Jekyll theme for documentation — writers

version 4.0

Table of Contents

Handling reviews

Publishing

Special layouts

Introduction

Overview

This site provides documentation, training, and other notes for the Jekyll Documentation theme. There's a lot of information about how to do a variety of things here, and it's not all unique to this theme. But by and large, understanding how to do things in Jekyll depends on how your theme is coded. As a result, these additional details are provided.

The instructions here are geared towards technical writers working on documentation. You may have a team of one or more technical writers working on documentation for multiple projects. You can use this same theme to author all of your documentation for each of your products. The theme is set up to push out documentation for multiple projects all from the same source. You can also share content across projects.

Survey of features

Some of the more prominent features of this theme include the following:

- Bootstrap framework
- Sidebar for table of contents
- Top navigation bar with drop-down menus
- PDF generation (through Prince XML utility)
- Build scripts to automate the workflow
- Notes, tips, and warning information notes
- A nifty system for creating links to different pages
- Tags for alternative nativation
- Content sharing across projects
- Emphasis on pages, not posts
- Relative (rather than absolute) link structure, so you can push the outputs anywhere and easily view them

I'm using this theme for my documentation projects, building about 20 different

Supported features

> Summary: If you're not sure whether Jekyll and this theme will support your requirements, this list provides a semi-comprehensive overview of available features.

Before you get into exploring Jekyll as a potential platform for help content, you may be wondering if it supports some basic features. The following table shows what is supported in Jekyll and this theme.

FEATURES	SUPPORTED	NOTES
Content re-use	Yes	Supports re-use through Liquid. You can re-use variables, snippets of code, entire pages, and more. In DITA speak, this includes conref and keyref.
Markdown	Yes	You can author content using Markdown syntax. This is a

FEATURES	SUPPORTED	NOTES
Collaboration	Yes	You collaborate with Jekyll projects the same way that developers collaborate with software projects. (You don't need a CMS.) Because you're working with text file for-

1. Build the default project

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.

2. After downloading the theme, note some unique aspects of the file structure:

    - Although there's a _config.yml file in the root directory, it's there only so that Github Pages will build the theme. Because the theme is set up for single sourcing, there's a separate configuration file for each unique output you're building.

    - All the configuration files are stored in the configs directory. Each configuration file has a different preview port. If you want, you can build and preview all your outputs simultaneously in different preview servers.

    - 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/

2. Add a new project

> Summary: You add a new project essentially by duplicating all the mydoc project files in the _data, _includes, configs, and other folders. You can add as many projects as you want in this theme.

2. Copy and customize the mydoc folder in configs

In the configs folder, copy the mydoc folder and its contents. Rename it to acme, and then rename each of the config_ files to the outputs you need for your acme project.

In this theme, each output requires a separate config file. If you have 10 audiences theyou want separate sites for each, then then you'll have 10 config files in this directory.

More details about customizing the settings in the configuration files will be explained later. For now you're just duplicating the necessary project files for your new project.

3. Create a includes folder

In the _includes/custom directory, add a new folder there called "acme." This folder should sit parallel to the mydoc folder. This is where you can store includes for your project.

4. Add an acme folder in the root directory

In the root directory, add a folder for your pages called acme (similar to the mydoc folder). Include two subfolders inside acme: files and images.

Inside the mydoc folder, copy the home.md file and add it to the acme folder. (With most Jekyll projects, they open up on the index.html file in the root directory. However, because the pages for each project are stored in subfolders, it was necessary to create a redirect from the index page to the home.md page.)

- mydoc_4_publish.sh
- mydoc_all.sh

6. Copy the URL generator text file

In the root directory, copy urls_mydoc.txt and duplicate it. Change the suffix to urls_acme.txt.

> Tip:

3. Decide on your project's attributes

> Summary: Each project has attributes that define the audience, platform, product, version, and output. These attributes are used in generating the outputs. The attributes function as filtering conditions jETBThe gine wETcontent gets included in the navigation.

4. Set the configuration options

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 — especially the product, platform, audience, and version.

Importance of the 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 most of the settings and other details unique to that site output, such as variables, titles, output directories, build folders, and more.

Configuration file variables

You can define arbitrary key-value pairs in the configuration file, and then you can access them through `PFQB.VLROHBV` , where `VLROHBV` is the name of the key.

However, some of the options you set in the configuration file determine theme settings. These options are required for this theme to work. The required settings are defined in the following tables.

