OFFENSIVE SECURITY

Penetration Test Report for Internal Lab

marry.poppins@pltravers.writer

OSID: 271828

Table of Contents

1.1. MkDocs 9

1.2. Getting Started with MkDocs 10

1.2.1. Installation 10

1.2.2. Creating a new project 10

1.2.3. Adding pages 13

1.2.4. Theming our documentation 14

1.2.5. Changing the Favicon Icon 15

1.2.6. Building the site 15

1.2.7. Other Commands and Options 16

1.2.8. Deploying 16

1.2.9. Getting help 16

2. Bloc de Code 17

2.3. Les blocs de Code MkDocs 18

2.3.1. Bloc de Code SuperFences MkDocs 18

2.3.2. Bloc de Code HTML MkDocs 20

2.4. Les blocs de Code HTML "OSCP" 21

2.4.1. Mise en évidence 21

2.4.2. Passage à la ligne 21

3. User Guide 23

3.5. User Guide 24

3.6. MkDocs Installation 25

Table of Contents

3.6.1. Requirements 25

3.6.2. Installing MkDocs 26

3.7. Writing your docs 28

3.7.1. File layout 28

3.7.2. Writing with Markdown 31

3.8. Choosing your Theme 38

3.8.1. mkdocs 38

3.8.2. readthedocs 40

3.8.3. Third Party Themes 43

3.9. Customizing Your Theme 44

3.9.1. Using the docs_dir 44

3.9.2. Using the theme custom_dir 45

3.10. Localizing Your Theme 49

3.10.1. Installation 49

3.10.2. Supported locales 49

3.10.3. Usage 49

3.10.4. Contributing theme translations 50

3.11. Configuration 51

3.11.1. Introduction 51

3.11.2. Project information 51

3.11.3. Documentation layout 54

Table of Contents

3.11.4. Build directories 55

3.11.5. Preview controls 58

3.11.6. Live Reloading 58

3.11.7. Formatting options 59

3.11.8. Environment Variables 64

3.11.9. Configuration Inheritance 65

3.12. Deploying your docs 68

3.12.1. GitHub Pages 68

3.12.2. Read the Docs 69

3.12.3. Other Providers 70

3.12.4. Local Files 70

3.12.5. 404 Pages 71

4. Developer Guide 73

4.13. Developer Guide 74

4.14. Developing Themes 75

4.14.1. Creating a custom theme 75

4.14.2. Basic theme 76

4.14.3. Theme Files 77

4.14.4. Template Variables 78

4.14.5. Template Filters 85

4.14.6. Search and themes 86

Table of Contents

4.14.7. Packaging Themes 88

4.14.8. Supporting theme Localization/Translation 92

4.15. Translations 97

4.15.1. Localization tooling prerequisites 97

4.15.2. Adding language translations to themes 97

4.15.3. Updating a theme translation 99

4.15.4. Updating theme documentation 100

4.15.5. Contributing translations 101

4.16. MkDocs Plugins 102

4.16.1. Installing Plugins 102

4.16.2. Using Plugins 102

4.16.3. Developing Plugins 103

5. About 112

5.17. Release Notes 113

5.17.1. Upgrading 113

5.17.2. Maintenance team 113

5.17.3. Version 1.2.3 (2021-10-12) 113

5.17.4. Version 1.2.2 (2021-07-18) 114

5.17.5. Version 1.2.1 (2021-06-09) 114

5.17.6. Version 1.2 (2021-06-04) 115

5.17.7. Version 1.1.2 (2020-05-14) 119

Table of Contents

5.17.8. Version 1.1.1 (2020-05-12)	119
5.17.9. Version 1.1 (2020-02-22)	119
5.17.10. Version 1.0.4 (2018-09-07)	121
5.17.11. Version 1.0.3 (2018-08-29)	121
5.17.12. Version 1.0.2 (2018-08-22)	121
5.17.13. Version 1.0.1 (2018-08-13)	121
5.17.14. Version 1.0 (2018-08-03)	122
5.17.15. Version 0.17.5 (2018-07-06)	127
5.17.16. Version 0.17.4 (2018-06-08)	127
5.17.17. Version 0.17.3 (2018-03-07)	127
5.17.18. Version 0.17.2 (2017-11-15)	127
5.17.19. Version 0.17.1 (2017-10-30)	127
5.17.20. Version 0.17.0 (2017-10-19)	128
5.17.21. Version 0.16.3 (2017-04-04)	131
5.17.22. Version 0.16.2 (2017-03-13)	131
5.17.23. Version 0.16.1 (2016-12-22)	131
5.17.24. Version 0.16 (2016-11-04)	132
5.17.25. Version 0.15.3 (2016-02-18)	136
5.17.26. Version 0.15.2 (2016-02-08)	136
5.17.27. Version 0.15.1 (2016-01-30)	136
5.17.28. Version 0.15.0 (2016-01-21)	137

Table of Contents

5.17.29. Version 0.14.0 (2015-06-09) 137

5.17.30. Version 0.13.3 (2015-06-02) 138

5.17.31. Version 0.13.2 (2015-05-30) 138

5.17.32. Version 0.13.1 (2015-05-27) 138

5.17.33. Version 0.13.0 (2015-05-26) 138

5.17.34. Version 0.12.2 (2015-04-22) 140

5.17.35. Version 0.12.1 (2015-04-14) 140

5.17.36. Version 0.12.0 (2015-04-14) 140

5.17.37. Version 0.11.1 (2014-11-20) 142

5.17.38. Version 0.11.0 (2014-11-18) 142

5.17.39. Version 0.10.0 (2014-10-29) 142

5.18. Contributing to MkDocs **144**

5.18.1. Code of Conduct 144

5.18.2. Reporting an Issue 144

5.18.3. Testing the Development Version 144

5.18.4. Installing for Development 144

5.18.5. Running the tests 145

5.18.6. Translating themes 145

5.18.7. Submitting Pull Requests 145

5.19. License **147**

5.19.1. Included projects 147

Table of Contents

5.19.2. MkDocs License (BSD) 147

1.1. MkDocs

Project documentation with Markdown.

MkDocs is a **fast**, **simple** and **downright gorgeous** static site generator that's geared towards building project documentation. Documentation source files are written in Markdown, and configured with a single YAML configuration file. Start by reading the introductory tutorial, then check the User Guide for more information.
Getting Started User Guide

1.1.1. Features

1.1.1.1. Great themes available

There's a stack of good looking themes available for MkDocs. Choose between the built in themes: mkdocs and readthedocs, select one of the third-party themes listed on the MkDocs Themes wiki page, or build your own.

1.1.1.2. Easy to customize

Get your project documentation looking just the way you want it by customizing your theme and/or installing some plugins. Modify Markdown's behavior with Markdown extensions. Many configuration options are available.

1.1.1.3. Preview your site as you work

The built-in dev-server allows you to preview your documentation as you're writing it. It will even auto-reload and refresh your browser whenever you save your changes.

1.1.1.4. Host anywhere

MkDocs builds completely static HTML sites that you can host on GitHub pages, Amazon S3, or anywhere else you choose.

1.2. Getting Started with MkDocs

An introductory tutorial!

1.2.1. Installation

To install MkDocs, run the following command from the command line:

```
pip install mkdocs
```

For more details, see the Installation Guide.

1.2.2. Creating a new project

Getting started is super easy. To create a new project, run the following command from the command line:

```
mkdocs new my-project
cd my-project
```

Take a moment to review the initial project that has been created for you.

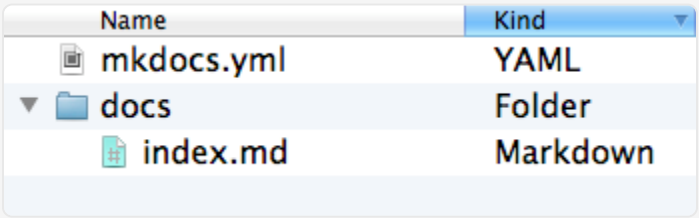

Figure 1 - The initial MkDocs layout

There's a single configuration file named `mkdocs.yml`, and a folder named `docs` that will contain your documentation source files (`docs` is the default value for the docs_dir configuration setting). Right now the `docs` folder just contains a single documentation page, named `index.md`.

MkDocs comes with a built-in dev-server that lets you preview your documentation as you work on it. Make sure you're in the same directory as the `mkdocs.yml` configuration file, and then start the server by running the `mkdocs serve` command:

```
$ mkdocs serve
INFO    -  Building documentation...
INFO    -  Cleaning site directory
[I 160402 15:50:43 server:271] Serving on http://127.0.0.1:8000
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
```

Listing 1 - Code with lines numbers and lines highlighted

Open up `http://127.0.0.1:8000/` in your browser, and you'll see the default home page being displayed:

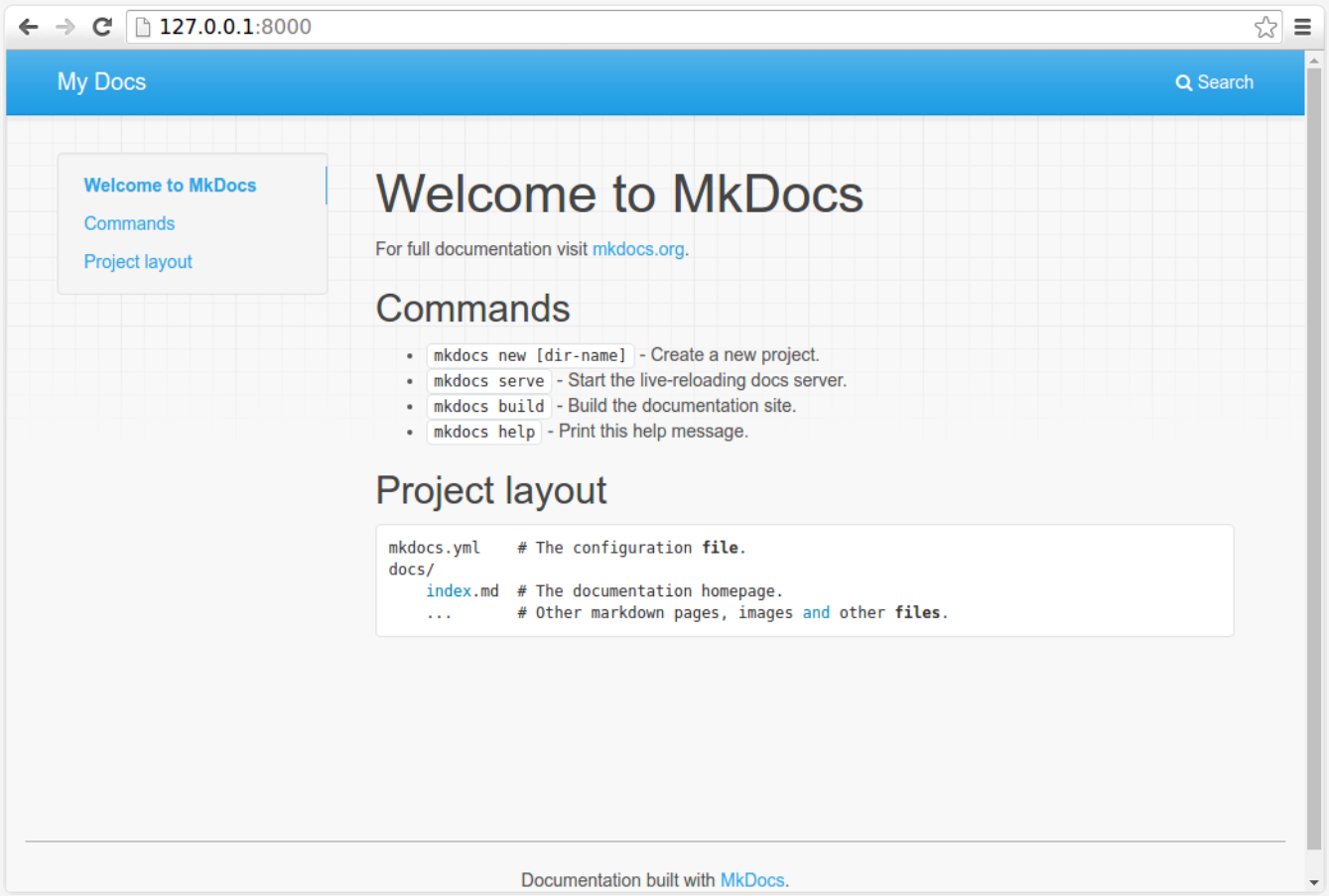

Figure 2 - The MkDocs live server

The `dev-server` also supports auto-reloading, and will rebuild your documentation whenever anything in the configuration file, documentation directory, or theme directory changes.

```
$ mkdocs serve
INFO    -  Building documentation...
INFO    -  Cleaning site directory
[I 160402 15:50:43 server:271] Serving on http://127.0.0.1:8000
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes <halt>
[I 160402 15:50:43 handlers:58] Start watching changes >
[I 160402 15:50:43 handlers:60] Start detecting changes >
```

Listing 2 - Direct HTML `pre code mark` with classes to highlight a part of code

Open the `docs/index.md` document in your text editor of choice, change the initial heading to `MkLorum`, and save your changes. Your browser will auto-reload and you should see your updated documentation immediately.

Now try editing the configuration file: `mkdocs.yml`. Change the `site_name` setting to `MkLorum` and save the file.

```
site_name: MkLorum
site_url: https://example.com/
```

Your browser should immediately reload, and you'll see your new site name take effect.

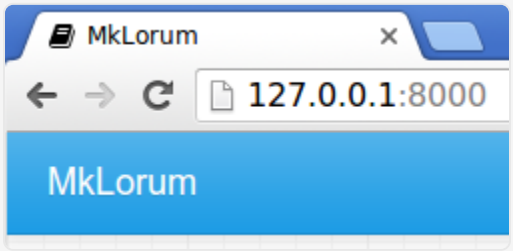

Figure 3 - The `site_name` setting

Note

The `site_name` and `site_url` configuration options are the only two required options in your configuration file. When you create a new project, the `site_url` option is assigned the placeholder value: `https://example.com`. If the final location is known, you can change the setting now to point to it. Or you may choose to leave it alone for now. Just be sure to edit it before you deploy your site to a production server.

1.2.3. Adding pages

Now add a second page to your documentation:

```
curl 'https://jaspervdj.be/lorem-markdownum/markdown.txt' > docs/about.md
```

As our documentation site will include some navigation headers, you may want to edit the configuration file and add some information about the order, title, and nesting of each page in the navigation header by adding a `nav` setting:

```
site_name: MkLorum
site_url: https://example.com/
nav:
    - Home: index.md
    - About: about.md
```

Save your changes and you'll now see a navigation bar with `Home` and `About` items on the left as well as `Search`, `Previous`, and `Next` items on the right.

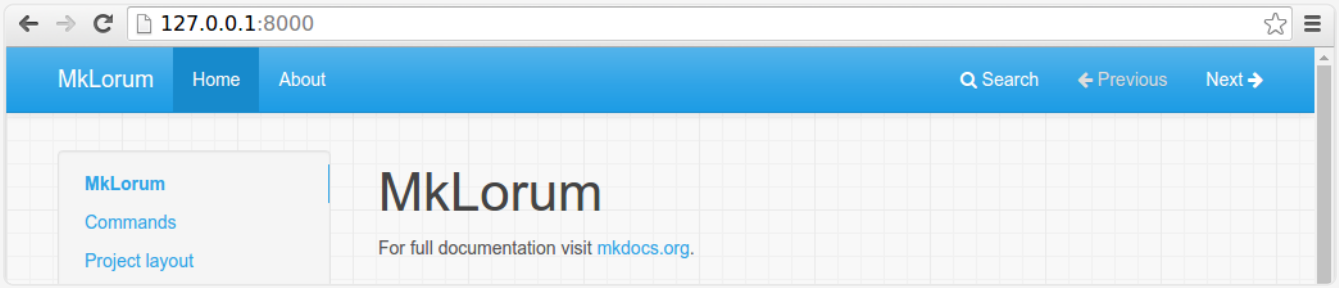

Figure 4 - Screenshot

Try the menu items and navigate back and forth between pages. Then click on `Search`. A search dialog will appear, allowing you to search for any text on any page. Notice that the search results include every occurrence of the search term on the site and links directly to the section of the page in which the search term appears. You get all of that with no effort or configuration on your part!

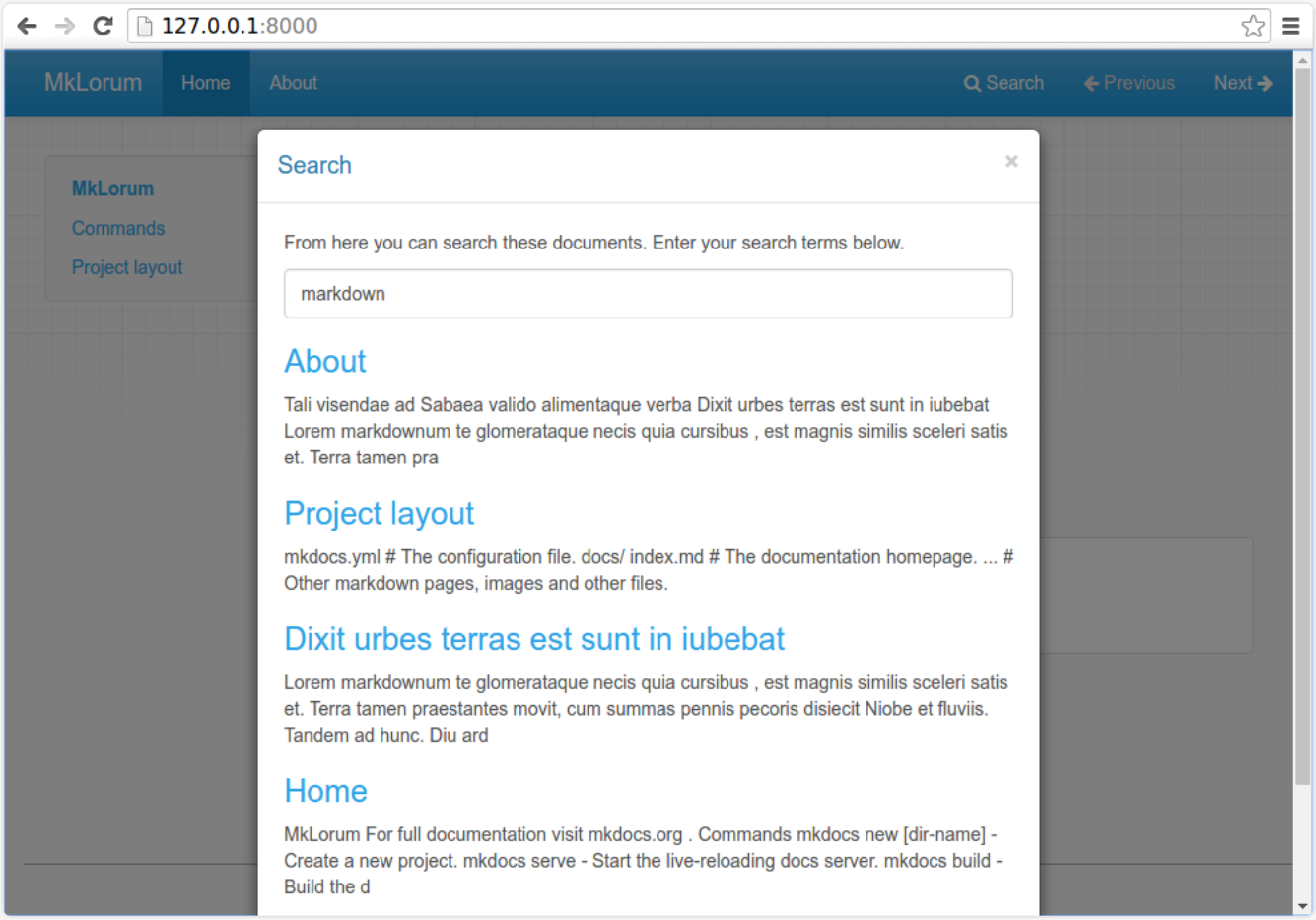

Figure 5 - Screenshot

1.2.4. Theming our documentation

Now change the configuration file to alter how the documentation is displayed by changing the theme. Edit the `mkdocs.yml` file and add a `theme` setting:

```
site_name: MkLorum
site_url: https://example.com/
nav:
    - Home: index.md
    - About: about.md
theme: readthedocs
```

Save your changes, and you'll see the ReadTheDocs theme being used.

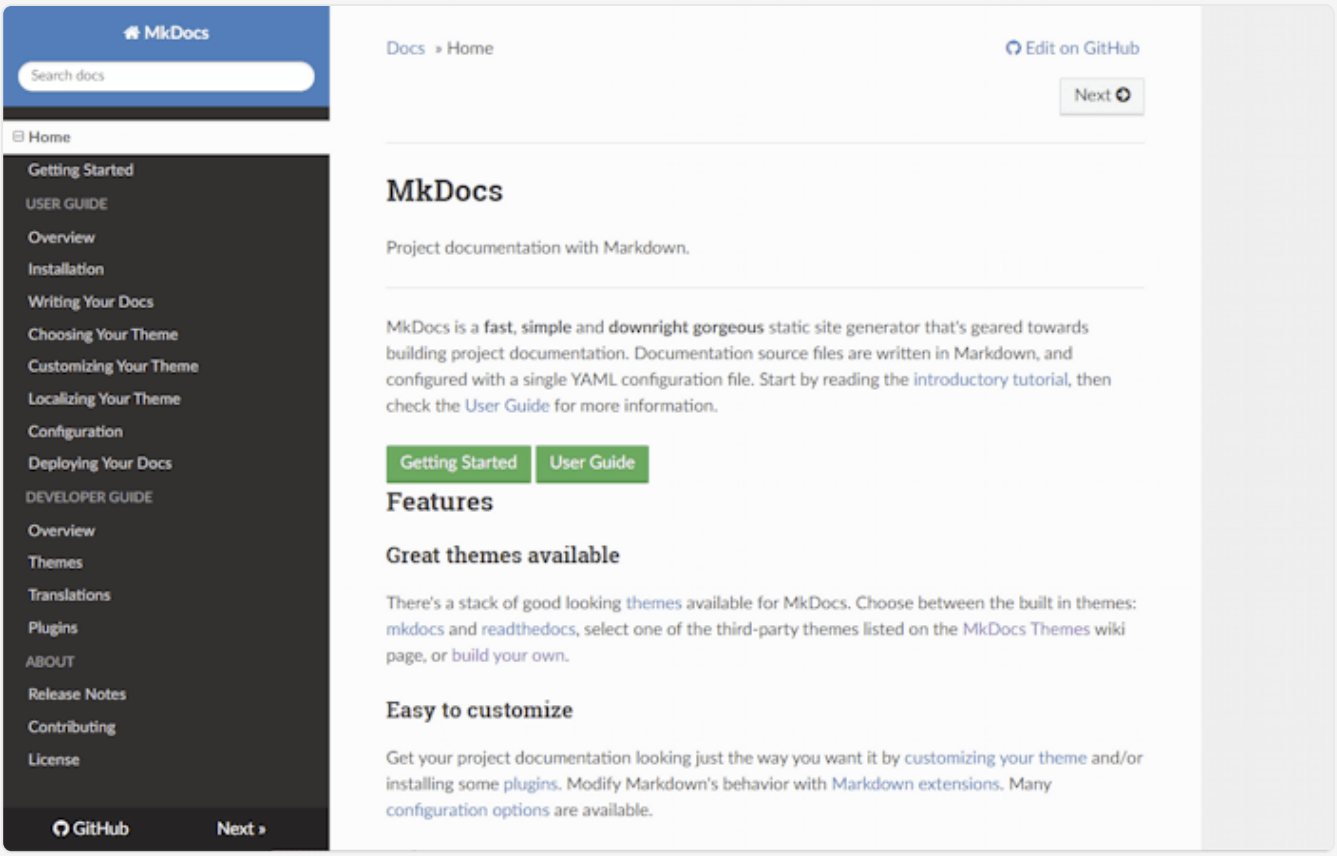

Figure 6 - Screenshot

1.2.5. Changing the Favicon Icon

By default, MkDocs uses the MkDocs favicon icon. To use a different icon, create an `img` subdirectory in the `docs` directory and copy your custom `favicon.ico` file to that directory. MkDocs will automatically detect and use that file as your favicon icon.

1.2.6. Building the site

That's looking good. You're ready to deploy the first pass of your `MkLorum` documentation. First build the documentation:

```
mkdocs build
```

This will create a new directory, named `site`. Take a look inside the directory:

```
$ ls site
about  fonts  index.html  license  search.html
css    img    js          mkdocs   sitemap.xml
```

Notice that your source documentation has been output as two HTML files named `index.html` and `about/index.html`. You also have various other media that's been copied into the `site` directory as part of the documentation theme. You even have a `sitemap.xml` file and `mkdocs/search_index.json`.

