Diátaxis Material for MkDocs Guide

Welcome to a practical documentation project for **Material for MkDocs**, organized using the **Diátaxis** framework.

> **Source code**
>
> The source code for this documentation is available here

What is MkDocs

MkDocs is a static site generator designed for project documentation. It builds a website from Markdown files and a simple configuration file, making it easy to create and maintain documentation.

What is Material for MkDocs

Material for MkDocs is a popular theme and feature-rich extension for MkDocs. It adds a modern dark-themed design, improved search, new UI features and more.

What is Diátaxis

Diátaxis is a documentation framework that organizes content into four distinct types: tutorials, how-to guides, reference, and explanation.

Overview

As the Diátaxis framework has four quadrants, this documentation is organized accordingly:

Tutorials

Hands-on learning

2026-07-09 GitHub 

Tutorials

- Getting started
- First site
- Adding pages and navigation
- Configuring your site
- Using Markdown extensions

2026-07-09 GitHub 

Getting started

This guide explains how to install Material for MkDocs. You will set up a Python environment, install the required packages, and verify that the `mkdocs` command is ready to use.

Prerequisites

- Python 3.8+
- `pip`

Create a virtual environment first

Using a virtual environment is highly recommended to isolate project dependencies.

Linux / macOS

```
python3 -m venv .venv
source .venv/bin/activate
```

Windows

```
python.exe -m venv .venv
.venv\Scripts\activate
```

Install Material for MkDocs

The `mkdocs-material` package includes MkDocs, Material for MkDocs, and all of their dependencies. Install it from PyPI using `pip`:

```
pip install mkdocs-material
```

Verify the installation

Check that the `mkdocs` command is available and see which version you installed:

```
mkdocs --version
```

Fixing live reload

A recent dependency update in MkDocs can prevent the live-reloading from working. This is a known issue (see #4032). If you run `mkdocs serve` and your browser does not automatically refresh after you change a file, you most likely need to downgrade the `click` package:

```
pip install "click==8.2.1"
```

You are now ready to create your first site.

2026-07-09 GitHub

Your first site

This tutorial guides you through creating a new documentation site, starting the local development server, editing a page, and building the static output.

Create a new project

The `mkdocs new` command creates a new project directory with a default `mkdocs.yml` file and a single `index.md` page.

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

Your folder now contains:

```
my-project/
├─ docs/
│  └─ index.md
└─ mkdocs.yml
```

- `mkdocs.yml`: This is your main configuration file. You'll use it to set your site's title, choose a theme, and configure navigation.
- `docs/`: This directory contains all of your documentation content, written as Markdown files. `index.md` is the default home page.

Start the local server

The `mkdocs serve` command starts a local development server that watches for file changes and automatically rebuilds your site.

```
mkdocs serve
```

Open `http://127.0.0.1:8000` in your browser.

Expected result

You should see a simple documentation site with a single home page.

Edit your first page

Open `docs/index.md` in your editor and replace the default content:

docs/index.md

```
# Hello, world!

This is my first MkDocs site.
```

When you save the file, the browser will automatically reload to show the new content.

What to notice

The page updates immediately because the development server watches your files for changes.

Build the static site

When you are ready to publish, use the `mkdocs build` command to generate the static HTML, CSS, and JavaScript files.

```
mkdocs build
```

This creates a `site/` directory containing the complete, self-contained website.

You have now created, previewed, and built your first documentation site.

2026-07-09 GitHub 

Adding pages and navigation

This tutorial shows you how to add new pages to your site and organize them into a structured navigation menu.

Create new pages

First, create a few new Markdown files inside your `docs` directory. For better organization, it's a good practice to group related pages into sub-folders.

