Blogging

03 Jekyll with Chirpy theme

Contents

[2 Overview 2](#_Toc63921991)

[3 Install JEKYLL 2](#_Toc63921992)

[3.1 Install JEKYLL on WSL (windows subsystem for linux) 3](#_Toc63921993)

[3.1.1 Once installed but want to run Jekyll on different folder 3](#_Toc63921994)

[3.1.2 Important 3](#_Toc63921995)

[4 Get started - Pick a theme for Jekyll 3](#_Toc63921996)

[5 Starting with Chirpy 4](#_Toc63921997)

[5.1 Make it work 4](#_Toc63921998)

[5.2 Start local development 5](#_Toc63921999)

[6 Jekyll – quick course 6](#_Toc63922000)

[6.1 FrontMatter 6](#_Toc63922001)

[6.2 Liquid 6](#_Toc63922002)

[6.2.1 Extra liquid filter plugins 6](#_Toc63922003)

[6.3 Basic Jekyll concepts 6](#_Toc63922004)

[6.3.1 Layouts 7](#_Toc63922005)

[6.3.2 Posts 7](#_Toc63922006)

[6.3.3 Pages 7](#_Toc63922007)

[6.3.4 Collections 7](#_Toc63922008)

[6.3.5 Includes 7](#_Toc63922009)

[6.3.6 Tags vs categories 7](#_Toc63922010)

[6.3.7 …and the rest 7](#_Toc63922011)

[6.4 Jekyll build order (limitations) 7](#_Toc63922012)

[7 Jekyll – Advanced 8](#_Toc63922013)

[7.1 Config file error - Rules 8](#_Toc63922014)

[7.2 Base URL 8](#_Toc63922015)

[7.3 Permalinks and site generation 8](#_Toc63922016)

[7.4 Collections vs arrays 8](#_Toc63922017)

[7.5 New configuration values inside \_config.yml 9](#_Toc63922018)

[7.6 Allowing static files inside \_posts folder 9](#_Toc63922019)

[8 Chirpy 9](#_Toc63922020)

[8.1 Configuration 9](#_Toc63922021)

[8.2 Basics 9](#_Toc63922022)

[8.3 Customization 10](#_Toc63922023)

[8.3.1 Extending liquid filters 10](#_Toc63922024)

[8.3.2 Allowing landing page 10](#_Toc63922025)

[8.3.3 Fixed pagination issues 11](#_Toc63922026)

[8.3.4 Menu via YAML + fixing the “is active” styling 11](#_Toc63922027)

[8.3.5 Archives layout 12](#_Toc63922028)

[8.3.6 Including special “partial” content 13](#_Toc63922029)

[8.3.7 Cookie consent integration 14](#_Toc63922030)

[8.3.8 Show article series 15](#_Toc63922031)

[8.4 Other extras 15](#_Toc63922032)

# Overview

As it was Karl Hughes who recommended Jekyll I gave it a try. Had a luck to found the most complete Chirpy theme, which helped me to learn a lot.

**The main issues with Jekyll are:**

* it requires Ruby and using and setting it up on windows is not the most convenient thing to do
* the build process (with watch) has a tendency to freeze often
* restarting windows linux subsystem (and thus freeing up the ports of those hung processes) is almost impossible
* **templating very limited**, you often need to set extra variables before if/then checks
* **if same filename is used multiple times and when permalinks use no dates, we might get non-working, duplicate urls**

The real pros:

* **language extensible via custom plugins**

Would consider not to use \_POSTS and go rather with a collection:

* **post permalinks can’t use front matter title**
* **BUT just posts do allow date-related variables to use in permalinks**
* post filenames must contain date-field and a name for an article (name is used for URL)

# Install JEKYLL

There are multiple ways on how to install Jekyll. Unfortunatelly some of the steps are a bit outdated (or just system dependent), see: <https://jekyllrb.com/docs/installation/windows/>

I did not manage to get Ruby work directly under windows, but **was able to make it work under the WSL.**

## Install JEKYLL on WSL (windows subsystem for linux)

Based on <https://jekyllrb.com/docs/installation/windows/>

Solved some issues, missing part in RED.