If you're using source code control such as `git` you probably don't want to check your documentation builds into the repository. Add a line containing `site/` to your `.gitignore` file.

```
echo "site/" >> .gitignore
```

If you're using another source code control tool you'll want to check its documentation on how to ignore specific directories.

1.2.7. Other Commands and Options

There are various other commands and options available. For a complete list of commands, use the `--help` flag:

```
mkdocs --help
```

To view a list of options available on a given command, use the `--help` flag with that command. For example, to get a list of all options available for the `build` command run the following:

```
mkdocs build --help
```

1.2.8. Deploying

The documentation site that you just built only uses static files so you'll be able to host it from pretty much anywhere. Simply upload the contents of the entire `site` directory to wherever you're hosting your website from and you're done. For specific instructions on a number of common hosts, see the Deploying your Docs page.

1.2.9. Getting help

See the User Guide for more complete documentation of all of MkDocs' features.

To get help with MkDocs, please use the GitHub discussions or GitHub issues.

2. Bloc de Code

2.1. Les blocs de Code MkDocs

Il y a deux types de blocs de code disponible dans MkDocs Material :

- bloc de code `SuperFences` (extension du parser Markdown de Python)
- bloc de code `HTML`

Références :

- Material for MkDocs > Code blocks
- Python-Markdown
- SuperFences

2.1.1. Bloc de Code `SuperFences` MkDocs

Ce type de bloc de code permet de :

- coloriser syntaxiquement le code si un langage est indiqué : ```` ```python ````
- mettre en évidence des lignes ```` ```hl_lines="2 3 10-15" ```` (lignes 2, 3 et de 10 à 15)
- numéroter les lignes ```` ```linenums="5" ```` (numérotation des lignes commençant à 5)

Ces différentes options peuvent être combinées :

````
```python hl_lines="2 5-6" linenums="1"
#!/usr/bin/python
# Note: see how to craft FEALIST in eternalblue_poc.
from impacket import smb
from struct import pack
import sys
import socket
```
````

Listing 3 - Syntaxe de code combinant les différentes options

```python
#!/usr/bin/python
# Note: see how to craft FEALIST in eternalblue_poc.
from impacket import smb
from struct import pack
import sys
import socket
```

Listing 4 - Résultat de code combinant les différentes options

> **Note**
>
> Les lignes mises évidences sont en **rouge OSCP** dans le rendu PDF.

```
#!/usr/bin/python
# Note: see how to craft FEALIST in eternalblue_poc.
from impacket import smb
from struct import pack
import sys
import socket
```

2.1.1.1. La gestion des retours à la ligne

Le retour à la ligne dans les blocs de code non numérotés est automatique :

```
ntfea10000 = pack('<BBH', 0, 0, 0xffdd) + 'A'*0xffde

ntfea11000 = (pack('<BBH', 0, 0, 0) + '\x00')*600  # with these fea, ntfea size is
0x1c20
ntfea11000 += pack('<BBH', 0, 0, 0xf3bd) + 'A'*0xf3be  # 0x10fe8 - 0x1c20 - 0xc =
0xf3bc

ntfea1f000 = (pack('<BBH', 0, 0, 0) + '\x00')*0x2494  # with these fea, ntfea size
is 0x1b6f0
ntfea1f000 += pack('<BBH', 0, 0, 0x48ed) + 'A'*0x48ee  # 0x1ffe8 - 0x1b6f0 - 0xc =
0x48ec
```

Listing 5 - Bloc de code avec retour à ligne automatique

La proprité CSS correspndante est `white-space: pre-wrap;`.

Attention

La notion de retour à la ligne "doux" n'existant pas dans un document PDF, un copier/coller du code ne donnera pas le résultat attendu.

```
ntfea10000 = pack('<BBH', 0, 0, 0xffdd) + 'A'*0xffde

ntfea11000 = (pack('<BBH', 0, 0, 0) + '\x00')*600  # with these fea, ntfea size i
ntfea11000 += pack('<BBH', 0, 0, 0xf3bd) + 'A'*0xf3be  # 0x10fe8 - 0x1c20 - 0xc =

ntfea1f000 = (pack('<BBH', 0, 0, 0) + '\x00')*0x2494  # with these fea, ntfea siz
ntfea1f000 += pack('<BBH', 0, 0, 0x48ed) + 'A'*0x48ee  # 0x1ffe8 - 0x1b6f0 - 0xc
```

Listing 6 - Bloc de code numéroté donc sans retour à ligne automatique

Remarque

La fonctionnalité n'est pas disponible pour les blocs de code numérotés car elle "casse" la numérotion des lignes aussi bien en rendu HTML qu'en rendu PDF.

2.1.2. Bloc de Code `HTML` MkDocs

Ce sont des blocs de code standard `HTML`, des éléments peuvent être mis en évidence en utilisant la balise `<mark>`.

> **Note**
>
> Il est nécessaire de traduire les chevrons ouvrants en entité HTML `<` s'il peuvent être interprété comme l'ouverture d'un balise HTML (exemple en ligne 7).

```
1  <pre><code>
2  <mark>$ mkdocs serve</mark>
3  INFO    -  Building documentation...
4  INFO    -  Cleaning site directory
5  [I 160402 15:50:43 server:271] <mark>Serving on http://127.0.0.1:8000</mark>
6  [I 160402 15:50:43 handlers:58] Start watching changes
7  [I 160402 15:50:43 handlers:60] Start detecting changes &lt;halt>
8  [I 160402 15:50:43 handlers:58] Start watching changes < halt>
9  [I 160402 15:50:43 handlers:60] Start detecting changes <halt>
10 </code></pre>
```

Listing 7 - Exemple d'un bloc de code HTML

```
$ mkdocs serve
INFO    -  Building documentation...
INFO    -  Cleaning site directory
[I 160402 15:50:43 server:271] Serving on http://127.0.0.1:8000
[I 160402 15:50:43 handlers:58] Start watching changes
[I 160402 15:50:43 handlers:60] Start detecting changes <halt>
[I 160402 15:50:43 handlers:58] Start watching changes < halt>
[I 160402 15:50:43 handlers:60] Start detecting changes
```

Listing 8 - Rendu d'un bloc de code HTML

2.2. Les blocs de Code HTML "OSCP"

2.2.1. Mise en évidence

Trois types de mise en évidence "OSCP" (`2012-code-block`) sont disponbibles en utilisant des balises spécifiques, les noms des balises correspondent à l'utilisation qui semblent en être faite dans le document de référence :

```
<pre><code>
kali@kali:~$ <oscp-cmd>searchsploit afd windows</oscp-cmd>
<oscp-parameter>local</oscp-parameter>
...
<oscp-hacked>Microsoft Windows XP/2003 | windows/local/6757.txt</oscp-hacked>
...
<!-- les anciennes balise mark + class oscp sont toujours disponibles-->
</code></pre>
```

Listing 9 - Exemple de mise en évidence d'une partie du listing

```
kali@kali:~$ searchsploit afd windows local
------------------------------------------------------------ ----------------------
 Exploit Title                                              |  Path
------------------------------------------------------------ ----------------------
Microsoft Windows (x86) - 'afd.sys' Local Privilege Escalation  | windows_x86/local/4
Microsoft Windows 7 (x86) - 'afd.sys' Dangling Pointer Privileg | windows_x86/local/3
Microsoft Windows XP - 'afd.sys' Local Kernel Denial of Service | windows/dos/17133.c
Microsoft Windows XP/2003 - 'afd.sys' Local Privilege Escalatio | windows/local/6757.
Microsoft Windows XP/2003 - 'afd.sys' Local Privilege Escalatio | windows/local/1817(
------------------------------------------------------------ ----------------------
Shellcodes: No Result
kali@kali:~$
```

Listing 10 - Résultat "OSCP"

2.2.2. Passage à la ligne

Le passage à la ligne n'est pas activé par défaut, car ce type de bloc de code est principalement utilisé pour illustrer le résultat de commande ; le passage à la ligne peut ne pas être le résultat attendu mais il est possible de l'activer à l'aide de la classe `fc-pdf-white-space-pre-wrap` :

```
<pre class="fc-pdf-white-space-pre-wrap"><code>
Most field in overwritten (corrupted) srvnet struct can be any value ...
```

Listing 11 - Exemple de bloc de code avec passage à la ligne forcée

```
Most field in overwritten (corrupted) srvnet struct can be any value because it
will be left without free (memory leak) after processing
Here is the important fields on x64
- offset 0x58 (VOID*) : pointer to a struct contained pointer to function. the
pointer to function is called when done receiving SMB request.
                         The value MUST point to valid (might be fake) struct.
- offset 0x70 (MDL)   : MDL for describe receiving SMB request buffer
  - 0x70 (VOID*)    : MDL.Next should be NULL
  - 0x78 (USHORT)   : MDL.Size should be some value that not too small
  - 0x7a (USHORT)   : MDL.MdlFlags should be 0x1004 (MDL_NETWORK_HEADER|
MDL_SOURCE_IS_NONPAGED_POOL)
  - 0x80 (VOID*)    : MDL.Process should be NULL
  - 0x88 (VOID*)    : MDL.MappedSystemVa MUST be a received network buffer
address. Controlling this value get arbitrary write.
                       The address for arbitrary write MUST be subtracted by a
number of sent bytes (0x80 in this exploit).
```

Listing 12 - Résultat du passage à la ligne forcée

3. User Guide

3.1. User Guide

Building Documentation with MkDocs

The MkDocs Developer Guide provides documentation for users of MkDocs. See Getting Started for an introductory tutorial. You can jump directly to a page listed below, or use the next and previous buttons in the navigation bar at the top of the page to move through the documentation in order.

- Installation
- Writing Your Docs
- Choosing Your Theme
- Customizing Your Theme
- Configuration
- Deploying Your Docs

3.2. MkDocs Installation

A detailed guide.

3.2.1. Requirements

MkDocs requires a recent version of Python and the Python package manager, pip, to be installed on your system.

You can check if you already have these installed from the command line:

```
$ python --version
Python 3.8.2
$ pip --version
pip 20.0.2 from /usr/local/lib/python3.8/site-packages/pip (python 3.8)
```

If you already have those packages installed, you may skip down to Installing MkDocs.

3.2.1.1. Installing Python

Install Python using your package manager of choice, or by downloading an installer appropriate for your system from python.org and running it.

> **Note**
> If you are installing Python on Windows, be sure to check the box to have Python added to your PATH if the installer offers such an option (it's normally off by default).

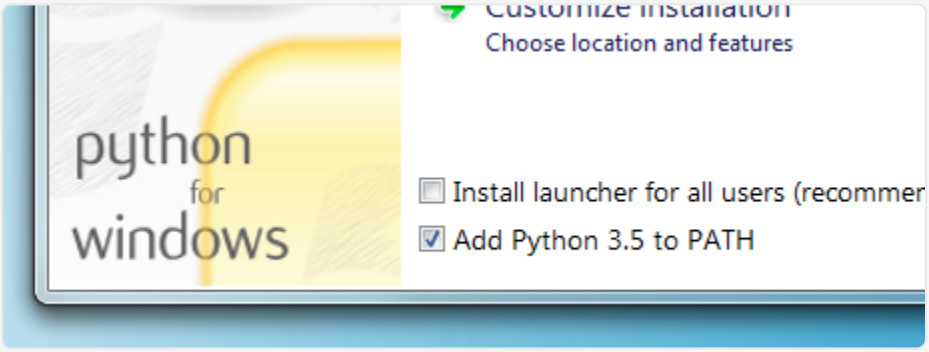

Figure 7 - Add Python to PATH

3.2.1.2. Installing pip

If you're using a recent version of Python, the Python package manager, pip, is most likely installed by default. However, you may need to upgrade pip to the lasted version:

```
pip install --upgrade pip
```

If you need to install pip for the first time, download get-pip.py. Then run the following command to install it:

```
python get-pip.py
```

3.2.2. Installing MkDocs

Install the `mkdocs` package using pip:

```
pip install mkdocs
```

You should now have the `mkdocs` command installed on your system. Run `mkdocs --version` to check that everything worked okay.

```
$ mkdocs --version
mkdocs, version 1.2.0 from /usr/local/lib/python3.8/site-packages/mkdocs (Python 3.
8)
```

Note

If you would like manpages installed for MkDocs, the click-man tool can generate and install them for you. Simply run the following two commands:

```
pip install click-man
click-man --target path/to/man/pages mkdocs
```

See the click-man documentation for an explanation of why manpages are not automatically generated and installed by pip.

Note

If you are using Windows, some of the above commands may not work out-of-the-box.

A quick solution may be to preface every Python command with `python -m` like this:

```
python -m pip install mkdocs
python -m mkdocs
```

For a more permanent solution, you may need to edit your `PATH` environment variable to include the `Scripts` directory of your Python installation. Recent versions of Python include a script to do this for you. Navigate to your Python installation directory (for example `C:\Python38\`), open the `Tools`, then `Scripts` folder, and run the `win_add2path.py` file by double clicking on it. Alternatively, you can download the script and run it (`python win_add2path.py`).

3.3. Writing your docs

How to layout and write your Markdown source files.

3.3.1. File layout

Your documentation source should be written as regular Markdown files (see Writing with Markdown below), and placed in the documentation directory. By default, this directory will be named `docs` and will exist at the top level of your project, alongside the `mkdocs.yml` configuration file.

The simplest project you can create will look something like this:

```
mkdocs.yml
docs/
    index.md
```

By convention your project homepage should be named `index.md` (see Index pages below for details). Any of the following file extensions may be used for your Markdown source files: `markdown`, `mdown`, `mkdn`, `mkd`, `md`. All Markdown files included in your documentation directory will be rendered in the built site regardless of any settings.

> **Note**
>
> Files and directories with names which begin with a dot (for example: `.foo.md` or `.bar/baz.md`) are ignored by MkDocs, which matches the behavior of most web servers. There is no option to override this behavior.

You can also create multi-page documentation, by creating several Markdown files:

```
mkdocs.yml
docs/
    index.md
    about.md
    license.md
```

The file layout you use determines the URLs that are used for the generated pages. Given the above layout, pages would be generated for the following URLs:

```
/
/about/
/license/
```

You can also include your Markdown files in nested directories if that better suits your documentation layout.

```
docs/
    index.md
    user-guide/getting-started.md
    user-guide/configuration-options.md
    license.md
```

Source files inside nested directories will cause pages to be generated with nested URLs, like so:

```
/
/user-guide/getting-started/
/user-guide/configuration-options/
/license/
```

Any files which are not identified as Markdown files (by their file extension) within the documentation directory are copied by MkDocs to the built site unaltered. See how to link to images and media below for details.

3.3.1.1. Index pages

When a directory is requested, by default, most web servers will return an index file (usually named `index.html`) contained within that directory if one exists. For that reason, the homepage in all of the examples above has been named `index.md`, which MkDocs will render to `index.html` when building the site.

Many repository hosting sites provide special treatment for README files by displaying the contents of the README file when browsing the contents of a directory. Therefore, MkDocs will allow you to name your index pages as `README.md` instead of `index.md`. In that way, when users are browsing your source code, the repository host can display the index page of that directory as it is a README file. However, when MkDocs renders your site, the file will be renamed to `index.html` so that the server will serve it as a proper index file.

If both an `index.md` file and a `README.md` file are found in the same directory, then the `index.md` file is used and the `README.md` file is ignored.

3.3.1.2. Configure Pages and Navigation

The nav configuration setting in your `mkdocs.yml` file defines which pages are included in the global site navigation menu as well as the structure of that menu. If not provided, the navigation will be automatically created by discovering all the Markdown files in the documentation directory. An automatically created navigation configuration will always be sorted alphanumerically by file

name (except that index files will always be listed first within a sub-section). You will need to manually define your navigation configuration if you would like your navigation menu sorted differently.

A minimal navigation configuration could look like this:

```
nav:
    - 'index.md'
    - 'about.md'
```

All paths in the navigation configuration must be relative to the `docs_dir` configuration option. If that option is set to the default value, `docs`, the source files for the above configuration would be located at `docs/index.md` and `docs/about.md`.

The above example will result in two navigation items being created at the top level and with their titles inferred from the contents of the Markdown file or, if no title is defined within the file, of the file name. To override the title in the `nav` setting add a title right before the filename.

```
nav:
    - Home: 'index.md'
    - About: 'about.md'
```

Note that if a title is defined for a page in the navigation, that title will be used throughout the site for that page and will override any title defined within the page itself.

Navigation sub-sections can be created by listing related pages together under a section title. For example:

```
nav:
    - Home: 'index.md'
    - 'User Guide':
        - 'Writing your docs': 'writing-your-docs.md'
        - 'Styling your docs': 'styling-your-docs.md'
    - About:
        - 'License': 'license.md'
        - 'Release Notes': 'release-notes.md'
```

Listing 13 - `nav` example

With the above configuration we have three top level items: "Home", "User Guide" and "About." "Home" is a link to the homepage for the site. Under the "User Guide" section two pages are listed: "Writing your docs" and "Styling your docs." Under the "About" section two more pages are listed: "License" and "Release Notes."

Note that a section cannot have a page assigned to it. Sections are only containers for child pages and sub-sections. You may nest sections as deeply as you like. However, be careful that you don't make it too difficult for your users to navigate through the site navigation by over-complicating the nesting. While sections may mirror your directory structure, they do not have to.

Any pages not listed in your navigation configuration will still be rendered and included with the built site, however, they will not be linked from the global navigation and will not be included in the `previous` and `next` links. Such pages will be "hidden" unless linked to directly.

3.3.2. Writing with Markdown

MkDocs pages must be authored in Markdown, a lightweight markup language which results in easy-to-read, easy-to-write plain text documents that can be converted to valid HTML documents in a predictable manner.

MkDocs uses the Python-Markdown library to render Markdown documents to HTML. Python-Markdown is almost completely compliant with the reference implementation, although there are a few very minor differences.

In addition to the base Markdown syntax which is common across all Markdown implementations, MkDocs includes support for extending the Markdown syntax with Python-Markdown extensions. See the MkDocs' markdown_extensions configuration setting for details on how to enable extensions.

MkDocs includes some extensions by default, which are highlighted below.

3.3.2.1. Internal links

MkDocs allows you to interlink your documentation by using regular Markdown links. However, there are a few additional benefits to formatting those links specifically for MkDocs as outlined below.

Linking to pages

When linking between pages in the documentation you can simply use the regular Markdown linking syntax, including the relative path to the Markdown document you wish to link to.

```
Please see the [project license](license.md) for further details.
```

When the MkDocs build runs, these Markdown links will automatically be transformed into an HTML hyperlink to the appropriate HTML page.

> **Warning**
>
> Using absolute paths with links is not officially supported. Relative paths are adjusted by MkDocs to ensure they are always relative to the page. Absolute paths are not modified at all. This means that your links using absolute paths might work fine in your local environment but they might break once you deploy them to your production server.

If the target documentation file is in another directory you'll need to make sure to include any relative directory path in the link.

```
Please see the [project license](../about/license.md) for further details.
```

The toc extension is used by MkDocs to generate an ID for every header in your Markdown documents. You can use that ID to link to a section within a target document by using an anchor link. The generated HTML will correctly transform the path portion of the link, and leave the anchor portion intact.

```
Please see the [project license](about.md#license) for further details.
```

Note that IDs are created from the text of a header. All text is converted to lowercase and any disallowed characters, including white-space, are converted to dashes. Consecutive dashes are then reduced to a single dash.

There are a few configuration settings provided by the toc extension which you can set in your `mkdocs.yml` configuration file to alter the default behavior:

- **`permalink`:**
  Generate permanent links at the end of each header. Default: `False`.

  When set to True the paragraph symbol (¶ or `¶`) is used as the link text. When set to a string, the provided string is used as the link text. For example, to use the hash symbol (`#`) instead, do:

```
markdown_extensions:
    - toc:
        permalink: "#"
```

- **`baselevel`:**
  Base level for headers. Default: `1`.

  This setting allows the header levels to be automatically adjusted to fit within the hierarchy of your HTML templates. For example, if the Markdown text for a page should not contain any headers higher than level 2 (`<h2>`), do:

```
markdown_extensions:
    - toc:
        baselevel: 2
```

Then any headers in your document would be increased by 1. For example, the header `# Header` would be rendered as a level 2 header (`<h2>`) in the HTML output.

- `separator` :
Word separator. Default: `-` .

Character which replaces white-space in generated IDs. If you prefer underscores, then do:

```
markdown_extensions:
    - toc:
        separator: "_"
```

Note that if you would like to define multiple of the above settings, you must do so under a single `toc` entry in the `markdown_extensions` configuration option.

```
markdown_extensions:
    - toc:
        permalink: "#"
        baselevel: 2
        separator: "_"
```

Linking to images and media

As well as the Markdown source files, you can also include other file types in your documentation, which will be copied across when generating your documentation site. These might include images and other media.

For example, if your project documentation needed to include a GitHub pages CNAME file and a PNG formatted screenshot image then your file layout might look as follows:

```
mkdocs.yml
docs/
    CNAME
    index.md
    about.md
    license.md
    img/
        screenshot.png
```

To include images in your documentation source files, simply use any of the regular Markdown image syntaxes:

```
Cupcake indexer is a snazzy new project for indexing small cakes.

![Screenshot](img/screenshot.png)

*Above: Cupcake indexer in progress*
```

Your image will now be embedded when you build the documentation, and should also be previewed if you're working on the documentation with a Markdown editor.

Linking from raw HTML

Markdown allows document authors to fall back to raw HTML when the Markdown syntax does not meets the author's needs. MkDocs does not limit Markdown in this regard. However, as all raw HTML is ignored by the Markdown parser, MkDocs is not able to validate or convert links contained in raw HTML. When including internal links within raw HTML, you will need to manually format the link appropriately for the rendered document.

3.3.2.2. Meta-Data

MkDocs includes support for both YAML and MultiMarkdown style meta-data (often called front-matter). Meta-data consists of a series of keywords and values defined at the beginning of a Markdown document, which are stripped from the document prior to it being processing by Python-Markdown. The key/value pairs are passed by MkDocs to the page template. Therefore, if a theme includes support, the values of any keys can be displayed on the page or used to control the page rendering. See your theme's documentation for information about which keys may be supported, if any.

In addition to displaying information in a template, MkDocs includes support for a few predefined meta-data keys which can alter the behavior of MkDocs for that specific page. The following keys are supported:

- `template`:
The template to use with the current page.

By default, MkDocs uses the `main.html` template of a theme to render Markdown pages. You can use the `template` meta-data key to define a different template file for that specific page. The template file must be available on the path(s) defined in the theme's environment.

- `title`:
The "title" to use for the document.

MkDocs will attempt to determine the title of a document in the following ways, in order:

1. A title defined in the nav configuration setting for a document.
2. A title defined in the `title` meta-data key of a document.

3. A level 1 Markdown header on the first line of the document body. Please note that Setext-style headers are not supported.
4. The filename of a document.

Upon finding a title for a page, MkDoc does not continue checking any additional sources in the above list.

YAML Style Meta-Data

YAML style meta-data consists of YAML key/value pairs wrapped in YAML style deliminators to mark the start and/or end of the meta-data. The first line of a document must be `---`. The meta-data ends at the first line containing an end deliminator (either `---` or `...`). The content between the deliminators is parsed as YAML.

```
---
title: My Document
summary: A brief description of my document.
authors:
    - Waylan Limberg
    - Tom Christie
date: 2018-07-10
some_url: https://example.com
---
This is the first paragraph of the document.
```

YAML is able to detect data types. Therefore, in the above example, the values of `title`, `summary` and `some_url` are strings, the value of `authors` is a list of strings and the value of `date` is a `datetime.date` object. Note that the YAML keys are case sensitive and MkDocs expects keys to be all lowercase. The top level of the YAML must be a collection of key/value pairs, which results in a Python `dict` being returned. If any other type is returned or the YAML parser encounters an error, then MkDocs does not recognize the section as meta-data, the page's `meta` attribute will be empty, and the section is not removed from the document.

MultiMarkdown Style Meta-Data

MultiMarkdown style meta-data uses a format first introduced by the MultiMarkdown project. The data consists of a series of keywords and values defined at the beginning of a Markdown document, like this:

```
Title:   My Document
Summary: A brief description of my document.
Authors: Waylan Limberg
         Tom Christie
Date:    January 23, 2018
blank-value:
some_url: https://example.com
```

```
This is the first paragraph of the document.
```

The keywords are case-insensitive and may consist of letters, numbers, underscores and dashes and must end with a colon. The values consist of anything following the colon on the line and may even be blank.

If a line is indented by 4 or more spaces, that line is assumed to be an additional line of the value for the previous keyword. A keyword may have as many lines as desired. All lines are joined into a single string.

