FREE eBook

LEARNING jekyll

Free unaffiliated eBook created from
Stack Overflow contributors.

#jekyll

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jekyll

It is an unofficial and free jekyll ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jekyll.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with jekyll

Remarks

Jekyll is a simple, blog-aware, static site generator. It takes a template directory containing raw text files in various formats, runs it through a converter (like Markdown) and its Liquid renderer, and spits out a complete, ready-to-publish static website suitable for serving with your favorite web server. Jekyll is also the engine behind GitHub Pages, which means you can use Jekyll to host your project's page, blog, or website from GitHub's servers for free.

Jekyll's website is at http://jekyllrb.com/, and documentation can be found at http://jekyllrb.com/docs/home/.

Latest Release:

Examples

Installation or Setup

Quickstart for Jekyll

```
 $ gem install jekyll
 $ jekyll new my-awesome-site
 $ cd my-awesome-site
~/my-awesome-site $ jekyll serve
```

Now browse to http://localhost:4000

Quickstart for Jekyll with Bundler

```
 $ gem install jekyll bundler
 $ jekyll new my-awesome-site
 $ cd my-awesome-site
~/my-awesome-site $ bundle exec jekyll serve
```

Now browse to http://localhost:4000

Create Jekyll Post And Pages

Create new Jekyll Post

To create a new Jekyll Post, create a new file on `_posts` directory with the format

```
YYYY-MM-DD-title.MARKUP
```

Replace `MARKUP` with the file extension for the language you want to use. This is usually Markdown(.md or .markdown) or HTML(.html).

```
_posts/2017-01-01-hello-jekyll.md
```

Create new Jekyll Page

To create a new Jekyll Page, create a new file on any folder or directory not excluded by Jekyll in your project directory.

```
about.html
contact/company_info.md
```

> **NOTE:** Both `Page` and `Post` files require Front Matter dashes to be considered for processing. Otherwise, they're simply designated as a `StaticFile`.
>
> Front Matter dashes should be at the very beginning, before your content, and simply look like this:
>
> ```
> ---
> ---
>
> < your content >
> ```

Basic Usage

The Jekyll gem makes a jekyll executable available to you in your Terminal window. You can use this command in a number of ways:

```
$ jekyll build
# => The current folder will be generated into ./_site

$ jekyll build --destination <destination>
# => The current folder will be generated into <destination>

$ jekyll build --source <source> --destination <destination>
# => The <source> folder will be generated into <destination>

$ jekyll build --watch
# => The current folder will be generated into ./_site,
#    watched for changes, and regenerated automatically.
```

Jekyll also comes with a built-in development server that will allow you to preview what the generated site will look like in your browser locally.

```
$ jekyll serve
# => A development server will run at http://localhost:4000/
# Auto-regeneration: enabled. Use `--no-watch` to disable.
```

Install Jekyll on Linux Mint 18

Install jekyll on Linux Mint 18 with the following steps:

```
sudo apt install ruby
sudo apt install build-essential
sudo apt install ruby-dev
sudo gem install jekyll
```

Install Jekyll on Windows

1. Open a command prompt with Administrator access
2. Install Chocolatey: `@powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-object net.webclient).DownloadString('https://chocolatey.org/install.ps1'))" && SET PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin`
3. Close the command prompt as Chocolatey will not be available until you close and reopen.
4. Open a command prompt with Administrator access
5. Intall Ruby: `choco install ruby -y`
6. Close and open a new command prompt with Administrator access
7. Install Jekyll: `gem install jekyll`

Found this guide here.

Read Getting started with jekyll online: https://riptutorial.com/jekyll/topic/2669/getting-started-with-jekyll

Chapter 2: Assets

Examples

Sass/SCSS

By default, all your `.scss` partials go in the `<source>/_sass` folder.

/_sass/base.scss

```
body {
  margin: 0;
}
```

Your main `.css` or `.scss` files go in the `<source>/css` folder. Note: the two first two lines of triple dashes are necessary in order for Jekyll to transpile your `.scss` file to `.css`.

