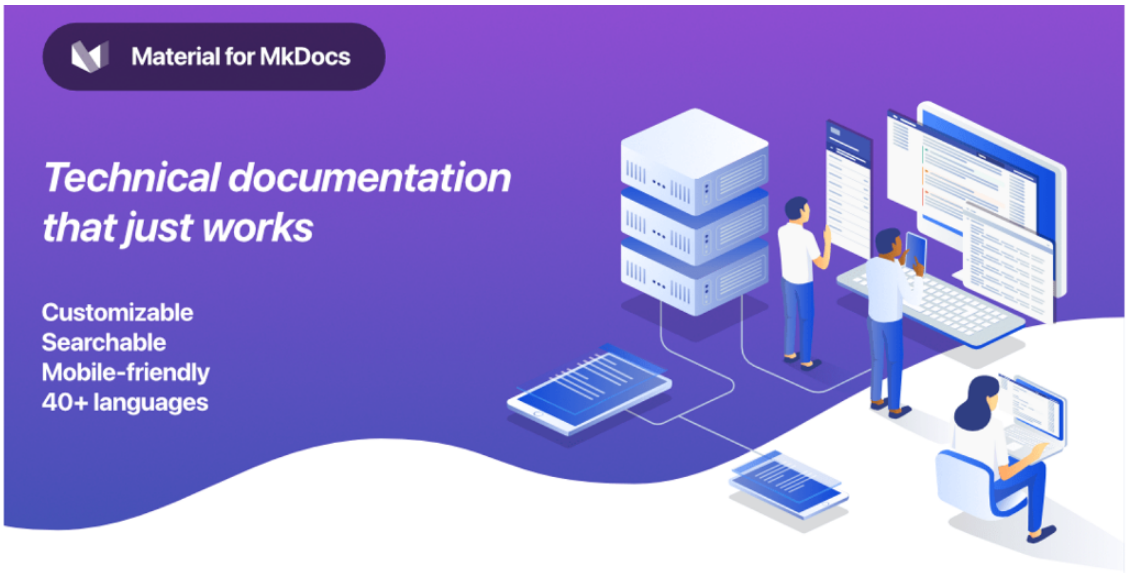
MkDocs Tutorial



Material for MkDocs

Technical documentation that just works

Содержание

- BackSlash escapes

Advanced

Advanced. Theme Material

- Formatting
  - Text with highlighting
  - Sub- and superscripts
  - Adding keyboard keys
- Icons, Emojis
- Grids
- Content tabs
- Tooltips
- Adding abbreviations
- Adding a glossary
- Admonitions
  - Admonition with custom title
  - Admonition, collapsible
  - Inline blocks
- Annotations
- Code blocks
  - Highlighting inline code blocks
  - Embedding external files
- Buttons
- Images
- Data tables
  - Import table from file
  - Readers
  - Embed File
- Diagrams
  - Flowcharts
  - Using sequence diagrams
  - Using state diagrams
  - Using class diagrams

Customization

- Custom colors
- Custom color schemes

Export docs

Plugin print-site

- Automated export using nodejs and chrome
- Export to
- Adding a PDF button to mkdocs-material theme
- Adding a print button to mkdocs theme
- Customize the cover page
- Adding configurable content
- Change the styling
- Customize the print site banner
- Adding configurable content for site banner
- Exclude content from print
  - Ignoring elements in a page
  - Ignoring admonitions
  - Ignoring an entire page

PDF Plugin with-pdf

PDF Plugin pdf-export

WeasyPrint

Publication

Project deploy

- Настройка сборки и публикации
- GitHub

FAQ

- Embedded files
- Useful plugins

Links

- DocOps
- MarkDown
- MkDocs Material

Home

Welcome to MkDocs

DocOps

- Выбор инструментов
- Системы документирования
- Статьи по теме

Getting started

Starting MkDocs

- Installation
- Creating your site
- Commands
- Project layout
- Project YML

Writing docs

Writing MkDocs. Markdown

- Paragraphs and Line breaks
- Headers
- New line
- Emphasis
- Code (text as is)
- Code blocks
- Fenced code blocks
- Horizontal rules
- Comments in md-files
- Links
  - Span element Links
  - Automatic links
  - Internal links
  - Images as Links
- Inline
- Blockquotes
- Lists
- Images
- BackSlash escapes

Advanced

Advanced. Theme Material

- Formatting
  - Text with highlighting
  - Sub- and superscripts
  - Adding keyboard keys
- Icons, Emojis
- Grids
- Content tabs
- Tooltips

- Adding abbreviations
- Adding a glossary
- Admonitions
  - Admonition with custom title
  - Admonition, collapsible
  - Inline blocks
- Annotations
- Code blocks
  - Highlighting inline code blocks
  - Embedding external files
- Buttons
- Images
- Data tables
  - Import table from file
  - Readers
  - Embed File
- Diagrams
  - Flowcharts
  - Using sequence diagrams
  - Using state diagrams
  - Using class diagrams

Customization

- Custom colors
- Custom color schemes

Export docs

Plugin print-site

- Automated export using nodejs and chrome
- Export to
- Adding a PDF button to mkdocs-material theme
- Adding a print button to mkdocs theme
- Customize the cover page
- Adding configurable content
- Change the styling
- Customize the print site banner
- Adding configurable content for site banner
- Exclude content from print
  - Ignoring elements in a page
  - Ignoring admonitions

- Ignoring an entire page

PDF Plugin with-pdf

PDF Plugin pdf-export

WeasyPrint

Publication

Project deploy

- Настройка сборки и публикации
- GitHub

FAQ

- Embedded files
- Useful plugins

Links

- DocOps
- MarkDown
- MkDocs Material

Home

Welcome to MkDocs

To realize DocOps idea.

- **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.

MkDocs site mkdocs.org.

Material for MkDocs mkdocs-material.

DocOps

Основная идея **DocOps** заключается в том, что создание технической документации осуществляется в тесном контакте с разработкой программного продукта. Рабочие процессы максимально синхронизируются и объединяются в общую систему, широко применяется автоматизация рутинных операций и упрощается совместная работа, что позволяет повысить общую эффективность процесса.

Перед техническими писателями, помимо прочего, почти всегда стоит задача разработки и публикации эксплуатационной документации для продукта — руководства пользователя, администратора и т. д. Если говорить о форматах публикации, как правило, необходим онлайн справочный портал, а также зачастую документы в формате для печати, например PDF. Кроме этого, возможно, что клиенты одновременно пользуются несколькими версиями продукта, — потребуется версионирование документации.

Выбор инструментов

Для решения поставленной задачи предлагается следующий базовый набор инструментов:

- **Система контроля версий (VCS)** для хранения исходных файлов (в формате MarkDown).
- **Сервис выполнения CI/CD-конвейеров** (пайплайнов) для автоматизации процессов сборки и публикации.

Git: GitHub GitLab. Кроме основных функций системы контроля версий, дают возможность запуска CI/CD-конвейеров, что позволяет автоматизировать, а значит упростить и ускорить процесс публикации.

- **Static Site Generator** для конвертации MarkDown-файлов в статические HTML-страницы и создания онлайн-версии документации с поддержкой поиска, версионирования и других базовых функций: MkDocs (mkdocs.org), Material for MkDocs (mkdocs-material).
- **Плагины** для генерации документации в PDF-формате: mkdocs-print-site-plugin.
- **IDE:** VS Code.
- На локальной машине должны быть установлены Python и PIP

Системы документирования *docs as code*

На сегодняшний день список самых популярных систем, способных сгенерировать статические сайты:

- *Jekyll* - Написан на Ruby, его использует Github для Github Pages.
- *Hugo* - Написан на Go, позиционирует себя как самый быстрый генератор статических сайтов.
- *Sphinx* - Написан на Python. Именно с его помощью сгенерирована документация к языку Python.
- **MkDocs** - Написан на Python. Популярный генератор на Github. Сгенерированный с его помощью сайт из коробки предоставляет гибкую верстку, легко читаемые элементы, встроенный быстрый поиск, не требует какой-либо серверной реализации. Популярная тема - **Material for MkDocs**.

Статьи по теме

Как разработать техническую документацию, которая точно будет работать. Часть 2. DocOps в действии

Что такое CI/CD? Разбираемся с непрерывной интеграцией и непрерывной поставкой

Непрерывную интеграция и непрерывная доставка. Основы CI/CD

Getting started

Starting MkDocs

Installation

Install **Python*.

Python Download https://www.python.org/downloads/windows/

Install **pip**

```
python.exe -m pip install --upgrade pip
```

Material for MkDocs is published as a Python package and can be installed with pip, ideally by using a virtual environment. Open up a terminal and install Material for MkDocs with:

```
pip install mkdocs-material
```

This will automatically install compatible versions of all dependencies: MkDocs, Markdown, Pygments and Python Markdown Extensions. Material for MkDocs always strives to support the latest versions, so there's no need to install those packages separately.

Creating your site

After you've installed Material for MkDocs, you can bootstrap your project documentation using the mkdocs executable. Go to the directory where you want your project to be located and enter:

```
mkdocs new .
```

or

```
cd /D c:\my-project
mkdocs new c:\my-project
```

Commands

- `mkdocs new [dir-name]` - Create a new project.
- `mkdocs serve` - Start the live-reloading docs server.
- `mkdocs build` - Build the documentation site.
- `mkdocs -h` - Print help message and exit.

Project layout

```
mkdocs.yml    # The configuration file.
docs/
    index.md  # The documentation homepage.
    ...       # Other markdown pages, images and other files.
```