sudo apt-get update -y && sudo apt-get upgrade -y

sudo apt-get install software-properties-common

sudo apt-add-repository ppa:brightbox/ruby-ng

sudo apt-get update

sudo apt-get install ruby2.5 ruby2.5-dev build-essential dh-autoreconf

sudo gem update

bundle exec jekyll –v

bundle exec jekyll serve –watch

### Once installed but want to run Jekyll on different folder

sudo gem install jekyll bundler

bundle init

bundle install

bundle exec jekyll serve --wathch --port 4001

Sometimes you might need to install additional gems or update existing ones:

sudo gem install jekyll-remote-theme

sudo gem update

### Important

When debugging, sometimes the serve process might stall. Even if you kill the terminal, the port won’t be released. Only a restart of the WSL helps, but there’s a quicker option: **kill the terminal and serve jekyll on a different port.**

**Services: LxssManager**

# Get started - Pick a theme for Jekyll

The best way to start with Jekyll is to download a full theme, I picked Chirpy and I must admit it was the best choice I could made. It’s an ideal theme to get started and do experiments and ideal for blogging.

<https://jamstackthemes.dev/themes/#css=bootstrap>

**For blogs & company sites (the functionality approach):**

* \*\*\* <http://jekyllthemes.org/themes/jekyll-theme-chirpy/> - a bit complex, but a **lot of useful things are given out of the box,** the CSS composition lacks reuse but **a very good start to learn things!**
* <https://github.com/thedevslot/WhatATheme/> - clean, SPA mixed with blog, missing tags and categories view, has social share…
* \*\*\* <https://github.com/tzuehlke/jekyll-uno-timeline> – misses tag/category menu (easy to add), otherwise **very impressing,** possibly the this is the one to use as a professional theme!
* <https://github.com/chrisbobbe/jekyll-theme-prologue> - clean design, interesting 2 level menu, no tags or categories, no heading navigation in posts
* [https://github.com/stackbithq/stackbit-theme-fresh/](https://jamstackthemes.dev/demo/theme/stackbit-fresh-unibit/) - very clean but missing tags ^ heading navigation, has **no social share…**
* <https://github.com/artemsheludko/flexible-jekyll> – lightweight (simple) yet impressive, but **misses main menu** as menu for tags/categories, otherwise a clean concept. **The picture preview makes this site look awesome!**
* <https://github.com/sfreytag/friday-theme> - very simple design, has map integration

**Nice styling:**

* modern design <https://chrisbobbe.github.io/jekyll-theme-prologue/>
* blog & doc in 1 sample <https://github.com/aksakalli/jekyll-doc-theme>
* material design <https://github.com/ShawnTeoh/matjek> no demo
* stylish design <https://jamstackthemes.dev/demo/theme/stackbit-fresh-unibit/>
* unique design <https://jamstackthemes.dev/demo/theme/stackbit-exto-unibit/>
* <https://jekyllthemes.io/theme/automatic-app-landing-page>
* <https://jekyllthemes.io/theme/wrap-onepage-jekyll-theme>