The first blank line ends all meta-data for the document. Therefore, the first line of a document must not be blank.

> **Note**
>
> MkDocs does not support YAML style deliminators (`---` or `...`) for MultiMarkdown style meta-data. In fact, MkDocs relies on the the presence or absence of the deliminators to determine whether YAML style meta-data or MultiMarkdown style meta-data is being used. If the deliminators are detected, but the content between the deliminators is not valid YAML meta-data, MkDocs does not attempt to parse the content as MultiMarkdown style meta-data.

3.3.2.3. Tables

The tables extension adds a basic table syntax to Markdown which is popular across multiple implementations. The syntax is rather simple and is generally only useful for simple tabular data.

A simple table looks like this:

```
First Header | Second Header | Third Header
------------ | ------------- | ------------
Content Cell | Content Cell  | Content Cell
Content Cell | Content Cell  | Content Cell
```

If you wish, you can add a leading and tailing pipe to each line of the table:

```
| First Header | Second Header | Third Header |
| ------------ | ------------- | ------------ |
| Content Cell | Content Cell  | Content Cell |
| Content Cell | Content Cell  | Content Cell |
```

Specify alignment for each column by adding colons to separator lines:

```
First Header | Second Header | Third Header
:----------- |:-------------:| -----------:
```

```
  Left         | Center        | Right
  Left         | Center        | Right
```

Note that table cells cannot contain any block level elements and cannot contain multiple lines of text. They can, however, include inline Markdown as defined in Markdown's syntax rules.

Additionally, a table must be surrounded by blank lines. There must be a blank line before and after the table.

3.3.2.4. Fenced code blocks

The fenced code blocks extension adds an alternate method of defining code blocks without indentation.

The first line should contain 3 or more backtick (`) characters, and the last line should contain the same number of backtick characters (`):

```
```
Fenced code blocks are like Standard
Markdown's regular code blocks, except that
they're not indented and instead rely on
start and end fence lines to delimit the
code block.
```
```

With this approach, the language can optionally be specified on the first line after the backticks which informs any syntax highlighters of the language used:

```
```python
def fn():
    pass
```
```

Note that fenced code blocks can not be indented. Therefore, they cannot be nested inside list items, blockquotes, etc.

3.4. Choosing your Theme

Selecting and configuring a theme.

MkDocs includes two built-in themes (mkdocs and readthedocs), as documented below. However, many third party themes are available to choose from as well.

To choose a theme, set the theme configuration option in your `mkdocs.yml` config file.

```
theme:
    name: readthedocs
```

3.4.1. mkdocs

The default theme, which was built as a custom Bootstrap theme, supports most every feature of MkDocs.

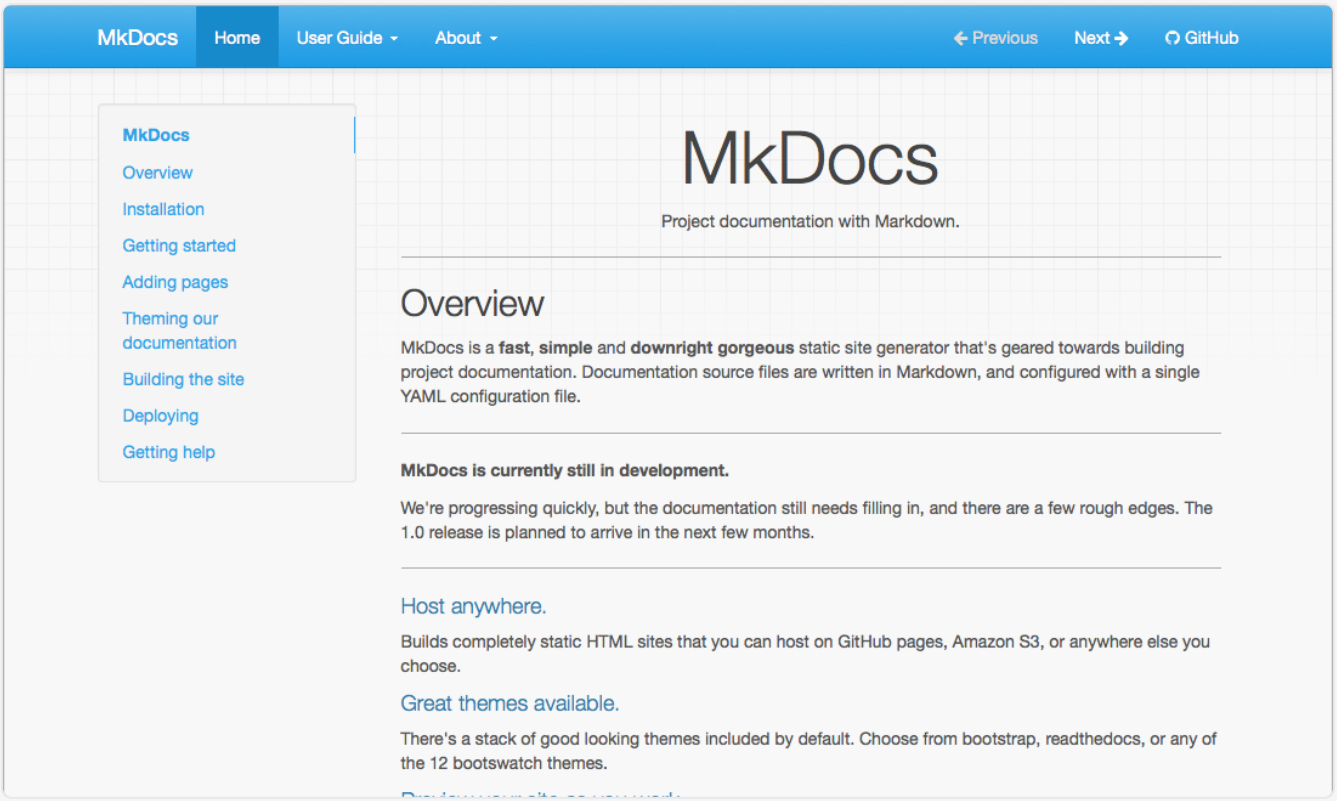

Figure 8 - mkdocs

In addition to the default theme configuration options, the `mkdocs` theme supports the following options:

- `highlightjs`: Enables highlighting of source code in code blocks using the highlight.js JavaScript library. Default: `True`.
- `hljs_style`: The highlight.js library provides 79 different styles (color variations) for highlighting source code in code blocks. Set this to the name of the desired style. Default: `github`.
- `hljs_languages`: By default, highlight.js only supports 23 common languages. List additional languages here to include support for them.

```
theme:
    name: mkdocs
    highlightjs: true
    hljs_languages:
        - yaml
        - rust
```

- `analytics`: Defines configuration options for an analytics service. Currently, only Google Analytics v4 is supported via the `gtag` option.
  - `gtag`: To enable Google Analytics, set to a Google Analytics v4 tracking ID, which uses the `G-` format. See Google's documentation to Set up Analytics for a website and/or app (GA4) or to Upgrade to a Google Analytics 4 property.

    ```
    theme:
        name: mkdocs
        analytics:
            gtag: G-ABC123
    ```

    When set to the default (`null`) Google Analytics is disabled for the site.
- `shortcuts`: Defines keyboard shortcut keys.

```
theme:
    name: mkdocs
    shortcuts:
        help: 191    # ?
        next: 78     # n
        previous: 80 # p
        search: 83   # s
```

All values must be numeric key codes. It is best to use keys which are available on all keyboards. You may use https://keycode.info/ to determine the key code for a given key.

- `help` : Display a help modal which lists the keyboard shortcuts. Default: `191` (?)
- `next` : Navigate to the "next" page. Default: `78` (n)
- `previous` : Navigate to the "previous" page. Default: `80` (p)
- `search` : Display the search modal. Default: `83` (s)

- `navigation_depth` : The maximum depth of the navigation tree in the sidebar. Default: `2` .
- `nav_style` : This adjusts the visual style for the top navigation bar; by default, this is set to `primary` (the default), but it can also be set to `dark` or `light` .

```
theme:
    name: mkdocs
    nav_style: dark
```

- `locale` : The locale (language/location) used to build the theme. If your locale is not yet supported, it will fallback to the default.

  The following locales are supported by this theme:

  - `en` : English (default)
  - `fr` : French
  - `es` : Spanish
  - `ja` : Japanese
  - `pt_BR` : Portuguese (Brazil)
  - `zh_CN` : Simplified Chinese
  - `de` : German

  See the guide on localizing your theme for more information.

3.4.2. readthedocs

A clone of the default theme used by the Read the Docs service, which offers the same restricted feature-set as its parent theme. Like its parent theme, only two levels of navigation are supported.

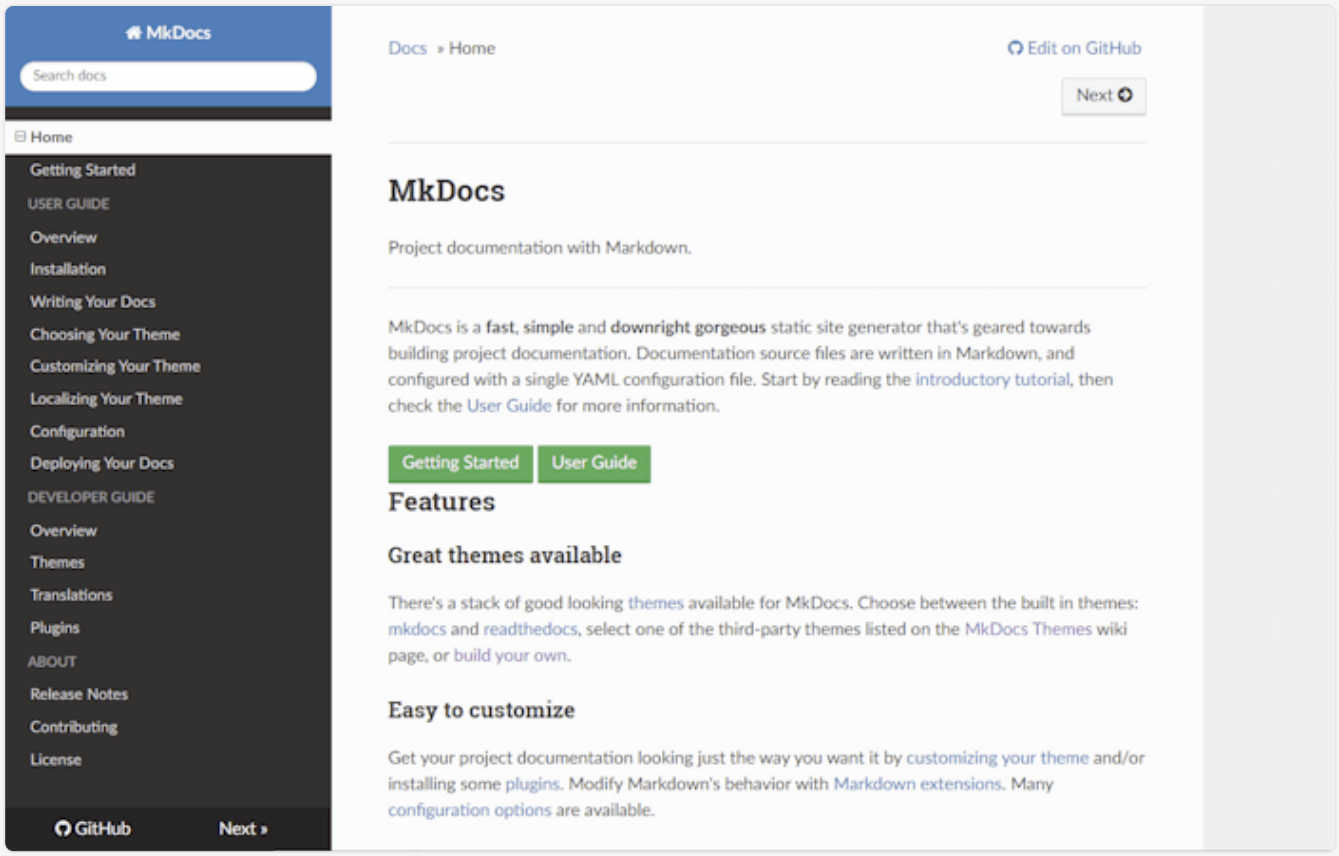

Figure 9 - ReadTheDocs

In addition to the default theme configuration options, the `readthedocs` theme supports the following options:

- `highlightjs` : Enables highlighting of source code in code blocks using the highlight.js JavaScript library. Default: `True` .
- `hljs_languages` : By default, highlight.js only supports 23 common languages. List additional languages here to include support for them.

```
theme:
    name: readthedocs
    highlightjs: true
    hljs_languages:
        - yaml
        - rust
```

- `analytics` : Defines configuration options for an analytics service.
  - `gtag` : To enable Google Analytics, set to a Google Analytics v4 tracking ID, which uses the `G-` format. See Google's documentation to Set up Analytics for a website and/or app (GA4) or to Upgrade to a Google Analytics 4 property.

```
theme:
    name: readthedocs
    analytics:
        gtag: G-ABC123
```

When set to the default (`null`) Google Analytics is disabled for the

- `anonymize_ip` : To enable anonymous IP address for Google Analytics, set this to `True` . Default: `False` .

- `include_homepage_in_sidebar` : Lists the homepage in the sidebar menu. As MkDocs requires that the homepage be listed in the `nav` configuration option, this setting allows the homepage to be included or excluded from the sidebar. Note that the site name/logo always links to the homepage. Default: `True` .
- `prev_next_buttons_location` : One of `bottom` , `top` , `both` , or `none` . Displays the “Next” and “Previous” buttons accordingly. Default: `bottom` .
- `navigation_depth` : The maximum depth of the navigation tree in the sidebar. Default: `4` .
- `collapse_navigation` : Only include the page section headers in the sidebar for the current page. Default: `True` .
- `titles_only` : Only include page titles in the sidebar, excluding all section headers for all pages. Default: `False` .
- `sticky_navigation` : If True, causes the sidebar to scroll with the main page content as you scroll the page. Default: `True` .
- `locale` : The locale (language/location) used to build the theme. If your locale is not yet supported, it will fallback to the default.

  The following locales are supported by this theme:

  - `en` : English (default)
  - `fr` : French
  - `es` : Spanish
  - `ja` : Japanese
  - `pt_BR` : Portuguese (Brazil)
  - `zh_CN` : Simplified Chinese
  - `de` : German

  See the guide on localizing your theme for more information.

- `logo` : To set a logo on your project instead of the plain text `site_name` , set this variable to be the location of your image. Default: `null` .

3.4.3. Third Party Themes

A list of third party themes can be found in the MkDocs community wiki. If you have created your own, please feel free to add it to the list.

3.5. Customizing Your Theme

Altering a theme to suit your needs.

If you would like to make a few tweaks to an existing theme, there is no need to create your own theme from scratch. For minor tweaks which only require some CSS and/or JavaScript, you can use the docs_dir. However, for more complex customizations, including overriding templates, you will need to use the theme custom_dir setting.

3.5.1. Using the docs_dir

The extra_css and extra_javascript configuration options can be used to make tweaks and customizations to existing themes. To use these, you simply need to include either CSS or JavaScript files within your documentation directory.

For example, to change the color of the headers in your documentation, create a file called `extra.css` and place it next to the documentation Markdown. In that file add the following CSS.

```
h1 {
  color: red;
}
```

> **Note**
>
> If you are deploying your documentation with ReadTheDocs. You will need to explicitly list the CSS and JavaScript files you want to include in your config. To do this, add the following to your mkdocs.yml.
>
> ```
> extra_css: [extra.css]
> ```

After making these changes, they should be visible when you run `mkdocs serve` - if you already had this running, you should see that the CSS changes were automatically picked up and the documentation will be updated.

> **Note**
>
> Any extra CSS or JavaScript files will be added to the generated HTML document after the page content. If you desire to include a JavaScript library, you may have better success including the library by using the theme custom_dir.

3.5.2. Using the theme custom_dir

The `theme.custom_dir` configuration option can be used to point to a directory of files which override the files in a parent theme. The parent theme would be the theme defined in the `theme.name` configuration option. Any file in the `custom_dir` with the same name as a file in the parent theme will replace the file of the same name in the parent theme. Any additional files in the `custom_dir` will be added to the parent theme. The contents of the `custom_dir` should mirror the directory structure of the parent theme. You may include templates, JavaScript files, CSS files, images, fonts, or any other media included in a theme.

> **Note**
>
> For this to work, the `theme.name` setting must be set to a known installed theme. If the `name` setting is instead set to `null` (or not defined), then there is no theme to override and the contents of the `custom_dir` must be a complete, standalone theme. See the Theme Developer Guide for more information.

For example, the mkdocs theme (browse source), contains the following directory structure (in part):

```
- css\
- fonts\
- img\
  - favicon.ico
  - grid.png
- js\
- 404.html
- base.html
- content.html
- nav-sub.html
- nav.html
- toc.html
```

To override any of the files contained in that theme, create a new directory next to your `docs_dir`:

```
mkdir custom_theme
```

And then point your `mkdocs.yml` configuration file at the new directory:

```
theme:
    name: mkdocs
    custom_dir: custom_theme/
```

To override the 404 error page ("file not found"), add a new template file named `404.html` to the `custom_theme` directory. For information on what can be included in a template, review the Theme Developer Guide.

To override the favicon, you can add a new icon file at `custom_theme/img/favicon.ico`.

To include a JavaScript library, copy the library to the `custom_theme/js/` directory.

Your directory structure should now look like this:

```
- docs/
  - index.html
- custom_theme/
  - img/
    - favicon.ico
  - js/
    - somelib.js
  - 404.html
- config.yml
```

> **Note**
>
> Any files included in the parent theme (defined in `name`) but not included in the `custom_dir` will still be utilized. The `custom_dir` will only override/replace files in the parent theme. If you want to remove files, or build a theme from scratch, then you should review the Theme Developer Guide.

3.5.2.1. Overriding Template Blocks

The built-in themes implement many of their parts inside template blocks which can be individually overridden in the `main.html` template. Simply create a `main.html` template file in your `custom_dir` and define replacement blocks within that file. Just make sure that the `main.html` extends `base.html`. For example, to alter the title of the MkDocs theme, your replacement `main.html` template would contain the following:

```
{% extends "base.html" %}

{% block htmltitle %}
<title>Custom title goes here</title>
{% endblock %}
```

In the above example, the `htmltitle` block defined in your custom `main.html` file will be used in place of the default `htmltitle` block defined in the parent theme. You may re-define as many blocks as you desire, as long as those blocks are defined in the parent. For example, you could replace the Google Analytics script with one for a different service or replace the search feature with

your own. You will need to consult the parent theme you are using to determine what blocks are available to override. The MkDocs and ReadTheDocs themes provide the following blocks:

- `site_meta` : Contains meta tags in the document head.
- `htmltitle` : Contains the page title in the document head.
- `styles` : Contains the link tags for stylesheets.
- `libs` : Contains the JavaScript libraries (jQuery, etc) included in the page header.
- `scripts` : Contains JavaScript scripts which should execute after a page loads.
- `analytics` : Contains the analytics script.
- `extrahead` : An empty block in the `<head>` to insert custom tags/scripts/etc.
- `site_name` : Contains the site name in the navigation bar.
- `site_nav` : Contains the site navigation in the navigation bar.
- `search_button` : Contains the search box in the navigation bar.
- `next_prev` : Contains the next and previous buttons in the navigation bar.
- `repo` : Contains the repository link in the navigation bar.
- `content` : Contains the page content and table of contents for the page.
- `footer` : Contains the page footer.

You may need to view the source template files to ensure your modifications will work with the structure of the site. See Template Variables for a list of variables you can use within your custom blocks. For a more complete explanation of blocks, consult the Jinja documentation.

3.5.2.2. Combining the custom_dir and Template Blocks

Adding a JavaScript library to the `custom_dir` will make it available, but won't include it in the pages generated by MkDocs. Therefore, a link needs to be added to the library from the HTML.

Starting the with directory structure above (truncated):

```
- docs/
- custom_theme/
  - js/
    - somelib.js
- config.yml
```

A link to the `custom_theme/js/somelib.js` file needs to be added to the template. As `somelib.js` is a JavaScript library, it would logically go in the `libs` block. However, a new `libs` block that only includes the new script will replace the block defined in the parent template and any

links to libraries in the parent template will be removed. To avoid breaking the template, a super block can be used with a call to `super` from within the block:

```
{% extends "base.html" %}

{% block libs %}
    {{ super() }}
    <script src="{{ base_url }}/js/somelib.js"></script>
{% endblock %}
```

Note that the base_url template variable was used to ensure that the link is always relative to the current page.

Now the generated pages will include links to the template provided libraries as well as the library included in the `custom_dir`. The same would be required for any additional CSS files included in the `custom_dir`.

3.6. Localizing Your Theme

Display your theme in your preferred language.

> **Note**
>
> Theme localization only translates the text elements of the theme itself (such as "next" and "previous" links), not the actual content of your documentation. If you wish to create multilingual documentation, you need to combine theme localization as described here with a third-party internationalization/localization plugin.

3.6.1. Installation

For theme localization to work, you must use a theme which supports it and enable `i18n` (internationalization) support by installing `mkdocs[i18n]`:

```
pip install mkdocs[i18n]
```

3.6.2. Supported locales

In most cases a locale is designated by the ISO-639-1 (2-letter) abbreviation for your language. However, a locale may also include a territory (or region or county) code as well. The language and territory must be separated by an underscore. For example, some possible locales for English might include `en`, `en_AU`, `en_GB`, and `en_US`.

For a list of locales supported by the theme you are using, see that theme's documentation.

- mkdocs
- readthedocs

> **Warning**
>
> If you configure a language locale which is not yet supported by the theme that you are using, MkDocs will fall back to the theme's default locale.

3.6.3. Usage

To specify the locale that MkDocs should use, set the locale parameter of the theme configuration option to the appropriate code.

For example, to build the `mkdocs` theme in French you would use the following in your `mkdocs.yml` configuration file:

```
theme:
    name: mkdocs
    locale: fr
```

3.6.4. Contributing theme translations

If a theme has not yet been translated into your language, feel free to contribute a translation using the Translation Guide.

3.7. Configuration

Guide to all available configuration settings.

3.7.1. Introduction

Project settings are configured by default using a YAML configuration file in the project directory named `mkdocs.yml`. You can specify another path for it by using the `-f` / `--config-file` option (see `mkdocs build --help`).

As a minimum, this configuration file must contain the `site_name` and `site_url` settings. All other settings are optional.

3.7.2. Project information

3.7.2.1. site_name

This is a **required setting**, and should be a string that is used as the main title for the project documentation. For example:

```
site_name: Marshmallow Generator
```

When rendering the theme this setting will be passed as the `site_name` context variable.

3.7.2.2. site_url

Set the canonical URL of the site. This will add a `link` tag with the `canonical` URL to the `head` section of each HTML page. If the 'root' of the MkDocs site will be within a subdirectory of a domain, be sure to include that subdirectory in the setting (`https://example.com/foo/`).

This setting is also used for `mkdocs serve`: the server will be mounted onto a path taken from the path component of the URL, e.g. `some/page.md` will be served from `http://127.0.0.1:8000/foo/some/page/` to mimic the expected remote layout.

default: `null`

3.7.2.3. repo_url

When set, provides a link to your repository (GitHub, Bitbucket, GitLab, ...) on each page.

```
repo_url: https://github.com/example/repository/
```

default: `null`

3.7.2.4. repo_name

When set, provides the name for the link to your repository on each page.

default: `'GitHub'`, `'Bitbucket'` or `'GitLab'` if the `repo_url` matches those domains, otherwise the hostname from the `repo_url`.

3.7.2.5. edit_uri

The path from the base `repo_url` to the docs directory when directly viewing a page, accounting for specifics of the repository host (e.g. GitHub, Bitbucket, etc), the branch, and the docs directory itself. MkDocs concatenates `repo_url` and `edit_uri`, and appends the input path of the page.

When set, and if your theme supports it, provides a link directly to the page in your source repository. This makes it easier to find and edit the source for the page. If `repo_url` is not set, this option is ignored. On some themes, setting this option may cause an edit link to be used in place of a repository link. Other themes may show both links.

The `edit_uri` supports query ('?') and fragment ('#') characters. For repository hosts that use a query or a fragment to access the files, the `edit_uri` might be set as follows. (Note the `?` and `#` in the URI...)

```
# Query string example
edit_uri: '?query=root/path/docs/'
```

```
# Hash fragment example
edit_uri: '#root/path/docs/'
```

For other repository hosts, simply specify the relative path to the docs directory.

```
# Query string example
edit_uri: root/path/docs/
```

Note

On a few known hosts (specifically GitHub, Bitbucket and GitLab), the `edit_uri` is derived from the 'repo_url' and does not need to be set manually. Simply defining a `repo_url` will automatically populate the `edit_uri` configs setting.

