Jekyll Documentation Theme for Designers

version 3.0

Last generated: August 11, 2015

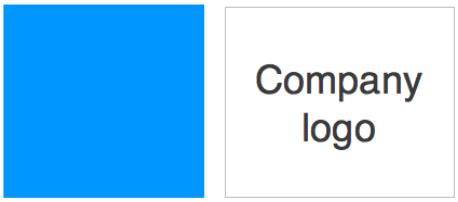

Table of Contents

Getting started

Step 1: Set up the prerequisites

Before you start installing the theme, make sure you have all of these prerequisites in place.

- **Mac computer (recommended)**. If you have a PC, you need to see Jekyll on Windows (http://jekyllrb.com/docs/windows/). Make sure you can get Jekyll working on Windows before proceeding.
- **Ruby (https://www.ruby-lang.org/en/)**. This should already be installed. Type `which ruby` to confirm.
- **Rubygems (https://rubygems.org/pages/download)**. This is a package manager for Ruby (gems). Type `which gem` to confirm.
- **Jekyllrb (http://jekyllrb.com/)**. To install: `gem install jekyll` . Type `which jekyll` to confirm.
- **Git (http://git-scm.com/download/mac)**. Type `which git` to see if you already have it installed.
- **Text editor** (some examples: Sublime Text, Atom, WebStorm)
- **iTerm (http://iterm.sourceforge.net/)** - Optional but recommended instead of Terminal.
- **pygments (http://pygments.org/download/)** - Pygments handles syntax highlighting. In my experiments, the Pygments highlighter seemed better than the default rouge highlighter. To install Pygments, you will need Python installed. (If you don't install pygments, you'll get an error when you build the theme.) To check if Python is installed, type `which python` . To install Pygments: `gem install pygments.rb` . If you want to use `rouge` instead, change `pygments` to `rouge` in the configuration files.

> ❗ **Warning:** These instructions are designed for users on Macs. Jekyllrb supposedly works on Windows but is not officially supported on that platform. See http://jekyllrb.com/docs/windows/ (page 0) for more details.

Step 2: Build the theme

Before you start customizing the theme, make sure you can build the theme with the default content and settings first.

1. Make sure you satisfy all the prerequisites as listed in the previous section, "Prerequisites."
2. Download the theme from the documentation-theme-jekyll Github repository (https://github.com/tomjohnson1492/documentation-theme-jekyll) and unzip it into your ~username/projects folder.

   You can either download the theme files directly by clicking the **Download Zip** button on the right of the repo, or use git to clone the repository to your local machine. Note, however, that you won't be using the pull command to update the theme since you'll be customizing it with your own content and won't want to overwrite those customizations, so there isn't a need to clone it.
3. After downloading the theme, note some unique aspects of the file structure:
   - There isn't a _config.yml file in the root directory because this theme is set up to single source multiple site outputs, not just one version.
   - In the configs directory, there's a separate configuration file for each unique output you're building.
   - Each configuration file has a different preview port.
   - Each configuration file specifies a different project and potentially a different audience, product, platform, and version. By setting unique values for these properties in the includes/custom/conditions.html file, you determine how the sidebar and top navigation is constructed.

> ☑ **Tip:** The main goal of this theme is to enable single sourcing. With single sourcing, you build multiple outputs from the same source, with somewhat different content in each site based on the particular product, platform, version, and audience. You don't have to use this theme for single sourcing, but most tech writing projects involve this requirement.

In general, you build the sites by using build commands that specify the configuration file, like this:

```
jekyll serve --config configs/config_writer.yml
```

The `--config` parameter specifies the location of the configuration file to be used in the build. The configuration file itself contains the destination location for where the site gets built.

There are two configuration files in this project: config_writer.yml and config_developer.yml. The idea is that there's an output specific to writers, and an output specific to developers.

4. In the root directory, you'll find a script called multibuild_web.sh that builds (with `jekyll build`, not `jekyll serve`) both projects with the same command.
5. Open a new tab in iTerm and build an additional output:

```
jekyll serve --config configs/config_designer.yml
```

Open a new tab in your browser and preview the site at the preview URL shown. Notice how the themes differ.

If the theme builds both outputs successfully, great. You can move on to the other sections. If you run into errors building the basic theme, you must solve them before moving on. See for more information.

☑ **Tip:** You can set up profiles in iTerm to initiate all your builds with one selection. See for details.

Questions

If you have questions, contact me at tomjohnson1492@gmail.com. My regular site is http://idratherbewriting.com (page 0). I'm eager to make these installation instructions as clear as possible, so please let me know if there are areas of confusion that need clarifying.

Setting up the configuration file

Summary: The configuration file contains important settings for your project. Some of the values you set here affect the display and functionality of the theme.

Importance of Configuration File

The configuration file serves important functions with single sourcing. For each site output, you create a unique configuration file for that output.

The configuration file contains all the settings and other details unique to that site output, such as variables, titles, output directories, build folders, and more.

Warning: This theme is coded to look for specific values set by the configuration file. If something isn't working correctly, check to make sure that you have the configuration values that are defined here properly configured.

Configuration file options

Some of the options you can set in the configuration file determine theme settings.