/css/main.scss

```
---
---

@import "base";

div {
  color: #000;
}
```

A transpiled `.css` file will then appear in `_site/css/` when you build your site:

/_site/css/main.css

```
body {
  margin: 0 }
div {
  color: #000 }
```

The css file can be referenced by your `.html` files like so:

/_layouts/home.html

```
<!DOCTYPE html>
<html>
<head>

<link rel="stylesheet" href="/css/main.css">
</head>

<body>
</body>
</html>
```

Read Assets online: https://riptutorial.com/jekyll/topic/9340/assets

Chapter 3: Building a Jekyll site folder

Syntax

- jekyll build [flag] [value] # Build the site with the setting specified by <flag> and <value>
  # cf. list of available settings for Jekyll

Remarks

If you edit `_config.yml` and you are using `--watch`, you need to restart the command to apply the changes.

Examples

Building the site to a folder

```
$ jekyll build
# The current site folder will be built into the ./_site directory

$ jekyll build --destination /var/www/
# The current site folder will be generated into /var/www/

$ jekyll build --watch
# The current site folder will be built into the ./_site directory and will be kept up to date
with the source until you press CTRL+C to kill the process
```

Building with a specific Jekyll environment

you can set a Jekyll environment and value, when build time

```
JEKYLL_ENV=production jekyll b
```

```
JEKYLL_ENV=production jekyll build
```

```
JEKYLL_ENV=production bundle exec jekyll build
```

---

if your code contains the bellow snippet, `analytics.html` will not be included unless your building with `JEKYLL_ENV=production`

```
{% if jekyll.environment == "production" %}

   {% include analytics.html %}

{% endif %}
```

Read Building a Jekyll site folder online: https://riptutorial.com/jekyll/topic/3534/building-a-jekyll-site-folder

Chapter 4: Collections

Examples

Configuring a New Collection

To create an albums collection, add the following to your `config.yml` file:

```
 collections:
- albums
```

Create a corresponding folder at the root of your Jekyll install, named exactly what you put in your `config.yml` file with an additional prepended underscore; in our example, `<source>/_albums`.

Adding documents to this folder will add items to your collection. Any variables included in a file's YAML front matter is read in as data attributes, and everything after it is included in the item's content attribute. If no YAML front matter is provided, Jekyll will not generate the file in your collection.

Collection metadata can be configured in `config.yml`:

```
collections:
  albums:
    type: media
```

In this example, `type: media` could be any arbitrary key-value pair.

Defaults for items within a collection can also be set within `config.yml`.

```
defaults:
  - scope:
      path: ""
      type: albums
    values:
      publisher: Me Publishers Inc
```

Given this default, any item within the `albums` collection that does not explicitly set `publisher` within its front matter will have its `publisher` variable set to `Me Publishers Inc` at build time.

Official Jekyll Collections Docs

Accessing A Specific Collection Item

As of Jekyll 3.2, you can use the filter `where_exp` to filter a collection by any of its properties.

Say you have the following collection item in an "albums" collection:

```
---
```

```
title: My Amazing Album
---
...
```

You can combine the `where_exp` and `first` filters to grab just that one item:

```
{% assign album = site.albums
    | where_exp:"album", "album.title == 'My Amazing Album'"
    | first %}
```

The `first` filter is necessary because `where_exp` returns an array of matched items.

You can then use it any way you'd like:

```
<h1>{{ album.title }}</h1>
```

Looping Through All Items in a Collection

Given an 'albums' collection, you can loop through and output each item:

```
{% for album in site.albums %}
    {{ album.content }}
{% endfor %}
```

Any custom front matter variables are also available within the loop.

```
{% for album in site.albums %}
    {{ album.title }}
    {{ album.content }}
{% endfor %}
```

Adding an Item to a Collection

Given an 'albums' collection, an item can be added by creating a file in the `<source>/_albums` directory. Note that files not including frontmatter will be ignored.

For instance, adding a file called `my_album.md` to the `_albums` directory would add it to the collection:

```
---
title: "My Album"
---
...
```

Everything after the second set of three dashes is included in `item.content`. Both an item's content and its front matter variables can be accessed like so:

```
{% for album in site.albums %}
    {{ album.title }}
    {{ album.content }}
{% endfor %}
```

If you don't wish to include any front matter variables in a collection item, two sets of three dashes are sufficient front matter to have an item included:

```
---
---
...
```

Read Collections online: https://riptutorial.com/jekyll/topic/4385/collections

Chapter 5: Front Matter

Remarks

Front Matter tells Jekyll to parse the page. It can contain properties for the page.