For example, for a GitHub- or GitLab-hosted repository, the `edit_uri` would be automatically set as `edit/master/docs/` (Note the `edit` path and `master` branch).

For a Bitbucket-hosted repository, the equivalent `edit_uri` would be automatically set as `src/default/docs/` (note the `src` path and `default` branch).

To use a different URI than the default (for example a different branch), simply set the `edit_uri` to your desired string. If you do not want any "edit URL link" displayed on your pages, then set `edit_uri` to an empty string to disable the automatic setting.

Warning

On GitHub and GitLab, the default "edit" path (`edit/master/docs/`) opens the page in the online editor. This functionality requires that the user have and be logged in to a GitHub/GitLab account. Otherwise, the user will be redirected to a login/signup page. Alternatively, use the "blob" path (`blob/master/docs/`) to open a read-only view, which supports anonymous access.

default: `edit/master/docs/` for GitHub and GitLab repos or `src/default/docs/` for a Bitbucket repo, if `repo_url` matches those domains, otherwise `null`

3.7.2.6. site_description

Set the site description. This will add a meta tag to the generated HTML header.

default: `null`

3.7.2.7. site_author

Set the name of the author. This will add a meta tag to the generated HTML header.

default: `null`

3.7.2.8. copyright

Set the copyright information to be included in the documentation by the theme.

default: `null`

3.7.2.9. remote_branch

Set the remote branch to commit to when using `gh-deploy` to deploy to GitHub Pages. This option can be overridden by a command line option in `gh-deploy`.

default: `gh-pages`

3.7.2.10. remote_name

Set the remote name to push to when using `gh-deploy` to deploy to GitHub Pages. This option can be overridden by a command line option in `gh-deploy`.

default: `origin`

3.7.3. Documentation layout

3.7.3.1. nav

This setting is used to determine the format and layout of the global navigation for the site. A minimal navigation configuration could look like this:

```
nav:
    - 'index.md'
    - 'about.md'
```

All paths in the navigation configuration must be relative to the `docs_dir` configuration option. See the section on configuring pages and navigation for a more detailed breakdown, including how to create sub-sections.

Navigation items may also include links to external sites. While titles are optional for internal links, they are required for external links. An external link may be a full URL or a relative URL. Any path which is not found in the files is assumed to be an external link. See the section about [Meta-Data] on how MkDocs determines the page title of a document.

```
nav:
    - Introduction: 'index.md'
    - 'about.md'
    - 'Issue Tracker': 'https://example.com/'
```

In the above example, the first two items point to local files while the third points to an external site.

However, sometimes the MkDocs site is hosted in a subdirectory of a project's site and you may want to link to other parts of the same site without including the full domain. In that case, you may use an appropriate relative URL.

```
site_url: https://example.com/foo/

nav:
    - Home: '../'
    - 'User Guide': 'user-guide.md'
    - 'Bug Tracker': '/bugs/'
```

In the above example, two different styles of external links are used. First, note that the `site_url` indicates that the MkDocs site is hosted in the `/foo/` subdirectory of the domain. Therefore, the `Home` navigation item is a relative link that steps up one level to the server root and effectively points to `https://example.com/`. The `Bug Tracker` item uses an absolute path from the server root and effectively points to `https://example.com/bugs/`. Of course, the `User Guide` points to a local MkDocs page.

default: By default `nav` will contain an alphanumerically sorted, nested list of all the Markdown files found within the `docs_dir` and its sub-directories. Index files will always be listed first within a sub-section.

3.7.4. Build directories

3.7.4.1. theme

Sets the theme and theme specific configuration of your documentation site. May be either a string or a set of key/value pairs.

If a string, it must be the string name of a known installed theme. For a list of available themes visit Choosing Your Theme.

An example set of key/value pairs might look something like this:

```
theme:
    name: mkdocs
    locale: en
    custom_dir: my_theme_customizations/
    static_templates:
        - sitemap.html
    include_sidebar: false
```

If a set of key/value pairs, the following nested keys can be defined:

> **name:**
>
> The string name of a known installed theme. For a list of available themes visit Choosing Your Theme.
>
> **locale:**
>
> A code representing the language of your site. See Localizing your theme for details.
>
> **custom_dir:**
>
> A directory containing a custom theme. This can either be a relative directory, in which case it is resolved relative to the directory containing your configuration file or it can be an absolute directory path from the root of your local file system.
>
> See Customizing Your Theme for details if you would like to tweak an existing theme.
>
> See the Theme Developer Guide if you would like to build your own theme from the ground up.
>
> **static_templates:**
>
> A list of templates to render as static pages. The templates must be located in either the theme's template directory or in the `custom_dir` defined in the theme configuration.
>
> **(theme specific keywords)**
>
> Any additional keywords supported by the theme can also be defined. See the documentation for the theme you are using for details.

default: `'mkdocs'`

3.7.4.2. docs_dir

The directory containing the documentation source markdown files. This can either be a relative directory, in which case it is resolved relative to the directory containing your configuration file, or it can be an absolute directory path from the root of your local file system.

default: `'docs'`

3.7.4.3. site_dir

The directory where the output HTML and other files are created. This can either be a relative directory, in which case it is resolved relative to the directory containing your configuration file, or it can be an absolute directory path from the root of your local file system.

default: `'site'`

> **Note:**
> If you are using source code control you will normally want to ensure that your build output files are not committed into the repository, and only keep the source files under version control. For example, if using `git` you might add the following line to your `.gitignore` file:
>
> ```
> site/
> ```
>
> If you're using another source code control tool, you'll want to check its documentation on how to ignore specific directories.

3.7.4.4. extra_css

Set a list of CSS files in your `docs_dir` to be included by the theme. For example, the following example will include the extra.css file within the css subdirectory in your docs_dir.

```
extra_css:
    - css/extra.css
    - css/second_extra.css
```

default: `[]` (an empty list).

3.7.4.5. extra_javascript

Set a list of JavaScript files in your `docs_dir` to be included by the theme. See the example in extra_css for usage.

default: `[]` (an empty list).

3.7.4.6. extra_templates

Set a list of templates in your `docs_dir` to be built by MkDocs. To see more about writing templates for MkDocs read the documentation about [custom themes] and specifically the section about the [available variables] to templates. See the example in extra_css for usage.

default: `[]` (an empty list).

3.7.4.7. extra

A set of key-value pairs, where the values can be any valid YAML construct, that will be passed to the template. This allows for great flexibility when creating custom themes.

For example, if you are using a theme that supports displaying the project version, you can pass it to the theme like this:

```
extra:
    version: 1.0
```

default: By default `extra` will be an empty key-value mapping.

3.7.5. Preview controls

3.7.6. Live Reloading

3.7.6.1. watch

Determines additional directories to watch when running `mkdocs serve`. Configuration is a YAML list.

```
watch:
- directory_a
- directory_b
```

Allows a custom default to be set without the need to pass it through the `-w` / `--watch` option every time the `mkdocs serve` command is called.

> **Note**
>
> The paths provided via the configuration file are relative to the configuration file.
>
> The paths provided via the `-w` / `--watch` CLI parameters are not.

3.7.6.2. use_directory_urls

This setting controls the style used for linking to pages within the documentation.

The following table demonstrates how the URLs used on the site differ when setting `use_directory_urls` to `true` or `false`.

Source file	use_directory_urls: true	use_directory_urls: false	more column
index.md	/	/index.html	/index.html
api-guide.md	/api-guide/	/api-guide.html	/api-guide.html

Source file	use_directory_urls: true	use_directory_urls: false	more column
about/license.md	/about/license/	/about/license.html	/about/license.html

The default style of `use_directory_urls: true` creates more user friendly URLs, and is usually what you'll want to use.

The alternate style can be useful if you want your documentation to remain properly linked when opening pages directly from the file system, because it creates links that point directly to the target file rather than the target directory.

default: `true`

3.7.6.3. strict

Determines how warnings are handled. Set to `true` to halt processing when a warning is raised. Set to `false` to print a warning and continue processing.

default: `false`

3.7.6.4. dev_addr

Determines the address used when running `mkdocs serve`. Must be of the format `IP:PORT`.

Allows a custom default to be set without the need to pass it through the `--dev-addr` option every time the `mkdocs serve` command is called.

default: `'127.0.0.1:8000'`

See also: site_url.

3.7.7. Formatting options

3.7.7.1. markdown_extensions

MkDocs uses the Python Markdown library to translate Markdown files into HTML. Python Markdown supports a variety of extensions that customize how pages are formatted. This setting lets you enable a list of extensions beyond the ones that MkDocs uses by default (`meta` , `toc` , `tables` , and `fenced_code`).

For example, to enable the SmartyPants typography extension, use:

```
markdown_extensions:
    - smarty
```

Some extensions provide configuration options of their own. If you would like to set any configuration options, then you can nest a key/value mapping (`option_name: option value`) of any options that a given extension supports. See the documentation for the extension you are using to determine what options they support.

For example, to enable permalinks in the (included) `toc` extension, use:

```
markdown_extensions:
    - toc:
        permalink: True
```

Note that a colon (`:`) must follow the extension name (`toc`) and then on a new line the option name and value must be indented and separated by a colon. If you would like to define multiple options for a single extension, each option must be defined on a separate line:

```
markdown_extensions:
    - toc:
        permalink: True
        separator: "_"
```

Add an additional item to the list for each extension. If you have no configuration options to set for a specific extension, then simply omit options for that extension:

```
markdown_extensions:
    - smarty
    - toc:
        permalink: True
    - sane_lists
```

In the above examples, each extension is a list item (starts with a `-`). As an alternative, key/value pairs can be used instead. However, in that case an empty value must be provided for extensions for which no options are defined. Therefore, the last example above could also be defined as follows:

```
markdown_extensions:
    smarty: {}
    toc:
        permalink: True
    sane_lists: {}
```

This alternative syntax is required if you intend to override some options via inheritance.

> **See Also:**
>
> The Python-Markdown documentation provides a list of extensions which are available out-of-the-box. For a list of configuration options available for a given extension, see the documentation for that extension.
>
> You may also install and use various third party extensions. Consult the documentation provided by those extensions for installation instructions and available configuration options.

default: `[]` (an empty list).

3.7.7.2. plugins

A list of plugins (with optional configuration settings) to use when building the site. See the Plugins documentation for full details.

If the `plugins` config setting is defined in the `mkdocs.yml` config file, then any defaults (such as `search`) are ignored and you need to explicitly re-enable the defaults if you would like to continue using them:

```
plugins:
    - search
    - your_other_plugin
```

To define options for a given plugin, use a nested set of key/value pairs:

```
plugins:
    - search
    - your_other_plugin:
        option1: value
        option2: other value
```

In the above examples, each plugin is a list item (starts with a `-`). As an alternative, key/value pairs can be used instead. However, in that case an empty value must be provided for plugins for which no options are defined. Therefore, the last example above could also be defined as follows:

```
plugins:
    search: {}
    your_other_plugin:
        option1: value
        option2: other value
```

This alternative syntax is required if you intend to override some options via inheritance.

To completely disable all plugins, including any defaults, set the `plugins` setting to an empty list:

```
plugins: []
```

default: `['search']` (the "search" plugin included with MkDocs).

Search

A search plugin is provided by default with MkDocs which uses lunr.js as a search engine. The following config options are available to alter the behavior of the search plugin:

separator

A regular expression which matches the characters used as word separators when building the index. By default whitespace and the hyphen (`-`) are used. To add the dot (`.`) as a word separator you might do this:

```
plugins:
    - search:
        separator: '[\s\-\.]+'
```

default: `'[\s\-]+'`

min_search_length

An integer value that defines the minimum length for a search query. By default searches shorter than 3 chars in length are ignored as search result quality with short search terms are poor. However, for some use cases (such as documentation about Message Queues which might generate searches for 'MQ') it may be preferable to set a shorter limit.

```
plugins:
    - search:
        min_search_length: 2
```

default: 3

lang

A list of languages to use when building the search index as identified by their ISO 639-1 language codes. With Lunr Languages, the following languages are supported:

- `ar` : Arabic
- `da` : Danish
- `nl` : Dutch
- `en` : English

- `fi` : Finnish
- `fr` : French
- `de` : German
- `hu` : Hungarian
- `it` : Italian
- `ja` : Japanese
- `no` : Norwegian
- `pt` : Portuguese
- `ro` : Romanian
- `ru` : Russian
- `es` : Spanish
- `sv` : Swedish
- `th` : Thai
- `tr` : Turkish
- `vi` : Vietnamese

You may contribute additional languages.

> **Warning**
>
> While search does support using multiple languages together, it is best not to add additional languages unless you really need them. Each additional language adds significant bandwidth requirements and uses more browser resources. Generally, it is best to keep each instance of MkDocs to a single language.

> **Note**
>
> Lunr Languages does not currently include support for Chinese or other Asian languages. However, some users have reported decent results using Japanese.

default: The value of `theme.locale` if set, otherwise `[en]`.

prebuild_index

Optionally generates a pre-built index of all pages, which provides some performance improvements for larger sites. Before enabling, confirm that the theme you are using explicitly supports using a prebuilt index (the builtin themes do). Set to `true` to enable.

> **Warning**
>
> This option requires that Node.js be installed and the command `node` be on the system path. If the call to `node` fails for any reason, a warning is issued and the build continues uninterrupted. You may use the `--strict` flag when building to cause such a failure to raise an error instead.

> **Note**
>
> On smaller sites, using a pre-built index is not recommended as it creates a significant increase is bandwidth requirements with little to no noticeable improvement to your users. However, for larger sites (hundreds of pages), the bandwidth increase is relatively small and your users will notice a significant improvement in search performance.

default: `False`

indexing

Configures what strategy the search indexer will use when building the index for your pages. This property is particularly useful if your project is large in scale, and the index takes up an enormous amount of disk space.

```
plugins:
    - search:
        indexing: 'full'
```

Options

Option	Description
`full`	Indexes the title, section headings, and full text of each page.
`sections`	Indexes the title and section headings of each page.
`titles`	Indexes only the title of each page.

default: `full`

3.7.8. Environment Variables

In most cases, the value of a configuration option is set directly in the configuration file. However, as an option, the value of a configuration option may be set to the value of an environment variable using the `!ENV` tag. For example, to set the value of the `site_name` option to the value of the variable `SITE_NAME` the YAML file may contain the following:

```
```

```
site_name: !ENV SITE_NAME
```

If the environment variable is not defined, then the configuration setting would be assigned a `null` (or `None` in Python) value. A default value can be defined as the last value in a list. Like this:

```
site_name: !ENV [SITE_NAME, 'My default site name']
```

Multiple fallback variables can be used as well. Note that the last value is not an environment variable, but must be a value to use as a default if none of the specified environment variables are defined.

```
site_name: !ENV [SITE_NAME, OTHER_NAME, 'My default site name']
```

Simple types defined within an environment variable such as string, bool, integer, float, datestamp and null are parsed as if they were defined directly in the YAML file, which means that the value will be converted to the appropriate type. However, complex types such as lists and key/value pairs cannot be defined within a single environment variable.

For more details, see the pyyaml_env_tag project.

3.7.9. Configuration Inheritance

Generally, a single file would hold the entire configuration for a site. However, some organizations may maintain multiple sites which all share a common configuration across them. Rather than maintaining separate configurations for each, the common configuration options can be defined in a parent configuration while which each site's primary configuration file inherits.

To define the parent for a configuration file, set the `INHERIT` (all caps) key to the path of the parent file. The path must be relative to the location of the primary file.

For configuration options to be merged with a parent configuration, those options must be defined as key/value pairs. Specifically, the markdown_extensions and plugins options must use the alternative syntax which does not use list items (lines which start with `-`).

For example, suppose the common (parent) configuration is defined in `base.yml`:

```
theme:
    name: mkdocs
    locale: en
    highlightjs: true

markdown_extensions:
    toc:
```

```
        permalink: true
    admonition: {}
```

Then, for the "foo" site, the primary configuration file would be defined at `foo/mkdocs.yml`:

```
INHERIT: ../base.yml
site_name: Foo Project
site_url: https://example.com/foo
```

When running `mkdocs build`, the file at `foo/mkdocs.yml` would be passed in as the configuration file. MkDocs will then parse that file, retrieve and parse the parent file `base.yml` and deep merge the two. This would result in MkDocs receiving the following merged configuration:

```
site_name: Foo Project
site_url: https://example.com/foo

theme:
    name: mkdocs
    locale: en
    highlightjs: true

markdown_extensions:
    toc:
        permalink: true
    admonition: {}
```

Deep merging allows you to add and/or override various values in your primary configuration file. For example, suppose for one site you wanted to add support for definition lists, use a different symbol for permalinks, and define a different separator. In that site's primary configuration file you could do:

```
INHERIT: ../base.yml
site_name: Bar Project
site_url: https://example.com/bar

markdown_extensions:
    def_list: {}
    toc:
        permalink:
        separator: "_"
```

In that case, the above configuration would be deep merged with `base.yml` and result in the following configuration:

```
site_name: Bar Project
site_url: https://example.com/bar
```

```yaml
theme:
    name: mkdocs
    locale: en
    highlightjs: true

markdown_extensions:
    def_list: {}
    toc:
        permalink:
        separator: "_"
    admonition: {}
```

Notice that the `admonition` extension was retained from the parent configuration, the `def_list` extension was added, the value of `toc.permalink` was replaced, and the value of `toc.separator` was added.

You can replace or merge the value of any key. However, any non-key is always replaced. Therefore, you cannot append items to a list. You must redefine the entire list.

As the nav configuration is made up of nested lists, this means that you cannot merge navigation items. Of course, you can replace the entire `nav` configuration with a new one. However, it is generally expected that the entire navigation would be defined in the primary configuration file for a project.

> **Warning**
>
> As a reminder, all path based configuration options must be relative to the primary configuration file and MkDocs does not alter the paths when merging. Therefore, defining paths in a parent file which is inherited by multiple different sites may not work as expected. It is generally best to define path based options in the primary configuration file only.

3.8. Deploying your docs

A basic guide to deploying your docs to various hosting providers

3.8.1. GitHub Pages

If you host the source code for a project on GitHub, you can easily use GitHub Pages to host the documentation for your project. There are two basic types of GitHub Pages sites: Project Pages sites, and User and Organization Pages sites. They are nearly identical but have some important differences, which require a different work flow when deploying.

3.8.1.1. Project Pages

Project Pages sites are simpler as the site files get deployed to a branch within the project repository (`gh-pages` by default). After you `checkout` the primary working branch (usually `master`) of the git repository where you maintain the source documentation for your project, run the following command:

```
mkdocs gh-deploy
```

That's it! Behind the scenes, MkDocs will build your docs and use the ghp-import tool to commit them to the `gh-pages` branch and push the `gh-pages` branch to GitHub.

Use `mkdocs gh-deploy --help` to get a full list of options available for the `gh-deploy` command.

Be aware that you will not be able to review the built site before it is pushed to GitHub. Therefore, you may want to verify any changes you make to the docs beforehand by using the `build` or `serve` commands and reviewing the built files locally.

> **Warning**
>
> You should never edit files in your pages repository by hand if you're using the `gh-deploy` command because you will lose your work the next time you run the command.

3.8.1.2. Organization and User Pages

User and Organization Pages sites are not tied to a specific project, and the site files are deployed to the `master` branch in a dedicated repository named with the GitHub account name. Therefore, you

need working copies of two repositories on our local system. For example, consider the following file structure:

```
my-project/
    mkdocs.yml
    docs/
orgname.github.io/
```

After making and verifying updates to your project you need to change directories to the `orgname.github.io` repository and call the `mkdocs gh-deploy` command from there:

```
cd ../orgname.github.io/
mkdocs gh-deploy --config-file ../my-project/mkdocs.yml --remote-branch master
```

Note that you need to explicitly point to the `mkdocs.yml` configuration file as it is no longer in the current working directory. You also need to inform the deploy script to commit to the `master` branch. You may override the default with the remote_branch configuration setting, but if you forget to change directories before running the deploy script, it will commit to the `master` branch of your project, which you probably don't want.

3.8.1.3. Custom Domains

GitHub Pages includes support for using a Custom Domain for your site. In addition to the steps documented by GitHub, you need to take one additional step so that MkDocs will work with your custom domain. You need to add a `CNAME` file to the root of your docs_dir. The file must contain a single bare domain or subdomain on a single line (see MkDocs' own CNAME file as an example). You may create the file manually, or use GitHub's web interface to set up the custom domain (under Settings / Custom Domain). If you use the web interface, GitHub will create the `CNAME` file for you and save it to the root of your "pages" branch. So that the file does not get removed the next time you deploy, you need to copy the file to your `docs_dir`. With the file properly included in your `docs_dir`, MkDocs will include the file in your built site and push it to your "pages" branch each time you run the `gh-deploy` command.

If you are having problems getting a custom domain to work, see GitHub's documentation on Troubleshooting custom domains.

3.8.2. Read the Docs

Read the Docs offers free documentation hosting. You can import your docs using any major version control system, including Mercurial, Git, Subversion, and Bazaar. Read the Docs supports MkDocs out-of-the-box. Follow the instructions on their site to arrange the files in your repository properly,

create an account and point it at your publicly hosted repository. If properly configured, your documentation will update each time you push commits to your public repository.

> **Note**
>
> To benefit from all of the features offered by Read the Docs, you will need to use the Read the Docs theme which ships with MkDocs. The various themes which may be referenced in Read the Docs' documentation are Sphinx specific themes and will not work with MkDocs.

3.8.3. Other Providers

Any hosting provider which can serve static files can be used to serve documentation generated by MkDocs. While it would be impossible to document how to upload the docs to every hosting provider out there, the following guidelines should provide some general assistance.

When you build your site (using the `mkdocs build` command), all of the files are written to the directory assigned to the site_dir configuration option (defaults to `"site"`) in your `mkdocs.yaml` config file. Generally, you will simply need to copy the contents of that directory to the root directory of your hosting provider's server. Depending on your hosting provider's setup, you may need to use a graphical or command line ftp, ssh or scp client to transfer the files.

For example, a typical set of commands from the command line might look something like this:

```
mkdocs build
scp -r ./site user@host:/path/to/server/root
```

Of course, you will need to replace `user` with the username you have with your hosting provider and `host` with the appropriate domain name. Additionally, you will need to adjust the `/path/to/server/root` to match the configuration of your hosts' file system.

See your host's documentation for specifics. You will likely want to search their documentation for "ftp" or "uploading site".

3.8.4. Local Files

Rather than hosting your documentation on a server, you may instead distribute the files directly, which can then be viewed in a browser using the `file://` scheme.

Note that, due to the security settings of all modern browsers, some things will not work the same and some features may not work at all. In fact, a few settings will need to be customized in very specific ways.

- site_url:

The `site_url` must be set to an empty string, which instructs MkDocs to build your site so that it will work with the `file://` scheme.

```
site_url: ""
```

- use_directory_urls:

  Set `use_directory_urls` to `false`. Otherwise, internal links between pages will not work properly.

  ```
  use_directory_urls: false
  ```

- search:

  You will need to either disable the search plugin, or use a third-party search plugin which is specifically designed to work with the `file://` scheme. To disable all plugins, set the `plugins` setting to an empty list.

  ```
  plugins: []
  ```

  If you have other plugins enabled, simply ensure that `search` is not included in the list.

When writing your documentation, it is imperative that all internal links use relative URLs as documented. Remember, each reader of your documentation will be using a different device and the files will likely be in a different location on that device.

If you expect your documentation to be viewed off-line, you may also need to be careful about which themes you choose. Many themes make use of CDNs for various support files, which require a live Internet connection. You will need to choose a theme which includes all support files directly in the theme.

When you build your site (using the `mkdocs build` command), all of the files are written to the directory assigned to the site_dir configuration option (defaults to `"site"`) in your `mkdocs.yaml` config file. Generally, you will simply need to copy the contents of that directory and distribute it to your readers. Alternatively, you may choose to use a third party tool to convert the HTML files to some other documentation format.

3.8.5. 404 Pages

When MkDocs builds the documentation it will include a 404.html file in the build directory. This file will be automatically used when deploying to GitHub but only on a custom domain. Other web

servers may be configured to use it but the feature won't always be available. See the documentation for your server of choice for more information.

4. Developer Guide

4.1. Developer Guide

Extending MkDocs

The MkDocs Developer Guide provides documentation for developers of third party themes and plugins. Please see the Contributing Guide for information on contributing to MkDocs itself. You can jump directly to a page listed below, or use the next and previous buttons in the navigation bar at the top of the page to move through the documentation in order.

- Themes
- Plugins

4.2. Developing Themes

A guide to creating and distributing custom themes.

> **Note**
>
> If you are looking for existing third party themes, they are listed in the MkDocs community wiki. If you want to share a theme you create, you should list it on the Wiki.