Create a new folder `docs/features` and add two files:

`docs/features/search.md`:

```
# Search

Our documentation has built-in search.
```

`docs/features/themes.md`:

```
# Themes

We use the Material for MkDocs theme.
```

Your project structure should now look like this:

```
my-project/
├─ docs/
│  ├─ features/
│  │  ├─ search.md
│  │  └─ themes.md
│  └─ index.md
└─ mkdocs.yml
```

If your `mkdocs serve` process is still running, you will notice that these new pages are not yet visible in the site's navigation.

Configure the navigation

To add the new pages to your site's navigation, you need to edit the `nav` section in your `mkdocs.yml` file.

Open `mkdocs.yml` and add the `nav` configuration:

mkdocs.yml

```yaml
site_name: My Docs
nav:
  - Home: index.md
  - Features:
    - Search: features/search.md
    - Themes: features/themes.md
```

Functionality

The top-level key `nav` tells MkDocs the structure of the navigation on the left side. With `Home: index.md`, it creates a top-level link named "Home" that points to `index.md`. `Features` will create a top-level navigation section, which has two subpages: `Search` and `Themes` with their own respective paths to markdown files.

Expected result

Your site now has a "Features" dropdown menu in the navigation bar, containing links to your "Search" and "Themes" pages.

You have now learned how to expand your site with new pages and control their order and structure in the navigation.

2026-07-09 GitHub 

Configuring your site

This tutorial explains how to configure your project by editing the `mkdocs.yml` file. You will learn how to set basic site information, apply the Material for MkDocs theme, and enable some of its key features.

Set basic site information

Let's configure the site by setting these core options in `mkdocs.yml`. Update your file to look like this:

mkdocs.yml

```yaml
site_name: My Docs
site_url: https://example.com/
site_description: A short description of my documentation site.
repo_url: https://github.com/your-username/your-repo

nav:
  - Home: index.md
```

- `site_name`: The title of your documentation site.
- `site_url`: The full public URL where your site will be hosted.
- `site_description`: A short summary used for search engine results and metadata.
- `repo_url`: The URL of your source code repository (e.g., GitHub, GitLab).

Apply the Material for MkDocs theme

Now, let's switch from the default theme to Material for MkDocs.

Add the `theme` section to your `mkdocs.yml`:

mkdocs.yml

```yaml
site_name: My Docs
# ... (other settings)

theme:
  name: material
```

Expected result

If `mkdocs serve` is running, your site will automatically reload with a new dark look. You are now using the Material for MkDocs theme.

Enable theme features

Material for MkDocs includes many features that can be enabled under the `theme.features` key. Here we will enable some features.

Update your `theme` configuration:

mkdocs.yml

```yaml
# ... (other settings)

theme:
  name: material
  features:
    - navigation.tabs
    - navigation.footer
    - navigation.top
```

- `navigation.tabs`: Turns the top-level navigation into tabs.
- `navigation.footer`: Adds "Previous" and "Next" page links in the footer.
- `navigation.top`: Adds a "Back to top" button that appears when you scroll down.

Enable and configure search

Search is a built-in plugin that is enabled by default. However, as soon as you add the `plugins` key to your configuration, you must explicitly include `search` to keep it.

Add the `plugins` section to your `mkdocs.yml`:

mkdocs.yml

```yaml
# ... (other settings)

plugins:
  - search:
```

- `plugins`: The list of plugins to enable.

Change the color palette

You can easily change the site's colors using the `palette` option.

Update your `theme` configuration again:

mkdocs.yml

```yaml
# ... (other settings)

theme:
  name: material
  features:
    - navigation.tabs
    - navigation.footer
    - navigation.top
  palette:
    scheme: slate
    primary: blue

plugins:
  - search:
```

- `scheme: slate`: Switches the site to a dark mode theme.
- `primary: blue`: Sets the primary accent color to blue.

You have now learned the basics of configuring your site, applying a theme, and customizing its features and appearance.

2026-07-09 GitHub 

Markdown extensions

This tutorial explains how to use some of Material for MkDocs most useful Markdown extensions to make your content more readable and interactive. You will learn how to add admonitions, styled code blocks, content tabs, collapsible blocks, mermaid diagrams, and abbreviations.