# Starting with Chirpy

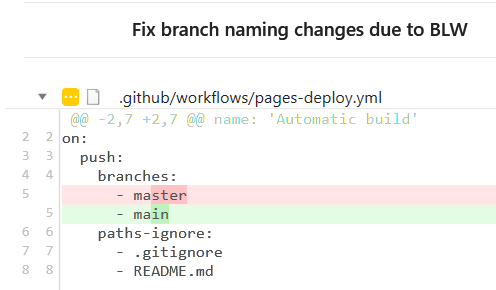
## Make it work

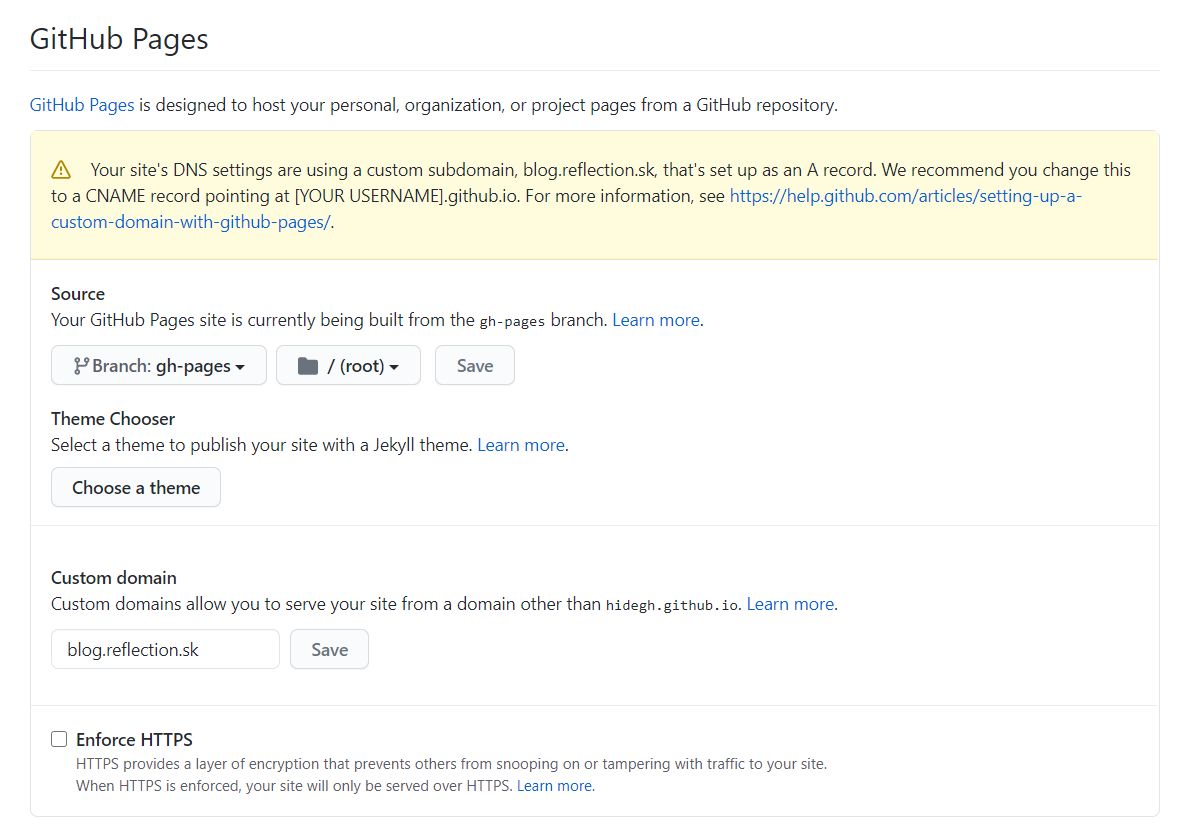
<http://jekyllthemes.org/themes/jekyll-theme-chirpy/>

<https://github.com/cotes2020/jekyll-theme-chirpy/>

Steps:

* Create YOUR OWN github pages
* Clone that REPO
* Copy the THEME into your REPO
* Need to execute the init script (bash works on latest windows) – without init there will be some build issues… **bash, then tools/init.sh**
* Fix the pages-deploy.yml (see screenshot below)
* Commit
* Wait until page builds
* Make sure to set up gh-pages correctly (see screenshot below)
* Load your page served on GH





<https://docs.github.com/en/github/working-with-github-pages/about-github-pages-and-jekyll>

## Start local development

Start VSCode, open up terminal, start bash then start serving the site:

bash

bundle exec jekyll serve --wathch --port 4001

# Jekyll – quick course

This link contains a lot of useful information to start with Jekyll: <https://jekyllrb.com/docs/step-by-step/01-setup/>

And also a tutorial is avail: <https://jekyllrb.com/tutorials/home/>

## FrontMatter

Any file that contains a YAML front matter block will be processed by Jekyll as a special file.