When creating a new theme, you can either follow the steps in this guide to create one from scratch or you can download the `mkdocs-basic-theme` as a basic, yet complete, theme with all the boilerplate required. **You can find this base theme on GitHub**. It contains detailed comments in the code to describe the different features and their usage.

4.2.1. Creating a custom theme

The bare minimum required for a custom theme is a `main.html` Jinja2 template file which is placed in a directory that is not a child of the docs_dir. Within `mkdocs.yml`, set the `theme.custom_dir` option to the path of the directory containing `main.html`. The path should be relative to the configuration file. For example, given this example project layout:

```
mkdocs.yml
docs/
    index.md
    about.md
custom_theme/
    main.html
    ...
```

... you would include the following settings in `mkdocs.yml` to use the custom theme directory:

```yaml
theme:
    name: null
    custom_dir: 'custom_theme/'
```

Note

Generally, when building your own custom theme, the theme.name configuration setting would be set to `null`. However, if the theme.custom_dir configuration value is used in combination with an existing theme, the theme.custom_dir can be used to replace only specific parts of a built-in theme. For example, with the above layout and if you set `name: "mkdocs"` then the `main.html` file in the theme.custom_dir would replace the file of the same name in the `mkdocs` theme but otherwise the `mkdocs` theme would remain unchanged. This is useful if you want to make small adjustments to an existing theme.

For more specific information, see Customizing Your Theme.

Warning

A theme's configuration defined in a `mkdocs_theme.yml` file is not loaded from `theme.custom_dir`. When an entire theme exists in `theme.custom_dir` and `theme.name` is set to `null`, then the entire theme configuration must be defined in the theme configuration option in the `mkdocs.yml` file.

However, when a theme is packaged up for distribution, and loaded using the `theme.name` configuration option, then a `mkdocs_theme.yml` file is required for the theme.

4.2.2. Basic theme

The simplest `main.html` file is the following:

```
<!DOCTYPE html>
<html>
  <head>
    <title>{% if page.title %}{{ page.title }} - {% endif %}{{ config.site_name }}
</title>
  </head>
  <body>
    {{ page.content }}
  </body>
</html>
```

The body content from each page specified in `mkdocs.yml` is inserted using the `{{ page.content }}` tag. Style-sheets and scripts can be brought into this theme as with a normal HTML file. Navbars and tables of contents can also be generated and included automatically, through the `nav` and `toc` objects, respectively. If you wish to write your own theme, it is recommended to start with one of the built-in themes and modify it accordingly.

Note

As MkDocs uses Jinja as its template engine, you have access to all the power of Jinja, including template inheritance. You may notice that the themes included with MkDocs make extensive use of template inheritance and blocks, allowing users to easily override small bits and pieces of the templates from the theme custom_dir. Therefore, the built-in themes are implemented in a `base.html` file, which `main.html` extends. Although not required, third party template authors are encouraged to follow a similar pattern and may want to define the same blocks as are used in the built-in themes for consistency.

4.2.3. Theme Files

There are various files which a theme treats special in some way. Any other files are simply copied from the theme directory to the same path in the `site_dir` when the site it built. For example image and CSS files have no special significance and are copied as-is. Note, however, that if the user provides a file with the same path in their `docs_dir`, then the user's file will replace the theme file.

4.2.3.1. Template Files

Any files with the `.html` extension are considered to be template files and are not copied from the theme directory or any subdirectories. Also, any files listed in static_templates are treated as templates regardless of their file extension.

4.2.3.2. Theme Meta Files

The various files required for packaging a theme are also ignored. Specifically, the `mkdocs_theme.yml` configuration file and any Python files.

4.2.3.3. Dot Files

Theme authors can explicitly force MkDocs to ignore files by starting a file or directory name with a dot. Any of the following files would be ignored:

```
.ignored.txt
.ignored/file.txt
foo/.ignored.txt
foo/.ignored/file.txt
```

4.2.3.4. Documentation Files

All documentation files are ignored. Specifically, any Markdown files (using any of the file extensions supported by MKDocs). Additionally, any README files which may exist in the theme directories are ignored.

4.2.4. Template Variables

Each template in a theme is built with a template context. These are the variables that are available to themes. The context varies depending on the template that is being built. At the moment templates are either built with the global context or with a page specific context. The global context is used for HTML pages that don't represent an individual Markdown document, for example a 404.html page or search.html.

4.2.4.1. Global Context

The following variables are available globally on any template.

config

The `config` variable is an instance of MkDocs' config object generated from the `mkdocs.yml` config file. While you can use any config option, some commonly used options include:

- config.site_name
- config.site_url
- config.site_author
- config.site_description
- config.theme.locale (See also Theme Configuration below)
- config.extra_javascript
- config.extra_css
- config.repo_url
- config.repo_name
- config.copyright
- config.google_analytics

nav

The `nav` variable is used to create the navigation for the documentation. The `nav` object is an iterable of navigation objects as defined by the nav configuration setting.

In addition to the iterable of navigation objects, the `nav` object contains the following attributes:

nav.homepage

The page object for the homepage of the site.

nav.pages

A flat list of all page objects contained in the navigation. This list is not necessarily a complete list of all site pages as it does not contain pages which are not included in the navigation. This list does match the list and order of pages used for all "next page" and "previous page" links. For a list of all pages, use the pages template variable.

Nav Example

Following is a basic usage example which outputs the first and second level navigation as a nested list.

```
{% if nav|length>1 %}
    <ul>
    {% for nav_item in nav %}
        {% if nav_item.children %}
            <li>{{ nav_item.title }}
                <ul>
                {% for nav_item in nav_item.children %}
                    <li class="{% if nav_item.active%}current{% endif %}">
                        <a href="{{ nav_item.url|url }}">{{ nav_item.title }}</a>
                    </li>
                {% endfor %}
                </ul>
            </li>
        {% else %}
            <li class="{% if nav_item.active%}current{% endif %}">
                <a href="{{ nav_item.url|url }}">{{ nav_item.title }}</a>
            </li>
        {% endif %}
    {% endfor %}
    </ul>
{% endif %}
```

base_url

The `base_url` provides a relative path to the root of the MkDocs project. While this can be used directly by prepending it to a local relative URL, it is best to use the url template filter, which is smarter about how it applies `base_url`.

mkdocs_version

Contains the current MkDocs version.

build_date_utc

A Python datetime object that represents the date and time the documentation was built in UTC. This is useful for showing how recently the documentation was updated.

pages

A flat list of `File` objects for all pages in the project. This list can contain pages not included in the global navigation and may not match the order of pages within that navigation. The page object for each `File` can be accessed from `file.page`.

page

In templates which are not rendered from a Markdown source file, the `page` variable is `None`. In templates which are rendered from a Markdown source file, the `page` variable contains a `page` object. The same `page` objects are used as `page` navigation objects in the global navigation and in the pages template variable.

All `page` objects contain the following attributes:

page.title

Contains the Title for the current page.

page.content

The rendered Markdown as HTML, this is the contents of the documentation.

page.toc

An iterable object representing the Table of contents for a page. Each item in the `toc` is an `AnchorLink` which contains the following attributes:

- `AnchorLink.title`: The text of the item.
- `AnchorLink.url`: The hash fragment of a URL pointing to the item.
- `AnchorLink.level`: The zero-based level of the item.
- `AnchorLink.children`: An iterable of any child items.

The following example would display the top two levels of the Table of Contents for a page.

```
<ul>
{% for toc_item in page.toc %}
    <li><a href="{{ toc_item.url }}">{{ toc_item.title }}</a></li>
    {% for toc_item in toc_item.children %}
        <li><a href="{{ toc_item.url }}">{{ toc_item.title }}</a></li>
    {% endfor %}
```

```
  {% endfor %}
  </ul>
```

page.meta

A mapping of the metadata included at the top of the markdown page. In this example we define a `source` property above the page title.

```
source: generics.py
        mixins.py

# Page title

Content...
```

A template can access this metadata for the page with the `meta.source` variable. This could then be used to link to source files related to the documentation page.

```
{% for filename in page.meta.source %}
  <a class="github" href="https://github.com/.../{{ filename }}">
    <span class="label label-info">{{ filename }}</span>
  </a>
{% endfor %}
```

page.url

The URL of the page relative to the MkDocs `site_dir`. It is expected that this be used with the url filter to ensure the URL is relative to the current page.

```
<a href="{{ page.url|url }}">{{ page.title }}</a>
```

page.abs_url

The absolute URL of the page from the server root as determined by the value assigned to the site_url configuration setting. The value includes any subdirectory included in the `site_url`, but not the domain. base_url should not be used with this variable.

For example, if `site_url: https://example.com/`, then the value of `page.abs_url` for the page `foo.md` would be `/foo/`. However, if `site_url: https://example.com/bar/`, then the value of `page.abs_url` for the page `foo.md` would be `/bar/foo/`.

page.canonical_url

The full, canonical URL to the current page as determined by the value assigned to the site_url configuration setting. The value includes the domain and any subdirectory included in the `site_url`. base_url should not be used with this variable.

page.edit_url

The full URL to the source page in the source repository. Typically used to provide a link to edit the source page. base_url should not be used with this variable.

page.is_homepage

Evaluates to `True` for the homepage of the site and `False` for all other pages. This can be used in conjunction with other attributes of the `page` object to alter the behavior. For example, to display a different title on the homepage:

```
{% if not page.is_homepage %}{{ page.title }} - {% endif %}{{ site_name }}
```

page.previous_page

The page object for the previous page or `None`. The value will be `None` if the current page is the first item in the site navigation or if the current page is not included in the navigation at all. When the value is a page object, the usage is the same as for `page`.

page.next_page

The page object for the next page or `None`. The value will be `None` if the current page is the last item in the site navigation or if the current page is not included in the navigation at all. When the value is a page object, the usage is the same as for `page`.

page.parent

The immediate parent of the page in the site navigation. `None` if the page is at the top level.

page.children

Pages do not contain children and the attribute is always `None`.

page.active

When `True`, indicates that this page is the currently viewed page. Defaults to `False`.

page.is_section

Indicates that the navigation object is a "section" object. Always `False` for page objects.

page.is_page

Indicates that the navigation object is a "page" object. Always `True` for page objects.

page.is_link

Indicates that the navigation object is a "link" object. Always `False` for page objects.

4.2.4.2. Navigation Objects

Navigation objects contained in the nav template variable may be one of section objects, page objects, and link objects. While section objects may contain nested navigation objects, pages and links do not.

Page objects are the full page object as used for the current page with all of the same attributes available. Section and Link objects contain a subset of those attributes as defined below:

Section

A `section` navigation object defines a named section in the navigation and contains a list of child navigation objects. Note that sections do not contain URLs and are not links of any kind. However, by default, MkDocs sorts index pages to the top and the first child might be used as the URL for a section if a theme chooses to do so.

The following attributes are available on `section` objects:

section.title

The title of the section.

section.parent

The immediate parent of the section or `None` if the section is at the top level.

section.children

An iterable of all child navigation objects. Children may include nested sections, pages and links.

section.active

When `True`, indicates that a child page of this section is the current page and can be used to highlight the section as the currently viewed section. Defaults to `False`.

section.is_section

Indicates that the navigation object is a "section" object. Always `True` for section objects.

section.is_page

Indicates that the navigation object is a "page" object. Always `False` for section objects.

section.is_link

Indicates that the navigation object is a "link" object. Always `False` for section objects.

Link

A `link` navigation object contains a link which does not point to an internal MkDocs page. The following attributes are available on `link` objects:

link.title

The title of the link. This would generally be used as the label of the link.

link.url

The URL that the link points to. The URL should always be an absolute URLs and should not need to have `base_url` prepended.

link.parent

The immediate parent of the link. `None` if the link is at the top level.

link.children

Links do not contain children and the attribute is always `None`.

link.active

External links cannot be "active" and the attribute is always `False`.

link.is_section

Indicates that the navigation object is a "section" object. Always `False` for link objects.

link.is_page

Indicates that the navigation object is a "page" object. Always `False` for link objects.

link.is_link

Indicates that the navigation object is a "link" object. Always `True` for link objects.

4.2.4.3. Extra Context

Additional variables can be passed to the template with the `extra` configuration option. This is a set of key value pairs that can make custom templates far more flexible.

For example, this could be used to include the project version of all pages and a list of links related to the project. This can be achieved with the following `extra` configuration:

```
extra:
    version: 0.13.0
    links:
        - https://github.com/mkdocs
        - https://docs.readthedocs.org/en/latest/builds.html#mkdocs
        - https://www.mkdocs.org/
```

And then displayed with this HTML in the custom theme.

```
{{ config.extra.version }}

{% if config.extra.links %}
  <ul>
  {% for link in config.extra.links %}
      <li>{{ link }}</li>
  {% endfor %}
  </ul>
{% endif %}
```

4.2.5. Template Filters

In addition to Jinja's default filters, the following custom filters are available to use in MkDocs templates:

4.2.5.1. url

Normalizes a URL. Absolute URLs are passed through unaltered. If the URL is relative and the template context includes a page object, then the URL is returned relative to the page object. Otherwise, the URL is returned with base_url prepended.

```
<a href="{{ page.url|url }}">{{ page.title }}</a>
```

4.2.5.2. tojson

Safety convert a Python object to a value in a JavaScript script.

```
<script>
    var mkdocs_page_name = {{ page.title|tojson|safe }};
</script>
```

4.2.6. Search and themes

As of MkDocs version 0.17 client side search support has been added to MkDocs via the `search` plugin. A theme needs to provide a few things for the plugin to work with the theme.

While the `search` plugin is activated by default, users can disable the plugin and themes should account for this. It is recommended that theme templates wrap search specific markup with a check for the plugin:

```
{% if 'search' in config['plugins'] %}
    search stuff here...
{% endif %}
```

At its most basic functionality, the search plugin will simply provide an index file which is no more than a JSON file containing the content of all pages. The theme would need to implement its own search functionality client-side. However, with a few settings and the necessary templates, the plugin can provide a complete functioning client-side search tool based on lunr.js.

The following HTML needs to be added to the theme so that the provided JavaScript is able to properly load the search scripts and make relative links to the search results from the current page.

```
<script>var base_url = '{{ base_url }}';</script>
```

With properly configured settings, the following HTML in a template will add a full search implementation to your theme.

```
<h1 id="search">Search Results</h1>

<form action="search.html">
  <input name="q" id="mkdocs-search-query" type="text" >
</form>

<div id="mkdocs-search-results">
  Sorry, page not found.
</div>
```

The JavaScript in the plugin works by looking for the specific ID's used in the above HTML. The form input for the user to type the search query must be identified with `id="mkdocs-search-query"` and the div where the results will be placed must be identified with `id="mkdocs-search-results"`.

The plugin supports the following options being set in the theme's configuration file, `mkdocs_theme.yml`:

4.2.6.1. include_search_page

Determines whether the search plugin expects the theme to provide a dedicated search page via a template located at `search/search.html`.

When `include_search_page` is set to `true`, the search template will be built and available at `search/search.html`. This method is used by the `readthedocs` theme.

When `include_search_page` is set to `false` or not defined, it is expected that the theme provide some other mechanisms for displaying search results. For example, the `mkdocs` theme displays results on any page via a modal.

4.2.6.2. search_index_only

Determines whether the search plugin should only generate a search index or a complete search solution.

When `search_index_only` is set to `false`, then the search plugin modifies the Jinja environment by adding its own `templates` directory (with a lower precedence than the theme) and adds its scripts to the `extra_javascript` config setting.

When `search_index_only` is set to `true` or not defined, the search plugin makes no modifications to the Jinja environment. A complete solution using the provided index file is the responsibility of the theme.

The search index is written to a JSON file at `search/search_index.json` in the site_dir. The JSON object contained within the file may contain up to three objects.

```
{
    config: {...},
    docs: [...],
    index: {...}
}
```

If present, the `config` object contains the key/value pairs of config options defined for the plugin in the user's `mkdocs.yml` config file under `plugings.search`. The `config` object was new in MkDocs version 1.0.

The `docs` object contains a list of document objects. Each document object is made up of a `location` (URL), a `title`, and `text` which can be used to create a search index and/or display search results.

If present, the `index` object contains a pre-built index which offers performance improvements for larger sites. Note that the pre-built index is only created if the user explicitly enables the

prebuild_index config option. Themes should expect the index to not be present, but can choose to use the index when it is available. The `index` object was new in MkDocs version 1.0.

4.2.7. Packaging Themes

MkDocs makes use of Python packaging to distribute themes. This comes with a few requirements.

To see an example of a package containing one theme, see the MkDocs Bootstrap theme and to see a package that contains many themes, see the MkDocs Bootswatch theme.

> **Note**
>
> It is not strictly necessary to package a theme, as the entire theme can be contained in the `custom_dir`. If you have created a "one-off theme," that should be sufficient. However, if you intend to distribute your theme for others to use, packaging the theme has some advantages. By packaging your theme, your users can more easily install it, they can rely on a default configuration being defined, and they can then take advantage of the custom_dir to make tweaks to your theme to better suit their needs.

4.2.7.1. Package Layout

The following layout is recommended for themes. Two files at the top level directory called `MANIFEST.in` and `setup.py` beside the theme directory which contains an empty `__init__.py` file, a theme configuration file (`mkdocs_theme.yml`), and your template and media files.

```
.
|-- MANIFEST.in
|-- theme_name
|   |-- __init__.py
|   |-- mkdocs_theme.yml
|   |-- main.html
|   |-- styles.css
`-- setup.py
```

The `MANIFEST.in` file should contain the following contents but with theme_name updated and any extra file extensions added to the include.

```
recursive-include theme_name *.ico *.js *.css *.png *.html *.eot *.svg *.ttf *.woff
recursive-exclude * __pycache__
recursive-exclude * *.py[co]
```

The `setup.py` should include the following text with the modifications described below.

```
from setuptools import setup, find_packages

VERSION = '0.0.1'

setup(
    name="mkdocs-themename",
    version=VERSION,
    url='',
    license='',
    description='',
    author='',
    author_email='',
    packages=find_packages(),
    include_package_data=True,
    entry_points={
        'mkdocs.themes': [
            'themename = theme_name',
        ]
    },
    zip_safe=False
)
```

Fill in the URL, license, description, author and author email address.

The name should follow the convention `mkdocs-themename` (like `mkdocs-bootstrap` and `mkdocs-bootswatch`), starting with MkDocs, using hyphens to separate words and including the name of your theme.

Most of the rest of the file can be left unedited. The last section we need to change is the entry_points. This is how MkDocs finds the theme(s) you are including in the package. The name on the left is the one that users will use in their mkdocs.yml and the one on the right is the directory containing your theme files.

The directory you created at the start of this section with the main.html file should contain all of the other theme files. The minimum requirement is that it includes a `main.html` for the theme. It **must** also include a `__init__.py` file which should be empty, this file tells Python that the directory is a package.

4.2.7.2. Theme Configuration

A packaged theme is required to include a configuration file named `mkdocs_theme.yml` which is placed in the root of your template files. The file should contain default configuration options for the theme. However, if the theme offers no configuration options, the file is still required and can be left blank. A theme which is not packaged does not need a `mkdocs_theme.yml` file as that file is not loaded from `theme.custom_dir`.

The theme author is free to define any arbitrary options deemed necessary and those options will be made available in the templates to control behavior. For example, a theme might want to make a sidebar optional and include the following in the `mkdocs_theme.yml` file:

```
show_sidebar: true
```

Then in a template, that config option could be referenced:

```
{% if config.theme.show_sidebar %}
<div id="sidebar">...</div>
{% endif %}
```

And the user could override the default in their project's `mkdocs.yml` config file:

```
theme:
    name: themename
    show_sidebar: false
```

In addition to arbitrary options defined by the theme, MkDocs defines a few special options which alters its behavior:

locale

This option mirrors the theme config option of the same name. If this value is not defined in the `mkdocs_theme.yml` file and the user does not set it in `mkdocs.yml` then it will default to `en` (English). The value is expected to match the language used in the text provided by the theme (such a "next" and "previous" links) and should be used as the value of the `<html>` tag's `lang` attribute. See Supporting theme localization/ translation for more information.

Note that during configuration validation, the provided string is converted to a `Locale` object. The object contains `Locale.language` and `Locale.territory` attributes and will resolve as a string from within a template. Therefore, the following will work fine:

```
<html lang="{ config.theme.locale }">
```

If the locale was set to `fr_CA` (Canadian French), then the above template would render as:

```
<html lang="fr_CA">
```

If you did not want the territory attribute to be included, then reference the `language` attribute directly:

```
<html lang="{ config.theme.locale.language }">
```

That would render as:

```
<html lang="fr">
```

static_templates

This option mirrors the theme config option of the same name and allows some defaults to be set by the theme. Note that while the user can add templates to this list, the user cannot remove templates included in the theme's config.

extends

Defines a parent theme that this theme inherits from. The value should be the string name of the parent theme. Normal Jinja inheritance rules apply.

Plugins may also define some options which allow the theme to inform a plugin about which set of plugin options it expects. See the documentation for any plugins you may wish to support in your theme.

4.2.7.3. Distributing Themes

With the above changes, your theme should now be ready to install. This can be done with pip, using `pip install .` if you are still in the same directory as the setup.py.

Most Python packages, including MkDocs, are distributed on PyPI. To do this, you should run the following command.

```
python setup.py register
```

If you don't have an account setup, you should be prompted to create one.

For a much more detailed guide, see the official Python packaging documentation for Packaging and Distributing Projects.

4.2.8. Supporting theme Localization/Translation

While the built-in themes provide support for localization/translation of templates, custom themes and third-party themes may choose not to. Regardless, the `locale` setting of the `theme` configuration option is always present and is relied upon by other parts of the system. Therefore, it is recommended that all third-party themes use the same setting for designating a language regardless of the system they use for translation. In that way, users will experience consistent behavior regardless of the theme they may choose.

The method for managing translations is up to the developers of a theme. However, if a theme developer chooses to use the same mechanisms used by the built-in themes, the sections below outline how to enable and make use of the same commands utilized by MkDocs.

4.2.8.1. Enabling the Localization/Translation commands

MkDocs includes some helper commands which are light wrappers around pybabel's commands. To use the commands on your own theme, add the following to your theme's `setup.py` script:

```
from mkdocs.commands.setup import babel_cmdclass

setup(
    ...
    cmdclass=babel_cmdclass
)
```

Note that `cmdclass=babel_cmdclass` was added an a parameter passed to the `setup` function.

> **Warning**
>
> As **pybabel is not installed by default** and most users will not have pybabel installed, theme developers and/or translators should make sure to have installed the necessary dependencies (using `pip install mkdocs[i18n]`) in order for the commands to be available for use.

4.2.8.2. Using the Localization/Translation commands

Since the translation commands are embedded in the `setup.py` script of your custom theme they should be called from the root of your theme's working tree as follows:

```
python setup.py <command_name> [OPTIONS]
```

Each command provides a detailed list of options available with the `-h/--help` option.

For an overview of the workflow used by MkDocs to translate the built-in themes, see the appropriate section of the Contributing Guide and the Translation Guide.

Default values for many of the options to the commands can be defined in a `setup.cfg` file. Create a section using the command name as the section name, and the long option name as the key. See MkDocs' own setup.cfg file for an example.

A summary of changes/additions to the behavior and options of the upstream pybabel commands are summarized below.

compile_catalog

The `-t/--theme` option has been added to this command. The `theme` specified must be a `theme` defined as a entry point in the same `setup.py` script. Other themes will not be recognized. If only one `theme` has been defined as an entry point, then that `theme` will be used as the default if none is specified by this option. If more than one `theme` is defined as entry points, then no default is set and a `theme` must be specified by this option. The command only operates on one theme at a time. Therefore, the command needs to be run once for each theme included in a package.

When a `theme` is specified, the directory of that `theme` as defined in the entry point is used to define a default value of the `-d/--directory` option. The `--directory` option is set to `{theme_dir}/locales`. If a `directory` is passed to the `--directory` option, then the `theme` option is ignored.

extract_messages

The `-t/--theme` option has been added to this command. The `theme` specified must be a `theme` defined as a entry point in the same `setup.py` script. Other themes will not be recognized. If only one `theme` has been defined as an entry point, then that `theme` will be used as the default if none

is specified by this option. If more than one `theme` is defined as entry points, then no default is set and a `theme` must be specified by this option. The command only operates on one theme at a time. Therefore, the command needs to be run once for each theme included in a package.

When a `theme` is specified, the directory of that `theme` as defined in the entry point is used to define a default value for the `--input-dirs` and `--output-file` options. The `--input-dirs` option is set to the `theme` directory and `--output-file` is set to `{theme_dir}/{domain}.pot`. If a path is provided to either option, then the `theme` option is ignored for that option.