MkDocs uses Python-Markdown, which implements John Gruber's original Markdown specification and also supports extensions for features like footnotes or admonitions. Furthermore, Material for MkDocs uses Pymdown Extensions, a collection of extensions for Python-Markdown to add even more capabilities.

Prerequisites

To use these extensions, enable the corresponding Markdown extensions in your `mkdocs.yml` file. For a full reference of all available extensions and their configuration, see the Material for MkDocs reference. Make sure your `markdown_extensions` section looks like this:

mkdocs.yml

```yaml
markdown_extensions:
  - admonition
  - abbr
  - pymdownx.details
  - pymdownx.superfences:
      custom_fences:
        - name: mermaid
          class: mermaid
          format: !!python/name:pymdownx.superfences.fence_code_format
  - pymdownx.tabbed:
      alternate_style: true
```

Admonitions

Admonitions are styled call-out blocks that are perfect for notes, warnings, or tips.

How to use them

Create an admonition by using `!!!` followed by a type qualifier.

```
!!! note
    This is a note.

!!! warning "Don't forget"
    This is a warning.
```

Example:

Quick fix

This is a tip.

Types of admonitions

- `note`
- `info`
- `tip`
- `question`
- `warning`
- `danger`
- `bug`
- `example`

Code blocks

You can add syntax highlighting and titles to your code blocks.

How to use them

To enable syntax highlighting, add the language identifier after the opening backticks. To add a title, use `title="Your Title"` after the language.

````
```python title="my_script.py"
import os

def main():
    print("Hello, world!")

if __name__ == "__main__":
    main()
```
````

Example:

my_script.py

```python
import os

def main():
    print("Hello, world!")

if __name__ == "__main__":
    main()
```

Content tabs

Content tabs are useful for grouping related but distinct information, such as installation instructions for different operating systems.

How to use them

Use `=== "Tab Title"` to create each tab.

````
=== "Linux / macOS"
    ```bash
    python3 -m venv .venv
    source .venv/bin/activate
    ```

=== "Windows"
    ```powershell
    python.exe -m venv .venv
    .venv\Scripts\activate
    ```
````

Example:

Linux / macOS

```
python3 -m venv .venv
source .venv/bin/activate
```

Windows

```
python.exe -m venv .venv
.venv\Scripts\activate
```

Collapsible blocks

You can hide less critical or lengthy content inside collapsible blocks, similar to admonitions.

How to use them

Use `???` for a block that is closed by default or `???+` for a block that is open by default.

```
??? note "Click to expand"
    This content is hidden by default. It's useful for supplementary information
or long log outputs.

???+ tip "This one is open"
    This content is visible by default but can be collapsed by the user.
```

Example:

Build log (collapsed)

```
INFO   Building documentation...
DEBUG  Reading file: docs/index.md
ERROR  Missing reference: api/usage.md
```

Mermaid diagrams

Mermaid diagrams are useful for visualizing processes, flows, and relationships directly in Markdown. In Material for MkDocs, they are enabled with a custom Superfences block.

How to use them

Add the Mermaid fence to your `mkdocs.yml` file:

mkdocs.yml

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

Example:

```
sequenceDiagram
    participant Client
    participant Server
    Client->>Server: POST /api/auth/login (username, password)
    Note over Server: Validate credentials
    Server-->>Client: 200 OK (Issuing JWT)
    Note over Client: Store token
    Client->>Server: GET /api/user with header Authorization: Bearer <token>
    Note over Server: Verify Signature & Expiration
    Server-->>Client: 200 OK (Data)
```

Abbreviations

Abbreviations let you define a short term once and show its full meaning when users hover over it.