NOTE: **this is important, as by default Jekyll ignores additional / extra files inside special directories!**

## Liquid

<https://shopify.github.io/liquid/>

Jekyll uses the Liquid templating language to process templates. It has some filters, tags…

### Extra liquid filter plugins

We’re free to add new liquid filters – we have to put them into the \_plugins directory:





## Basic Jekyll concepts

### Layouts

Jekyll uses layouts to generate static HTML files. This post describes how layouts work: <https://jekyllrb.com/tutorials/convert-site-to-jekyll/#how-layouts-work>

Recap:

* **Layouts do stack.** The actual layout’s output (content) is projected into the upper (parent) one…
* The **defaults** section of the **\_config.yml** file contains a setup, where we can define for each type of content and/or each path the default layout that should be used. This layout can be overridden by (inside) the FrontMatter.

### Posts

The \_posts folder is where your blog posts live. You typically write posts in Markdown, HTML is also supported. To create a post, add a file to your \_posts directory with the following format: YEAR-MONTH-DAY-title.MARKUP

Now we can have the **\_post** folder in the root, but **we can have multiple \_post folder inside other folders** – in the latter case the other folder names are automatically used as categories (which will be expanded via the categories defined in the font matter).

### Pages

Are similar to posts, they are just a different collection… Every file that has a FontMatter and is processed (and is not a post) will be put into this collection.

### Collections

If **taxonomies** (categories and tags) would not be tied just to posts, we could use a custom collection for those posts. This would allow us to get rid of the naming convention (limitation) of posts. Custom collections are accessible from the global site variable.



### Includes

**Unfortunately, those are very limited but well suited to help us to build the layout in a more grained way.**

### Tags vs categories

Tags and categories are keywords (list of keywords). The difference between tags & categories is that **categories can be used in the URL,** while tags can’t.

### …and the rest

Make sure to check out: <https://jekyllrb.com/docs/>

## Jekyll build order (limitations)

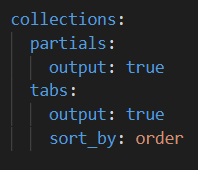
<https://github.com/cotes2020/jekyll-theme-chirpy/issues/266#issuecomment-770428733>

<https://github.com/jekyll/jekyll/issues/8565>

**Especially important when including other processed pages in our content.** As rule of thumb:

1. Use \_partials dir.
2. If you use custom collection, make sure the one from you intend to import content is defined as the 1st (or before the others that use them) – **see our custom \_partial collection/directory.**

**See \_config.yml:**



# Jekyll – Advanced

## Config file error - Rules

This error means that your site failed to build because the \_config.yml file contains syntax errors.

**To troubleshoot, make sure that your \_config.yml file follows these rules:**

* Use spaces instead of tabs.
* Include a space after the : for each key value pair, like timezone: Africa/Nairobi.
* Use only UTF-8 characters.
* Quote any special characters, such as :, like title: "my awesome site: an adventure in parse errors".
* For multi-line values, use | to create newlines and > to ignore newlines.

To identify any errors, you can copy and paste the contents of your YAML file into a YAML linter, such as **YAML Validator.**

## Base URL

<https://byparker.com/blog/2014/clearing-up-confusion-around-baseurl/>

## Permalinks and site generation

Good article, also explaining permalinks and Jekyll site generation: <https://nhoizey.github.io/jekyll-postfiles/>

## Collections vs arrays

The **site.categories** is a hash object. There’s no out of the box solution to convert it to an array of key/values, nor a way to fetch keys.

**But there’s a simple solution to fetch all keys and thus enable easy filtering:** <https://github.com/MichaelCurrin/jekyll-keys-filter>

NOTE: keys filter might be already a part of the filters avail. in Jekyll (based on Shopify API docs)

{% assign collection\_keys\_filtered = site.categories | keys | filter\_exp: "key", "key != 'Company'" %}

See: 8.2.1

## New configuration values inside \_config.yml

By adding new config values to \_config.yml – NOTE: **they had to be unique and non-conflicting with existing global values** – we managed to tweak whether related articles and post-navigation is shown after the content.