The `--domain` option has been added to this command and can be used to override the `domain` used for the `output-file` based on the `theme`. Defaults to `messages`.

The `-F/--mapping-file` option defaults to the mapping file used by MkDocs' built-in themes. However, if that mapping file does not meet your theme's needs to can override it by providing your own and passing the path of that file into the option.

init_catalog

The `-t/--theme` option has been added to this command. The `theme` specified must be a `theme` defined as a entry point in the same `setup.py` script. Other themes will not be recognized. If only one `theme` has been defined as an entry point, then that `theme` will be used as the default if none is specified by this option. If more than one `theme` is defined as entry points, then no default is set and a `theme` must be specified by this option. The command only operates on one theme at a time. Therefore, the command needs to be run once for each theme included in a package.

When a `theme` is specified, the directory of that `theme` as defined in the entry point is used to define a default value for the `-i/--input-file` and `-d/--output-dir` options. The `--input-file` option is set to `{theme_dir}/{domain}.pot` (`domain` defaults to `messages`) and `--output-dir` is set to `{theme_dir}/locales`. If a path is provided to either option, then the `theme` option is ignored for that option.

update_catalog

The `-t/--theme` option has been added to this command. The `theme` specified must be a `theme` defined as a entry point in the same `setup.py` script. Other themes will not be recognized. If only one `theme` has been defined as an entry point, then that `theme` will be used as the default if none is specified by this option. If more than one `theme` is defined as entry points, then no default is set and a `theme` must be specified by this option. The command only operates on one theme at a time. Therefore, the command needs to be run once for each theme included in a package.

When a `theme` is specified, the directory of that `theme` as defined in the entry point is used to define a default value for the `-i/--input-file` and `-d/--output-dir` options. The `--input-file` option is set to `{theme_dir}/{domain}.pot` (`domain` defaults to `messages`)

and `--output-dir` is set to `{theme_dir}/locales`. If a path is provided to either option, then the `theme` option is ignored for that option.

4.2.8.3. Example custom theme Localization/Translation workflow

> **Note**
>
> If your theme inherits from an existing theme which already provides translation catalogs, your theme's translations will be merged with the parent theme's translations during a MkDocs build.
>
> This means that you only need to concentrate on the added translations. Yet, you will still benefit from the translations of the parent theme. At the same time, you may override any of parent theme's translations!

Let's suppose that you're working on your own fork of the mkdocs-basic-theme and want to add translations to it.

You would first modify the `setup.py` like this:

```
--- a/setup.py
+++ b/setup.py
@@ -1,4 +1,5 @@
 from setuptools import setup, find_packages
+from mkdocs.commands.setup import babel_cmdclass

 VERSION = '1.1'

@@ -18,5 +19,6 @@ setup(
             'basictheme = basic_theme',
         ]
     },
-    zip_safe=False
+    zip_safe=False,
+    cmdclass=babel_cmdclass
 )
```

Next, you would edit the templates by wrapping text in your HTML sources with `{% trans %}` and `{% endtrans %}` as follows:

```
--- a/basic_theme/base.html
+++ b/basic_theme/base.html
@@ -88,7 +88,7 @@

 <body>

-  <h1>This is an example theme for MkDocs.</h1>
+  <h1>{% trans %}This is an example theme for MkDocs.{% endtrans %}</h1>

   <p>
     It is designed to be read by looking at the theme HTML which is heavily
```

Then you would follow the Translation Guide as usual to get your translations running.

4.2.8.4. Packaging Translations with your theme

While the Portable Object Template (`pot`) file created by the `extract_messages` command and the Portable Object (`po`) files created by the `init_catalog` and `update_catalog` commands are useful for creating and editing translations, they are not used by MkDocs directly and do not need to be included in a packaged release of a theme. When MkDocs builds a site with translations, it only makes use of the binary `mo` files(s) for the specified locale. Therefore, when packaging a theme, you would need to make the following addition to your `MANIFEST.in` file:

```
recursive-include theme_name *.mo
```

Then, before building your Python package, you will want to ensure that the binary `mo` file for each locale is up-to-date by running the `compile_catalog` command for each locale. MkDocs expects the binary `mo` files to be located at `locales/<locale>/LC_MESSAGES/messages.mo` , which the `compile_catalog` command automatically does for you. See Testing theme translations for details.

> **Note**
>
> As outlined in our Translation Guide, the MkDocs project has chosen to include the `pot` and `po` files in our code repository, but not the `mo` files. This requires us to always run `compile_catalog` before packaging a new release regardless of whether any changes were made to a translation or not. However, you may chose an alternate workflow for your theme. At a minimum, you need to ensure that up-to-date `mo` files are included at the correct location in each release. However, you may use a different process for generating those `mo` files if you chose to do so.

4.3. Translations

Theme localization guide.

The built-in themes that are included with MkDocs provide support for translations. This is a guide for translators, which documents the process for contributing new translations and/or updating existing translations. For guidance on modifying the existing themes, see the Contributing Guide. To enable a specific translation see the documentation about the specific theme you are using in the User Guide. For translations of third-party themes, please see the documentation for those themes. For a third-party theme to make use of MkDocs' translation tools and methods, that theme must be properly configured to make use of those tools.

> **Note**
>
> Translations only apply to text contained within a theme's template, such as "next" and "previous" links. The Markdown content of a page is not translated. If you wish to create multilingual documentation, you need to combine theme localization with a third-party internationalization/localization plugin.

4.3.1. Localization tooling prerequisites

Theme localization makes use of the babel project for generation and compilation of localization files. Custom commands are available from the MkDocs' `setup.py` script as described below to assist with the process of updating and contributing translations. You will need to be working from the git working tree on your local machine to make use of the helper scripts.

See the Contributing Guide for direction on how to Install for Development and Submit a Pull Request. The instructions in this document assume that you are working from a properly configured development environment.

Make sure translation requirements are installed in your environment:

```
pip install mkdocs[i18n]
```

4.3.2. Adding language translations to themes

If your favorite language locale is not yet supported on one (or both) of the built-in themes (`mkdocs` and `readthedocs`), you can easily contribute a translation by following the steps below.

Here is a quick summary of what you'll need to do:

1. Initialize new localization catalogs for your language (if a translation for your locale already exists, follow the instructions for updating theme localization files instead).
2. Add a translation for every text placeholder in the localized catalogs.
3. Locally serve and test the translated themes for your language.
4. Update the documentation about supported translations for each translated theme.
5. Contribute your translation through a Pull Request.

> **Note**
>
> Translation locales are usually identified using the ISO-639-1 (2-letter) language codes. While territory/region/county codes are also supported, location specific translations should only be added after the general language translation has been completed and the regional dialect requires use of a term which differs from the general language translation.

4.3.2.1. Initializing the localization catalogs

The templates for each theme contain text placeholders that have been extracted into a Portable Object Template (`messages.pot`) file, which is present in each theme's folder.

Initializing a catalog consists of running a command which will create a directory structure for your desired language and prepare a Portable Object (`messages.po`) file derived from the `pot` file of the theme.

Use the `init_catalog` command on each theme (`-t <theme>`) and provide the appropriate language code (`-l <language>`). For example, to add a translation for the Spanish `es` language to the `mkdocs` theme, run the following command:

```
python setup.py init_catalog -t mkdocs -l es
```

The above command will create the following file structure:

```
mkdocs/themes/mkdocs/locales
├── es
│   └── LC_MESSAGES
│       └── messages.po
```

You can now move on to the next step and add a translation for every text placeholder in the localized catalog.

4.3.3. Updating a theme translation

If a theme's `messages.pot` template file has been updated since the `messages.po` was last updated for your locale, follow the steps below to update the theme's `messages.po` file:

1. Update the theme's translation catalog to refresh the translatable text placeholders of each theme.
2. Translate the newly added translatable text placeholders on every `messages.po` catalog file language you can.
3. Locally serve and test the translated themes for your language.
4. Contribute your translation through a Pull Request.

4.3.3.1. Updating the translation catalogs

This step should be completed after a theme template have been updated for each language that you are comfortable contributing a translation for.

To update the `fr` translation catalog of the `mkdocs` theme, use the following command:

```
python setup.py update_catalog -t mkdocs -l fr
```

You can now move on to the next step and add a translation for every updated text placeholder in the localized catalog.

4.3.3.2. Translating the MkDocs themes

Now that your localized `messages.po` files are ready, all you need to do is add a translation in each `msgstr` item for each `msgid` item in the file.

```
msgid "Next"
msgstr "Siguiente"
```

> **Warning**
>
> Do not modify the `msgid` as it is common to all translations. Just add its translation in the `msgstr` item.

Once you have finished translating all of the terms listed in the `po` file, you'll want to test your localized theme.

4.3.3.3. Testing theme translations

To test a theme with translations, you need to first compile the `messages.po` files of your theme into `messages.mo` files. The following command will compile the `es` translation for the `mkdocs` theme.

```
python setup.py compile_catalog -t mkdocs -l es
```

The above command results in the following file structure:

```
mkdocs/themes/mkdocs/locales
├── es
│   └── LC_MESSAGES
│       ├── messages.mo
│       └── messages.po
```

Note that the compiled `messages.mo` file was generated based on the `messages.po` file that you just edited.

Then modify the `mkdocs.yml` file at the root of the project to test the new and/or updated locale:

```
theme:
    name: mkdocs
    locale: es
```

Finally, run `mkdocs serve` to check out your new localized version of the theme.

> **Note**
>
> The build and release process takes care of compiling and distributing all locales to end users so you only have to worry about contributing the actual text translation `messages.po` files (the rest is ignored by git).
>
> After you have finished testing your work, be sure to undo the change to the `locale` setting in the `mkdocs.yml` file before submitting your changes.

4.3.4. Updating theme documentation

Update the lists of supported translations for each translated theme located at Choosing your theme (`docs/user-guide/choosing-your-theme.md`), in their `locale` options.

4.3.5. Contributing translations

It is now time for you to contribute your nice work to the project. Thank you!

4.4. MkDocs Plugins

A Guide to installing, using and creating MkDocs Plugins

4.4.1. Installing Plugins

Before a plugin can be used, it must be installed on the system. If you are using a plugin which comes with MkDocs, then it was installed when you installed MkDocs. However, to install third party plugins, you need to determine the appropriate package name and install it using `pip`:

```
pip install mkdocs-foo-plugin
```

Once a plugin has been successfully installed, it is ready to use. It just needs to be enabled in the configuration file. The MkDocs Plugins wiki page has a growing list of plugins that you can install and use.

4.4.2. Using Plugins

The `plugins` configuration option should contain a list of plugins to use when building the site. Each "plugin" must be a string name assigned to the plugin (see the documentation for a given plugin to determine its "name"). A plugin listed here must already be installed.

```
plugins:
    - search
```

Some plugins may provide configuration options of their own. If you would like to set any configuration options, then you can nest a key/value mapping (`option_name: option value`) of any options that a given plugin supports. Note that a colon (`:`) must follow the plugin name and then on a new line the option name and value must be indented and separated by a colon. If you would like to define multiple options for a single plugin, each option must be defined on a separate line.

```
plugins:
    - search:
        lang: en
        foo: bar
```

For information regarding the configuration options available for a given plugin, see that plugin's documentation.

For a list of default plugins and how to override them, see the configuration documentation.

4.4.3. Developing Plugins

Like MkDocs, plugins must be written in Python. It is generally expected that each plugin would be distributed as a separate Python module, although it is possible to define multiple plugins in the same module. At a minimum, a MkDocs Plugin must consist of a BasePlugin subclass and an entry point which points to it.

4.4.3.1. BasePlugin

A subclass of `mkdocs.plugins.BasePlugin` should define the behavior of the plugin. The class generally consists of actions to perform on specific events in the build process as well as a configuration scheme for the plugin.

All `BasePlugin` subclasses contain the following attributes:

config_scheme

A tuple of configuration validation instances. Each item must consist of a two item tuple in which the first item is the string name of the configuration option and the second item is an instance of `mkdocs.config.config_options.BaseConfigOption` or any of its subclasses.

For example, the following `config_scheme` defines three configuration options: `foo`, which accepts a string; `bar`, which accepts an integer; and `baz`, which accepts a boolean value.

```
class MyPlugin(mkdocs.plugins.BasePlugin):
    config_scheme = (
        ('foo', mkdocs.config.config_options.Type(str, default='a default value')),
        ('bar', mkdocs.config.config_options.Type(int, default=0)),
        ('baz', mkdocs.config.config_options.Type(bool, default=True))
    )
```

When the user's configuration is loaded, the above scheme will be used to validate the configuration and fill in any defaults for settings not provided by the user. The validation classes may be any of the classes provided in `mkdocs.config.config_options` or a third party subclass defined in the plugin.

Any settings provided by the user which fail validation or are not defined in the `config_scheme` will raise a `mkdocs.config.base.ValidationError`.

config

A dictionary of configuration options for the plugin, which is populated by the `load_config` method after configuration validation has completed. Use this attribute to access options provided by the user.

```
def on_pre_build(self, config):
    if self.config['bool_option']:
        # implement "bool_option" functionality here...
```

All `BasePlugin` subclasses contain the following method(s):

load_config(options)

Loads configuration from a dictionary of options. Returns a tuple of `(errors, warnings)`. This method is called by MkDocs during configuration validation and should not need to be called by the plugin.

on_<event_name>()

Optional methods which define the behavior for specific events. The plugin should define its behavior within these methods. Replace `<event_name>` with the actual name of the event. For example, the `pre_build` event would be defined in the `on_pre_build` method.

Most events accept one positional argument and various keyword arguments. It is generally expected that the positional argument would be modified (or replaced) by the plugin and returned. If nothing is returned (the method returns `None`), then the original, unmodified object is used. The keyword arguments are simply provided to give context and/or supply data which may be used to determine how the positional argument should be modified. It is good practice to accept keyword arguments as `**kwargs`. In the event that additional keywords are provided to an event in a future version of MkDocs, there will be no need to alter your plugin.

For example, the following event would add an additional static_template to the theme config:

```
class MyPlugin(BasePlugin):
    def on_config(self, config, **kwargs):
        config['theme'].static_templates.add('my_template.html')
        return config
```

4.4.3.2. Events

There are three kinds of events: Global Events, Page Events and Template Events.

Global Events

Global events are called once per build at either the beginning or end of the build process. Any changes made in these events will have a global effect on the entire site.

on_serve

The `serve` event is only called when the `serve` command is used during development. It is passed the `Server` instance which can be modified before it is activated. For example, additional files or directories could be added to the list of "watched" files for auto-reloading.

- **Parameters:**

server: `livereload.Server` instance
config: global configuration object
builder: a callable which gets passed to each call to `server.watch`

- **Returns:**

`livereload.Server` instance

on_config

The `config` event is the first event called on build and is run immediately after the user configuration is loaded and validated. Any alterations to the config should be made here.

- **Parameters:**

config: global configuration object

- **Returns:**

global configuration object

on_pre_build

The `pre_build` event does not alter any variables. Use this event to call pre-build scripts.

- **Parameters:**

config: global configuration object

on_files

The `files` event is called after the files collection is populated from the `docs_dir`. Use this event to add, remove, or alter files in the collection. Note that Page objects have not yet been associated with the file objects in the collection. Use Page Events to manipulate page specific data.

- **Parameters:**

files: global files collection
config: global configuration object

- **Returns:**
global files collection

on_nav

The `nav` event is called after the site navigation is created and can be used to alter the site navigation.

- **Parameters:**
nav: global navigation object
config: global configuration object
files: global files collection

- **Returns:**
global navigation object

on_env

The `env` event is called after the Jinja template environment is created and can be used to alter the Jinja environment.

- **Parameters:**
env: global Jinja environment
config: global configuration object
files: global files collection

- **Returns:**
global Jinja Environment

on_post_build

The `post_build` event does not alter any variables. Use this event to call post-build scripts.

- **Parameters:**
config: global configuration object

on_build_error

The `build_error` event is called after an exception of any kind is caught by MkDocs during the build process. Use this event to clean things up before MkDocs terminates. Note that any other events which were scheduled to run after the error will have been skipped. See Handling Errors for more details.

- **Parameters:**
error: exception raised

Template Events

Template events are called once for each non-page template. Each template event will be called for each template defined in the extra_templates config setting as well as any static_templates defined in the theme. All template events are called after the env event and before any page events.

on_pre_template

The `pre_template` event is called immediately after the subject template is loaded and can be used to alter the template.

- **Parameters:**

template: a Jinja2 Template object
template_name: string filename of template
config: global configuration object

- **Returns:**

a Jinja2 Template object

on_template_context

The `template_context` event is called immediately after the context is created for the subject template and can be used to alter the context for that specific template only.

- **Parameters:**

context: dict of template context variables
template_name: string filename of template
config: global configuration object

- **Returns:**

dict of template context variables

on_post_template

The `post_template` event is called after the template is rendered, but before it is written to disc and can be used to alter the output of the template. If an empty string is returned, the template is skipped and nothing is is written to disc.

- **Parameters:**

output_content: output of rendered template as string
template_name: string filename of template
config: global configuration object

- **Returns:**

output of rendered template as string

Page Events

Page events are called once for each Markdown page included in the site. All page events are called after the post_template event and before the post_build event.

on_pre_page

The `pre_page` event is called before any actions are taken on the subject page and can be used to alter the `Page` instance.

- **Parameters:**

page: `mkdocs.nav.Page` instance
config: global configuration object
files: global files collection

- **Returns:**

`mkdocs.nav.Page` instance

on_page_read_source

The `on_page_read_source` event can replace the default mechanism to read the contents of a page's source from the filesystem.

- **Parameters:**

page: `mkdocs.nav.Page` instance
config: global configuration object

- **Returns:**

The raw source for a page as unicode string. If `None` is returned, the default loading from a file will be performed.

on_page_markdown

The `page_markdown` event is called after the page's markdown is loaded from file and can be used to alter the Markdown source text. The meta- data has been stripped off and is available as `page.meta` at this point.

- **Parameters:**

markdown: Markdown source text of page as string
page: `mkdocs.nav.Page` instance
config: global configuration object
files: global files collection

- **Returns:**

Markdown source text of page as string

on_page_content

The `page_content` event is called after the Markdown text is rendered to HTML (but before being passed to a template) and can be used to alter the HTML body of the page.

- **Parameters:**

html: HTML rendered from Markdown source as string
page: `mkdocs.nav.Page` instance
config: global configuration object
files: global files collection

- **Returns:**

HTML rendered from Markdown source as string

on_page_context

The `page_context` event is called after the context for a page is created and can be used to alter the context for that specific page only.

- **Parameters:**

context: dict of template context variables
page: `mkdocs.nav.Page` instance
config: global configuration object
nav: global navigation object

- **Returns:**

dict of template context variables

on_post_page

The `post_page` event is called after the template is rendered, but before it is written to disc and can be used to alter the output of the page. If an empty string is returned, the page is skipped and nothing is written to disc.

- **Parameters:**

output: output of rendered template as string
page: `mkdocs.nav.Page` instance
config: global configuration object

- **Returns:**

output of rendered template as string

4.4.3.3. Handling Errors

MkDocs defines four error types:

`mkdocs.exceptions.MkDocsException`

The base class which all MkDocs exceptions inherit from. This should not be raised directly. One of the subclasses should be raised instead.

`mkdocs.exceptions.ConfigurationError`

This error is raised by configuration validation when a validation error is encountered. This error should be raised by any configuration options defined in a plugin's config_scheme.

`mkdocs.exceptions.BuildError`

This error may be raised by MkDocs during the build process. Plugins should not raise this error.

`mkdocs.exceptions.PluginError`

A subclass of `mkdocs.exceptions.BuildError` which can be raised by plugin events.

Unexpected and uncaught exceptions will interrupt the build process and produce typical Python tracebacks, which are useful for debugging your code. However, users generally find tracebacks overwhelming and often miss the helpful error message. Therefore, MkDocs will catch any of the errors listed above, retrieve the error message, and exit immediately with only the helpful message displayed to the user.

Therefore, you might want to catch any exceptions within your plugin and raise a `PluginError`, passing in your own custom-crafted message, so that the build process is aborted with a helpful message.

The on_build_error event will be triggered for any exception.

For example:

```
from mkdocs.exceptions import PluginError
from mkdocs.plugins import BasePlugin


class MyPlugin(BasePlugin):
    def on_post_page(self, output, page, config, **kwargs):
        try:
            # some code that could throw a KeyError
            ...
        except KeyError as error:
            raise PluginError(str(error))

    def on_build_error(self, error):
```

```
        # some code to clean things up
        ...
```

4.4.3.4. Entry Point

Plugins need to be packaged as Python libraries (distributed on PyPI separate from MkDocs) and each must register as a Plugin via a setuptools `entry_points`. Add the following to your `setup.py` script:

```
entry_points={
    'mkdocs.plugins': [
        'pluginname = path.to.some_plugin:SomePluginClass',
    ]
}
```

The `pluginname` would be the name used by users (in the config file) and `path.to.some_plugin:SomePluginClass` would be the importable plugin itself (`from path.to.some_plugin import SomePluginClass`) where `SomePluginClass` is a subclass of BasePlugin which defines the plugin behavior. Naturally, multiple Plugin classes could exist in the same module. Simply define each as a separate entry point.

```
entry_points={
    'mkdocs.plugins': [
        'featureA = path.to.my_plugins:PluginA',
        'featureB = path.to.my_plugins:PluginB'
    ]
}
```

Note that registering a plugin does not activate it. The user still needs to tell MkDocs to use it via the config.

5. About

5.1. Release Notes

5.1.1. Upgrading

To upgrade MkDocs to the latest version, use pip:

```
pip install -U mkdocs
```

You can determine your currently installed version using `mkdocs --version`:

```
$ mkdocs --version
mkdocs, version 1.0 from /path/to/mkdocs (Python 3.6)
```

5.1.2. Maintenance team

The current and past members of the MkDocs team.

- @tomchristie
- @d0ugal
- @waylan
- @oprypin
- @ultrabug

5.1.3. Version 1.2.3 (2021-10-12)

- Built-in themes now also support these languages:
  - Simplified Chinese (#2497)
  - Japanese (#2525)
  - Brazilian Portuguese (#2535)
  - Spanish (#2545, previously #2396)
- Third-party plugins will take precedence over built-in plugins with the same name (#2591)
- Bugfix: Fix ability to load translations for some languages: core support (#2565) and search plugin support with fallbacks (#2602)
- Bugfix (regression in 1.2): Prevent directory traversal in the dev server (#2604)

- Bugfix (regression in 1.2): Prevent webserver warnings from being treated as a build failure in strict mode (#2607)
- Bugfix: Correctly print colorful messages in the terminal on Windows (#2606)
- Bugfix: Python version 3.10 was displayed incorrectly in `--version` (#2618)

Other small improvements; see commit log.

5.1.4. Version 1.2.2 (2021-07-18)

- Bugfix (regression in 1.2): Fix serving files/paths with Unicode characters (#2464)
- Bugfix (regression in 1.2): Revert livereload file watching to use polling observer (#2477)

  This had to be done to reasonably support usages that span virtual filesystems such as non-native Docker and network mounts.

  This goes back to the polling approach, very similar to that was always used prior, meaning most of the same downsides with latency and CPU usage.
- Revert from 1.2: Remove the requirement of a `site_url` config and the restriction on `use_directory_urls` (#2490)
- Bugfix (regression in 1.2): Don't require trailing slash in the URL when serving a directory index in `mkdocs serve` server (#2507)

  Instead of showing a 404 error, detect if it's a directory and redirect to a path with a trailing slash added, like before.
- Bugfix: Fix `gh_deploy` with config-file in the current directory (#2481)
- Bugfix: Fix reversed breadcrumbs in "readthedocs" theme (#2179)
- Allow "mkdocs.yaml" as the file name when '--config' is not passed (#2478)
- Stop treating ";" as a special character in URLs: urlparse -> urlsplit (#2502)
- Improve build performance for sites with many pages (partly already done in 1.2) (#2407)

5.1.5. Version 1.2.1 (2021-06-09)

- Bugfix (regression in 1.2): Ensure 'gh-deploy' always pushes.

5.1.6. Version 1.2 (2021-06-04)

5.1.6.1. Major Additions to Version 1.2

Support added for Theme Localization (#2299)

The `mkdocs` and `readthedocs` themes now support language localization using the `theme.locale` parameter, which defaults to `en` (English). The only other supported languages in this release are `fr` (French) and `es` (Spanish). For details on using the provided translations, see the user guide. Note that translation will not happen by default. Users must first install the necessary dependencies with the following command:

```
pip install mkdocs[i18n]
```

Translation contributions are welcome and detailed in the Translation Guide.

Developers of third party themes may want to review the relevant section of the Theme Development Guide.

Contributors who are updating the templates to the built-in themes should review the Contributing Guide.

The `lang` setting of the `search` plugin now defaults to the language specified in `theme.locale`.