Configuration settings for web outputs

FIELD	REQUIRED?	DESCRIPTION
project	Required	A unique name for the project. The _includes/custom/conditions.html file will use this project name to determine what sidebar and top nav data

FIELD	REQUIRED?	DESCRIPTION
homepage_title	Required	You set the title for your homepage via this setting. This is because multiple projects are all using the same index.md as their homepage. Because index.md has `ELJBM>DB: QORB` in the frontmatter, the "page" layout will use the `ELJBM>DB<QFQIB`

FIELD	REQUIRED?	DESCRIPTION
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. Tooltips are useful for creating UI content. For more information about creating tooltips for UI text, see Help APIs and UI tooltips (page 130).
output	Optional	Boolean. Whether this build is `TB?` or `MAC` . This setting allows you to run conditions in your content such as `X% FC PFQB.LRQMRQ == MAC %Z` do this... `X% BKAFC %Z` . Limit the options to just `TB?` or `MAC` for this setting.
github_editme_path	Optional	A path to configure the Github Edit Me button. Put the path to the branch on Github where you want to edit the theme. Here's a sample: `QLJGLEKPLK1492/AL@RJBKQ>QFLK-QEBJB-GBHVII/BAFQ/OBSFBTP` . In this case, "re-

FIELD	REQUIRED?	DESCRIPTION
footer_image_locatioi	Optioial	The image used in the footer to brand your site. Store this image in the common_images folder so that

5. Customize the conditions file

Summary: The conditions file is included in various parts of the theme. Its purpose is to set attributes as variables that affect how the

You need to duplicate this block for each output you have.

Once you've duplicated the block, make a few customizations:

- In each place that "mydoc" appears, change "mydoc" to "acme".
- Use the same attributes for project, audi1nce, version, product, and platform that you used in your configuration file. (If you don't have a specific attribute value that you need, just put "all".) The values here have to exactly match those in the configuration file.

> Tip: If you want to create signposts in the code as shown in the

6. Configure the sidebar

Summary: The sidebar and top navigation bar read their values from yml files. The navigation components are one of the most unique

```
BKQOFBP
  PR?@>QBDLOFBP
    FQBJP
      QEFOAIBSBI
        QEFOAIBSBIFQBJP
```

Within these levels, you add your content. You can only have two levels in the sidebar. Here's an example:

```
)/ÀR@Ķ
 -> 'BQĶ PQ>OQBA
 -> &B>QROBP
 -> #ÄE$âK
    -> /MQFÄ
    -> !RQLJ>QFÄ
```

"Introduction" is a heading — it's the first level. Beneath it are Getting started and

Configuration is a heading announcing a second level. Below it are Options and Automation — these are on the second level.

more than two levels of navigation anyway, since it creates a paralysis of choice

(If you need deeper sublevels, I recommend creating different sidebars for different pages, which is logic that I haven't coded into the theme but which could probably be added fairly easily. Additionally, if you wanted to create a third level, you could do so by following the same pattern as the second level and customizing a few things. However, the theme is not coded to support additional levels.)

features — these are sub-items for the first level.

You can't add more than two levels. In general, it's a best practice not to create

All of the attributes (which are defined in the conditions.html file) must be met in order to displaw4ausavigacond. T of the attrible) muspresent on bothaw4aet in

```
-    QFQIB:
     >RAFBK@B: TOFQBOP, ABPFDKBOP
     MI>QCLOJ: >II
     MOLAR@Q: >II
     SBOPFLK: >II
     LRQMRQ: MAC
     QVMB: COLKQJ>QQBO
     FQBJP:
     - QFQIB:

       >RAFBK@B: TOFQBOP, ABPFDKBOP
       MI>QCLOJ: >II
       MOLAR@Q: >II
       SBOPFLK: >II
       LRQMRQ: MAC
       QVMB: COLKQJ>QQBO
     - QFQIB:
       ROI: /QL@M>DB.EQJI
       >RAFBK@B: TOFQBOP, ABPFDKBOP
```

7. Configure the top navigation

8. Customize the URL generator

> Summary: You need to customize the URL generator with your project's name. This generator helps you make quick links within your content.

9. Set up Prince XML

> Summary:

Each of your configuration files needs to have a destination like this: `../AL@<LRQMRQP/JVAL@/>AQORQE-G>S>`. That is, the project should build in the doc_outputs folder, in a subfolder that matches the project name.

The purpose of this script is to make a version of the HTML output that is friendly to the Prince XML PDF generator. This version of the output strips out the sidebar, topnav, and other components to just render a bare-bones HTML representation of the content.

=OFK à=OFK à

This is one way the configuration file for the PDF-friendly output differs from the HTML output. (If the PDF isn't building, it'22.4 causethe Pprince-ile -list.txt inthe

The doc_outputs/mydoc/designers contains the site output from mydoc, so when you push content from this folder into Github, you're actually pushing the HTML site output into Github, not the mydoc source filem.

Your delivery team can also grab the site output from these repos. After downloading it, the person unzips the folder and sees the website folders inside.

mydoc_all.sh

Here's what this script looks like:

```
. ABSF@BFKPFDEQ<1<JRIQFPBOSB<MAC.PE； . ABSF@BFKPFDEQ<2<JRIQF?RF
IA<MAC.PE； . ABSF@BFKPFDEQ<3<JRIQF?RFIA<TB?.PE； . ABSF@BFKPFDE
Q<4<MR?IFPE.PE；
```

This script simply runs the other scripts. To sequence the commands, you just separate them with semicolons. (If you added the optional script, be sure to include it here.)