Project YML

mkdocs.yml

```yaml
site_name: MkDocs Tutorial

# Navigation menu
nav:
  - Home:
    - index.md
    #- About: '_about/about.md'
    #- Install: '10_start/install.md'
    #- Writing MkDocs standart:  '20_syntax/standart.md'
    #- Advanced. Theme Material: '20_syntax/advance.md'
    #- Export docs: '30_addition/export_print_site.md'
    #- Publication: '30_addition/publication.md'
    #- Links: '_about/links.md'
    #- FAQ:   '_about/faq.md'

  - About: '_about/about.md'

  - Getting started:
    - 10_start/install.md

  - Writing docs:
    - 20_syntax/standart.md

  - Advanced:
    - 20_syntax/advance.md
    #- Diagrams: '20_syntax/21_diagrams.md'
    - 20_syntax/customize.md

  - Export docs:
    - 30_addition/export_print_site.md
    - 30_addition/export_with_pdf.md
    - 30_addition/export_pdf_export.md
    - 30_addition/weasyprint.md

  - Publication:
    - 30_addition/publication.md

  - FAQ: '_about/faq.md'
  - Links: '_about/links.md'

theme:

  name: material #mkdocs #readthedocs  #material
  custom_dir: overrides

  #palette:
    # Palette toggle for automatic mode
    #- media: "(prefers-color-scheme)"
    #  toggle:
    #    icon: material/brightness-auto
    #    name: Switch to light mode

    # Palette toggle for light mode
    #- media: "(prefers-color-scheme: light)"
    # Palette toggle for dark mode
    #- media: "(prefers-color-scheme: dark)"

  palette:
    # Palette toggle for light mode
    - scheme: default
```

```
      toggle:
        icon: material/weather-sunny
        name: Light. Switch to dark mode
      primary: teal
      accent: blue

    # Palette toggle for dark mode
    - scheme: slate
      toggle:
        icon: material/weather-night
        name: Dark. Switch to light mode
      primary: blue grey
      accent: lime

  icon:
    annotation: material/arrow-right-circle

  logo: assets/logo.png
  favicon: assets/logo_fav.png

  language: en

  features:
    - navigation.instant # ускорение загрузки (SPA)
    - navigation.instant.prefetch # eature that will start to fetch a page once the user hovers
over a link
    - navigation.tracking # динамический URL страницы
    - navigation.tabs # перенос названий документов в горизонтальное меню
    - navigation.tabs.sticky # горизонтальное меню не скрывается при прокрутке
    - navigation.indexes # главные страницы разделов не отображаются в меню
    - navigation.top # вкл. кнопку возврата наверх
    - navigation.sections # When sections are enabled, top-level sections are rendered as groups
in the sidebar for viewports above 1220px, but remain as-is on mobile
    - navigation.expand # When expansion is enabled, the left sidebar will expand all collapsible
subsections by default, so the user doesn't have to open subsections manually
    - navigation.path # When navigation paths are activated, a breadcrumb navigation is rendered
above the title of each page, which might make orientation easier for users visiting your
documentation on devices with smaller screens
    - toc.follow # When anchor following for the table of contents is enabled, the sidebar is
automatically scrolled so that the active anchor is always visible
    - toc.integrate
    - navigation.footer
    - content.tooltips # Improved tooltips
    - content.code.copy
    - content.code.select
    - header.autohide
    - announce.dismiss

markdown_extensions:
  - md_in_html
  - def_list
  - footnotes

  - markdown_include.include:
      base_path: docs

  # Font icons
  - pymdownx.emoji:
      emoji_index: !!python/name:materialx.emoji.twemoji
      emoji_generator: !!python/name:materialx.emoji.to_svg
```

```
  # Formatting
  - pymdownx.critic
  - pymdownx.caret
  - pymdownx.keys
  - pymdownx.mark
  - pymdownx.tilde

  # List
  - pymdownx.tasklist:
      custom_checkbox: true

  # Content tabs
  - pymdownx.tabbed:
      alternate_style: true

  # Snippets
  - abbr
  - attr_list

  - pymdownx.snippets:
      base_path: ["."]  #alternative - "./docs"
      auto_append: ["./docs/includes/abbreviations.md"] # in directory ./docs/_about/

  # Diagrams
  - pymdownx.superfences:
      custom_fences:
        - name: mermaid
          class: mermaid
          format: !!python/name:pymdownx.superfences.fence_code_format


  # Aadmonition, Annotations
  - admonition
  - pymdownx.details
  - pymdownx.superfences

  # Data tables
  - tables

  # Code blocks
  - pymdownx.highlight:
      anchor_linenums: true
      line_spans: __span
      pygments_lang_class: true
  - pymdownx.inlinehilite

  # table of content
  #- toc:
      #permalink: "#"
      #baselevel: 2
      #separator: "-"

plugins:
  - glightbox: # Enlarge image
      touchNavigation: true # Enable or disable the touch navigation (swipe)
      loop: false # Loop slides on end
      effect: zoom #  (zoom, fade, none)
      slide_effect: slide # (slide, zoom, fade, none)
      width: 100% # 90% or 100vw for full width
      height: auto # 90%, 100vh or auto.
      zoomable: true # Enable or disable zoomable images
```

```
      draggable: true # Enable or disable mouse drag to go prev and next slide.
      skip_classes: # Disable lightbox of those image with specific custom class name.
        - custom-skip-class-name
        - img_index
      auto_caption: false # Enable or disable using alt of image as caption title automatically
      caption_position: bottom  # Default captions position. (bottom, top, left, right)

  - search:
      lang:
        - en
        - ru

  # - toc-md:
  #     toc_page_title: Contents
  #     #toc_page_description: Usage mkdocs-toc-md
  #     header_level: 3  #Header level (depth) to render.
  #     pickup_description_meta: false
  #     pickup_description_class: false
  #     output_path: index.md
  #     output_log: false
  #     ignore_page_pattern: index.*.md$
  #     remove_navigation_page_pattern: index.*.md$
  #     template_dir_path: custom_template
  #     integrate_mkdocs_static_i18n: true
  #     languages:
  #       en:
  #         toc_page_title: Contents
  #         #toc_page_description: Usage mkdocs-toc-md
  #       ru:
  #         toc_page_title: Содержание
  #         #toc_page_description:
  #     shift_header: after_h1_of_index
  #     extend_module: true
  #     output_comment: html

  - table-reader
  # - pdf-export:
  #     verbose: true
  #     media_type: print
  #     enabled_if_env: 0
  #     combined: 1
      #combined_output_path: pdf/combined.pdf

  # Export PDF 1
  - print-site:
      add_to_navigation: true # Пункт "Распечатать PDF" в меню, см. след. парам.
      print_page_title: 'To PDF' # Название пункта, см. пред. парам.
      add_print_site_banner: true # Выключает ненужный:) баннер на странице печати
      print_site_banner_template: "docs/assets/templates/print_site_banner.tpl"
      add_table_of_contents: true # Включает отображение содержания
      toc_title: 'Содержание' # Название раздела с содержанием
      toc_depth: 6 # Количество уровней заголовков в содержании
      add_full_urls: false # Отображение адресов ссылок, например: ссылка (https://site)
      enumerate_headings: false # Отключает нумерацию заголовков
      enumerate_figures: true # Включает нумерацию рисунков
      add_cover_page: true # Включает отображение титульной страницы
      cover_page_template: "docs/assets/templates/pdf_cover_page.tpl" # Путь к шаблону титульной
страницы
      path_to_pdf: "MkDocs_Tutorial.pdf" # Путь к сгенерированному PDF-документу
      include_css: true # Включает переопределение дефолтных CSS с целью кастомизации
      # Включает плагин
```

```
      enabled: true # /!ENV [ENABLED_PRINT_SITE, True] #Commands: export ENABLED_PRINT_SITE=false;
mkdocs serve
      #exclude: # Исключает страницы из PDF
      #- index.md

  # Export PDF 2
  #- with-pdf

  # Export PDF 3
  #



extra:
  version:
    provider: mike
    default: latest
  social:
    - icon: fontawesome/solid/paper-plane
      link: mailto:<ar4develop@gmail.com>
      name: ar4develop@gmail.com

  generator: false # displays a Made with Material for MkDocs notice

extra_css:
  - stylesheets/extra.css
  - stylesheets/glightbox.min.css
  #- https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css

extra_javascript:
  - javascripts/extra.js
  - javascripts/tablesort.min.js  #- https://unpkg.com/tablesort@5.3.0/dist/tablesort.min.js
  - javascripts/tablesort.js

copyright: Copyright &copy; 2023 ar4develop@gmail.com
```

Writing docs

Writing MkDocs. Markdown

Writing your docs

Markdown: Syntax

Paragraphs and Line breaks

A paragraph is simply one or more consecutive lines of text, separated by one or more blank lines. (A blank line is any line that looks like a blank line — a line containing nothing but spaces or tabs is considered blank.) Normal paragraphs should not be indented with spaces or tabs.

When you do want to insert a `<br/>` break tag using Markdown, you end a line with two or more spaces, then type return.

Headers

Setext-style headers are “underlined” using equal signs (for first-level headers) and dashes (for second-level headers). For example:

```
This is an H1
=============

This is an H2
-------------
```

Any number of underlining =’s or -’s will work.

Atx-style headers use 1-6 hash characters at the start of the line, corresponding to header levels 1-6. For example:

```
# This is an H1

## This is an H2

###### This is an H6
```

New line

New line with two or more spaces in end of line or tag `<br/>`.

```
One line'  '
Two line  <br/>
Three line
```

One line
Two line
Three line

Emphasis

Markdown treats asterisks (*) *and underscores (_) as indicators of emphasis. Text wrapped with one * or _ will be wrapped with an HTML* `<em>` *tag; double* *'s or _'s will be wrapped with an HTML `<strong>` tag. E.g., this input:

```
*single asterisks*
**double asterisks**
```

single asterisks
single underscores
double asterisks
double underscores

Emphasis can be used in the middle of a word:

```
un**frigging**believable
```

un**frigging**believable

To produce a literal asterisk or underscore at a position where it would otherwise be used as an emphasis delimiter, you can backslash escape it:

```
\*this text is surrounded by literal asterisks\*
```

this text is surrounded by literal asterisks

Code (text as is)

To indicate a span of code, wrap it with backtick quotes (`). Unlike a pre-formatted code block, a code span indicates code within a normal paragraph. For example:

```
Use the `printf()` function.
```

Use the `printf()` function.

To include a literal backtick character within a code span, you can use multiple backticks as the opening and closing delimiters:

```
``There is a literal backtick (`) here.``
```

`There is a literal backtick (`) here.`

A backtick-delimited string in a code span: `` `foo` ``

```
`` `foo` ``
```

Code blocks

One level of indentation — 4 spaces or 1 tab — is removed from each line of the code block. For example, this:

Here is an example of AppleScript:

```
tell application "Foo"
    beep
end tell
```

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.
```

Horizontal rules

Each of the following lines will produce a horizontal rule tag `<hr/>` :

```
* * *
***
*****
- - -
---------------------------------------
```

Comments in md-files

```
[comment]: <> (This is a comment, it will not be included)
[comment]: <> (in  the output file unless you use it in)
[comment]: <> (a reference style link.)
```

Or you could go further:

```
[//]: <> (This is also a comment.)
```

To improve platform compatibility (and to save one keystroke) it is also possible to use # (which is a legitimate hyperlink target) instead of <>:

```
[//]: # (This may be the most platform independent comment)
```

Links

Span element Links

```
This is [an example](http://example.com/ "Title") inline link.

[This link](http://example.net/) has no title attribute.
```

This is an example inline link.
This link has no title attribute.

```
See my [About](../_about/about/) page for details.
```

See my About page for details.

```
This is [an example][id] reference-style link.
```

Then, anywhere in the document, you define your link label like this, on a line by itself:

```
[id]: http://example.com/  "Optional Title Here"
```

Example:

```
I get 10 times more traffic from [Google] [1] than from [Yahoo] [2] or [MSN] [3].

[1]: http://google.com/        "Google"
[2]: http://search.yahoo.com/  "Yahoo Search"
[3]: http://search.msn.com/    "MSN Search"
```

I get 10 times more traffic from Google than from Yahoo or MSN.

Automatic links

Markdown supports a shortcut style for creating “automatic” links for URLs and email addresses: simply surround the URL or email address with angle brackets. What this means is that if you want to show the actual text of a URL or email address, and also have it be a clickable link, you can do this:

```
<http://example.com/>
<address@example.com>
```

In text link: http://example.com/
In text e-mail: address@example.com

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

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

Please see the about for further details.

Link to a section within a target document by using an anchor link.

```
[Commands](../10_start/install.md#Commands)
```

Please see the for use MkDocs Commands.

Images as Links

To use an image as a link, use the following syntax:

```
[![ImageCaption A kitten](../_images/kitten_300.jpg)](http://www.linktarget.com)
```

Inline HTML

Markdown's syntax is intended for one purpose: to be used as a format for writing for the web.

The only restrictions are that block-level HTML elements – e.g. `<div>, <table>, <pre>, <p>, etc.` – must be separated from surrounding content by blank lines, and the start and end tags of the block should not be indented with tabs or spaces. Markdown is smart enough not to add extra (unwanted) `<p>` tags around HTML block-level tags.

For example, to add an HTML table to a Markdown article:

```
<h2>Header h2</h2>
```

This is a table:

```
<table>
<thead>
    <tr>
        <th colspan="2">The table header</th>
    </tr>
</thead>
<tbody>
    <tr>
      <td>The table body</td>
    <td>with two columns</td>
    </tr>
</tbody>
</table>
```

The table header	
The table body	with two columns

This is another regular paragraph.

Blockquotes

```
> This is a blockquote with two paragraphs. Lorem ipsum dolor sit amet,
> consectetuer adipiscing elit. Aliquam hendrerit mi posuere lectus.
> Vestibulum enim wisi, viverra nec, fringilla in, laoreet vitae, risus.
>
> Donec sit amet nisl. Aliquam semper ipsum sit amet velit. Suspendisse
> id sem consectetuer libero luctus adipiscing.
```

or

```
> This is a blockquote with two paragraphs. Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Aliquam hendrerit mi posuere lectus.
Vestibulum enim wisi, viverra nec, fringilla in, laoreet vitae, risus.