## Allowing static files inside \_posts folder

**Note:** by default Jekyll generates a totally new structure based on your \_config.yml into the \_site folder. But any assets from \_posts folder are not copied over.

**There’s a custom plugin, which does the copy of extra asset files.** Custom plugins were not supported by github pages – **but now they are, via github actions!**

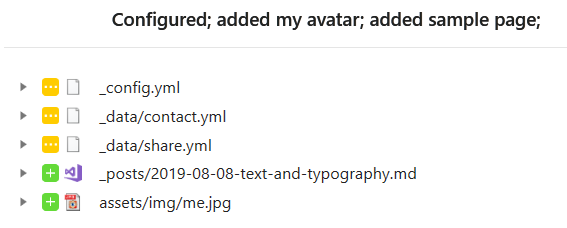
**We just need to add   gem "jekyll-postfiles" into the gemfile!**

<https://jekyllrb.com/docs/continuous-integration/github-actions/>

In the past similar extra steps were needed: <http://joshfrankel.me/blog/deploying-a-jekyll-blog-to-github-pages-with-custom-plugins-and-travisci/>, <https://www.travis-ci.com/signup>

# Chirpy

## Configuration



**Some values has to be set on multiple places,** like some social platform URL / CONFIG VALUES (names, etc.). URL usually is used to link the page, the other values are then used in case of sharing and so, …

## Basics

<https://keirwhitaker.com/blog/setting-development-variables-in-jekyll/>

#### CSS / JS

**css-selector.html** – loads just some CSS files, based on rel. path

**js-selector.html** – same as CSS just for JS files

#### Layouts / partials

The **index.html** is the entry. Those layouts do stack: <https://jekyllrb.com/tutorials/convert-site-to-jekyll/#how-layouts-work> so **index.html** -> **\_layout/home.html** -> **\_layout/page.html** -> **\_layout/default.html** (this contains all the main HTML parts) -> \_layout/compress.html (which does just some extra processing over the generated HTML content)…

* **default.html** – HTML page structure, with head, sidebar, topbar, content, footer, search-loader, etc. included
* **page.html** – is a content that is injected into the default.html, it is responsible for displaying a page, with a dynamic title and with disqus for commenting (if page.comment is truthy)
* **post.html** – just like page.html, it’s content is injected into default.html, it is responsible for displaying posts, with all the extras, like: time-ago, readtime, title (also series title), image, main content, navigation (related posts, also post series, navigation…) – it fulfills the main blogging functionality…
* **home.html** – it’s originally the main page, it lists the pinned posts then the rest of the posts with a paginator at the bottom.
* The **sidebar.html** contains the main menu on the left. It’s the **\_tab** folder (collection) from where the content is picked up.

If there’s no layout defined, the \_layout/default.html is used, so the content of the MD is injected into that layout. See original: \_tabs/about.md

## Customization

### Extending liquid filters

* Collection filtering: keys, skip, take
* String operations: startswith, endswith
* Object filters: property (selector)

### Allowing landing page

Originally we have the list of blogs on the index page. Also the official paginator is allowed just on a single page. **But with Github Actions in use we can use any other paginator plugin, we do not need to stick with the “safe” ones.**

1. The original post-paginator.html had to be fixed so it accepts the **paginate\_path** value from **\_config.yml**
2. Set paginate\_path from /:num to: **/blog/:num**
3. Create a **/blob/index.html** that has the layout set to **posts** in the Front Matter
4. Change original /index.html to use the **home** layout
5. Create the landing page content inside the **\_layouts/home.html** file

For advanced pagination see: <https://github.com/sverrirs/jekyll-paginate-v2>

Before github actions, this was the way to bypass plugin limitations: <https://blog.rafael.picanco.nom.br/2015/09/06/jekyll-pagination-on-github-pages-without-plugins.html>

### Fixed pagination issues

**post-paginator.html**

**The current paginator in the CHIRPY theme had a BUG:** it’s not working correctly, when the paginate\_path is set in the \_config.yml: e.g. try value /posts/:num/ - so we introduced a fix on it: <https://github.com/hidegh/hidegh.github.io/commit/a97a0aae9e31a16bc596aefe55d215dcfbb4978e>