After you've configured all the scripts, you can run them all by running

include it 0ere.)

Markdown or HTML format

Pages can be either Markdown or HTML format (specified through either an .md or .html file extension).

If you use Markdown, you can also include HTML formatting where needed. But not vice versa — if you use HTML (as your file extension), you can't insert Markdown content.

Also, if you use HTML inside a Markdown file, you cannot use Markdown inside of HTML. But you can use HTML inside of Markdown.

For your Markdown files, note that a space or two indent will set text off as code or blocks, so avoid spacing indents unless intentional.

Where to save pages

Store all your pages inside the root directory. This is because the site is built with relative links. There aren't any permalinks or baseurls used in the link architecture.

REDCARPET EXTENSION	DESCRIPTION
no_intra_emphasis	don't italicize words with underscores
fenced_code_blocks	allow three backticks before and after code blocks instead of `<MOB>` tags
tables	allow table syntax
with_toc_data	add ID tags to headings automatically

You can also add "autolink" as an option if you want links such as http://google.com to automatically be converted into links.

> Note: Make sure you leav0 the `TFQE<QL@<A>Q>` option included. This auto-creates an ID for each Markdown-formatted heading, which then gets injected into the mini-TOC. Without this auto-creation of IDs, the mini-TOC won't include the heading. If you ever use HTML formatting for headings, you need to manually add an ID attribute to the heading in order for the heading to appear in the mini-TOC.

Automatic mini-TOCs

By default, a mini-TOC appears at the top of your pages and posts. If you don't want this, you can remov0 the `X% FK@IRAB QL@.EQJI %Z` from the layouts/page.html file.

If you don't want the TOC to appear for a specific page, add `QL@: C>IPB` in the frontmatter of the page.

The mini-TOC requires you to use the `##` syntax for headings. If you use `<E2>` elements, then you must add an ID attribute for the h2 element in order for it to appear in the mini-TOC.

Specify a particular page layout

The configuration file sets the default layout for pages as the "page" layout.

You can create other layouts inside the layouts folder. If you create a new layout, you can specify that your page use your new layout by adding `I>VLRQ: JVI>VLRQ.EQJI` in the pageout,

WebStorm Text Editor

> Summary: You can use a variety of text editors when working with a Jekyll project. WebStorm from IntelliJ offers a lot of project-specific features, such as find and replace, that make it ideal for working with tech comm projects.

About text editors and WebStorm

There are a variety of text editors available, but I like WebStorm the best because

COMMAND	SHORTCUTS
Shift + Command + R	Replace in whoii5troject. (Again, WebStorm calls it "Replace in path.")
Command + F	Find on page
Shift + R	Replace on page
Right-click > Add to Favorites	Allows you to add files to a Favorites section, which expands Ti2 T8bf list ofd files in8bf 5trojecn pne.o
Shift +tab-	

[redacted]

Shift +Funectiot +F6-

RepDeleteft +tab-

Conditional logic

> Summary: You can implement advanced conditional logic that includes if statements, or statements, unless, and more. This conditional logic facilitates single sourcing scenarios in which you're outputting the same content for different audiences.

About Liquid and conditional statements

If you want to create different outputs for different audiences, you can do all of this using a combination of Jekyll's Liquid markup and values in your configuration file.

You can then incorporate conditional statements that check the values in the configuration files.

Tip: Definitely check out Liquid's documentation (http://docs.shopify.com/themes/liquid-documentation/basics) for more details about how to use op1rators and other liquid markup. The notes here are a small, somewhat sup1rficial sample from the site.

Where to store filtering values

You can filter content based on values that you have set either in your config file or in a file in your _data folder. If you set the attribute in your config file, you need to restart the Jekyll server to see the changes. If you set the value in a file in your

Storing conditions in the _data folder

Here's an example of using conditional logic based on a value in a data file:

```
X% FC PFQB.A>Q>.LMQFLKP.LRQMRQ == ">IME>" %Z
PELT QEFP @LKQBKQ...
X% BIPFC PFQB.A>Q>.LMQFLKP.LRQMRQ == "?BQ>" %Z
PELT QEFP @LKQBKQ...
X% BIPB %Z
QEFP PELTP FC KBFQEBO LC QEB >?LSB QTL FC @LKAFQFLKP >OB JBQ.
X% BKAFC %Z
```

You can see that the `BUQBOK>I<ROI` is a condition that applies a different formatting. Although this feature is available, I recommend putting any external navigation links in the top navigation bar instead of the side navigation bar.

Sidebar item highlighting

The sidebar.html file inserts an `>@QFSB` class into the sidebar element when the `ROI` attribute in the sidebar data file matches the page URL.

