to your site are published automatically when the changes are merged into your site's publishing source. If you are publishing from a custom GitHub Actions workflow, changes are published whenever your workflow is triggered (typically by a push to the default branch). If you want to preview your changes first, you can make the changes locally instead of on GitHub Enterprise Cloud. Then, test your site locally. For more information, see "Testing your GitHub Pages site locally with Jekyll."