Table of Contents

GitBook Documentation: v3.0.0-pre.0

FAQ

There are questions that are asked quite often, check this out before creating an issue.

Setup and Installation of GitBook

Getting GitBook installed and ready-to-go should only take a few minutes.

GitBook.com

GitBook.com is an easy to use solution to write, publish and host books. It's best and easier solution for publishing your content and collaborate on it.

It integrates well with the GitBook Editor.

Local Installation

Requirements

Installing GitBook is easy and straight-forward, but there are a few requirements you'll need to make sure your system has before you start.

- NodeJS (v4.0.0 and above are adviced)
- Windows, Linux, Unix, or Mac OS X

Install with NPM

The best way to install GitBook is via **NPM**. At the terminal prompt, simply run the following command to install GitBook:

```
$ npm install gitbook-cli -g
```

`gitbook-cli` is an utility to install and use multiple versions of GitBook on the same system. It will automatically install the required version to build a book.

Using pre-releases

`gitbook-cli` makes it easy to install and test other versions of GitBook with your book:

```
$ gitbook install beta
```

Directory structure

```
.
├── book.json
├── README.md
├── SUMMARY.md
├── chapter-1/
|   ├── README.md
|   └── something.md
└── chapter-2/
    ├── README.md
    └── something.md
```

An overview of what each of these does:

File	Description
`book.json`	Stores configuration data (**optional**)
`README.md`	Preface / Introduction for your book (**required**)
`SUMMARY.md`	Table of Contents

Sub-directory

Configuration

GitBook allows you to customize your book using a flexible configuration. These options are specified in a `book.json` file.

Configuration Settings

Variable	Description
`title`	Title of your book, default value is extracted from the README. On GitBook.com this field is pre-filled.
`description`	Description of your book, default value is extracted from the README. On GitBook.com this field is pre-filled.
`author`	Name of the author. On GitBook.com this field is pre-filled.
`isbn`	ISBN of the book
`language`	ISO code of the book's language, default value is `en`
`direction`	`rtl` or `ltr` , default value depends on the value of `language`
`gitbook`	SemVer condition to validate which GitBook version should be used
`plugins`	List of plugins to load, See the plugins section for more details
`pluginsConfig`	Configuration for plugins, See the plugins section for more details

Plugins

Plugins and their configurations are specified in the `book.json` .

Templating

GitBook uses the Nunjucks templating language to process pages and templates.

The Nunjucks syntax is very similar to Jinja2 and Liquid.

Content References

Content referencing (conref) is a convenient mechanism for reuse of content from other files or books.

Importing local files

Importing an other file's content is really easy using the `include` tag:

```
{% include "./test.md" %}
```

Importing file from another book

GitBook can also resolve the include path by using git:

```
{% include "git+https://github.com/GitbookIO/documentation.git/README.md#0.0.1" %}
```

The format of git url is:

```
git+https://user@hostname/project/blah.git/file#commit-ish
```

The real git url part should finish with `.git`, the filename to import is extracted after the `.git` till the fragment of the url.

The `commit-ish` can be any tag, sha, or branch which can be supplied as an argument to `git checkout`. The default is `master`.

Inheritance

Template inheritance is a way to make it easy to reuse templates. When writing a template, you can define "blocks" that child templates can override. The inheritance chain can be as long as you like.

`block` defines a section on the template and identifies it with a name. Base templates can specify blocks and child templates can override them with new content.

```
{% extends "./mypage.md" %}

{% block pageContent %}
# This is my page content
{% endblock %}
```

In the file `mypage.md`, you should specify the blocks that can be extent:

```
{% block pageContent %}
This is the default content
{% endblock %}

# License

{% import "./LICENSE" %}
```

Variables

The following is a reference of the available data during book's parsing and theme generation.

Global Variables

Variable	Description
`book`	Bookwide information + configuration settings from `book.json` . See below for details.
`gitbook`	GitBook specific information
`page`	Current page specific information
`file`	File associated with the current page specific information
`summary`	Information about the table of contents
`config`	Dump of the `book.json`

Book Variables

Variable	Description
`book.[CONFIGURATION_DATA]`	All the `variables` set via the `book.json` are available through the book variable.

GitBook Variables

Variable	Description
`gitbook.time`	The current time (when you run the `gitbook` command).
`gitbook.version`	Version of GitBook used to generate the book

File Variables

Variable	Description
`file.path`	The path to the raw page
`file.mtime`	Modified Time, Time when file data last modified
`file.type`	The name of the parser used to compile this file (ex: `markdown` , `asciidoc` , etc)

Page Variables

Variable	Description
`page.title`	Title of the page
`page.previous`	Previous page in the Table of Contents (can be `null`)
`page.next`	Next page in the Table of Contents (can be `null`)
`page.dir`	Text direction, based on configuration or detected from content (`rtl` or `ltr`)

Table of Contents Variables

Variable	Description
`summary.parts`	List of sections in the Table of Contents

Thw whole table of contents (`SUMMARY.md`) can be accessed:

`summary.parts[0].articles[0].title` will return the title of the first article.

Plugins

Plugins are the best way to extend GitBook functionalities (ebook and website). There exist plugins to do a lot of things: bring math formulas display support, track visits using Google Analytic, etc.

How to find plugins?

Plugins can be easily searched on plugins.gitbook.com.

How to install a plugin?

Once you find a plugin that you want to install, you need to add it to your book.json:

```
{
    "plugins": ["myPlugin", "anotherPlugin"]
}
```

You can also specify a specific version using: "myPlugin@0.3.1". By default GitBook will resolve the latest version of the plugin compatbile with the current GitBook version.

Plugins are automatically installed on GitBook.com. Locally, run `gitbook install` to install and prepare all plugins for your books.

Theming

Since version 3.0.0, GitBook can be easily themed. Books are using by default the gitbook-theme-default.

The theme to use is specified in the book's configuration using key `theme`.

> **Caution**: Custom theming can block some plugins from working correctly.

Structure of a theme

A theme is a folder containing templates and assets. All the templates are optionnal, since theme are always extending the default theme.

Folder	Description
`_layouts`	Main folder containing all the templates
`_layouts/website/page.html`	Template for a normal page
`_layouts/ebook/page.html`	Template for a normal page during ebook generation (PDF< ePub, Mobi)

Extend/Customize theme in a book

Authors can extend the templates of a theme directly from the book source (without creating an external theme). Templates will be resolved in the `_layouts` folder of the book first, then in

Publish a theme

Themes are published as plugins (see related docs) with a `theme-` prefix. For example the theme `awesome` will be loaded from `theme-awesome` plugin, and then from `gitbook-plugin-theme-awesome` NPM package.

GitBook FAQ

Should I use an `.html` or `.md` extensions in my links?

You should always use `.md` extensions for your relative links, GitBook will automatically replace these links by the right value when the pointing file is referenced in the Table of Contents.

Can I create a GitBook in a sub-directory of my repository?

Yes, GitBooks can be created in sub-directories. GitBook.com and the CLI also looks by default in a series of folders.