Note: In the _includes/mydoc folder, there's a taglogic.html file. This file (included in each tag archive file) has common logic for getting the tags and listing out the pages containing the tag in a table with summaries or truncated excerpts. You don't have to do anything with the file — just leave it there because the tag archive

```
#BQQ9×ÁĘOR×A MTA:<RI>7C̀LOMTAGÀFR×.MTA%Z7C̀LO4T@GÀMTAƃD%Z7C̀0G04T@===="D9×ÁĘOR×A"%Z<IF><P3T
```

%Z
X%

%Z
X%

%Z
X%

```
'BQQFKD PQ>OQBA M>DBP:
<RI>
X% >PPFDK PLOQBA<M>DBP = (PFQB.M>DBP Y PLOQ: 'QFQIB') %Z
X% CLO M>DB FK PLOQBA<M>DBP %Z
X% CLO Q>D FK M>DB.Q>DP %Z
X% FC                                                  %Z
```

M>DB2P

With posts (instead of pages), since you can access just the posts inside
`MLPQP.Q>D.Q>DK>JB`

> Summary: You can automatically link together topics belonging to

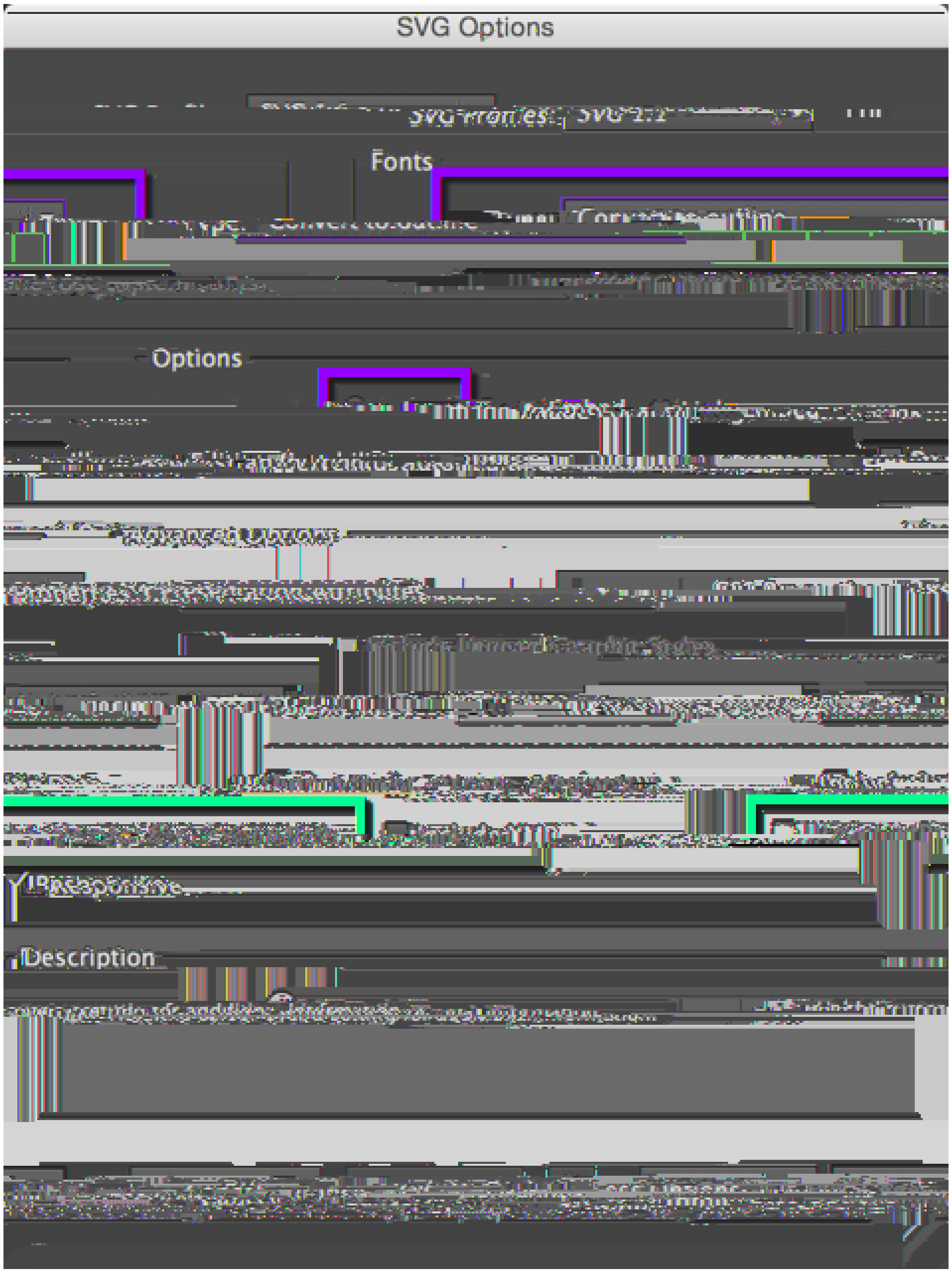
dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

80051787

callout_default: Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard

With Font Awesome, you always use the `F` tag with the appropriate class. You also implement `C>` as a base class first. You can use font awesome icons inside

You can use any of the following:



SVG Options

SVG Profiles: SVG 1.1

Fonts

Type: Convert to outline

Options

Advanced Options

Responsive

Description

) & ! #$ # ' ! "& &#" * ($& $% % $ '

Labels

> Summary: Labels are just a simple Bootstrap component that you can include in your pages as needed. They represent one of many Bootstrap options you can include in your theme.