> Donec sit amet nisl. Aliquam semper ipsum sit amet velit. Suspendisse
id sem consectetuer libero luctus adipiscing.
```

> This is a blockquote with two paragraphs. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aliquam hendrerit mi posuere lectus. Vestibulum enim wisi, viverra nec, fringilla in, laoreet vitae, risus.
>
> Donec sit amet nisl. Aliquam semper ipsum sit amet velit. Suspendisse id sem consectetuer libero luctus adipiscing.

Nested:

```
> This is the first level of quoting.
>
> > This is nested blockquote.
>
> Back to the first level.
```

> This is the first level of quoting.
>
> > This is nested blockquote.
>
> Back to the first level.

Blockquotes can contain other Markdown elements, including headers, lists, and code blocks:

```
> ### Blockquotes header.
>
> 1.   This is the first list item.
> 2.   This is the second list item.
>
> Here's some example code:
>
>     return shell_exec("echo $input | $markdown_script");
```

> Blockquotes header
>
> 1. This is the first list item.
> 2. This is the second list item.
>
> Here's some example code:
>
> ```
> return shell_exec("echo $input | $markdown_script");
> ```

Lists

Markdown supports ordered (numbered) and unordered (bulleted) lists.

Unordered lists use asterisks, pluses, and hyphens — interchangably — as list markers:

```
*   Red
*   Green
*   Blue
```

is equivalent to:

```
+   Red
+   Green
+   Blue
```

and:

```
-   Red
-   Green
-   Blue
```

- Red
- Green
- Blue

Ordered lists use numbers followed by periods:

```
1.  Bird
2.  McHale
3.  Parish
```

1. Bird
2. McHale
3. Parish

It's important to note that the actual numbers you use to mark the list have no effect on the HTML output Markdown produces. The HTML Markdown produces from the above list is:

```
<ol>
<li>Bird</li>
<li>McHale</li>
<li>Parish</li>
</ol>
```

If you instead wrote the list in Markdown like this:

```
1.  Bird
1.  McHale
1.  Parish
```

or even:

```
3. Bird
1. McHale
8. Parish
```

you'd get the exact same HTML output.

Images

Inline image syntax looks like this:

```
![Alt text](/path/to/img.jpg)
```

Reference-style image syntax looks like this:

```
![Alt text][id]
```

Where “id” is the name of a defined image reference. Image references are defined using syntax identical to link references:

```
[id]: url/to/image  "Optional title attribute"
```

BackSlash escapes

Markdown allows you to use backslash escapes to generate literal characters which would otherwise have special meaning in Markdown's formatting syntax. For example, if you wanted to surround a word with literal asterisks (instead of an HTML `<em>` tag), you can use backslashes before the asterisks, like this:

```
\*literal asterisks\*
```

Markdown provides backslash escapes for the following characters:

```
\   backslash
`   backtick
*   asterisk
_   underscore
{}  curly braces
[]  square brackets
()  parentheses
#   hash mark
+   plus sign
-   minus sign (hyphen)
.   dot
!   exclamation mark
```

Advanced

Advanced. Theme Material

Material for MkDocs link: https://squidfunk.github.io/mkdocs-material/

Local article Getting started.

Formatting

Source link: https://squidfunk.github.io/mkdocs-material/reference/formatting/

Local article Formatting.

Configuration

This configuration enables support for keyboard keys, tracking changes in documents, defining sub- and superscript and highlighting text. Add the following lines to mkdocs.yml:

markdown_extensions: - pymdownx.critic - pymdownx.caret - pymdownx.keys - pymdownx.mark - pymdownx.tilde

Text with highlighting

```
- ==This was marked==
- ^^This was inserted^^
- ~~This was deleted~~
```

- This was marked
- This was inserted
- ~~This was deleted~~

Sub- and superscripts

When Caret & Tilde are enabled, text can be sub- and superscripted with a simple syntax, which is more convenient than directly using the corresponding sub and sup HTML tags:

Text with sub- and superscripts

```
- H~2~O
- A^T^A
```

- H_2O
- A^TA

Adding keyboard keys

When Keys is enabled, keyboard keys can be rendered with a simple syntax. Consult the Python Markdown Extensions documentation to learn about all available shortcodes:

Keyboard keys

```
++ctrl+alt+del++
```

⌃ Ctrl + ⎇ Alt + ⌦ Del

Icons, Emojis

Source link: https://squidfunk.github.io/mkdocs-material/reference/icons-emojis/

Local article Icons, Emojis.

One of the best features of Material for MkDocs is the possibility to use more than 10,000 icons and thousands of emojis in your project documentation with practically zero additional effort. Moreover, custom icons can be added and used in mkdocs.yml, documents and templates.

```
markdown_extensions:
  - attr_list
  - pymdownx.emoji:
      emoji_index: !!python/name:materialx.emoji.twemoji
      emoji_generator: !!python/name:materialx.emoji.to_svg
```

The following icon sets are bundled with Material for MkDocs:

– 🏳 Material Design https://materialdesignicons.com/

– 🏴 FontAwesome https://fontawesome.com/search?m=free

– 🏳 Octicons https://octicons.github.com/

– 🏳 Simple Icons https://simpleicons.org/

Using emojis Emojis can be integrated in Markdown by putting the shortcode of the emoji between two colons. If you're using Twemoji (recommended), you can look up the shortcodes at Emojipedia:

Emoji

```
:smile:
```

😄

Icon
`:fontawesome-regular-face-laugh-wink:`

😄

Icon with color

```
:fontawesome-brands-twitter:{ .twitter }
```

Grids

Material for MkDocs makes it easy to arrange sections into grids, grouping blocks that convey similar meaning or are of equal importance. Grids are just perfect for building index pages that show a brief overview of a large section of your documentation.

Configuration

This configuration enables the use of grids, allowing to bring blocks of identical or different types into a rectangular shape. Add the following lines to mkdocs.yml:

```
markdown_extensions:
  - attr_list
  - md_in_html
```

Usage

Grids come in two flavors: card grids, which wrap each element in a card that levitates on hover, and generic grids, which allow to arrange arbitrary block elements in a rectangular shape.

Card grids wrap each grid item with a beautiful hover card that levitates on hover. They come in two slightly different syntaxes: list and block syntax, adding support for distinct use cases.

List syntax The list syntax is essentially a shortcut for card grids, and consists of an unordered (or ordered) list wrapped by a div with both, the grid and cards classes:

Card grid

```
<div class="grid cards" markdown>

- :fontawesome-brands-html5: __HTML__ for content and structure
- :fontawesome-brands-js: __JavaScript__ for interactivity
- :fontawesome-brands-css3: __CSS__ for text running out of boxes
- :fontawesome-brands-internet-explorer: __Internet Explorer__ ... huh?

</div>
```

-  **HTML** for content and structure
-  **JavaScript** for interactivity
-  **CSS** for text running out of boxes
-  **Internet Explorer** ... huh?

Card grid, complex example

```
<div class="grid cards" markdown>

-   :material-clock-fast:{ .lg .middle } __Set up in 5 minutes__

    ---

    Install [`mkdocs-material`](#) with [`pip`](#) and get up
    and running in minutes

    [:octicons-arrow-right-24: Getting started](#)

-   :fontawesome-brands-markdown:{ .lg .middle } __It's just Markdown__

    ---

    Focus on your content and generate a responsive and searchable static site

    [:octicons-arrow-right-24: Reference](#)

-   :material-format-font:{ .lg .middle } __Made to measure__

    ---

    Change the colors, fonts, language, icons, logo and more with a few lines

    [:octicons-arrow-right-24: Customization](#)

-   :material-scale-balance:{ .lg .middle } __Open Source, MIT__

    ---

    Material for MkDocs is licensed under MIT and available on [GitHub]

    [:octicons-arrow-right-24: License](#)

</div>
```

- **Set up in 5 minutes**

Install `mkdocs-material` with `pip` and get up and running in minutes

→ Getting started

- **It's just Markdown**

Focus on your content and generate a responsive and searchable static site

→ Reference

- **Made to measure**

Change the colors, fonts, language, icons, logo and more with a few lines

→ Customization

- ⚖ **Open Source, MIT**

Material for MkDocs is licensed under MIT and available on [GitHub]

→ License

Card grid, blocks

```
<div class="grid" markdown>

:fontawesome-brands-html5: __HTML__ for content and structure
{ .card }

:fontawesome-brands-js: __JavaScript__ for interactivity
{ .card }

:fontawesome-brands-css3: __CSS__ for text running out of boxes
{ .card }

> :fontawesome-brands-internet-explorer: __Internet Explorer__ ... huh?

</div>
```

HTML for content and structure

JavaScript for interactivity

CSS for text running out of boxes

> **Internet Explorer** ... huh?

Content tabs

Sometimes, it's desirable to group alternative content under different tabs, e.g. when describing how to access an API from different languages or environments. Material for MkDocs allows for beautiful and functional tabs, grouping code blocks and other content.

Configuration

This configuration enables content tabs, and allows to nest arbitrary content inside content tabs, including code blocks. Add the following lines to mkdocs.yml:

```
markdown_extensions:
- pymdownx.superfences
- pymdownx.tabbed:
      alternate_style: true
```

Anchor links

In order to link to content tabs and share them more easily, Insiders adds an anchor link to each content tab automatically, which you can copy via right click or open in a new tab...

Linked content tabs

When enabled, all content tabs across the whole documentation site will be linked and switch to the same label when the user clicks on a tab. Add the following lines to mkdocs.yml:

```
theme:
features:
    - content.tabs.link
```

Content tabs are linked based on their label, not offset. This means that all tabs with the same label will be activated when a user clicks a content tab regardless of order inside a container. Furthermore, this feature is fully integrated with instant loading and persisted across page loads.

Usage

Grouping code blocks

````
=== "C"

    ``` c
    #include <stdio.h>

    int main(void) {
      printf("Hello world!\n");
      return 0;
    }
    ```

=== "C++"

    ``` c++
    #include <iostream>

    int main(void) {
      std::cout << "Hello world!" << std::endl;
      return 0;
    }
    ```
````

C

```c
#include <stdio.h>

int main(void) {
  printf("Hello world!\n");
  return 0;
}
```

C++

```cpp
#include <iostream>

int main(void) {
  std::cout << "Hello world!" << std::endl;
  return 0;
}
```

Grouping other content

```
    === "Unordered list"

        * Sed sagittis eleifend rutrum
        * Donec vitae suscipit est
        * Nulla tempor lobortis orci

    === "Ordered list"

        1. Sed sagittis eleifend rutrum
        2. Donec vitae suscipit est
        3. Nulla tempor lobortis orci
```

Unordered list

- Sed sagittis eleifend rutrum
- Donec vitae suscipit est
- Nulla tempor lobortis orci

Ordered list

1. Sed sagittis eleifend rutrum
2. Donec vitae suscipit est
3. Nulla tempor lobortis orci

Embedded content. Content tabs in admonition

```
!!! example "Content tabs in Admonition for List"

    === "Unordered List"


        * Sed sagittis eleifend rutrum
        * Donec vitae suscipit est
        * Nulla tempor lobortis orci

    === "Ordered List"

        1. Sed sagittis eleifend rutrum
        2. Donec vitae suscipit est
        3. Nulla tempor lobortis orci
```

Content tabs in Admonition for List

Unordered List

- Sed sagittis eleifend rutrum
- Donec vitae suscipit est
- Nulla tempor lobortis orci

Ordered List

1. Sed sagittis eleifend rutrum
2. Donec vitae suscipit est
3. Nulla tempor lobortis orci

Tooltips

Link with tooltip, inline syntax

```
[Hover me](https://example.com "I'm a tooltip!")
```

Hover me

Tooltips can also be added to link references:

```
[Hover me again][example]
[example]: https://example.com "I'm a tooltip too!"
```

Hover me again

Hover me from abbreviations

Adding abbreviations

Abbreviations can be defined by using a special syntax similar to URLs and footnotes, starting with a * and immediately followed by the term or acronym to be associated in square brackets:

Text with abbreviations

```
The HTML specification is maintained by the W3C.