Support added for Environment Variables in the configuration file (#1954)

Environments variables may now be specified in the configuration file with the `!ENV` tag. The value of the variable will be parsed by the YAML parser and converted to the appropriate type.

```
somekey: !ENV VAR_NAME
otherkey: !ENV [VAR_NAME, FALLBACK_VAR, 'default value']
```

See Environment Variables in the Configuration documentation for details.

Support added for Configuration Inheritance (#2218)

A configuration file may now inherit from a parent configuration file. In the primary file set the `INHERIT` key to the relative path of the parent file.

```
INHERIT: path/to/base.yml
```

The two files will then be deep merged. See Configuration Inheritance for details.

Update `gh-deploy` command (#2170)

The vendored (and modified) copy of ghp_import has been replaced with a dependency on the upstream library. As of version 1.0.0, ghp-import includes a Python API which makes it possible to call directly.

MkDocs can now benefit from recent bug fixes and new features, including the following:

- A `.nojekyll` file is automatically included when deploying to GitHub Pages.
- The `--shell` flag is now available, which reportedly works better on Windows.
- Git author and committer environment variables should be respected (#1383).

Rework auto-reload and HTTP server for `mkdocs serve` (#2385)

`mkdocs serve` now uses a new underlying server + file watcher implementation, based on http.server from standard library and watchdog. It provides similar functionality to the previously used livereload library (which is now dropped from dependencies, along with tornado).

This makes reloads more responsive and consistent in terms of timing. Multiple rapid file changes no longer cause the site to repeatedly rebuild (issue #2061).

Almost every aspect of the server is slightly different, but actual visible changes are minor. The logging outputs are only similar to the old ones. Degradations in behavior are not expected, and should be reported if found.

Offset the local site root according to the sub-path of the `site_url` (#2424)

When using `mkdocs serve` and having the `site_url` specified as e.g. `http://example.org/sub/path/`, now the root of the locally served site becomes `http://127.0.0.1:8000/sub/path/` and all document paths are offset accordingly.

A `build_error` event was added (#2103)

Plugin developers can now use the `on_build_error` hook to execute code when an exception is raised while building the site.

See `on_build_error` in the Plugins documentation for details.

Three new exceptions: BuildError PluginError and Abort (#2103)

MkDocs now has tree new exceptions defined in `mkdocs.exceptions`: `BuildError`, `PluginError`, and `Abort`:

- `PluginError` can be raised from a plugin to stop the build and log an error message without traceback.

- `BuildError` should not be used by third-party plugins developers and is reserved for internal use only.
- `Abort` is used internally to abort the build and display an error without a traceback.

See `Handling errors` in the Plugins documentation for details.

Search Indexing Strategy configuration

Users can now specify which strategy they wish to use when indexing their site for search. A user can select between the following options:

- **full**: Adds page title, section headings, and full page text to the search index.
- **sections**: Adds page titles and section headings only to the search index.
- **titles**: Adds only the page titles to the search index.

See `Search Indexing` in the configuration documentation for details.

5.1.6.2. Backward Incompatible Changes in 1.2

- The site_url configuration option is now **required**. If it is not set, a warning will be issued. In a future release an error will be raised (#2189).

  The use_directory_urls configuration option will be forced to `false` if site_url is set to an empty string. In that case, if `use_directory_urls` is not explicitly set to `false`, a warning will be issued (#2189).

  > **Note**
  >
  > This was reverted in release 1.2.2

- The `google_analytics` configuration option is deprecated as Google appears to be phasing it out in favor of its new Google Analytics 4 property. See the documentation for your theme for alternatives which can be configured as part of your theme configuration. For example, the mkdocs and readthedocs themes have each added a new `theme.analytics.gtag` configuration option which uses the new Google Analytics 4 property. See Google's documentation on how to Upgrade to a Google Analytics 4 property. Then set `theme.analytics.gtag` to the "G-" ID and delete the `google_analytics` configuration option which contains a "UA-" ID. So long as the old "UA-" ID and new "G-" ID are properly linked in your Google account, and you are using the "G-" ID, the data will be made available in both the old and new formats by Google Analytics. See #2252.
- A theme's files are now excluded from the list of watched files by default when using the `--livereload` server. This new default behavior is what most users need and provides better performance when editing site content. Theme developers can enable the old behavior with the `--watch-theme` option. (#2092).

- The `mkdocs` theme now removes the sidebar when printing a page. This frees up horizontal space for better rendering of content like tables (#2193).
- The `mkdocs.config.DEFAULT_SCHEMA` global variable has been replaced with the function `mkdocs.config.defaults.get_schema()`, which ensures that each instance of the configuration is unique (#2289).
- The `mkdocs.utils.warning_filter` is deprecated and now does nothing. Plugins should remove any reference to is as it may be deleted in a future release. To ensure any warnings get counted, simply log them to the `mkdocs` log (i.e.: `mkdocs.plugins.pluginname`).
- The `on_serve` event (which receives the `server` object and the `builder` function) is affected by the server rewrite. `server` is now a `mkdocs.livereload.LiveReloadServer` instead of `livereload.server.Server`. The typical action that plugins can do with these is to call `server.watch(some_dir, builder)`, which basically adds that directory to watched directories, causing the site to be rebuilt on file changes. That still works, but passing any other function to `watch` is deprecated and shows a warning. This 2nd parameter is already optional, and will accept only this exact `builder` function just for compatibility.
- The `python` method of the `plugins.search.prebuild_index` configuration option is pending deprecation as of version 1.2. It is expected that in version 1.3 it will raise a warning if used and in version 1.4 it will raise an error. Users are encouraged to use an alternate method to generate a prebuilt index for search.
- The `lunr` and `lunr[languages]` dependencies are no longer installed by default. The dependencies are only needed for the rare user who pre-builds the search index and uses the `python` option, which is now pending deprecation. If you use this feature, then you will need to manually install `lunr` and `lunr[languages]`. A warning is issued if the dependencies are needed but not installed.

5.1.6.3. Other Changes and Additions to Version 1.2

- Bugfix: Properly process navigation child items in `_get_by_type` when filtering for sections (#2203).
- Official support for Python 3.9 has been added and support for Python 3.5 has been dropped.
- Bugfix: Fixes an issue that would result in a partially cut-off navigation item in the ReadTheDocs theme (#2297).
- Structure Files object now has a `remove` method to help plugin developers manipulate the Files tree. The corresponding `src_paths` has become a property to accommodate this possible dynamic behavior. See #2305.
- Updated highlight.js to 10.5.0. See #2313.
- Bugfix: Search plugin now works with Japanese language. See #2178.

- Documentation has been refactored (#1629).
- Restore styling of tables in the `readthedocs` theme (#2028).
- Ensure `site_url` ends with a slash (#1785).
- Correct documentation of `pages` template context variable (#1736).
- The `lunr` dependency has been updated to 0.5.9, and `lunr.js` to the corresponding 2.3.9 version (#2306).
- Color is now used in log messages to identify errors, warnings and debug messages.
- Bugfix: Identify homepage when `use_directory_urls` is `False` (#2362).

5.1.7. Version 1.1.2 (2020-05-14)

- Bugfix: Normalize IP addresses and change unsupported address error to a warning (#2108).

5.1.8. Version 1.1.1 (2020-05-12)

- Bugfix: Allow compressed sitemap to be deterministic by supporting the `SOURCE_DATE_EPOCH` environment variable (#2100).
- Bugfix: Use `README.md` as `index.html` even if `use_directory_urls` is false (#2081).
- Bugfix: Ignore links which start with a backslash (#1680).
- Bugfix: Pass `builder` to the `on_serve` event so that it can be passed to `server.watch` by plugins (#1952).
- Bugfix: Use `lunr[languages]==0.5.8` to avoid `nltk` incompatibilities (#2062).
- Bugfix: Ensure wheel is Python 3 only (#2021).
- Bugfix: Clean up `dev_addr` validation and disallow `0.0.0.0` (#2022).
- Add support for `min_search_length` parameter for search plugin (#2014).
- Bugfix: `readthedocs` theme `code` colors (#2027).

5.1.9. Version 1.1 (2020-02-22)

5.1.9.1. Major Additions to Version 1.1

Support for Lunr.py as `prebuild_index` engine

Mkdocs now supports pre-building indices using Lunr.py, a pure Python implementation of Lunr.js, allowing the user to avoid installing a NodeJS environment if so desired. For more information please read the `prebuild_index` documentation.

`readthedocs` theme updated with upstream (#588 and #1374)

The `readthedocs` theme now more closely matches the upstream Sphinx theme (version 0.4.1). A number of new theme configuration settings were added which mirror the upstream configuration options. See the theme documentation for details.

Update `mkdocs` theme to Bootswatch 4.1.3 (#1563)

The `mkdocs` theme now supports all the features of Bootswatch 4.1. Additionally, 2 filenames were changed in this update. If you are using a theme which inherits from the `mkdocs` theme, the theme developer may need to update these filenames as follows.

```
css/bootstrap-custom.min.css => css/bootstrap.min.css
js/bootstrap-3.0.3.min.js => js/bootstrap.min.js
```

Improved configuration support on the command line (#1401)

The `build`, `serve`, and `gh-deploy` subcommands now support flags to control whether directory URLs should be created: `--use-directory-urls` / `--no-directory-urls`. In addition, the `gh-deploy` subcommand now supports all the configuration options that `build` and `serve` do, adding `--strict`, `--theme`, `--theme-dir`, and `--site-dir`.

Updated lunr-languages support (#1729)

The `lunr-languages` plugin has been updated to 1.4.0, adding support for Arabic (`ar`) and Vietnamese (`vi`) languages. In addition, the Dutch and Japanese language codes have been changed to their standard values: `nl` and `ja`, respectively. The old language codes (`du` and `jp`) remain as aliases but may be removed in a future version of MkDocs.

5.1.9.2. Other Changes and Additions to Version 1.1

- Bugfix: Ensure nested dot files in themes are ignored and document behavior (#1981).
- Update minimum dependency to Markdown 3.2.1.
- Updated minimum dependency to Jinja 2.10.1 to address security concerns (#1780).
- Update to lunr.js 2.3.8 (#1989).
- Add support for Python 3.8.
- Drop support for Python 3.4.
- Drop support for Python 2.7. MkDocs is PY3 only now (#1926).
- Bugfix: Select appropriate asyncio event loop on Windows for Python 3.8+ (#1885).
- Bugfix: Ensure nested index pages do not get identified as the homepage (#1919).

- Bugfix: Properly identify deployment version (#1879).
- Bugfix: Properly build `ValidationError` message for `custom_dir` (#1849).
- Bugfix: Exclude Markdown files and READMEs from theme (#1766).
- Bugfix: Account for encoded URLs (#1670).
- Bugfix: Ensure theme files do not override `docs_dir` files (#1671).
- Bugfix: Do not normalize URL fragments (#1655).
- Bugfix: Skip external URLs in sitemap.xml (#1742).
- Bugfix: Ensure theme files do not override docs_dir files on Windows (#1876)
- Add canonical tag to `readthedocs` theme (#1669).
- Improved error message for when `git` is not available.
- Add support for `nav_style` theme option for the `mkdocs` theme (#1930).
- Bugfix: Long/nested dropdowns now behave more consistently for the `mkdocs` theme (#1234).
- Bugfix: Multi-row nav headers in the `mkdocs` theme no longer obscure the document content (#716).
- Add support for `navigation_depth` theme option for the `mkdocs` theme (#1970).
- `level` attribute in `page.toc` items is now 1-indexed to match the level in `<hN>` tags (#1970).

5.1.10. Version 1.0.4 (2018-09-07)

- Bugfix: Ignore absolute links in Markdown (#1621).

5.1.11. Version 1.0.3 (2018-08-29)

- Bugfix: Warn on relative paths in navigation (#1604).
- Bugfix: Handle empty `theme_config.yml` files correctly (#1602).

5.1.12. Version 1.0.2 (2018-08-22)

- Bugfix: Provide absolute `base_url` to error templates (#1598).

5.1.13. Version 1.0.1 (2018-08-13)

- Bugfix: Prevent page reload when [Enter] is pressed in search box (#1589).
- Bugfix: Avoid calling `search` until all assets are ready (#1584).
- Bugfix: Exclude `README.md` if `index.md` is present (#1580).

- Bugfix: Fix `readthedocs` theme navigation bug with homepage (#1576).

5.1.14. Version 1.0 (2018-08-03)

5.1.14.1. Major Additions to Version 1.0

Internal Refactor of Pages, Files, and Navigation

Internal handling of pages, files and navigation has been completely refactored. The changes included in the refactor are summarized below.

- Support for hidden pages. All Markdown pages are now included in the build regardless of whether they are included in the navigation configuration (#699).
- The navigation can now include links to external sites (#989 #1373 & #1406).
- Page data (including titles) is properly determined for all pages before any page is rendered (#1347).
- Automatically populated navigation now sorts index pages to the top. In other words, The index page will be listed as the first child of a directory, while all other documents are sorted alphanumerically by file name after the index page (#73 & #1042).
- A `README.md` file is now treated as an index file within a directory and will be rendered to `index.html` (#608).
- The URLs for all files are computed once and stored in a files collection. This ensures all internal links are always computed correctly regardless of the configuration. This also allows all internal links to be validated, not just links to other Markdown pages. (#842 & #872).
- A new url template filter smartly ensures all URLs are relative to the current page (#1526).
- An on_files plugin event has been added, which could be used to include files not in the `docs_dir`, exclude files, redefine page URLs (i.e. implement extensionless URLs), or to manipulate files in various other ways.

Backward Incompatible Changes

As part of the internal refactor, a number of backward incompatible changes have been introduced, which are summarized below.

URLs have changed when `use_directory_urls` is `False`

Previously, all Markdown pages would be have their filenames altered to be index pages regardless of how the use_directory_urls setting was configured. However, the path munging is only needed when `use_directory_urls` is set to `True` (the default). The path mangling no longer happens when `use_directory_urls` is set to `False`, which will result in different URLs for all pages that

were not already index files. As this behavior only effects a non-default configuration, and the most common user-case for setting the option to `False` is for local file system (`file://`) browsing, its not likely to effect most users. However, if you have `use_directory_urls` set to `False` for a MkDocs site hosted on a web server, most of your URLs will now be broken. As you can see below, the new URLs are much more sensible.

Markdown file	Old URL	New URL
index.md	index.html	index.html
foo.md	foo/index.html	foo.html
foo/bar.md	foo/bar/index.html	foo/bar.html

Table 1 - Old and New ULR

Note that there has been no change to URLs or file paths when `use_directory_urls` is set to `True` (the default), except that MkDocs more consistently includes an ending slash on all internally generated URLs.

The `pages` configuration setting has been renamed to `nav`

The `pages` configuration setting is deprecated and will issue a warning if set in the configuration file. The setting has been renamed `nav` . To update your configuration, simply rename the setting to `nav` . In other words, if your configuration looked like this:

```
pages:
    - Home: index.md
    - User Guide: user-guide.md
```

Simply edit the configuration as follows:

```
nav:
    - Home: index.md
    - User Guide: user-guide.md
```

Listing 14 - `pages` become `nav`

In the current release, any configuration which includes a `pages` setting, but no `nav` setting, the `pages` configuration will be copied to `nav` and a warning will be issued. However, in a future release, that may no longer happen. If both `pages` and `nav` are defined, the `pages` setting will be ignored.

Template variables and `base_url`

In previous versions of MkDocs some URLs expected the base_url template variable to be prepended to the URL and others did not. That inconsistency has been removed in that no URLs are modified before being added to the template context.

For example, a theme template might have previously included a link to the `site_name` as:

```
<a href="{{ nav.homepage.url }}">{{ config.site_name }}</a>
```

And MkDocs would magically return a URL for the homepage which was relative to the current page. That "magic" has been removed and the url template filter should be used:

```
<a href="{{ nav.homepage.url|url }}">{{ config.site_name }}</a>
```

This change applies to any navigation items and pages, as well as the `page.next_page` and `page.previous_page` attributes. For the time being, the `extra_javascript` and `extra_css` variables continue to work as previously (without the `url` template filter), but they have been deprecated and the corresponding configuration values (`config.extra_javascript` and `config.extra_css` respectively) should be used with the filter instead.

```
{% for path in config['extra_css'] %}
    <link href="{{ path|url }}" rel="stylesheet">
{% endfor %}
```

Note that navigation can now include links to external sites. Obviously, the `base_url` should not be prepended to these items. However, the `url` template filter is smart enough to recognize the URL is absolute and does not alter it. Therefore, all navigation items can be passed to the filter and only those that need to will be altered.

```
{% for nav_item in nav %}
    <a href="{{ nav_item.url|url }}">{{ nav_item.title }}</a>
{% endfor %}
```

Path Based Settings are Relative to Configuration File (#543)

Previously any relative paths in the various configuration options were resolved relative to the current working directory. They are now resolved relative to the configuration file. As the documentation has always encouraged running the various MkDocs commands from the directory that contains the configuration file (project root), this change will not affect most users. However, it

will make it much easier to implement automated builds or otherwise run commands from a location other than the project root.

Simply use the `-f/--config-file` option and point it at the configuration file:

```
mkdocs build --config-file /path/to/my/config/file.yml
```

As previously, if no file is specified, MkDocs looks for a file named `mkdocs.yml` in the current working directory.

Added support for YAML Meta-Data (#1542)

Previously, MkDocs only supported MultiMarkdown style meta-data, which does not recognize different data types and is rather limited. MkDocs now also supports YAML style meta-data in Markdown documents. MkDocs relies on the the presence or absence of the deliminators (`---` or `...`) to determine whether YAML style meta-data or MultiMarkdown style meta-data is being used.

Previously MkDocs would recognize MultiMarkdown style meta-data between the deliminators. Now, if the deliminators are detected, but the content between the deliminators is not valid YAML meta-data, MkDocs does not attempt to parse the content as MultiMarkdown style meta-data. Therefore, MultiMarkdown's style meta-data must not include the deliminators. See the MultiMarkdown style meta-data documentation for details.

Prior to version 0.17, MkDocs returned all meta-data values as a list of strings (even a single line would return a list of one string). In version 0.17, that behavior was changed to return each value as a single string (multiple lines were joined), which some users found limiting (see #1471). That behavior continues for MultiMarkdown style meta-data in the current version. However, YAML style meta-data supports the full range of "safe" YAML data types. Therefore, it is recommended that any complex meta-data make use of the YAML style (see the YAML style meta-data documentation for details). In fact, a future version of MkDocs may deprecate support for MultiMarkdown style meta-data.

Refactor Search Plugin

The search plugin has been completely refactored to include support for the following features:

- Use a web worker in the browser with a fallback (#1396).
- Optionally pre-build search index locally (#859 & #1061).
- Upgrade to lunr.js 2.x (#1319).
- Support search in languages other than English (#826).
- Allow the user to define the word separators (#867).
- Only run searches for queries of length > 2 (#1127).

- Remove dependency on require.js (#1218).
- Compress the search index (#1128).

Users can review the configuration options available and theme authors should review how search and themes interact.

`theme_dir` Configuration Option fully Deprecated

As of version 0.17, the custom_dir option replaced the deprecated `theme_dir` option. If users had set the `theme_dir` option, MkDocs version 0.17 copied the value to the `theme.custom_dir` option and a warning was issued. As of version 1.0, the value is no longer copied and an error is raised.

5.1.14.2. Other Changes and Additions to Version 1.0

- Keyboard shortcuts changed to not conflict with commonly used accessibility shortcuts (#1502.)
- User friendly YAML parse errors (#1543).
- Officially support Python 3.7.
- A missing theme configuration file now raises an error.
- Empty `extra_css` and `extra_javascript` settings no longer raise a warning.
- Add highlight.js configuration settings to built-in themes (#1284).
- Close search modal when result is selected (#1527).
- Add a level attribute to AnchorLinks (#1272).
- Add MkDocs version check to gh-deploy script (#640).
- Improve Markdown extension error messages. (#782).
- Drop official support for Python 3.3 and set `tornado>=5.0` (#1427).
- Add support for GitLab edit links (#1435).
- Link to GitHub issues from release notes (#644).
- Expand {sha} and {version} in gh-deploy commit message (#1410).
- Compress `sitemap.xml` (#1130).
- Defer loading JS scripts (#1380).
- Add a title attribute to the search input (#1379).
- Update RespondJS to latest version (#1398).
- Always load Google Analytics over HTTPS (#1397).
- Improve scrolling frame rate (#1394).
- Provide more version info. (#1393).

- Refactor `writing-your-docs.md` (#1392).
- Workaround Safari bug when zooming to < 100% (#1389).
- Remove addition of `clicky` class to body and animations. (#1387).
- Prevent search plugin from re-injecting `extra_javascript` files (#1388).
- Refactor `copy_media_files` util function for more flexibility (#1370).
- Remove PyPI Deployment Docs (#1360).
- Update links to Python-Markdown library (#1360).
- Document how to generate manpages for MkDocs commands (#686).

5.1.15. Version 0.17.5 (2018-07-06)

- Bugfix: Fix Python 3.7 and PEP 479 incompatibility (#1518).

5.1.16. Version 0.17.4 (2018-06-08)

- Bugfix: Add multi-level nesting support to sitemap.xml (#1482).

5.1.17. Version 0.17.3 (2018-03-07)

- Bugfix: Set dependency `tornado>=4.1,<5.0` due to changes in 5.0 (#1428).

5.1.18. Version 0.17.2 (2017-11-15)

- Bugfix: Correct `extra_*` config setting regressions (#1335 & #1336).

5.1.19. Version 0.17.1 (2017-10-30)

- Bugfix: Support `repo_url` with missing ending slash. (#1321).
- Bugfix: Add length support to `mkdocs.toc.TableOfContext` (#1325).
- Bugfix: Add some theme specific settings to the search plugin for third party themes (#1316).
- Bugfix: Override `site_url` with `dev_addr` on local server (#1317).

5.1.20. Version 0.17.0 (2017-10-19)

5.1.20.1. Major Additions to Version 0.17.0

Plugin API. (#206)

A new Plugin API has been added to MkDocs which allows users to define their own custom behaviors. See the included documentation for a full explanation of the API.

The previously built-in search functionality has been removed and wrapped in a plugin (named "search") with no changes in behavior. When MkDocs builds, the search index is now written to `search/search_index.json` instead of `mkdocs/search_index.json`. If no plugins setting is defined in the config, then the `search` plugin will be included by default. See the configuration documentation for information on overriding the default.

Theme Customization. (#1164)

Support had been added to provide theme specific customizations. Theme authors can define default options as documented in Theme Configuration. A theme can now inherit from another theme, define various static templates to be rendered, and define arbitrary default variables to control behavior in the templates. The theme configuration is defined in a configuration file named `mkdocs_theme.yml` which should be placed at the root of your template files. A warning will be raised if no configuration file is found and an error will be raised in a future release.

Users can override those defaults under the theme configuration option of their `mkdocs.yml` configuration file, which now accepts nested options. One such nested option is the custom_dir option, which replaces the now deprecated `theme_dir` option. If users had previously set the `theme_dir` option, a warning will be issued, with an error expected in a future release.

If a configuration previously defined a `theme_dir` like this:

```
theme: mkdocs
theme_dir: custom
```

Then the configuration should be adjusted as follows:

```
theme:
    name: mkdocs
    custom_dir: custom
```

See the theme configuration option documentation for details.

Previously deprecated Template variables removed. (#1168)

Page Template

The primary entry point for page templates has been changed from `base.html` to `main.html`. This allows `base.html` to continue to exist while allowing users to override `main.html` and extend `base.html`. For version 0.16, `base.html` continued to work if no `main.html` template existed, but it raised a deprecation warning. In version 1.0, a build will fail if no `main.html` template exists.

Context Variables

Page specific variable names in the template context have been refactored as defined in Custom Themes. The old variable names issued a warning in version 0.16, but have been removed in version 1.0.

Any of the following old page variables should be updated to the new ones in user created and third-party templates:

Old Variable Name	New Variable Name
current_page	page
page_title	page.title
content	page.content
toc	page.toc
meta	page.meta
canonical_url	page.canonical_url
previous_page	page.previous_page
next_page	page.next_page

Additionally, a number of global variables have been altered and/or removed and user created and third-party templates should be updated as outlined below:

Old Variable Name	New Variable Name or Expression
current_page	page

Old Variable Name	New Variable Name or Expression
include_nav	nav|length>1
include_next_prev	(page.next_page or page.previous_page)
site_name	config.site_name
site_author	config.site_author
page_description	config.site_description
repo_url	config.repo_url
repo_name	config.repo_name
site_url	config.site_url
copyright	config.copyright
google_analytics	config.google_analytics
homepage_url	nav.homepage.url
favicon	{{ base_url }}/img/favicon.ico

Auto-Populated extra_css and extra_javascript Fully Deprecated. (#986)

In previous versions of MkDocs, if the `extra_css` or `extra_javascript` config settings were empty, MkDocs would scan the `docs_dir` and auto-populate each setting with all of the CSS and JavaScript files found. On version 0.16 this behavior was deprecated and a warning was issued. In 0.17 any unlisted CSS and JavaScript files will not be included in the HTML templates, however, a warning will be issued. In other words, they will still be copied to the `site-dir`, but they will not have any effect on the theme if they are not explicitly listed.