About labels

OR

```
<> EOBC="P>JMIB.EQJI">2>JMIB</>>
```

However, what happens when you change the page's title or link? Jekyll doesn't automatically pull in the page's title when you create links.

In my experience, coding links like this results in a lot of broken links.

Managed links

For internal links, I've found that it's a best practice to store the link in a YAML file that is derived from the table of contents.

The theme has a file called urls_mydoc.txt. This e pagg links l3n_6Aull in t1mydoc.txt. The a1mydo5000 49

Navtabs

> Summary: Navtabs provide a tab-based navagation directly in your content, allowing users to click from tab to tab to see different panels of content. Navtabs are especially helpful for showing code samples for different programming languages. The only downside to using navtabs is that you must use HTML instead of Markdown.

Common uses

Navtabs are particularly useful for scenarios where you want to show a variety of options, such as code samples for Java, .NET, or PHP, on the same page.

While you could resort to single-source publishing to provide different outputs for each unique programming language or role, you could also use navtabs to allow

Video embeds

> Summary: You can embed files with a Video JS wrapper by adding 'video: true' in the frontmatter. Alternatively, you can just fall back on the default video wrapper in the browser.

e has the video.js (http://www.videojs.com/) player integrated. But the ly appear on a page or post if you have certain frontmatter in that page.

About Video JS

play by ade

If you want the player button in the upper-left corner (which is the default), remove the `SGP-?FD-MI>V-@BKQBOBA` from the video class.

Here are more details on this video player from Video JS (https://github.com/videojs/video.js/blob/stable/docs/guides/setup.md).

Note that if some of the js doesn't load correctly, the default fallback player is the regular HTML5 video player available via the browser. Here's an example of the built-in browser video wrapper:

Your browser does not support the video tag.

However, I don't think the built-in browser video players work very well (you can't easily scrud pladd)

Commenting on files

> Summary: You can add a button to your pages that allows people to

About the review process

If you're using the doc ascCode approach, you might alsocConsider using the same techniques for reviewing the doc ascmng on use increviewing Code. Ths capproach

is c.mng on filmakecCommens. incanoeasier interface.

Basics

Name: Killall Jekyll

Themes

> Summary: You can choose between two different themes (one green, the other blue) for your projects. The theme CSS is stored in the CSS folder and configured in the configuration file for each project.

Theme options

You can choose a green or blue theme, or you can create your own. In the css folder, there are two theme files: theme-blue.css and theme-green.css. These files have the most common CSS elements extracted in their own CSS file. Just change the hex colors to the ones you want.

In the configuration file, specify the theme file you want the output to use — for

Note that in order for the title-checker file to run correctly, it has to detect a match between the URL listed in the sidebar or top navigation with the URL for the page (based on the file name). If you have the wrong URL, it won't tell you if the page titles match. Therefore you should always click through all the topics in your

Check page title consistency

> Summary: title checker...

To make sure your page titles match the sidebar titles, there's a file called "title-checker.html." After your site builds, view this file in your browser. It wil7 tel7 if you

Generating PDFs

> Summary: You can generate a PDF from your Jekyll project. You do this by creating a web version of your project that is printer friendly. You then use utility called Prince to iterate through the pages and create a PDF from them. It works quite well and gives you complete control to customize the PDF output through CSS, including page directives and dynamic tags from Prince.

PDF overview

This process for creating a PDF relies on Prince XML to transform the HTML content into PDF. Prince costs about $500 per license. That might seem like a lot, but if you're creating a PDF, you're probably working for a company that sells a product, so you likely have access to some resources.

The basic approach is to generate a list of all pages that need to be added to the PDF, and then add leverage Prince to package them up into a PDF.