*[HTML]: Hyper Text Markup Language
*[W3C]: World Wide Web Consortium
```

The HTML specification is maintained by the W3C.

Adding a glossary

The Snippets extension can be used to implement a simple glossary by moving all abbreviations in a dedicated file, and auto-append this file to all pages with the following configuration:

mkdocs.yml

```
markdown_extensions:
- pymdownx.snippets:
      auto_append:
        - includes/abbreviations.md
```

Admonitions

> *Source link:* https://squidfunk.github.io/mkdocs-material/reference/admonitions/
>
> *Local article* Admonitions.

Configuration

This configuration enables admonitions, allows to make them collapsible and to nest arbitrary content inside admonitions. Add the following lines to mkdocs.yml:

mkdocs.yml

```
markdown_extensions:
- admonition
- pymdownx.details
- pymdownx.superfences
```

Admonition with custom title

Admonitions follow a simple syntax: a block starts with !!!, followed by a single keyword used as a type qualifier. The content of the block follows on the next line, indented by four spaces:

```
!!! note "Phasellus posuere in sem ut cursus"

    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod
    nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor
    massa, nec semper lorem quam in massa.
```

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Removing the title

```
!!! note ""
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Admonition, collapsible

When Details is enabled and an admonition block is started with ??? instead of !!!, the admonition is rendered as a collapsible block with a small toggle on the right side:

```
??? note "Author note"

    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod
    nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor
    massa, nec semper lorem quam in massa.
```

Author note

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Admonition, collapsible and initially expanded

```
???+ note
```

Note

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

> **Type qualifiers:**
> *note, abstract, info, success, question, warning, failure, danger, bug, example, quote.*

Inline blocks

```
!!! info inline "Lorem ipsum"

    1 Lorem ipsum dolor sit amet, consectetur
    adipiscing elit.
```

and

```
!!! info inline end "Lorem ipsum"

    2 Lorem ipsum dolor sit amet, consectetur
    adipiscing elit.
```

Lorem ipsum

2 Lorem ipsum dolor sit amet, consectetur adipiscing elit.

2 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Annotations

One of the flagship features of Material for MkDocs is the ability to inject annotations – little markers that can be added almost anywhere in a document and expand a tooltip containing arbitrary Markdown on click or keyboard focus.

Configuration

This configuration allows to add annotations to all inline- and block-level elements, as well as code blocks, and nest annotations inside each other. Add the following lines to mkdocs.yml:

```
markdown_extensions:
- attr_list
- md_in_html
- pymdownx.superfences
```

Annotation icons

The annotation icon can be changed to any icon bundled with the theme, or even a custom icon, e.g. to material/arrow-right-circle:. Simply add the following lines to mkdocs.yml:

```
theme:
icon:
    annotation: material/arrow-right-circle
```

Some popular choices:

⊕ - material/plus-circle • - material/circle-medium ⊙ - material/record-circle ⊕ - material/arrow-right-circle ⊕ - material/arrow-right-circle-outline ⊕ - material/chevron-right-circle ⊕ - material/star-four-points-circle ⊕ - material/plus-circle-outline

Using annotations

Annotations consist of two parts: a marker, which can be placed anywhere in a block marked with the annotate class, and content located in a list below the block containing the marker:

Text with annotations:

```
Lorem ipsum dolor sit amet, (1) consectetur adipiscing elit.
{ .annotate }

1.  :man_raising_hand: I'm an annotation! I can contain `code`, __formatted
    text__, images, ... basically anything that can be expressed in Markdown.
```

Lorem ipsum dolor sit amet, (1) consectetur adipiscing elit.

(1) 🙋 I'm an annotation! I can contain `code`, **formatted text**, images, ... basically anything that can be expressed in Markdown.

In admonitions

The titles and bodies of admonitions can also host annotations by adding the annotate modifier after the type qualifier, which is similar to how inline blocks work:

Admonition with annotations

```
!!! note annotate "Phasellus posuere in sem ut cursus (1)"

    Lorem ipsum dolor sit amet, (2) consectetur adipiscing elit.
    Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus,
    justo purus auctor massa, nec semper lorem quam in massa.

1.  :man_raising_hand: I'm an annotation!
2.  :woman_raising_hand: I'm an annotation as well!
```

Phasellus posuere in sem ut cursus (1)

Lorem ipsum dolor sit amet, (2) consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

(1) 🙋‍♂️ I'm an annotation!

(2) 🙋‍♀️ I'm an annotation as well!

In content tabs

Content tabs can host annotations by adding the annotate class to the block of a dedicated content tab (and not to the container, which is not supported):

Content tabs with annotations

```
=== "Tab 1"

    Lorem ipsum dolor sit amet, (1) consectetur adipiscing elit.
    { .annotate }

    1.  :man_raising_hand: I'm an annotation!

=== "Tab 2"

    Phasellus posuere in sem ut cursus (1)
    { .annotate }

    1.  :woman_raising_hand: I'm an annotation as well!
```

Tab 1

Lorem ipsum dolor sit amet, (1) consectetur adipiscing elit.

(1) 🙋‍♂️ I'm an annotation!

Tab 2

Phasellus posuere in sem ut cursus (1)

(1) 🙋‍♀️ I'm an annotation as well!

In everything else

The Attribute Lists extension is the key ingredient for adding annotations to most elements, but it has some limitations. However, it's always possible to leverage the Markdown in HTML extension to wrap arbitrary elements with a div with the annotate class:

HTML with annotations

```
<div class="annotate" markdown>

> Lorem ipsum dolor sit amet, (1) consectetur adipiscing elit.

</div>

1.  :man_raising_hand: I'm an annotation!
```

> Lorem ipsum dolor sit amet, (1) consectetur adipiscing elit.

(1) 🙋 I'm an annotation!

Code blocks

Code blocks and examples are an essential part of technical project documentation. Material for MkDocs provides different ways to set up syntax highlighting for code blocks, either during build time using Pygments or during runtime using a JavaScript syntax highlighter.

Configuration

This configuration enables syntax highlighting on code blocks and inline code blocks, and allows to include source code directly from other files. Add the following lines to mkdocs.yml:

```
markdown_extensions:
  - pymdownx.highlight:
      anchor_linenums: true
      line_spans: __span
      pygments_lang_class: true
  - pymdownx.inlinehilite
  - pymdownx.snippets
  - pymdownx.superfences
```

Code copy button

Code blocks can automatically render a button on the right side to allow the user to copy a code block's contents to the clipboard. Add the following to mkdocs.yml to enable them globally:

```
theme:
features:
    - content.code.copy
```

Code selection button

Code blocks can include a button to allow for the selection of line ranges by the user, which is perfect for linking to a specific subsection of a code block. This allows the user to apply line highlighting dynamically. Add the following to mkdocs.yml to enable it globally:

```
theme:
features:
    - content.code.select
```

Enabling or disabling code copy and select buttons for a specific code block:

````
``` { .yaml .copy .select}
# Code block content
```
````

or

````
``` { .yaml .no-copy .no-select}
# Code block content
```
````

Enable

```
# Code block content
```

Disable

```
# Code block content
```

Code block with title, line numbers and highlighted lines

````
``` py title="bubble_sort.py" linenums="1" hl_lines="2 3"
def bubble_sort(items):
    for i in range(len(items)):
        for j in range(len(items) - 1 - i):
            if items[j] > items[j + 1]:
                items[j], items[j + 1] = items[j + 1], items[j]
```
````

bubble_sort.py

```
1  def bubble_sort(items):
2      for i in range(len(items)):
3          for j in range(len(items) - 1 - i):
4              if items[j] > items[j + 1]:
5                  items[j], items[j + 1] = items[j + 1], items[j]
```

Highlighting inline code blocks

When InlineHilite is enabled, syntax highlighting can be applied to inline code blocks by prefixing them with a shebang, i.e. #!, directly followed by the corresponding language shortcode.

````
The `#!python range()` function is used to generate a sequence of numbers.
````

The `range()` function is used to generate a sequence of numbers.

Embedding external files

When Snippets is enabled,

```
markdown_extensions:
- pymdownx.snippets
```

content from other files (including source files) can be embedded by using the --8<-- notation directly from within a code block:

````
``` title="data.src"
`--8<-- "./_files/data.src"`
```

``` title="data.src"

```
````

or

```
/--8<--
./_files/data.src
./docs/includes/example_file.html
/--8<--
```

Example

//Code sample from data.src

if(true) Console.Writeln("Hello world"); <!DOCTYPE html>

```
<h1>Example Document HTML</h1>

<hr>

<h2>Lorem ipsum dolor sit amet</h2>

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugit rem atque labore quas consequatur.
Soluta quaerat voluptatibus veritatis eveniet enim dolor, earum asperiores libero incidunt atque eum
maxime perspiciatis nesciunt.
```

More information: https://facelessuser.github.io/pymdown-extensions/extensions/snippets/

Buttons

Material for MkDocs provides dedicated styles for primary and secondary buttons that can be added to any link, label or button element. This is especially useful for documents or landing pages with dedicated call-to-actions.

Configuration

This configuration allows to add attributes to all inline- and block-level elements with a simple syntax, turning any link into a button. Add the following lines to mkdocs.yml:

```
markdown_extensions:
- attr_list
```

Usage

In order to render a link as a button, suffix it with curly braces and add the .md-button class selector to it. The button will receive the selected primary color and accent color if active.

Button

```
[Subscribe to our newsletter](#){ .md-button }
```

Subscribe to our newsletter

Button, primary

```
[Subscribe to our newsletter](#){ .md-button .md-button--primary }
```

Subscribe to our newsletter

Button with icon. Go to data-table

```
[Go to tables :fontawesome-solid-paper-plane:](#data-tables){ .md-button }
```

Go to tables

Images

Source link: https://squidfunk.github.io/mkdocs-material/reference/images/

Local article: Images

This configuration adds the ability to align images, add captions to images (rendering them as figures), and mark large images for lazy-loading. Add the following lines to mkdocs.yml:

```
markdown_extensions:
- attr_list
- md_in_html
```

If you want to add image zoom functionality to your documentation, the glightbox plugin is an excellent choice, as it integrates perfectly with Material for MkDocs. Install it with pip:

```
pip install mkdocs-glightbox
```

Then, add the following lines to mkdocs.yml:

```
plugins:
- glightbox:
   touchNavigation: true
   loop: false
   effect: zoom
   slide_effect: slide
   width: 100%
   height: auto
   zoomable: true
   draggable: true
   skip_classes:
     - custom-skip-class-name
   auto_caption: false
   caption_position: bottom
```

Configuration options

When Attribute Lists is enabled, images can be aligned by adding the respective alignment directions via the align attribute, i.e. align=left or align=right:

```
![Image title](https://dummyimage.com/600x400/eee/aaa){ align=left loading=lazy width=150}
```

Image with caption

```
<figure markdown>
![Image title](https://dummyimage.com/600x400/){ width="300" }
<figcaption>Image caption</figcaption>
</figure>
```

Light and dark mode

```
![Image title](https://dummyimage.com/600x400/f5f5f5/aaaaaa#only-light){ width="300" }
![Image title](https://dummyimage.com/600x400/21222c/d5d7e2#only-dark){ width="300" }
```

Data tables

Material for MkDocs defines default styles for data tables – an excellent way of rendering tabular data in project documentation. Furthermore, customizations like sortable tables can be achieved with a third-party library and some additional JavaScript.