All CSS and JavaScript files in the `docs_dir` should be explicitly listed in the `extra_css` or `extra_javascript` config settings going forward.

5.1.20.2. Other Changes and Additions to Version 0.17.0

- Add "edit Link" support to MkDocs theme (#1129)
- Open files with `utf-8-sig` to account for BOM (#1186)
- Symbolic links are now followed consistently (#1134)

- Support for keyboard navigation shortcuts added to included themes (#1095)
- Some refactoring and improvements to config_options (#1296)
- Officially added support for Python 3.6 (#1296)
- 404 Error page added to readthedocs theme (#1296))
- Internal refactor of Markdown processing (#713)
- Removed special error message for mkdocs-bootstrap and mkdocs-bootswatch themes (#1168)
- The legacy pages config is no longer supported (#1168)
- The deprecated `json` command has been removed (#481)
- Support for Python 2.6 has been dropped (#165)
- File permissions are no longer copied during build (#1292)
- Support query and fragment strings in `edit_uri` (#1224 & #1273)

5.1.21. Version 0.16.3 (2017-04-04)

- Fix error raised by autoscrolling in the readthedocs theme (#1177)
- Fix a few documentation typos (#1181 & #1185)
- Fix a regression to livereload server introduced in 0.16.2 (#1174)

5.1.22. Version 0.16.2 (2017-03-13)

- System root (`/`) is not a valid path for site_dir or docs_dir (#1161)
- Refactor readthedocs theme navigation (#1155 & #1156)
- Add support to dev server to serve custom error pages (#1040)
- Ensure nav.homepage.url is not blank on error pages (#1131)
- Increase livereload dependency to 2.5.1 (#1106)

5.1.23. Version 0.16.1 (2016-12-22)

- Ensure scrollspy behavior does not affect nav bar (#1094)
- Only "load" a theme when it is explicitly requested by the user (#1105)

5.1.24. Version 0.16 (2016-11-04)

5.1.24.1. Major Additions to Version 0.16.0

Template variables refactored. (#874)

Page Context

Page specific variable names in the template context have been refactored as defined in Custom Themes. The old variable names will issue a warning but continue to work for version 0.16, but may be removed in a future version.

Any of the following old page variables should be updated to the new ones in user created and third-party templates:

Old Variable Name	New Variable Name
current_page	page
page_title	page.title
content	page.content
toc	page.toc
meta	page.meta
canonical_url	page.canonical_url
previous_page	page.previous_page
next_page	page.next_page

Table 2 - Caption of the second table

Global Context

Additionally, a number of global variables have been altered and/or deprecated and user created and third-party templates should be updated as outlined below:

Previously, the global variable `include_nav` was altered programmatically based on the number of pages in the nav. The variable will issue a warning but continue to work for version 0.16, but may be removed in a future version. Use `{% if nav|length>1 %}` instead.

Previously, the global variable `include_next_prev` was altered programmatically based on the number of pages in the nav. The variable will issue a warning but continue to work for version 0.16, but may be removed in a future version. Use `{% if page.next_page or page.previous_page %}` instead.

Previously the global variable `page_description` was altered programmatically based on whether the current page was the homepage. Now it simply maps to `config['site_description']`. Use `{% if page.is_homepage %}` in the template to conditionally change the description.

The global variable `homepage_url` maps directly to `nav.homepage.url` and is being deprecated. The variable will issue a warning but continue to work for version 0.16, but may be removed in a future version. Use `nav.homepage.url` instead.

The global variable `favicon` maps to the configuration setting `site_favicon`. Both the template variable and the configuration setting are being deprecated and will issue a warning but continue to work for version 0.16, and may be removed in a future version. Use `{{ base_url }}/img/favicon.ico` in your template instead. Users can simply save a copy of their custom favicon icon to `img/favicon.ico` in either their `docs_dir` or `theme_dir`.

A number of variables map directly to similarly named variables in the `config`. Those variables are being deprecated and will issue a warning but continue to work for version 0.16, but may be removed in a future version. Use `config.var_name` instead, where `var_name` is the name of one of the configuration variables.

Below is a summary of all of the changes made to the global context:

Old Variable Name	New Variable Name or Expression
current_page	page
include_nav	nav|length>1
include_next_prev	(page.next_page or page.previous_page)
site_name	config.site_name
site_author	config.site_author
page_description	config.site_description
repo_url	config.repo_url
repo_name	config.repo_name

Old Variable Name	New Variable Name or Expression
site_url	config.site_url
copyright	config.copyright
google_analytics	config.google_analytics
homepage_url	nav.homepage.url
favicon	{{ base_url }}/img/favicon.ico

Increased Template Customization. (#607)

The built-in themes have been updated by having each of their many parts wrapped in template blocks which allow each individual block to be easily overridden using the `theme_dir` config setting. Without any new settings, you can use a different analytics service, replace the default search function, or alter the behavior of the navigation, among other things. See the relevant documentation for more details.

To enable this feature, the primary entry point for page templates has been changed from `base.html` to `main.html`. This allows `base.html` to continue to exist while allowing users to override `main.html` and extend `base.html`. For version 0.16, `base.html` will continue to work if no `main.html` template exists, but it is deprecated and will raise a warning. In version 1.0, a build will fail if no `main.html` template exists. Any custom and third party templates should be updated accordingly.

The easiest way for a third party theme to be updated would be to simply add a `main.html` file which only contains the following line:

```
{% extends "base.html" %}
```

That way, the theme contains the `main.html` entry point, and also supports overriding blocks in the same manner as the built-in themes. Third party themes are encouraged to wrap the various pieces of their templates in blocks in order to support such customization.

Auto-Populated `extra_css` and `extra_javascript` Deprecated. (#986)

In previous versions of MkDocs, if the `extra_css` or `extra_javascript` config settings were empty, MkDocs would scan the `docs_dir` and auto-populate each setting with all of the CSS and JavaScript files found. This behavior is deprecated and a warning will be issued. In the next release, the auto-populate feature will stop working and any unlisted CSS and JavaScript files will not be

included in the HTML templates. In other words, they will still be copied to the `site-dir`, but they will not have any effect on the theme if they are not explicitly listed.

All CSS and JavaScript files in the `docs_dir` should be explicitly listed in the `extra_css` or `extra_javascript` config settings going forward.

Support for dirty builds. (#990)

For large sites the build time required to create the pages can become problematic, thus a "dirty" build mode was created. This mode simply compares the modified time of the generated HTML and source markdown. If the markdown has changed since the HTML then the page is re-constructed. Otherwise, the page remains as is. This mode may be invoked in both the `mkdocs serve` and `mkdocs build` commands:

```
mkdocs serve --dirtyreload
```

```
mkdocs build --dirty
```

It is important to note that this method for building the pages is for development of content only, since the navigation and other links do not get updated on other pages.

Stricter Directory Validation

Previously, a warning was issued if the `site_dir` was a child directory of the `docs_dir`. This now raises an error. Additionally, an error is now raised if the `docs_dir` is set to the directory which contains your config file rather than a child directory. You will need to rearrange you directory structure to better conform with the documented layout.

5.1.24.2. Other Changes and Additions to Version 0.16.0

- Bugfix: Support `gh-deploy` command on Windows with Python 3 (#722)
- Bugfix: Include .woff2 font files in Python package build (#894)
- Various updates and improvements to Documentation Home Page/Tutorial (#870)
- Bugfix: Support livereload for config file changes (#735)
- Bugfix: Non-media template files are no longer copied with media files (#807)
- Add a flag (-e/--theme-dir) to specify theme directory with the commands `mkdocs build` and `mkdocs serve` (#832)
- Fixed issues with Unicode file names under Windows and Python 2. (#833)
- Improved the styling of in-line code in the MkDocs theme. (#718)

- Bugfix: convert variables to JSON when being passed to JavaScript (#850)
- Updated the ReadTheDocs theme to match the upstream font sizes and colors more closely. (#857)
- Fixes an issue with permalink markers showing when the mouse was far above them (#843)
- Bugfix: Handle periods in directory name when automatically creating the pages config. (#728)
- Update searching to Lunr 0.7, which comes with some performance enhancements for larger documents (#859)
- Bugfix: Support SOURCE_DATE_EPOCH environment variable for "reproducible" builds (#938)
- Follow links when copying media files (#869).
- Change "Edit on..." links to point directly to the file in the source repository, rather than to the root of the repository (#975), configurable via the new `edit_uri` setting.
- Bugfix: Don't override config value for strict mode if not specified on CLI (#738).
- Add a `--force` flag to the `gh-deploy` command to force the push to the repository (#973).
- Improve alignment for current selected menu item in readthedocs theme (#888).
- `http://user.github.io/repo` => `https://user.github.io/repo/` (#1029).
- Improve installation instructions (#1028).
- Account for wide tables and consistently wrap inline code spans (#834).
- Bugfix: Use absolute URLs in nav & media links from error templates (#77).

5.1.25. Version 0.15.3 (2016-02-18)

- Improve the error message the given theme can't be found.
- Fix an issue with relative symlinks (#639)

5.1.26. Version 0.15.2 (2016-02-08)

- Fix an incorrect warning that states external themes will be removed from MkDocs.

5.1.27. Version 0.15.1 (2016-01-30)

- Lower the minimum supported Click version to 3.3 for package maintainers. (#763)

5.1.28. Version 0.15.0 (2016-01-21)

5.1.28.1. Major Additions to Version 0.15.0

Add support for installable themes

MkDocs now supports themes that are distributed via Python packages. With this addition, the Bootstrap and Bootswatch themes have been moved to external git repositories and python packages. See their individual documentation for more details about these specific themes.

- MkDocs Bootstrap
- MkDocs Bootswatch

They will be included with MkDocs by default until a future release. After that they will be installable with pip: `pip install mkdocs-bootstrap` and `pip install mkdocs-bootswatch`

See the documentation for Customizing Your Theme for more information about using and customizing themes and Custom themes for creating and distributing new themes

5.1.28.2. Other Changes and Additions to Version 0.15.0

- Fix issues when using absolute links to Markdown files. (#628)
- Deprecate support of Python 2.6, pending removal in 1.0.0. (#165)
- Add official support for Python version 3.5.
- Add support for site_description and site_author to the ReadTheDocs theme. (#631)
- Update FontAwesome to 4.5.0. (#789)
- Increase IE support with X-UA-Compatible. (#785)
- Added support for Python's `-m` flag. (#706)
- Bugfix: Ensure consistent ordering of auto-populated pages. (#638)
- Bugfix: Scroll the tables of contents on the MkDocs theme if it is too long for the page. (#204)
- Bugfix: Add all ancestors to the page attribute `ancestors` rather than just the initial one. (#693)
- Bugfix: Include HTML in the build output again. (#691)
- Bugfix: Provide filename to Read the Docs. (#721 and RTD#1480)
- Bugfix: Silence Click's unicode_literals warning. (#708)

5.1.29. Version 0.14.0 (2015-06-09)

- Improve Unicode handling by ensuring that all config strings are loaded as Unicode. (#592)

- Remove dependency on the six library. (#583)
- Remove dependency on the ghp-import library. (#547)
- Add `--quiet` and `--verbose` options to all sub-commands. (#579)
- Add short options (`-a`) to most command line options. (#579)
- Add copyright footer for readthedocs theme. (#568)
- If the requested port in `mkdocs serve` is already in use, don't show the user a full stack trace. (#596)
- Bugfix: Fix a JavaScript encoding problem when searching with spaces. (#586)
- Bugfix: gh-deploy now works if the mkdocs.yml is not in the git repo root. (#578)
- Bugfix: Handle (pass-through instead of dropping) HTML entities while parsing TOC. (#612)
- Bugfix: Default extra_templates to an empty list, don't automatically discover them. (#616)

5.1.30. Version 0.13.3 (2015-06-02)

- Bugfix: Reduce validation error to a warning if the site_dir is within the docs_dir as this shouldn't cause any problems with building but will inconvenience users building multiple times. (#580)

5.1.31. Version 0.13.2 (2015-05-30)

- Bugfix: Ensure all errors and warnings are logged before exiting. (#536)
- Bugfix: Fix compatibility issues with ReadTheDocs. (#554)

5.1.32. Version 0.13.1 (2015-05-27)

- Bugfix: Fix a problem with minimal configurations which only contain a list of paths in the pages config. (#562)

5.1.33. Version 0.13.0 (2015-05-26)

5.1.33.1. Deprecations to Version 0.13.0

Deprecate the JSON command

In this release the `mkdocs json` command has been marked as deprecated and when used a deprecation warning will be shown. It will be removed in a future release of MkDocs, version 1.0 at the latest. The `mkdocs json` command provided a convenient way for users to output the

documentation contents as JSON files but with the additions of search to MkDocs this functionality is duplicated.

A new index with all the contents from a MkDocs build is created in the site_dir, so with the default value for the `site_dir` It can be found in `site/mkdocs/search_index.json`.

This new file is created on every MkDocs build (with `mkdocs build`) and no configuration is needed to enable it.

Change the pages configuration

Provide a new way to define pages, and specifically nested pages, in the mkdocs.yml file and deprecate the existing approach, support will be removed with MkDocs 1.0.

Warn users about the removal of builtin themes

All themes other than mkdocs and readthedocs will be moved into external packages in a future release of MkDocs. This will enable them to be more easily supported and updates outside MkDocs releases.

5.1.33.2. Major Additions to Version 0.13.0

Search

Support for search has now been added to MkDocs. This is based on the JavaScript library lunr.js. It has been added to both the `mkdocs` and `readthedocs` themes. See the custom theme documentation on supporting search for adding it to your own themes.

New Command Line Interface

The command line interface for MkDocs has been re-written with the Python library Click. This means that MkDocs now has an easier to use interface with better help output.

This change is partially backwards incompatible as while undocumented it was possible to pass any configuration option to the different commands. Now only a small subset of the configuration options can be passed to the commands. To see in full commands and available arguments use `mkdocs --help` and `mkdocs build --help` to have them displayed.

Support Extra HTML and XML files

Like the extra_javascript and extra_css configuration options, a new option named extra_templates has been added. This will automatically be populated with any `.html` or `.xml` files in the project docs directory.

Users can place static HTML and XML files and they will be copied over, or they can also use Jinja2 syntax and take advantage of the global variables.

By default MkDocs will use this approach to create a sitemap for the documentation.

5.1.33.3. Other Changes and Additions to Version 0.13.0

- Add support for Markdown extension configuration options. (#435)
- MkDocs now ships Python wheels. (#486)
- Only include the build date and MkDocs version on the homepage. (#490)
- Generate sitemaps for documentation builds. (#436)
- Add a clearer way to define nested pages in the configuration. (#482)
- Add an extra config option for passing arbitrary variables to the template. (#510)
- Add `--no-livereload` to `mkdocs serve` for a simpler development server. (#511)
- Add copyright display support to all themes (#549)
- Add support for custom commit messages in a `mkdocs gh-deploy` (#516)
- Bugfix: Fix linking to media within the same directory as a markdown file called index.md (#535)
- Bugfix: Fix errors with Unicode filenames (#542).

5.1.34. Version 0.12.2 (2015-04-22)

- Bugfix: Fix a regression where there would be an error if some child titles were missing but others were provided in the pages config. (#464)

5.1.35. Version 0.12.1 (2015-04-14)

- Bugfix: Fixed a CSS bug in the table of contents on some browsers where the bottom item was not clickable.

5.1.36. Version 0.12.0 (2015-04-14)

- Display the current MkDocs version in the CLI output. (#258)
- Check for CNAME file when using gh-deploy. (#285)
- Add the homepage back to the navigation on all themes. (#271)
- Add a strict more for local link checking. (#279)
- Add Google analytics support to all themes. (#333)
- Add build date and MkDocs version to the ReadTheDocs and MkDocs theme outputs. (#382)
- Standardize highlighting across all themes and add missing languages. (#387)

- Add a verbose flag. (-v) to show more details about what the build. (#147)
- Add the option to specify a remote branch when deploying to GitHub. This enables deploying to GitHub pages on personal and repo sites. (#354)
- Add favicon support to the ReadTheDocs theme HTML. (#422)
- Automatically refresh the browser when files are edited. (#163)
- Bugfix: Never re-write URLs in code blocks. (#240)
- Bugfix: Don't copy dotfiles when copying media from the `docs_dir`. (#254)
- Bugfix: Fix the rendering of tables in the ReadTheDocs theme. (#106)
- Bugfix: Add padding to the bottom of all bootstrap themes. (#255)
- Bugfix: Fix issues with nested Markdown pages and the automatic pages configuration. (#276)
- Bugfix: Fix a URL parsing error with GitHub enterprise. (#284)
- Bugfix: Don't error if the mkdocs.yml is completely empty. (#288)
- Bugfix: Fix a number of problems with relative URLs and Markdown files. (#292)
- Bugfix: Don't stop the build if a page can't be found, continue with other pages. (#150)
- Bugfix: Remove the site_name from the page title, this needs to be added manually. (#299)
- Bugfix: Fix an issue with table of contents cutting off Markdown. (#294)
- Bugfix: Fix hostname for BitBucket. (#339)
- Bugfix: Ensure all links end with a slash. (#344)
- Bugfix: Fix repo links in the readthedocs theme. (#365)
- Bugfix: Include jQuery locally to avoid problems using MkDocs offline. (#143)
- Bugfix: Don't allow the docs_dir to be in the site_dir or vice versa. (#384)
- Bugfix: Remove inline CSS in the ReadTheDocs theme. (#393)
- Bugfix: Fix problems with the child titles due to the order the pages config was processed. (#395)
- Bugfix: Don't error during live reload when the theme doesn't exist. (#373)
- Bugfix: Fix problems with the Meta extension when it may not exist. (#398)
- Bugfix: Wrap long inline code otherwise they will run off the screen. (#313)
- Bugfix: Remove HTML parsing regular expressions and parse with HTMLParser to fix problems with titles containing code. (#367)
- Bugfix: Fix an issue with the scroll to anchor causing the title to be hidden under the navigation. (#7)
- Bugfix: Add nicer CSS classes to the HTML tables in bootswatch themes. (#295)

- Bugfix: Fix an error when passing in a specific config file with `mkdocs serve`. (#341)
- Bugfix: Don't overwrite index.md files with the `mkdocs new` command. (#412)
- Bugfix: Remove bold and italic from code in the ReadTheDocs theme. (#411)
- Bugfix: Display images inline in the MkDocs theme. (#415)
- Bugfix: Fix problems with no-highlight in the ReadTheDocs theme. (#319)
- Bugfix: Don't delete hidden files when using `mkdocs build --clean`. (#346)
- Bugfix: Don't block newer versions of Python-markdown on Python >= 2.7. (#376)
- Bugfix: Fix encoding issues when opening files across platforms. (#428)

5.1.37. Version 0.11.1 (2014-11-20)

- Bugfix: Fix a CSS wrapping issue with code highlighting in the ReadTheDocs theme. (#233)

5.1.38. Version 0.11.0 (2014-11-18)

- Render 404.html files if they exist for the current theme. (#194)
- Bugfix: Fix long nav bars, table rendering and code highlighting in MkDocs and ReadTheDocs themes. (#225)
- Bugfix: Fix an issue with the google_analytics code. (#219)
- Bugfix: Remove `__pycache__` from the package tar. (#196)
- Bugfix: Fix markdown links that go to an anchor on the current page. (#197)
- Bugfix: Don't add `prettyprint well` CSS classes to all HTML, only add it in the MkDocs theme. (#183)
- Bugfix: Display section titles in the ReadTheDocs theme. (#175)
- Bugfix: Use the polling observer in watchdog so rebuilding works on filesystems without inotify. (#184)
- Bugfix: Improve error output for common configuration related errors. (#176)

5.1.39. Version 0.10.0 (2014-10-29)

- Added support for Python 3.3 and 3.4. (#103)
- Configurable Python-Markdown extensions with the config setting `markdown_extensions`. (#74)
- Added `mkdocs json` command to output your rendered documentation as json files. (#128)

- Added `--clean` switch to `build`, `json` and `gh-deploy` commands to remove stale files from the output directory. (#157)
- Support multiple theme directories to allow replacement of individual templates rather than copying the full theme. (#129)
- Bugfix: Fix `<ul>` rendering in readthedocs theme. (#171)
- Bugfix: Improve the readthedocs theme on smaller displays. (#168)
- Bugfix: Relaxed required python package versions to avoid clashes. (#104)
- Bugfix: Fix issue rendering the table of contents with some configs. (#146)
- Bugfix: Fix path for embedded images in sub pages. (#138)
- Bugfix: Fix `use_directory_urls` config behavior. (#63)
- Bugfix: Support `extra_javascript` and `extra_css` in all themes. (#90)
- Bugfix: Fix path-handling under Windows. (#121)
- Bugfix: Fix the menu generation in the readthedocs theme. (#110)
- Bugfix: Fix the mkdocs command creation under Windows. (#122)
- Bugfix: Correctly handle external `extra_javascript` and `extra_css`. (#92)
- Bugfix: Fixed favicon support. (#87)

5.2. Contributing to MkDocs

An introduction to contributing to the MkDocs project.

The MkDocs project welcomes, and depends, on contributions from developers and users in the open source community. Contributions can be made in a number of ways, a few examples are:

- Code patches via pull requests
- Documentation improvements
- Bug reports and patch reviews

For information about available communication channels please refer to the README file in our GitHub repository.

5.2.1. Code of Conduct

Everyone interacting in the MkDocs project's codebases, issue trackers, chat rooms, and mailing lists is expected to follow the PyPA Code of Conduct.

5.2.2. Reporting an Issue

Please include as much detail as you can. Let us know your platform and MkDocs version. If the problem is visual (for example a theme or design issue) please add a screenshot and if you get an error please include the full error and traceback.

5.2.3. Testing the Development Version

If you want to just install and try out the latest development version of MkDocs you can do so with the following command. This can be useful if you want to provide feedback for a new feature or want to confirm if a bug you have encountered is fixed in the git master. It is **strongly** recommended that you do this within a virtualenv.

```
pip install https://github.com/mkdocs/mkdocs/archive/master.tar.gz
```

5.2.4. Installing for Development

First you'll need to fork and clone the repository. Once you have a local copy, run the following command. It is **strongly** recommended that you do this within a virtualenv.

```
pip install --editable .
```

This will install MkDocs in development mode which binds the `mkdocs` command to the git repository.

5.2.5. Running the tests

To run the tests, it is recommended that you use tox.

Install Tox using pip by running the command `pip install tox`. Then the test suite can be run for MkDocs by running the command `tox` in the root of your MkDocs repository.

It will attempt to run the tests against all of the Python versions we support. So don't be concerned if you are missing some and they fail. The rest will be verified by Github Actions when you submit a pull request.

5.2.6. Translating themes

To localize a theme to your favorite language, follow the guide on Translating Themes. We welcome translation Pull Requests!

5.2.7. Submitting Pull Requests

If you're considering a large code contribution to MkDocs, please prefer to open an issue first to get early feedback on the idea.

Once you think the code is ready to be reviewed, push it to your fork and send a pull request. For a change to be accepted it will most likely need to have tests and documentation if it is a new feature.

5.2.7.1. Submitting changes to the builtin themes

When installed with `i18n` support (`pip install mkdocs[i18n]`), MkDocs allows themes to support being translated into various languages (referred to as locales) if they respect Jinja's i18n extension by wrapping text placeholders with `{% trans %}` and `{% endtrans %}` tags.

Each time a translatable text placeholder is added, removed or changed in a theme template, the theme's Portable Object Template (`pot`) file needs to be updated by running the `extract_messages` command. For example, to update the `pot` file of the `mkdocs` theme, run the following command:

```
python setup.py extract_messages -t mkdocs
```

The updated `pot` file should be included in a PR with the updated template. The updated `pot` file will allow translation contributors to propose the translations needed for their preferred language. See the guide on Translating Themes for details.

> **Note**
>
> Contributors are not expected to provide translations with their changes to a theme's templates. However, they are expected to include an updated `pot` file so that everything is ready for translators to do their job.

5.3. License

The legal stuff.

5.3.1. Included projects

Themes used under license from the ReadTheDocs projects.

- ReadTheDocs theme - View license.

Many thanks to the authors and contributors of those wonderful projects.

5.3.2. MkDocs License (BSD)

Copyright © 2014, Tom Christie. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Did I try enough harder? ;-)