2. Create a new configuration file for each of your PDF targets

The PDF configuration file will build on the settings in the regular configuration file

```
?LAV.COLKQJ>QQBO X M>DB: COLKQJ>QQBO Z
?LAV.COLKQJ>QQBO X@LRKQBO-OBPBQ: M>DB 1Z


 M>DB COLKQJ>QQBO X
     QLM-IBCQ X
        @LKQBKQ: MOFK@B-P@OFMQ(DRFAB.>JB);
    Z
     QLM-OFDEQ X
        @LKQBKQ: MOFK@B-P@OFMQ(A>QBPQ>JM);
    Z
     ?LQQLJ-OFDEQ X
        @LKQBKQ); );
```

(

OLM-IBCO

?LAV.CFOPQ<M>DB X@LRKQBO-OBPBQ:

JavaScript conflicts

```
 JBAF> MOFKQX*,:>CQBO,:4%ÃÄOO/$ÂàÀ3%U/å#Sà
ALT:KLKB!FJMLOQ>KQ;/*?>@HDOLR !:O O!FJMLOQ>KQ*/;-TB4¢/Âå#Sà
ALT:KLKB!FJMLOQ>KQ;?LU-PE>ALT:KLKB!FJMLOQ>KQZ
```

If you update Bootrap, make sure you make this edit. (Sorry, admittedly I couldn't figure out how to simply overwrite the `*` selector with a later style.)

I did, however, remove the color from the alerts and lighten the background shading for `MOB` elements. The printstyles.css has this setting.

In your exclude list for your alpha project, specify it as follows:

```
BU@IRAB:
- ?BQ><*
```

If you have more sophisticated exclusion, add another level to your file names. For example, if you have different programming languages you want to filter by, add this:

- alpha_java_sample.html
- alpha_cpp_sample.html

Then you exclude files for your Alpha C++ project as follows:

```
BU@IRAB:

- >IME><G>S><*
- ?BQ><*
```

And you exclude files for your Alpha Java project as follows:

```
BU@IRAB:

- >IME><@MM<*
- >IME><?BQ><*
```

When you exclude folders, include the trailing slash at the end of the folder name:

```
BU@IRAB:
- FJ>DBP/>IME>/
```

There isn't a way to automatically exclude anything. By default, everything is included unless you explicitly list it under the exclude section.

Excluding draft content

If you're working on a draft, put it inside the _drafts folder or add `MR?IFPEBA: C>IPB` in the frontmatter. The _drafts folder is excluded by default, so you don't have to specify it in your exclude list.

Limitations

What if a file should appear in two projects but not the third? This can get tricky. For some files, rather than using a wildcard, you may need to manually specify the entire filename that you're excluding instead of excluding it by way of a wildcard pattern.

Help APIs and UI tooltips

> Summary: You can loop through files and generate a JSON file that developers can consume like a help API. Developers can pull in values from the JSON into interface elements, styling them as popovers for user interface text, for example. The beauty of this method is that the UI text remains in the help system (or at least in a single JSON file delivered to the dev team) and isn't hard-coded into the UI.

The product attribute is required in the frontmatter to distinguish the tooltips produced here from the tooltips for other products in the same _tooltips folder. When creating the JSON file, Jekyll will iterate through all the pages inside _tooltips, regardless oor ti637ubolder. inscludd here

```
X
  "BKQOFBP": 8
    X
      "FA": "?>PB?>II",
      "?LAV": "">PB?>II FP @LKPFABOBA !JBOF@>'P M>PQQFJB PMLO
Q, QELRDE QE>Q J>V ?B JLOB LC > EFPQLOF@>I QBOJ QE>K > @ROOBKQ
LKB. 3EBOB'P > ILQ JLOB BU@FQBJBKQ >?LRQ CLLQ?>II QE>K ?>PB?>I
```

You can store your mydoc_tooltips_source.json file anywhere you want, but to me it make sense to store it inside a tooltips folder for your specific project. This way it will automatically be excluded from other projects that are already excluding that project directory.

Note that you can create different JSON files that specialize in different content. For example, suppose you have some getting started information. You could put that into a different JSON file. Using the same structure, you might add an `FC` tag that checks whether the page has frontmatter that says `QVMB: DBQQFKD<PQ>OQBA` or something. Or you could put the content into separate collection entirely (different from tooltips).

By chunking up your JSON files, you can provide a quicker lookup, though I'm not sure how big the JSON file can be before you experience any latency with the jQuery lookup.

The `B>@E` method looks through all the JSON content to find the item whose `M>DB.FA` is equal to `?>PHBQ?>II` . It then looks for an element on the page named `#?>PHBQ?>II` and adds a `A>Q>-@LKQBKQ` attribute to that element.