Configuration

This configuration enables Markdown table support, which should normally be enabled by default, but to be sure, add the following lines to mkdocs.yml:

markdown_extensions: - tables

Usage

Left

```
| Method      | Description                          |
| :---------- | :----------------------------------- |
| `GET`       | :material-check:     Fetch resource  |
| `PUT`       | :material-check-all: Update resource |
| `DELETE`    | :material-close:     Delete resource |
```

Center

```
| Method      | Description                          |
| :---------: | :----------------------------------: |
| `GET`       | :material-check:     Fetch resource  |
| `PUT`       | :material-check-all: Update resource |
| `DELETE`    | :material-close:     Delete resource |
```

Right

```
| Method      | Description                          |
| ----------: | -----------------------------------: |
| `GET`       | :material-check:     Fetch resource  |
| `PUT`       | :material-check-all: Update resource |
| `DELETE`    | :material-close:     Delete resource |
```

Sortable tables

If you want to make data tables sortable, you can add tablesort, which is natively integrated with Material for MkDocs and will also work with instant loading via additional JavaScript:

docs/javascripts/tablesort.js

```
document$.subscribe(function() {
  var tables = document.querySelectorAll("article table:not([class])")
  tables.forEach(function(table) {
    new Tablesort(table)
  })
})
```

mkdocs.yml

```
extra_javascript:
  - https://unpkg.com/tablesort@5.3.0/dist/tablesort.min.js
  - javascripts/tablesort.js
```

Table Centered, Sorted:

Method	Description
`GET`	✓ Fetch resource
`PUT`	✓✓ Update resource
`DELETE`	✕ Delete resource

Import table from file

You can also import data from a CSV or Excel file using the plugin mkdocs-table-reader-plugin.

First, you will need to install it with pip:

```
pip install mkdocs-table-reader-plugin
```

Then extend the mkdocs.yml file like this:

```
plugins:
- table-reader
```

Then, it is a simple process to import the data in to the Markdown files.

Import data from CSV file

data.csv

```
col1,col2,col3
r1c1,r1c2,r1c3
r2c1,r2c2,r2c3
r3c1,r3c2,r3c3
```

add it to your Markdown page this (withot symbol slash (/)):

/{/{ read_csv('./docs/_files/data.csv') /}/}

Result from data.csv

{{ read_csv('./docs/_files/data.csv') }}

Import data from Excel file

```
pip install openpyxl

./docs/_files/data.xlsx

add it to your Markdown page this (withot symbol slash (/)):

/{/{ read_excel('./docs/_files/data.xlsx', engine='openpyxl') /}/}

For document with sheets.

/{/{ read_excel('./Book1.xlsx', engine='openpyxl', sheet_name="Sheet1") /}/}
```

Result from data.xlsx

{{ read_excel('./docs/_files/data.xlsx', engine='openpyxl') }}

Import data from other file types

The plugin mkdocs-table-reader-plugin also provides readers for other formats:

read_csv, read_fwf, read_yaml, read_table, read_json, read_excel, read_raw

You can read more on their Docs website: mkdocs-table-reader-plugin

Readers

Install the plugin using pip:

```
pip install mkdocs-table-reader-plugin
```

mkdocs.yml

```
    plugins:
    - table-reader
```

The following table reader functions are available:

read_csv

```
/{/{ read_csv('tables/basic_table.csv') /}/}
```

read_fwf

```
/{/{ read_fwf('tables/fixedwidth_table.txt') /}/}
```

read_yaml

```
/{/{ read_yaml('tables/yaml_table.yml') /}/}
```

read_table

```
/{/{ read_table('tables/basic_table.csv', sep = ',') /}/}
```

read_json

```
/{/{ read_json('tables/data.json', orient='split') /}/}
```

read_excel

```
/{/{ read_excel('tables/excel_table.xlsx', engine='openpyxl') /}/}
```

read_raw

{{ read_raw() }} inserts contents from a file directly. This is great if you have a file with a table already in markdown format. It could also replace a workflow where you use the snippets extension to embed external files.

```
/{/{ read_raw('tables/markdown_table.md') /}/}
```

Embed File

```
/--8<--
/./docs/20_syntax/21_tables.md
/./docs/20_syntax/21_diagrams.md
/--8<--
```

Diagrams

Source links:
About Mermaid A diagrams plugin for Mkdocs
MagicSpace Draw. Diagrams MkDocs Material. Diagrams

Local article MkDocs Material. Diagrams

Diagrams help to communicate complex relationships and interconnections between different technical components, and are a great addition to project documentation. Material for MkDocs integrates with Mermaid.js, a very popular and flexible solution for drawing diagrams.

Configuration

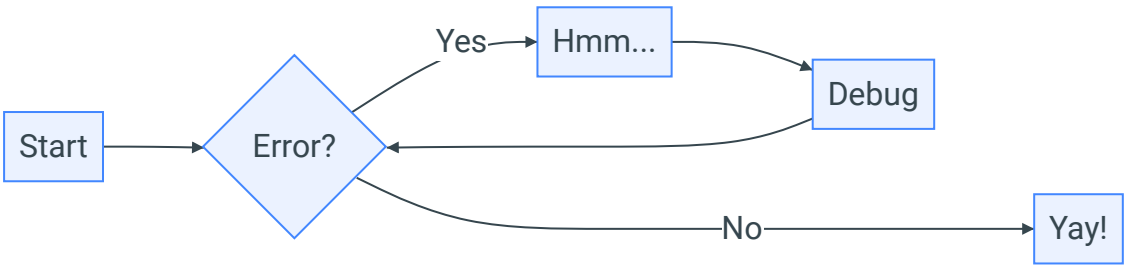
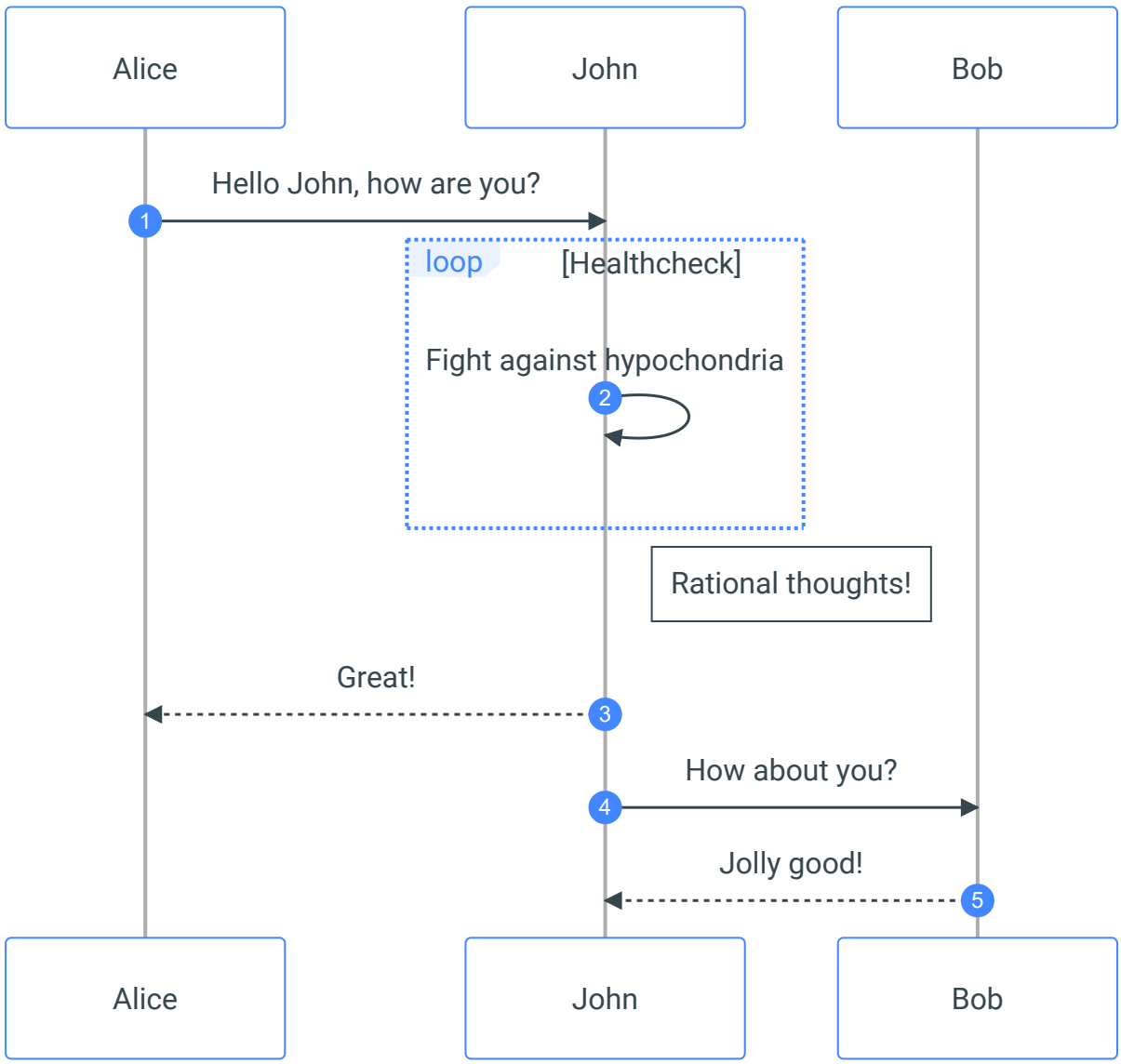
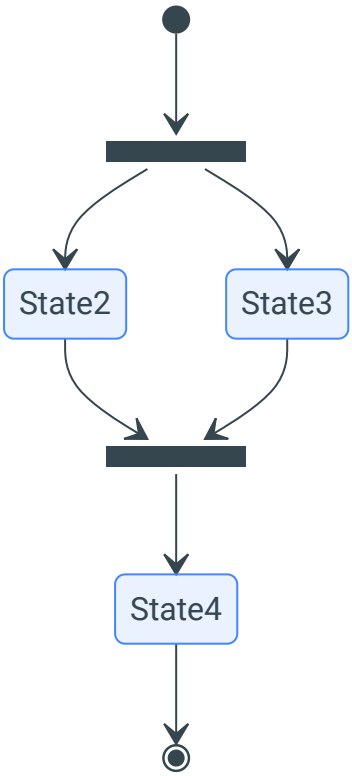
This configuration enables native support for Mermaid.js diagrams. Material for MkDocs will automatically initialize the JavaScript runtime when a page includes a mermaid code block:

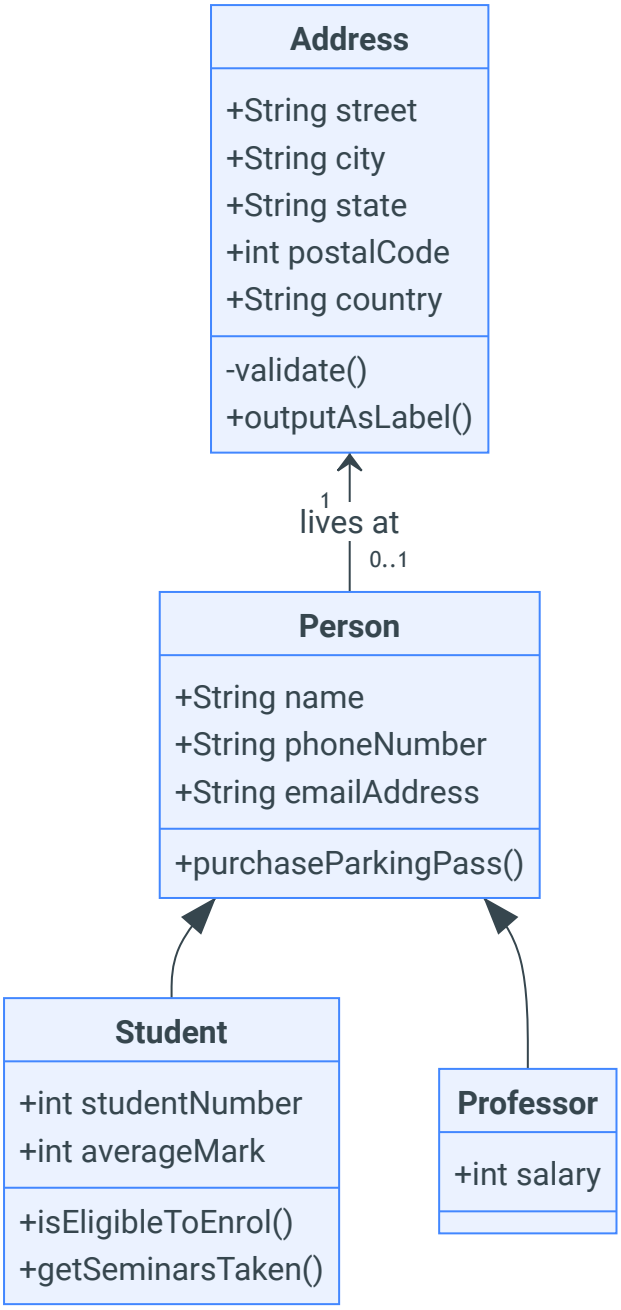
```
markdown_extensions:
  - pymdownx.superfences:
      custom_fences:
        - name: mermaid
          class: mermaid
          format: !!python/name:pymdownx.superfences.fence_code_format
```

Usage

Flowcharts

Flowcharts are diagrams that represent workflows or processes. The steps are rendered as nodes of various kinds and are connected by edges, describing the necessary order of steps: Flow chart

```
``` mermaid
graph LR
A[Start] --> B{Error?};
B -->|Yes| C[Hmm...];
C --> D[Debug];
D --> B;
B ---->|No| E[Yay!];
```
```



Using sequence diagrams

Sequence diagrams describe a specific scenario as sequential interactions between multiple objects or actors, including the messages that are exchanged between those actors:

```
``` mermaid title="Sequence diagram"
sequenceDiagram
autonumber
Alice->>John: Hello John, how are you?
loop Healthcheck
    John->>John: Fight against hypochondria
end
Note right of John: Rational thoughts!
John-->>Alice: Great!
John->>Bob: How about you?
Bob-->>John: Jolly good!
```
```



Using state diagrams

State diagrams are a great tool to describe the behavior of a system, decomposing it into a finite number of states, and transitions between those states:

```
``` mermaid
stateDiagram-v2
  state fork_state <<fork>>
    [*] --> fork_state
    fork_state --> State2
    fork_state --> State3

    state join_state <<join>>
    State2 --> join_state
    State3 --> join_state
    join_state --> State4
    State4 --> [*]
```
```



Using class diagrams

Class diagrams are central to object oriented programing, describing the structure of a system by modelling entities as classes and relationships between them:

```
``` mermaid
classDiagram
  Person <|-- Student
  Person <|-- Professor
  Person : +String name
  Person : +String phoneNumber
  Person : +String emailAddress
  Person: +purchaseParkingPass()
  Address "1" <-- "0..1" Person:lives at
  class Student{
    +int studentNumber
    +int averageMark
    +isEligibleToEnrol()
    +getSeminarsTaken()
  }
  class Professor{
    +int salary
  }
  class Address{
    +String street
    +String city
    +String state
    +int postalCode
    +String country
    -validate()
    +outputAsLabel()
  }
```
```


Address
+String street
+String city
+String state
+int postalCode
+String country
-validate()
+outputAsLabel()
1
lives at
0..1
Person
+String name
+String phoneNumber
+String emailAddress
+purchaseParkingPass()
Student
+int studentNumber
+int averageMark
+isEligibleToEnrol()
+getSeminarsTaken()
Professor
+int salary

Customization

Material for MkDocs link: https://squidfunk.github.io/mkdocs-material/customization/

Local article Customization.

Custom colors

Material for MkDocs implements colors using CSS variables (custom properties). If you want to customize the colors beyond the palette (e.g. to use your brand-specific colors), you can add an additional style sheet and tweak the values of the CSS variables.

First, set the primary or accent values in mkdocs.yml to custom, to signal to the theme that you want to define custom colors, e.g., when you want to override the primary color:

```
theme:
  palette:
    primary: custom
```

Let's say you're YouTube, and want to set the primary color to your brand's palette. Just add:

mkdocs.yml

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

docs/stylesheets/extra.css

```
:root {
  --md-primary-fg-color:        #EE0F0F;
  --md-primary-fg-color--light: #ECB7B7;
  --md-primary-fg-color--dark:  #90030C;
}
```

See the file containing the color definitions for a list of all CSS variables.

Custom color schemes

Besides overriding specific colors, you can create your own, named color scheme by wrapping the definitions in a `[data-md-color-scheme="..."]` attribute selector, which you can then set via mkdocs.yml as described in the color schemes section:

``` title="docs/stylesheets/extra.css" [data-md-color-scheme="youtube"] { --md-primary-fg-color: #EE0F0F; --md-primary-fg-color--light: #ECB7B7; --md-primary-fg-color--dark: #90030C; }
```

````
Additionally, the slate color scheme defines all of it's colors via hsla color functions and
deduces its colors from the --md-hue CSS variable. You can tune the slate theme with:

``` title="extra.css"
[data-md-color-scheme="slate"] {
  --md-hue: 210;
}
````
````

Export docs

Plugin print-site

Source link: Documentation mkdocs-print-site-plugin https://timvink.github.io/mkdocs-print-site-plugin/print_page.html#

Local article: Print Site

Installation

Install the package with pip:

```
pip install mkdocs-print-site-plugin
```

Enable the plugin in your mkdocs.yml as **last** plugin:

```
plugins:
    ...
    - print-site:
        add_to_navigation: true # Пункт "Распечатать PDF" в меню, см. след. парам.
        print_page_title: 'Распечатать PDF' # Название пункта, см. пред. парам.
        add_print_site_banner: false # Выключает ненужный:) баннер на странице печати
        add_table_of_contents: true # Включает отображение содержания
        toc_title: 'Содержание' # Название раздела с содержанием
        toc_depth: 6 # Количество уровней заголовков в содержании
        add_full_urls: false # Отображение адресов ссылок, например: ссылка (https://site)
        enumerate_headings: false # Отключает нумерацию заголовков
        enumerate_figures: true # Включает нумерацию рисунков
        add_cover_page: true # Включает отображение титульной страницы
        cover_page_template: "" # Путь к шаблону титульной страницы
        path_to_pdf: "" # Путь к сгенерированному PDF-документу
        include_css: true # Включает переопределение дефолтных CSS с целью кастомизации
        enabled: true # Включает плагин
        #exclude: # Исключает страницы из PDF
        #- index.md
```

Options

`add_to_navigation` Default is false. Adds a link 'Print Site' to your site navigation. You can also set to false and explicitly include the link in your navigation (/print_page or /print_page.html).

`print_page_title` Default is 'Print Site'. When add_to_navigation is set to true this setting controls the name of the print page in the navigation of the site. This setting is ignored when add_to_navigation is set to false.

`add_table_of_contents` Default is true. Adds a table of contents section at the beginning of the print page (in print version, the HTML version has a different sidebar ToC).

`toc_title` Default is 'Table of Contents'. When add_table_of_contents is set to true this setting controls the name of the table of contents of the print version of the print page. This setting is ignored when add_table_of_contents is set to false.

`toc_depth` Default is 3. When add_table_of_contents is set to true this setting controls the depth of the table of contents in the print version of the print page. This setting is ignored when add_table_of_contents is set to false.

`add_full_urls` Default is false. When printing a page, you cannot see the target of a link. This option adds the target url in parenthesis behind a link. For example "google.com" will be replaced by "google.com (https://www.google.com)"

`enumerate_headings` Default true. This will add numbering (enumeration) to all headings and sections, as well as the table of contents. Note this will only enumerate the print site page; if you want to enumerate the entire site, you can use mkdocs-enumerate-headings-plugin.

```
Example "1.2 A chapter subsection".
```

`enumerate_headings_depth` Default 6. If enumerate_headings, the depth until which headings and sections are enumerated.

`enumerate_figures` Default true. This will add numbering to all figure captions (for example "Figure 1: "). Works especially well with mkdocs-img2fig-plugin.

`add_cover_page` Default false. When enabled, a cover page is added to the print page, displaying the site_title and other information from the mkdocs.yml file. See also Customizing the cover page

`cover_page_template` Default "". The path to a custom cover page template to use. See Customizing the Cover Page for more info.

`add_print_site_banner` Default false. When enabled, a banner is added to the top of the HTML print page, explaining to users the current page contains all site pages. See Customizing the print site banner for more info.

`print_site_banner_template` Default "". The path to a custom print site banner template to use. See Customizing the print site banner for more info.

`path_to_pdf` Default is empty. Option to make it easier to add a link to the PDF version of the site on each page. See Adding a PDF button for more info.

`include_css` Default is true. When disabled the print-site stylesheets are not included. This makes it easy to overwrite the CSS with your own stylesheets, using the extra_css option in your mkdocs.yml file.

`enabled` Default is true. Enables you to deactivate this plugin. A possible use case is local development where you might want faster build times. It's recommended to use this option with an environment variable together with a default fallback (introduced in mkdocs v1.2.1, see docs). Example:

mkdocs.yml

```
    plugins:
        - print-site:
            enabled: !ENV [ENABLED_PRINT_SITE, True]
```

Which enables you do disable the plugin locally using:

```
export ENABLED_PRINT_SITE=false
mkdocs serve
```

`exclude` Default is empty. Allows to specify a list of page source paths that should not be included in the print page. See Do Not Print for more info.

Usage

- Navigate to /print_page/ or print_page.html
- Export to standalone HTML (see export to HTML)
- Export to PDF using your browser using File > Print > Save as PDF (see export to PDF)

> **mkdocs serve** URL http://127.0.0.1:8000/**print_page**/, Press `^ Ctrl` + `P` and *"Save as PDF"*.
> **GitHub:** .github/workflows/main.yml Add string - run: pip install mkdocs-print-site-plugin.

Automated export using nodejs and chrome

We can use nodejs together with the puppeteer headless chrome node.js package:

- Install nodejs
- Download puppeteer in the root of your project using the node package manager: `npm i --save puppeteer`
- Save the script `export_to_pdf.js` (see below) in the root of your project
- Start a webserver with your site (`mkdocs serve`)
- In a separate terminal window, you can now run the PDF export with url (to your print page), pdfPath (name of output file) and title arguments:

  node exportpdf.js http://localhost:8000/print_page.html out.pdf 'title'

export_to_pdf.js

```
(async() => {
const browser = await puppeteer.launch({
    headless: true,
    executablePath: process.env.CHROME_BIN || null,
    args: ['--no-sandbox', '--headless', '--disable-gpu', '--disable-dev-shm-usage']
});

const page = await browser.newPage();
await page.goto(url, { waitUntil: 'networkidle2' });
await page.pdf({
    path: pdfPath, // path to save pdf file
    format: 'A4', // page format
    displayHeaderFooter: true, // display header and footer (in this example, required!)
    printBackground: true, // print background
    landscape: false, // use horizontal page layout
    headerTemplate: headerHtml, // indicate html template for header
    footerTemplate: footerHtml,
    scale: 1, //Scale amount must be between 0.1 and 2
    margin: { // increase margins (in this example, required!)
        top: 80,
        bottom: 80,
        left: 30,
        right: 30
    }
});

await browser.close();
})();
```

Export to HTML

After enabling the print-site plugin in your mkdocs.yml, you will have your entire site combined into a single page. That allows you to create a standalone HTML page: a single self-contained file that has all images, styling and scripts embedded. This means you could send a site as an email attachment, a use case common within companies where deploying static sites might be more involved.