Examples

Basic Front Matter in Jekyll

Front matter tells Jekyll to parse a file. You add predefined variables, which are YAML sets, to the front matter. Then, you can use Liquid tags in your files to access the front matter.

Front matter is indicated with two triple-dashed lines. You must place the variables between the two triple-dashed lines, and you must place front matter at the top of the file.

For example, the front matter for two posts in a blog about learning different musical instruments might look like this:

```
---
layout: post
title: "Learning to Play the Violin by Self-Study"
date: 2016-07-25
tags: [violin, self-study, beginner]
---
```

and

```
---
layout: post
title: "Taking Lessons on the Violin as a Beginner"
date: 2016-07-25
tags: [violin, lessons, beginner]
---
```

For more information about front matter, please see Jekyll Tips: Front Matter.

Using Custom Variables

You can also put custom variables in the front matter. These can be reused in the page layout.

For example, if your front matter looks like this:

```
---
layout: post
title: "Using Custom Variables!"
date: 2016-07-25
chicken: "I like Chicken."
---
```

You can use the `chicken` variable in the post layout like this:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>{{ post.chicken }}</title>
  </head>
  <body>
    ...
```

The `chicken` variable will be substituted in instead of `{{ post.chicken }}`

Predefined Variables

There are a number of predefined global variables that you can set in the front matter of a page or post.

Variable	Description
`layout`	If set, this specifies the layout file to use. Use the layout file name without the file extension. Layout files must be placed in the `_layouts` directory.
`permalink`	If you need your processed blog post URLs to be something other than the site-wide style (default `/year/month/day/title.html`), then you can set this variable and it will be used as the final URL.
`published`	Set to false if you don't want a specific post to show up when the site is generated.

There are also predefined variables specifically for posts.

Variable	Description
`date`	A date here overrides the date from the name of the post. This can be used to ensure correct sorting of posts. A date is specified in the format `YYYY-MM-DD HH:MM:SS +/-TTTT`; hours, minutes, seconds, and timezone offset are optional.
`category` `categories`	Instead of placing posts inside of folders, you can specify one or more categories that the post belongs to. When the site is generated the post will act as though it had been set with these categories normally. Categories (plural key) can be specified as a YAML list or a comma-separated string.
`tags`	Similar to categories, one or multiple tags can be added to a post. Also like categories, tags can be specified as a YAML list or a comma-separated string.

Read Front Matter online: https://riptutorial.com/jekyll/topic/3756/front-matter

Chapter 6: Hosting

Examples

GitHub Pages

GitHub offers unlimited hosting for users or organizations and project site. Both Jekyll and static files are available.

Here are the steps in hosting your Jekyll blog on Github.

Setup

Users or organizations site

1. Create a repository named **username.github.io**, where username is your username (or organization name) on GitHub.
2. Clone the repository onto your computer:

   ```
   $ git clone https://github.com/username/username.github.io
   ```

3. Enter the project folder, bootstrap, design and debug your site:

   ```
   $ cd username.github.io
   $ bundle install
   $ bundle exec jekyll serve
   ```

4. Commit and push the repository:

   ```
   $ git add --all
   $ git commit -m "Initial commit"
   $ git push -u origin master
   ```

Now you should be able to go to **username.github.io** to see your blog.

Project site

Project site can be enabled in every repository including private repositories.

1. Enable project site.

   Go to Settings-GitHub Pages-Sources, choose a source to switch on GitHub Pages for the repository.

2. Build site

You may build a Jekyll site from scratch or use Theme Chooser to find a theme for your project site.

3. Edit content

4. Commit

Now you should be able to go to **username.github.io/your-project** to see your project site.

Custom Domains

1. Open Settings->GitHub Pages->Custom domain, add your custom domain.
2. Create a `CNAME` file:

```
$ cd username.github.io
$ echo "example.com" > CNAME
```

3. Commit and push

```
$ git commit -m "Add CNAME" CNAME
$ git push -u origin master
```

Restrictions

Plugins

Jekyll has a plugin system with hooks that allow you to create custom generated content specific to your site. However, GitHub Pages only allows a white list of plugins for security reasons.

Here is the white list:

- Jekyll Sitemap
- Jekyll SEO Tag
- github-metadata
- Jekyll Feed
- Jekyll Redirect From
- Jemoji
- Jekyll Mentions

To avoid the inconsistency with GitHub Pages, you may use `--safe` to serve in local.

You can still use all plugins by publishing your generated site to GitHub Pages, by converting the site locally and pushing the generated static files to your GitHub repository instead of the Jekyll source files.

Markdown Engine

Since 01/05/2016, GitHub Pages supports only kramdown as Markdown engine.

See https://github.com/blog/2100-github-pages-now-faster-and-simpler-with-jekyll-3-0 for more detail.

Sources

GitHub allows you to set Jekyll sources to either `master` branch, `gh-pages` branch or `/docs` folder in `master` branch.

> A full tutorial is available at https://pages.github.com/

Local machine

For testing purposes, you can host your blog on your local machine. After setting up and making any changes, Jekyll can server the blog to http://localhost:4000. On the command line in the root directory of the project, run:

```
$ bundle exec jekyll serve
```

The `bundle exec` part is optional, but if you use Bundler, it ensures the gem dependacies are up-to-date. For a quicker Edit-Build-Test loop, use the `--draft` option to build articles from the `_drafts` directory:

```
$ bundle exec jekyll serve --draft --detach
```

Using `--detach` puts the process in the background so that the command prompt can be used for something else.

(As of version 2.4, the `--watch` option is enabled by default. If, by chance, you have an older version you'll need to add that option so that changes are monitored.)

You can also set the destination directory to a directory on a different web server such as Apache, nginx, or Lighttpd:

```
$ jekyll build --destination /path/to/web_server/root
```

CloudCannon hosting (and CMS)

CloudCannon offers hosting and a CMS for Jekyll applications. Here are the steps in hosting your Jekyll application on CloudCannon (http://cloudcannon.com).

Without version control:

- Create your blog locally using some local files and `jekyll serve`
- Create a CloudCannon account and create a new site
- Drag your site to the 'File Browser' within CloudCannon

With version control:

- Create a repository on Github or Bitbucket
- Create your blog locally using some local files and `jekyll serve`
- Create a CloudCannon account and create a new site
- Connect your Github or Bitbucket repository

Note that CloudCannon is not completely free. In the free plan you can use CloudCannon as a graphical CMS, but you will need external hosting. In the paid plan hosting is also included.

Read Hosting online: https://riptutorial.com/jekyll/topic/3614/hosting

Chapter 7: Importing

Remarks

More information can be found at http://import.jekyllrb.com/

Examples

Introduction

If you're switching to Jekyll from another blogging system, Jekyll's importers can help you with the move. Most methods listed on this page require read access to the database from your old system to generate posts for Jekyll. Each method generates `.markdown` posts in the `_posts` directory based on the entries in the foreign system.

Installation

Because the importers have many of their own dependencies, they are made available via a separate gem called jekyll-import. To use them, all you need to do is install the gem, and they will become available as part of Jekyll's standard command line interface.

```
$ gem install jekyll-import
```

Usage

You should now be all set to run the importers with the following incantation:

```
$ ruby -rubygems -e 'require "jekyll-import";
    JekyllImport::Importers::MyImporter.run({
      # options for this importer
    })'
```

Where `MyImporter` is the name of the specific importer.

Read Importing online: https://riptutorial.com/jekyll/topic/4225/importing

Credits

S. No	Chapters	Contributors
1	Getting started with jekyll	ashmaroli, Community, ganesshkumar, geek1011, Jackie Chen, Jacob Linney, JoostS, yaitloutou, Yana
2	Assets	Etrain
3	Building a Jekyll site folder	geek1011, yaitloutou
4	Collections	irowe, Nathan Arthur
5	Front Matter	geek1011, hillary.fraley, Jackie Chen
6	Hosting	approxiblue, David Zhang, geek1011, Jackie Chen, Jacob Linney, Jon Ericson, JoostS
7	Importing	Jackie Chen