Links used to fix this plugin: <https://blog.webjeda.com/jekyll-pagination/#pagination-can-be-done-only-on-indexhtml>

### Menu via YAML + fixing the “is active” styling

<https://stackoverflow.com/questions/8340170/jekyll-automatically-highlight-current-tab-in-menu-bar>

**By default sidemenu.html is rendered from the \_tabs folder/collection** and in some cases selected item highlighting does not works, e.g.: if you select a concrete category/tag/post…

  <ul class="w-100">

    <!-- home -->

    <li class="nav-item{% if page.layout == 'home' %}{{ " active" }}{% endif %}">

      <a href="{{ '/' | relative\_url }}" class="nav-link">

        <i class="fa-fw fas fa-home ml-xl-3 mr-xl-3 unloaded"></i>

        <span>{{ "HOME" }}</span>

      </a>

    </li>

    <!-- blog -->

    {% assign hasSeparateBlogLink = site.paginate\_path | startswith: "/blog/" %}

    {% if hasSeparateBlogLink %}

    <li class="nav-item{% if page.layout == 'posts' %}{{ " active" }}{% endif %}">

      <a href="{{ '/blog' | relative\_url }}" class="nav-link">

        <i class="fa-fw fas fa-home ml-xl-3 mr-xl-3 unloaded"></i>

        <span>{{ "BLOG POSTS" }}</span>

      </a>

    </li>

    {% endif %}

    <!-- the real tabs -->

    {% for tab in site.tabs %}

    <li class="nav-item{% if tab.url == page.url %}{{ " active" }}{% endif %}">

      <a href="{{ tab.url | relative\_url }}" class="nav-link">

        <i class="fa-fw {{ tab.icon }} ml-xl-3 mr-xl-3 unloaded"></i>

        <span>{{ tab.title | upcase }}</span>

      </a>

    </li> <!-- .nav-item -->

    {% endfor %}

  </ul> <!-- ul.nav.flex-column -->

#### Tweaks

1. We define a menu inside the \_data/menu.yml
2. We are allowed to add multiple match urls – even to use \* for partial match
3. We can have also multi-level menu

If we define a *navigation list* on your **\_config.yml** like this:

# menu.html include configuration

# has advanced URL matching and thus allows to match on partial URL's

# always enter the desired redirection url as 1st, then add any other URLs, feel free to append \* for partial matching

# items with matched URL will have an active class set

-

  title: Home

  url: /

  icon: fas fa-home

  # sample for multiple menu nesting

  # menu:

  #   -

  #     title: Home intro

  #     url: /home/intro/\*

  #     icon: fas fa-home

  #     menu:

  #       -

  #         title: Home intro - inner

  #         url: /home/intro/inner\*

  #         icon: fas fa-home

  #   -

  #     title: Home outro

  #     url: /home/outro/\*

  #     icon: fas fa-home

-

  title: Posts

  url: /blog/\*, /posts/\*

  icon: fas fa-blog

-

  title: Categories

  url: /tabs/categories/, /categories/\*

  icon: fas fa-stream

-

  title: Tags

  url: /tabs/tags/, /tags/\*

  icon: fas fa-tags

-

  title: Archives

  url: /tabs/archives/

  icon: fas fa-archive

-

  title: About

  url: /tabs/about/

  icon: fas fa-info

We implemented a new **\_includes/menu.html** template to work with the configuration above.

And inside the **\_includes/sidebar.html** we replaced the menu snippet with the following:

<!-- navigation via nestable, reusable include -->

<div class="w-100">

{% include menu.html menu = site.data.menu multi\_level = true %}

</div>

NOTE: there were some other solutions available out there, they used contains to do the partial match, but they always did a partial match and they always matched just against a single URL only.

### Archives layout

As the original CSS did not allowed to include extra lines below the blogs title, we had to re-create the entire timeline layout (new HTML and new CSS and extended content).