Warning: Note:

won't work. I use the nam260 492.6260000ribute 1our JSON file(https://centm..

```
<IFKH OBI="PQVIBPEBBQ" EOBC==
```

<

```
<E2>1BRPB $BJL</E2>


<Q>?IB>
<QEB>A>
<QO>
<QE>2MLOQ</QE>
<QE>#LJJBKQP</QE>
</QO>
</QEB>A>
<Q?LAV>

<QO>
<QA>">PHBQ?>II</QA>
<QA>#ë 2âÄÄÄÄ$4êp%2Ä                                   </QA>
</QO>

<QO>
<QA>">PB?>II</QA>
<QA>#ë 2âÄÄÄÄ$4êp#Ä                                   </QA>
</QO>

<QO>
<QA>&LLQ?>II</QA>
<QA>#ë 2âÄÄÄÄ$4êpÄ2Ä                                  </QA>
</QO>

<QO>
<QA>2L@@BO</QA>
<QA>#ë 2âÄÄÄÄ$4êp@@BOZZ                         </QA>
</QO>
</Q?LAV>
</Q>?IB>
```

And here's the code:

Reuse Demo

SPORT	COMMENTS
Basketball	Basketball is a sport involving two teams of five players each competing to put a ball through a small circular rim 10 feet above the ground. Basketball requires players to be in top physical condition requires plaular

Search configuration

> Summary: The search feature uses JavaScript to look for keyword matches in a JSON file. The results show instant matches, but it doesn't provide a search results page like Google. Also, sometimes invalid formatting can break the JSON file.

iTerm profiles

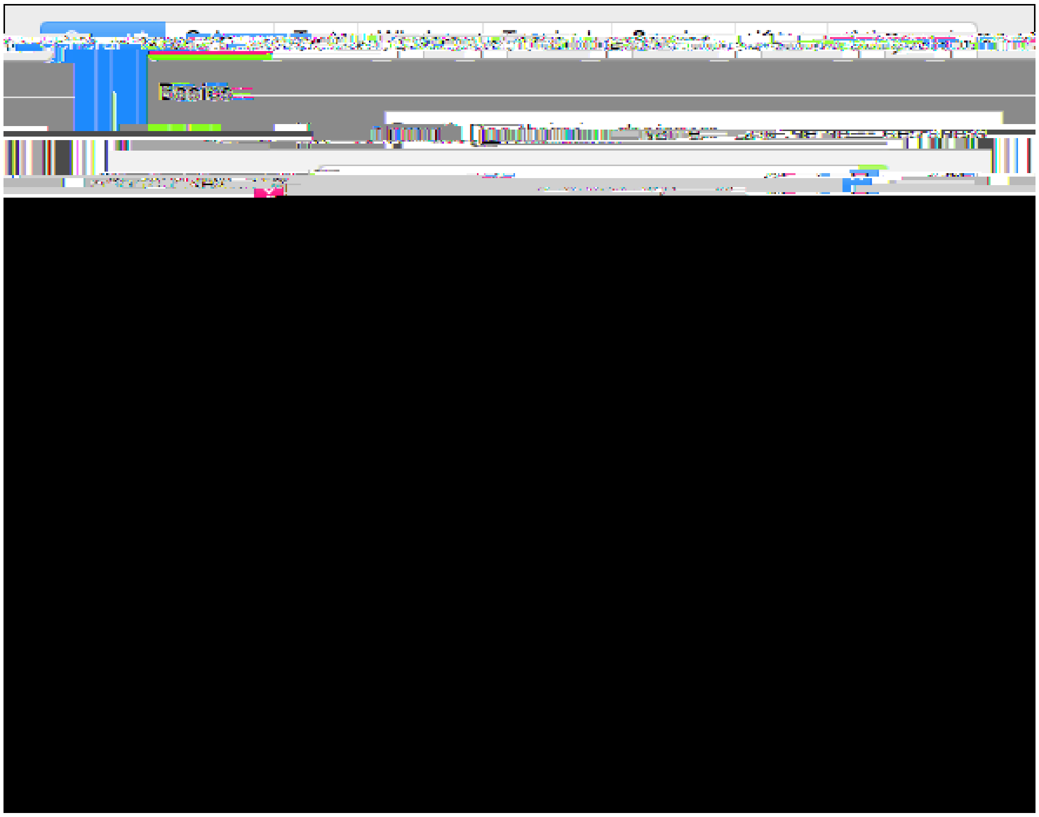
> Summary: Set up profiles in iTerm to facilitate the build process with just a few clicks. This can make it a lot easier to quickly build multiple outputs.

About iTerm profiles

When you're working with tech docs, a lot of times you're single sourcing multiple outputs. It can be a hascle to fire up each one of thece outputs using the build files containing the shell scripts. Instead, it's easier to configure iTerm with profiles that initiate the scripts.

Set up profiles

1. Open iTerm and go to



Launching a profile

1. In iTerm, make sure the Toolbar is shown. Go to View > Toggle Toolbar.
2. Click the New button and select your profile.

> Tip: When you're done with the session, make sure to click **Ctrl+C**.

Getting around the password prompts in SCP

You can publish your docs via SSH through a Terminal window or more likely, via a shell script that you simply execute as part of the publishing process. However, you will be prompted for your password with each file transfer unless you configure passwordless SSH.

The basic process for setting up password less SSH is to create a key on your own machine that you also transfer to the remote machine. When you use the SCP command, the remote machine checks that you have the authorized key and

Knowledge-base layout

> Summary: This shows a sample layout for a knowledge base. Each square could link to a tag archive page. In this example, font icons from Font Awesome are enlarged to a to a esize. You can also add captions below each icon.