In order to create a self-contained, standalone HTML file from the print page, we will need to embed images, CSS and javascript using data URLs. We can do this quite easily using the htmlark python package:

```
pip install http html5lib requests
pip install htmlark
```

To create the export:

```
mkdocs build

cd site/

# when mkdocs.yml has use_directory_urls: true (the default)
htmlark print_page/index.html -o standalone.html

# when mkdocs.yml has use_directory_urls: false
htmlark print_page.html -o standalone.html
```

Adding a PDF button to mkdocs-material theme

In the mkdocs-material theme you can create an override for main.html (see customization).

mkdocs.yml

```
theme:
name: material
custom_dir: docs/overrides

plugins:
    - print-site:
        - path_to_pdf: "assets/the_name_of_your_file.pdf"
```

docs/overrides/main.html

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

{% block content %}

{% if page.url_to_pdf %}
    <a href="{{ page.url_to_pdf }}" title="Site PDF" class="md-content__button md-icon">
        {% include ".icons/material/file-pdf-box.svg" %}
    </a>
{% endif %}

{{ super() }}
{% endblock content %}
```

Adding a print button to mkdocs theme

You can also customize the base mkdocs theme, by overriding main.html.

mkdocs.yml

```
theme:
    name: mkdocs
    custom_dir: docs/overrides

plugins:
    - print-site:
        - path_to_pdf: "assets/the_name_of_your_file.pdf"

docs/overrides/main.html
```

docs/overrides/main.html

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

{% block repo %}
    {% if page.url_to_pdf %}
        <li class="nav-item">
            <a href="{{ page.url_to_pdf }}" title="Site PDF" class="nav-link">
            <i class="fas fa-file-pdf"></i> PDF
            </a>
        </li>
    {% endif %}

{{ super() }}
{% endblock repo %}
```

Customize the cover page

By default the add_cover_page option is set to true, which will add a cover page to the print page. You might want to customize it more to your liking.

You can do that by specifying the path to a custom cover page template in the mkdocs.yml file. This file should be a standard jinja2 template where you can combine HTML and jinja2 variables. The information specified in mkdocs.yml will already by available as jinja2 variables (see mkdocs project information).

mkdocs.yml

```
plugins:
    - print-site:
        add_cover_page: true
        cover_page_template: "docs/assets/templates/custom_cover_page.tpl"
```

docs/assets/templates/custom_cover_page.tpl

```
{% if config.site_name %}
    <h1>{{ config.site_name }}</h1>
{% endif %}
<h2>This is my custom print cover page</h2>

Adding images
<img src="/assets/img/example.png" />
```

Adding configurable content

You might want to add some content to your cover page that's not yet specified in your mkdocs.yml file. Of course you could just hard-code it in your custom template file, but you could also make use of MkDocs's extra context feature, allowing you to use custom variables from your config file with `{{ config.extra.<your variable> }}`.

mkdocs.yml

```
plugins:
    - print-site:
        add_cover_page: true
        cover_page_template: "docs/assets/templates/custom_cover_page.tpl"

extra:
    abstract: This is a report about a topic
```

docs/assets/templates/custom_cover_page.tpl

```
{% if config.site_name %}
<h1>{{ config.site_name }}</h1>
{% endif %}
<p>{{ config.extra.abstract }}</p>
```

Change the styling

You'll likely also want to change the styling of the cover page to your liking. You can add your own CSS file using the extra_css option from MkDocs. mkdocs-print-site-plugin wraps the cover page in a `<section id="print-site-cover-page">`. You should use this in your CSS to ensure not affecting other pages.

mkdocs.yml

``` plugins: - print-site: add_cover_page: true

extra_css: - docs/assets/css/my_cover_page.css

**docs/assets/css/my_cover_page.css**

```
#print-site-cover-page h1 {
    color: blue;
}
```

## Customize the print site banner

When a user visits the print page, it might not be immediately obvious how to use it. You can set the add_print_site_banner option to true to add a banner to the top of the HTML print page that will be hidden when printing.
```

You might want to customize this banner, for example by translating it to your language. You can do that by specifying the path to a custom banner template in the mkdocs.yml file. This file should be a standard jinja2 template where you can combine HTML and jinja2 variables. The information specified in mkdocs.yml will already by available as jinja2 variables (see mkdocs project information).

mkdocs.yml

```
plugins:
    - print-site:
        add_print_site_banner: true
        print_site_banner_template: "docs/assets/templates/custom_banner.tpl"
```

docs/assets/templates/custom_banner.tpl

```
<p>
    <em>This box will disappear when printing</em>
    <span style="float: right"><a href="https://timvink.github.io/mkdocs-print-site-
plugin/">mkdocs-print-site-plugin</a></span>
</p>
<p>This page has combined all site pages into one. You can export to PDF using <b>File > Print >
Save as PDF</b>.</p>
<p>See also [export to PDF](https://timvink.github.io/mkdocs-print-site-plugin/how-to/export-
PDF.html) and [export to standalone HTML](https://timvink.github.io/mkdocs-print-site-plugin/how-
to/export-HTML.html).</p>
```

Adding configurable content for site banner

You might want to add some content to your cover page that's not yet specified in your mkdocs.yml file. Of course you could just hard-code it in your custom template file, but you could also make use of MkDocs's extra context feature, allowing you to use custom variables from your config file with {{ config.extra. }}.

mkdocs.yml

```
plugins:
    - print-site:
        add_print_site_banner: true
        print_site_banner_template: "docs/assets/templates/custom_banner.tpl"

extra:
    banner_message: "Save this page using File > Print > Save as PDF"
```

docs/assets/templates/custom_banner.tpl

```
        <p>
        <em>This box will disappear when printing</em>
        <span style="float: right"><a href="https://timvink.github.io/mkdocs-print-site-plugin/">mkdocs-print-site-
plugin</a></span>
    </p>
    <p>{{ config.extra.banner_message }}</p>
```

Exclude content from print

You might want to exclude certain parts of you website from the print site page. This can be useful when you don't want to include certain page, large images, tables, certain admonitions or appendixes to your site exports.

Ignoring elements in a page

mkdocs-print-site-plugin offers the CSS class .print-site-plugin-ignore, that will ignore certain elements.

The Attribute Lists extension, which is part of the standard Markdown library, allows to add HTML attributes and CSS classes to Markdown elements, and needs to be enabled in your mkdocs.yml.

To apply the .print-site-plugin-ignore class to an element you can use {: .print-site-plugin-ignore } on many different markdown elements, as explained in the attr_list docs. attr_list does not support all markdown elements (see limitations), but remember Markdown is a subset of HTML and anything which cannot be expressed in Markdown can always be expressed with raw HTML directly.

mkdocs.yml

```
plugins:
    - print-site

markdown_extensions:
    - attr_list
```

docs/example.md

```
/# Example page

This paragraph will not be part of the print site page.
{: .print-site-plugin-ignore }

![ignored image](some/path/image.png){: .print-site-plugin-ignore }

You can also use HTML to hide things from printing:
<span class="print-site-plugin-ignore">hello</span>
```

Ignoring admonitions

Adding a class to admonitions is not supported by attr_list. You can use the .print-site-plugin-ignore class directly on admonitions however.

```
    !!! info print-site-plugin-ignore

    As an example, this admonition will not be printed. Go have a look at the [print site page]
(/print_page.html) and you'll find it missing.
```

Ignoring an entire page

In the plugin configuration in mkdocs.yml you can specify a list of page source paths (one per line) that should not be included in the print page (excluded from processing by this plugin). This can be useful for example to exlude large appendixes that you only want to display on the web version. The source path of a page is relative to your docs/ folder. You can also use globs instead of full source paths. To exclude docs/subfolder/page.md specify in your mkdocs.yml a line under exclude: with - subfolder/page.md.

mkdocs.yml

```
plugins:
  - print-site:
      exclude:
        - index.md
        - subfolder/page.md
        - another_page.md
        - folder/*
```

PDF Plugin with-pdf

Source link GitHub: https://github.com/orzih/mkdocs-with-pdf

Sorce link pypi.org: https://pypi.org/project/mkdocs-with-pdf/

Installation

Install the package with pip:

```
pip install mkdocs-with-pdf
```

Enable the plugin in your mkdocs.yml:

```
    plugins:
        - with-pdf
```

Options

```
plugins:
    - with-pdf:
      author: Автор Документа # Автор документа
      copyright: ©2023 Your company # Копирайт
      # cover: false # Отключает отображение обложки (C1 – титульный лист)
      # back_cover: true # Включает отображение обложки (C4 – задняя часть). На ней может
отображаться QR-code со ссылкой на вашу документацию, см. ниже
      cover_title: Документация для Наш продукт # Заголовок на обложке C1
      cover_subtitle: Руководство по установке • Руководство администратора # Подзаголовок на
обложке C1
      cover_logo: docs/assets/logo_fav.png # Логотип на обложку
      custom_template_path: templates # Директория с кастомизированными CSS
      toc_title: Содержание # Название раздела с содержанием
      # heading_shift: false # Отключает вложенность заголовков. Просто раскомментируйте этот
параметр и посмотрите, как изменится содержание
      #toc_level: 3 # Уровни заголовков в содержании. К сожалению, максимальное значение – 3
      ordered_chapter_level: 3 # Максимальный уровень нумеруемых заголовков. К сожалению,
максимальное значение – 3
      #excludes_children: # Исключает отдельные документы из генерируемого PDF
      #    - 'release-notes/:upgrading' # см. секцию nav в этом файле
      #    - 'release-notes/:changelog' # см. секцию nav в этом файле
      #exclude_pages: # Исключает отдельные папки из PDF
      #    - 'ig/' # см. секцию nav в этом файле
      #    - 'ag/contribute/' # см. секцию nav в этом файле
      #convert_iframe: # Конвертирует тег iframe в картинку со ссылкой
      #    - src: IFRAME SRC #
      #      img: POSTER IMAGE URL #
      #      text: ALTERNATE TEXT #
      #    - src: ... #
      two_columns_level: 3 # Включает двухколоночную верстку, начиная с третьего уровня заголовков
      #render_js: true # Включает рендер результатов JS (требуется Headless Chrome)
      #headless_chrome_path: headless-chromium # путь к Headless Chrome
      output_path: ../site/Doc.pdf # Директория, в которую сохраняется PDF. В данном случае, чтобы
просмотреть документ достаточно будет добавить к адресу в браузере Docs.pdf
      # enabled_if_env: PDF_EXPORT # Отключение генерации PDF, например при разработке (чтобы
включить, комментируем эту строку)
      debug_html: true # Создает html-файл со всеми разделами. Незаменим при создании/
редактировании стилей. Для вывода в файл, а не в терминал используем команду  mkdocs build >
pdf_print.html
      # verbose: true # Подробное логирование процесса генерации
```

PDF Plugin pdf-export

The pdf-export plugin will export all markdown pages in your MkDocs repository as PDF files using **WeasyPrint**. The exported documents support many advanced features missing in most other PDF exports, such as a PDF Index and support for CSS paged media module.

Source link: https://pythonrepo.com/repo/zhaoterryy-mkdocs-pdf-export-plugin-python-documentation

MkDocs plugin that adds a page to your site combining all pages, allowing your site visitors to File > Print > Save as PDF the entire site.