### Including special “partial” content

**Not all folders content are copied to \_site when “compiling”.**

Regardless of if you’re in a collection folder (\_prefix) or a regular static content folder, Jekyll will:

* **ignore folders** with: \_ (underscore) – except if configured as a collection, # (hash), . (dot), ~
* but include folders with: !, $, + prefix

For such includes we had to define a simple layout” \_layouts/simple.html

#### Including collection content into the page

Collection objects are pre-processed and do have an output property, which is rendered as default when using {{ }}.

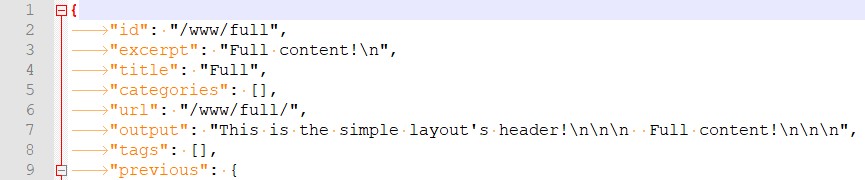
{% assign wf = site.partials | find: "path", "\_partials/full.md" %}

Www full layout: {{ wf.layout }}

{{ wf }}

<textarea>{{ wf | jsonify | strip\_html }}</textarea>

{{ site.partials | find: "path", "\_partials/full.md" }}



The partial content has to be processed before the content that uses it! Otherwise the output property will be empty and the non-processed content property will be displayed.

NOTE: In this example, we used simple layout for the content (and not the full “page” layout)!

#### Including static content into the page

Although static files might be also processed via Front Matter, when working them inside the framework (so when using Jekyll/Liquid templating)…

…we won’t have access to the processed results, the only thing that we will be able to use is the original, unprocessed HTML/MD content!

Here’s a sample on how to include static content:

{% assign p =  site.pages | find: "path", "www/www.md" %}

Www pages Layout {{ p.layout }}

{{ p.content | markdownify }}

<textarea>{{ p | jsonify | strip\_html }}</textarea>

{{ site.pages | find: "path", "www/www.md" }}

To make it simpler, we added some new filters (via plugins), with them it’s enough to write:

{{ site.pages | find: "path", "www/www.md" | property: "content" | markdownify }}

### Cookie consent integration

Jekyll sample: <https://ptc-it.de/enabling-cookie-consent-with-jekyll-minimal-mistakes/>

**Osano cookie consent:**

<https://www.osano.com/cookieconsent/documentation/javascript-api/>

Design your JS code: <https://www.osano.com/cookieconsent/download/> - select: start coding (free edition)!

**Quick overview:**

* Every script (like google analytics, disqus) has to be added as a function (see below)
* Consent initialization has to be tweaked

**Steps:**

1. We created an **/assets/js/scripts\_config.js** – where we IIFE we defined
   1. Default script load
   2. Script load when cookies are disabled
   3. Script loading for enabled cookies
   4. Cookie consent set-up
2. Inside **\_layouts/default.html** we added:

<!-- scripts - allowing consent -->

<script src="/assets/js/scripts\_config.js"></script>

<script>

  (function() {

    var enableConsentEngine = JSON.parse('{{ site.cookie\_consent.enabled }}');

    var consentUrl = '{{ site.cookie\_consent.consent\_url }}';

    scripts\_config.loadScripts(enableConsentEngine, consentUrl);

  }());

</script>

1. We also had to change the scripts for GA / Disqus to change so they don’t get called immediately, just via our scrip load methods…

### Show article series

This function is implemented inside the: **\_includes/related-series-posts.html**

A small change inside **\_layouts/post.html** was necessary:

1. Include series heading (title) above the post
2. Add the new series paginator below the post and disable the default related post paginator

{% if page.series != nil %}

  {% include related-series-posts.html %}

{% endif %}

<!-- https://shopify.github.io/liquid/basics/operators/#order-of-operations -->

{% if site.post\_view.show\_related and page.series == nil or site.post\_view.disable\_related\_on\_series == false %}

  {% include related-posts.html %}

{% endif %}

## Other extras

Subscribing via MailChimp: <https://blog.webjeda.com/jekyll-subscribe-form/>