Getting Started

Navigation

single_sourcing

Publishing

Special layouts

- Support (page 0)
- Supported features (page 3)
- Troubleshooting (page 139)
- WebStorm Text Editor

Scroll layout

> Summary: This page demonstrates how you the integration of a script called ScrollTo, which is used here to link definitions of a JSON

```
X
  ">MMIBP" (M>DB 157): "OBA CORFQ >Q QEB PQLOB",
  "?>K>K>8Y,WG=(M>DB 157): "VBIILT ?>K>K>32å$#P0Đ
  "@>OOLQ8Y,g7}(M>DB 157): "LO>KDB SBDBQ>?IBP QE>Q DOLT FK QE
B DOLRKA",
:: "> -BAFQBOD#,ç8$ <QLM(E/6QLT F37Q3Å#1H,5E3<O!a"> QÔ3Ä3#S$3$å$Ä2ä
LJMRQBO",
  "BDDP" (M>DB 157): "@EF@HBK3Å3T$Pá3ÔÔ3#å43T

  "C>I>CBI8Y,X6=(M>DB 158): -BAFQBOO>KB>K P>KATF@E @LKFPQFK
D LC ILQ3Ä33d4$$9LQ3Ä33d4$$9LIç30$P4$0Å3434$ž7ãõD2ia
```

: "> -BAFQBOO>KB>ã?RKA9Ô1Ô9ëA<ORFOC(E"WG="?>K>K>8Y,$26bDÔ!ã

```
B >IME>?BQ, TEF@E MOLSBA QL ?B > ERDB PQBM CLOT>OA CLO ERJ>
K QELRDEQ >KA BUMOBPPFLK.",
  "?BQ>" (M>DB 160)
```

falafel

Praesent auctor a enim non lacinia. Integer sodales aliquet mi vel dapibus.

paranoia

Aenean hendrerit mauris ipsum, non laoreet ipsum luctus vel. Curabitur tristique auctor elit ut pulvinar. Quisque arcu arcu, condimentum aliquam sodales nec, dignissim nec justo. Nunc tristique sem felis, pharetra euismod lorem volutpat sed. Ut porttitor metus sit amet elit rhoncus semper.

qui

Quisque rhoncus cursus felis vel elementum. Vestibulum dignissim molestie

Note:

Shuffle layout

> Summary: This layout shows an example of a knowledge-base style navigation system, where there is no hierarchy, just groups of pages that have certain tags.

Note:

- Pages (page 44)
- Sidebar Navigation (page 62)
- Support (page 0)
- Supported features (page 3)
- Troubleshooting (page 169)
- WebStorm Text Editor (page 50)

These topics get into formatting syntax, such as images and tables, that you'll use on each of your pages:

- Tooltips (page 74)
- Alerts (page 75)
- Glossary layout (page 166)10TjETBT0 g357.5750 468.6162 Td(•) TjETBT0.0000 0
- Links (page 89)
- Icons (page 79)
- Images (page 85)
- Labels (page 88)
- Navtabs (page 94)
- Pages (page 44)
- Syntax highlighting (page 105)
- Tables (page 101)
- Video embeds (page 97)

Single Sourcing

These topics cover strategies for single_sourcing. Single sourcing refers to strategies for re-using the same source in different outputs for different audiences or purposes.

- Conditional logic (page 53)
- 4. Set the configuration options (page 17)
- Content reuse (page 58)

Publishing

When you're building, publishing, and deploying your Jekyll site, you might fi02 15000 Tf(r,) TjETBT329.2500 226.3702 Tdhelp

If you squish your screen small enough, at a certain breakpoint this style reverts to the regular `AI` class.

Although I like the side-by-side view for shorter definitions, I found it problematic with longer definitions.

```
GBHVII PBOSB --@LKCFD @LKCFD<?>PB.VJI,@LKCFD<ABPFDKBO.VJI
```

shell file not executable

If you run into permissions errors trying to run a shell script file (such as mydoc_multibuild_web.sh), you may need to change the file permissions to make the sh file executable. Browse to the directory containing the shell script and run the following:

```
@EJLA +U ?RFIA<TOFQBO.PE
```

Pygments not installed

The config file requires pygments for the highlighter. You must download and install Pygments (http://pygments.org/download/), which requires Python, in order to use this syntax highlighter. If you don't want to bother with Pygments, open the configuration file and change

Look in _includes/custom/conditions.html and make sure the conditional values there match up with the values declared in the configuration file. Specifically, you need to make sure you've declared a value for project, product, platform, and version.

If you don't have any values for these properties, you still need to keep them in your configuration file. Just put something like `>II` as the value.

> Note: This theme is designed for single sourcing. If you're only building one site, you can remove these values from the _includes/sidebar.html file