GitHub: https://timvink.github.io/mkdocs-print-site-plugin/index.html

**Installation*

Install the package with pip:

```
pip install mkdocs-pdf-export-plugin
```

Enable the plugin in your mkdocs.yml:

```
plugins:
    - pdf-export:
        verbose: true
        media_type: print
        enabled_if_env: ENABLE_PDF_EXPORT
        combined: 1
```

Options

You may customize the plugin by passing options in mkdocs.yml:

`verbose` Setting this to true will show all WeasyPrint debug messages during the build. Default is false.

`media_type` This option allows you to use a different CSS media type (or a custom one like pdf-export) for the PDF export. Default is print.

`enabled_if_env` Setting this option will enable the build only if there is an environment variable set to 1. This is useful to disable building the PDF files during development, since it can take a long time to export all files. Default is not set.

`combined` Setting this to true will combine all pages into a single PDF file. All download links will point to this file. Default is false.

`combined_output_path`

This option allows you to use a different destination for the combined PDF file. Has no effect when combined is set to false. Default is pdf/combined.pdf.

`theme_handler_path`

This option allows you to specify a custom theme handler module. This path must be relative to your project root (See example below). Default is not set.

```
project-root
├── theme-handler.py
├── docs
├── mkdocs.yml
├── site
.
```

WeasyPrint

The pdf-export plugin will export all markdown pages in your MkDocs repository as PDF files using **WeasyPrint**. The exported documents support many advanced features missing in most other PDF exports, such as a PDF Index and support for CSS paged media module.

WeasyPrint Documentation: https://doc.courtbouillon.org/weasyprint/stable/first_steps.html#installation

Blog: How to build your GTK+ application on Windows

GitHub project: GTK+ for Windows Runtime Environment Installer: 64-bit

- Install GTK-for-Windows-Runtime-Environment-Installer from: https://github.com/tschoonj/GTK-for-Windows-Runtime-Environment-Installer/releases
- Create environment variable: **WEASYPRINT_DLL_DIRECTORIES = C:\Program Files\GTK3-Runtime Win64\bin**

> *gobject-2.0-0 error message: OSError: cannot load library 'gobject-2.0-0'*
>
> The error means that the gobject-2.0.0 library, which is part of GTK3+, cannot be found. Did you follow the installation instructions (https://doc.courtbouillon.org/weasyprint/stable/first_steps.html), which include installation of GTK3+? If no, do that. If yes, then the problem is, that the GTK3+ DLLs are not where Python is looking for them. For this, you need to add the directory containing the DLLs (e.g. C:\Program Files\GTK3-Runtime Win64\bin on Windows) to your PATH environment variable. That directory contains the relevant libgobject-2.0-0.dll library.

Publication

Project deploy

Source links:
MkDocs.Deploying your docs
MkDocs MagicSpace. Publish on GitHub Pages
Как разработать техническую документацию, которая точно будет работать. Часть 2. DocOps в действии

Local article
Как разработать техническую документацию, которая точно будет работать. Часть 2. DocOps в действии

Настройка сборки и публикации

Примечание: Прежде чем продолжить, убедитесь, что на вашей машине установлен Git.

```
git --version
```

Примечание: После установки Git вероятно потребуется перезапустить VS Code.

GitHub

1. В VS Code перейдите на вкладку Source Control (Ctrl+Shift+G) и нажмите на кнопку Initialize Repository.
2. На вкладке Source Control нажмите … › Remote › Add remote…, а затем выберите Add remote from GitHub.
3. После успешной авторизации в браузере и возвращения в VS Code выберите ранее созданный репозиторий и назначьте ему имя.
4. В поле Message вкладки Source Control введите комментарий, например init, и нажмите на кнопку Commit.

Примечание: Если попытка выгрузки коммита закончилась неудачно, выполните в терминале две следующие команды, указав в кавычках свои данные, и опубликуйте коммит вновь.

```
git config --global user.email "you@mail.ru"
git config --global user.name "Your Name"
```

1. На вкладке Source Control нажмите на появившуюся кнопку Publish Branch.

> Примечание: В некоторых ОС на этом этапе можно столкнуться с затруднениями, но их легко преодолеть, следуя рекомендациям разработчиков VS Code и GitHub, а также многоуважаемого сетевого комьюнити.

1. Открыв в браузере соответствующий репозиторий, убеждаемся, что файлы успешно загружены.
2. В GitHub переходим на вкладку **Actions** и нажимаем на кнопку **Configure** на карточке **Simple workflow**.
3. Переименуем файл *mkdoks.yml* в *main.yml*, вставим следующие строки и нажмем на кнопку **Commit changes…**.

```
name: ci
on:
  push:
    branches:
      - main
permissions:
  contents: write
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-python@v4
        with:
          python-version: 3.x
      - run: echo "cache_id=$(date --utc '+%V')" >> $GITHUB_ENV
      - uses: actions/cache@v3
        with:
          key: mkdocs-material-${{ env.cache_id }}
          path: .cache
          restore-keys: |
            mkdocs-material-
      - run: pip install mkdocs-material
      - run: mkdocs gh-deploy --force
```

1. В открывшемся окне просто нажимаем на кнопку Commit changes.
2. Переходим на вкладку Actions и убеждаемся, что пайплайн отработал без ошибок.
3. Переходим на вкладку Settings.
4. Выбираем пункт меню Pages.
5. Выбираем ветку для публикации.
6. Нажимаем на кнопку Save.
7. Спустя минуту нужно перезагрузить страницу (Cmd+R) и нажать на кнопку Visit site, появившуюся в ее верхней части.

Теперь после размещения в репозитории каждого нового коммита будет автоматически запускаться сборка и публикация справочного портала на GitHub Pages.

FAQ

Embedded files

Use without first slash `(/)`.

```
/--8<--
/./docs/20_syntax/21_tables.md
/./docs/20_syntax/21_diagrams.md
/--8<--
```

Use without slash `(/)` before `{`.

```
/{/{ read_raw('./docs/20_syntax/21_diagrams.md') /}/}
```

Useful plugins

mkdocs.yaml

```
plugins:
  - glightbox
  - search:
      lang:
        - en
        - ru

  - table-reader
  - print-site
```

- *Table reader*
  https://github.com/timvink/mkdocs-table-reader-plugin
  https://timvink.github.io/mkdocs-table-reader-plugin/
  `pip install mkdocs-table-reader-plugin`
- *Import data from Excel file*
  https://github.com/shshe/openpyxl
  https://openpyxl.readthedocs.io/en/stable/tutorial.html
  `pip install openpyxl`
- *Export PDF*
  mkdocs-print-site-plugin
  https://github.com/timvink/mkdocs-print-site-plugin
  https://timvink.github.io/mkdocs-print-site-plugin/print_page.html
  `pip install mkdocs-print-site-plugin`

- *Lightbox For Enlarge image*
  https://github.com/blueswen/mkdocs-glightbox
  https://github.com/biati-digital/glightbox#lightbox-options
  `pip install mkdocs-glightbox`

Links

DocOps

- Как разработать техническую документацию, которая точно будет работать. Часть 2. DocOps в действии
  https://habr.com/ru/companies/swordfish_security/articles/754780/
  Local article DocOps Part-2.

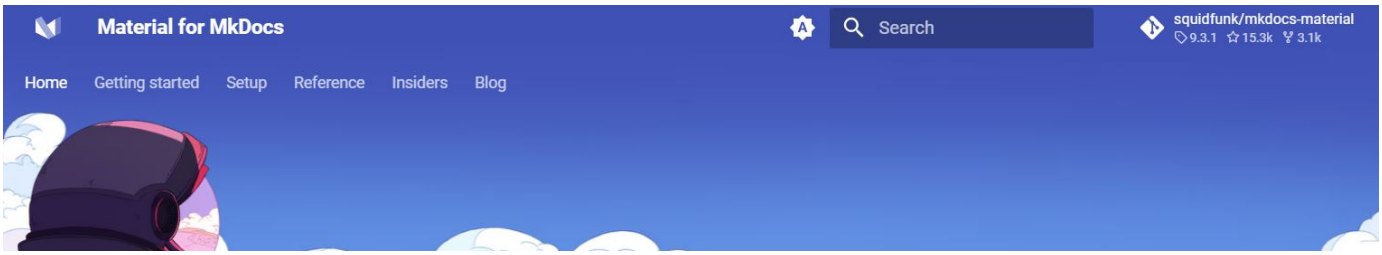
MarkDown

- Язык разметки Markdown: шпаргалка по синтаксису с примерами
  https://skillbox.ru/media/code/yazyk-razmetki-markdown-shpargalka-po-sintaksisu-s-primerami/#stk-19
  Local article Markdown_SkillboxMedia-1.

MkDocs Material

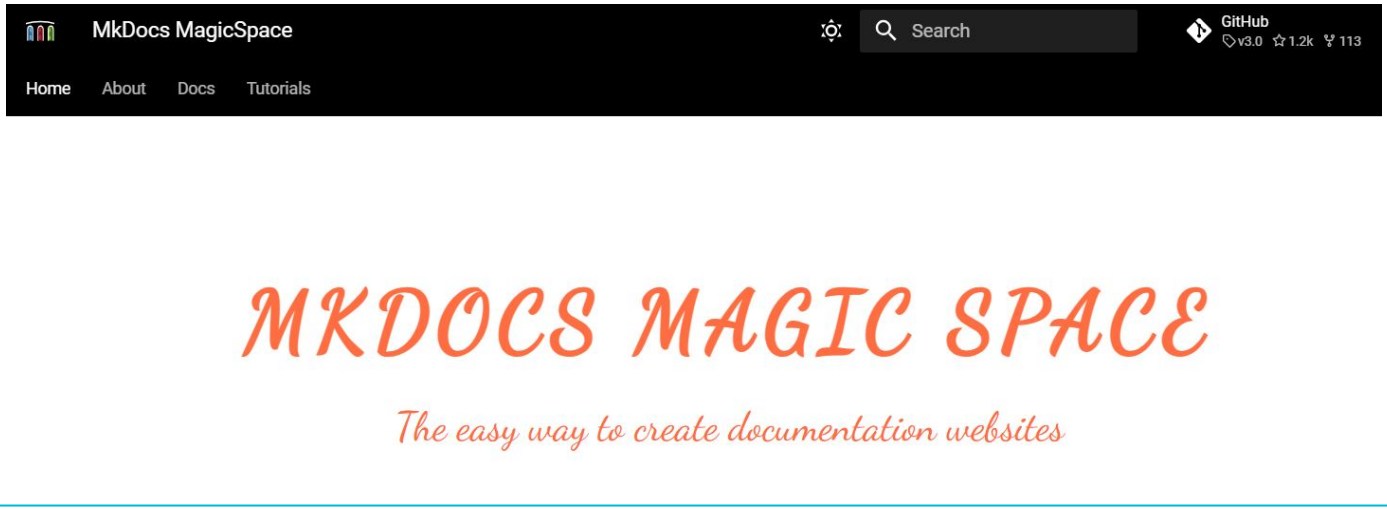
- **Material for MkDocs**
  Official site https://squidfunk.github.io/mkdocs-material/

Info



- **MKDOCS MAGIC SPACE**
  The easy way to create documentation websites
  https://mkdocs-magicspace.alnoda.org/

Info



- **PyMdown Extensions Documentation** https://facelessuser.github.io/pymdown-extensions/
- **MkDocs Material Extensions**
  https://pypi.org/project/mkdocs-material-extensions/ Local article MkDocs Material Extensions.
- **MkDocs TOC**
  https://pypi.org/project/mkdocs-toc-md/
- **Reproducible Reports with MkDocs**
  https://timvink.nl/reproducible-reports-with-mkdocs/