Note that you can arbitrary key value pairs in the configuration file, and then you can access them through `site.yourkey` . However, the values in these tables are used to control different aspects of the theme and are not arbitrary key-value pairs.

Cascading configuration files

When you build or serve Jekyll, you specify the configuration file that you want Jekyll to use. If you have multiple sites you're building, you can put all the common configuration values into a base configuration file. For the unique configuration files, you put them into another configuration file. Jekyll will read multiple configuration files.

The build command needs to look like this:

```
jekyll serve --config config_base.yml,config_designer.yml
```

Later configuration files will override settings in earlier configuration files.

Warning: You cannot use a space after the period.

Configuration settings

FIELD	REQUIRED?	DESCRIPTION
project	Required	A unique name for the project. The _includes/custom/{project}/conditions.html file will use this project name to determine what sidebar and top nav data files to use. Make this value unique.
audience	Required	The audience for the output. Each sidebar entry in the _data/sidebar.yml files needs to have an audience attribute that matches the value here in order for the sidebar item to be included.
platform	Required	The platform for the output. The function is the same as audience -- the sidebar file matches up with attributes defined in the configuration file to determine what sidebar items are included in the build.
product	Required	Same function as audience, platform, and version.
version	Required	Same function as audience, platform, and product.

FIELD	REQUIRED?	DESCRIPTION
destination	Required	The folder where the site is built. If you put this into your same folder as your other files, Jekyll may start building and rebuilding in an infinite loop because it detects more files in the project folder. Make sure you specify a folder outside your project folder, by using `../` or by specifying the absolute path, such as /Applications/XAMPP/xamppfiles/ht-docs/myfolder.
sidebar_tagline	Optional	Appears above the sidebar. Usually you put some tag related to the site specific build, such as the audience.
sidebar_version	Optional	Appears below the sidebar_tagline in a smaller font. Usually specifying the version of the documentation.
topnav_title	Required	Appears next to the home button in the top nav bar.
homepage_title	Required	You set the title for your homepage via this setting. This is because multiple files are being displayed on the index.md page depending on the project. Because index.md has `homepage: true` in the frontmatter, then the page layout will use the `homepage_title` property from the configuration file instead of the traditional title in the frontmatter.
site_title	Appears in the webpage title area (on the browser tab, not in the page viewing area).	

FIELD	REQUIRED?	DESCRIPTION
port	Required	The port used in the preview mode. This is only for the live preview and doesn't affect the published output. If you serve multiple outputs simultaneously, the port must be unique.
feedback_email	Gets configured as the email address in the Send Feedback button in the top navigation bar.	
sidebar_accordion	Optional	Boolean. Whether you want the navigation sidebar to use the accordion effect or not. The accordion effect means when you expand a section, the other sections auto-collapse. If you put false, then you can expand multiple sections at once. At the bottom of the navigation sidebar, two links will appear: Collapse All and Expand All. The default is true.
disqus_shortname	Optional	The disqus site shortname, which is used for comments. If you don't want comment forms via disqus, leave this blank or omit it altogether.
markdown	Required	The processor to use for Markdown. This is a Jekyll-specific setting.
redcarpet	Required	Extensions used with redcarpet. You can read more about them by searching for redcarpet extensions online.

FIELD	REQUIRED?	DESCRIPTION
highlighter	Optional	The syntax highlighter used. Rouge is also an option. I think Pygments does a better job. You will need to install Pygments (http://pygments.org/download/) on your machine or else you will see an error.
exclude	Optional	A list of files and directories that you want excluded from the build. By default, all the content in your project is included in the output.
permalink	Required	The structure used for the link URLs. To read more about permalinks, see Permalinks (http://jekyllrb.com/docs/permalinks/).
defaults	Optional	Here you can set default values for frontmatter based on the content type (page, post, or collection).
sass	Required	The directory for sass files. This is a Jekyll-specific settings. Sass isn't customized in this theme.
collections	Optional	Any specific collections (custom content types that extend beyond pages or posts) that you want to define. This theme defines a collection called tooltips. You access this collection by using site.tooltips instead of site.pages or site.posts. Put the tooltip content types inside a folder in your project called _tooltips.
print	Optional	Boolean. Whether this build is a print build or not. This setting allows you to run conditions in your content such as {% if site.print == true %} do this... {% endif %} .

FIELD	REQUIRED?	DESCRIPTION
suffix	Optional	If you publish on Salesforce's site.com, you have to add index.html to the permalink or else the page won't render. If you add `suffix: index.html` in your config file, this suffix will be appended in the homepage URL. If you're not publishing to Salesforce, don't add this property to your configuration file.

Where to store configuration files

In this theme, the configuration files are listed in the configs directory. There are some build scripts in the root directory that simply reference the configuration files.

Series pages

Summary: You can automatically link together topics belonging to the same series. This helps users know the context within a particular process.

You create a series by looking for all pages within a tag namespace that contain certain frontmatter. Here's a demo: .

1. Create the series button

First create an include that contains your series button:

```
<div class="seriesContext">
    <div class="btn-group">
        <button type="button" data-toggle="dropdown" class="bt
n btn-primary dropdown-toggle">Series Demo <span class="care
t"></span></button>
        <ol class="dropdown-menu">
            {% assign pages = site.pages | sort:"weight"  %}
            {% for p in pages %}
            {% if p.series == "ACME series" %}
            {% if p.url == page.url %}
            <li class="active"> → {{p.weight}}. {{p.title}}</l
i>
            {% else %}
            <li>
                <a href="{{p.url}}">{{p.weight}}. {{p.titl
e}}</a>
            </li>
            {% endif %}
            {% endif %}
            {% endfor %}
        </ol>
    </div>
</div>
```

Change "ACME series" to the name of your series.

Save this in your _includes folder as something like series_acme.html.

Note that with pages, there isn't a universal namespace created from tags or categories like there is with Jekyll posts. As a result, you have to loop through all pages. If you have a lot of pages in your site (e.g., 1,000+), then this looping will create a slow build time. If this is the case, you will need to rethink the approach to looping here.

2. Create the "next" include

This will be the next button at the bottom of the page:

```
<p>{% assign series_pages = site.tags.series_acme %}
    {% for p in pages %}
    {% if p.series == "ACME series" %}
    {% assign nextTopic = page.weight | plus: "0.1"  %}
    {% if p.weight == nextTopic  %}
    <a href="{{p.url}}"><button type="button" class="btn btn-pr
imary">Next: {{p.weight}}  {{p.title}}</button></a>
    {% endif %}
    {% endif %}
    {% endfor %}
</p>
```

Change "acme" to the name of your series.

Save this in your _includes folder as series_acme_next.html.

3. Add the right frontmatter to each of your series pages

Now add the following frontmatter to each page in the series:

```
series: "ACME series"
weight: 1.0
```

With weight, you could use 1, 2, 3, etc.., but Jekyll will treat 10 as coming after 1. This is why I use 1.0 and 1.1, 1.2, etc.

If you do use whole numbers, change the `plus: "0.1"` to `plus: "1".

4. Add links to the series button and next button on each page.

On each series page, add a link to the series button at the top and a link to the next button at the bottom.

```
{% include custom/doc/series_acme.html %}

<!-- your page content goes here ... -->

{% include custom/doc/series_acme_next.html %}
```

Changing the series drop-down color

The Bootstrap menu uses the `primary` class for styling. If you change this class in your theme, the Bootstrap menu should automatically change color as well.

Series pages

Summary: You can automatically link together topics belonging to the same series. This helps users know the context within a particular process.

You create a series by looking for all pages within a tag namespace that contain certain frontmatter. Here's a demo: .

1. Create the series button

First create an include that contains your series button:

```
<div class="seriesContext">
    <div class="btn-group">
        <button type="button" data-toggle="dropdown" class="bt
n btn-primary dropdown-toggle">Series Demo <span class="care
t"></span></button>
        <ol class="dropdown-menu">
            {% assign pages = site.pages | sort:"weight"  %}
            {% for p in pages %}
            {% if p.series == "ACME series" %}
            {% if p.url == page.url %}
            <li class="active"> → {{p.weight}}. {{p.title}}</l
i>
            {% else %}
            <li>
                <a href="{{p.url}}">{{p.weight}}. {{p.titl
e}}</a>
            </li>
            {% endif %}
            {% endif %}
            {% endfor %}
        </ol>
    </div>
</div>
```

Change "ACME series" to the name of your series.

Save this in your _includes folder as something like series_acme.html.

Note that with pages, there isn't a universal namespace created from tags or categories like there is with Jekyll posts. As a result, you have to loop through all pages. If you have a lot of pages in your site (e.g., 1,000+), then this looping will create a slow build time. If this is the case, you will need to rethink the approach to looping here.

2. Create the "next" include

This will be the next button at the bottom of the page:

```
<p>{% assign series_pages = site.tags.series_acme %}
    {% for p in pages %}
    {% if p.series == "ACME series" %}
    {% assign nextTopic = page.weight | plus: "0.1"  %}
    {% if p.weight == nextTopic  %}
    <a href="{{p.url}}"><button type="button" class="btn btn-pr
imary">Next: {{p.weight}}  {{p.title}}</button></a>
    {% endif %}
    {% endif %}
    {% endfor %}
</p>
```

Change "acme" to the name of your series.

Save this in your _includes folder as series_acme_next.html.

3. Add the right frontmatter to each of your series pages

Now add the following frontmatter to each page in the series:

```
series: "ACME series"
weight: 1.0
```

With weight, you could use 1, 2, 3, etc.., but Jekyll will treat 10 as coming after 1. This is why I use 1.0 and 1.1, 1.2, etc.

If you do use whole numbers, change the `plus: "0.1"` to `plus: "1".

4. Add links to the series button and next button on each page.

On each series page, add a link to the series button at the top and a link to the next button at the bottom.

```
{% include custom/doc/series_acme.html %}

<!-- your page content goes here ... -->

{% include custom/doc/series_acme_next.html %}
```

Changing the series drop-down color

The Bootstrap menu uses the `primary` class for styling. If you change this class in your theme, the Bootstrap menu should automatically change color as well.