How to use them

Enable `abbr` in `mkdocs.yml`, then write the abbreviation definitions at the end of the page in the syntax `*[Abbreviation]: Definition`.

```
MkDocs and Mermaid are used throughout this guide.

*[MkDocs]: A static site generator geared towards project documentation.
*[Mermaid]: A diagram syntax for creating flowcharts and sequence diagrams.
```

Example:

MkDocs and Mermaid are used throughout this guide.

2026-07-09 GitHub 

How-To

- Generate a PDF file out of MkDocs
- Deploy to GitHub Pages
- Dark/Light Mode Toggle
- Show information about pages

2026-07-09 GitHub

Deploy MkDocs to Github Pages

This page describes how to deploy MkDocs documentation to Github Pages.

Github pages

Github Pages is a static site hosting service that allows to publish documentation directly from a GitHub repository.

> **Plan Availability and Usage**
>
> GitHub Pages is available for free in public repositories. GitHub Pages is not allowed for commercial use. See GitHub Pages limits for more details.

Deploying to Github Pages

To deploy to Github Pages, Github Actions is used. This workflow automatically builds and deploys the documentation on pushes or manuall triggers.

Setting up GitHub Actions

Create a new workflow file at `.github/workflows/deploy-docs.yml` in the repository with the following content:

```yaml
name: deploy-docs

on: # (1)
  workflow_dispatch:
  push:
    branches:
      - master
    paths:
      - 'docs/**'
      - 'mkdocs.yml'

permissions:
  contents: write

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Git Credentials
        run: |
          git config user.name github-actions[bot]
          git config user.email 41898282+github-
actions[bot]@users.noreply.github.com
      - uses: actions/setup-python@v5
        with:
          python-version: 3.x
      - run: echo "cache_id=$(date --utc '+%V')" >> $GITHUB_ENV
      - uses: actions/cache@v4
        with:
          key: mkdocs-material-${{ env.cache_id }}
          path: ~/.cache
          restore-keys: |
            mkdocs-material-
      - run: pip install -r requirements.txt
      - run: mkdocs gh-deploy --force
```

1. It can be freely choosen, when to trigger the workflow

Enabling GitHub Pages

After the first successful workflow run, the repository needs to be configured:

1. Go to the repository Settings → Pages
2. Under "Build and deployment", select "Deploy from a branch"
3. Choose `gh-pages` as the branch and `/ (root)` as the folder

The documentation will now be automatically deployed to GitHub Pages at `https://<username>.github.io/<repository>/` whenever the workflow is triggered.

Deployment

A trigger will start the Github Actions workflow. In this configuration it is either manual or by a push. MkDocs then builds the documentation. After the build, the workflow pushes the generated site to the `gh-pages` branch, which is why write permission for repository is required. Finally, GitHub Pages serves the site from that branch.

2026-07-09 GitHub 

Generate a PDF file out of MkDocs

This page describes how to generate a PDF file from the MkDocs documentation using mkdocs-exporter.

PDF Export from MkDocs

MkDocs does not natively support PDF generation. However, the MkDocs plugins provides several solutions to export the documentation as a PDF. Popular plugins include `mkdocs-pdf-export-plugin` and `mkdocs-with-pdf`, However this how-to focuses on `mkdocs-exporter`.

Installing mkdocs-exporter

First, install the mkdocs-exporter package using pip:

```
pip install mkdocs-exporter
```

Additionally, Playwright needs to install browser binaries that it uses to render pages. Run the following to install those browsers:

```
playwright install
```

Configuring mkdocs-exporter

Add the plugin to the `mkdocs.yml` configuration file with the basic settings:

```yaml
plugins:
  - exporter:
      formats:
        pdf:
          enabled: true
          concurrency: 8
```

The `concurrency` setting controls how many pages are rendered in parallel, speeding up the PDF generation process.

Aggregating into a Single PDF

By default, the exporter generates individual PDF files for each page. To combine all pages into a single PDF document, enable the aggregator option:

```yaml
plugins:
  - exporter:
      formats:
        pdf:
          enabled: true
          concurrency: 8
          aggregator:
            enabled: true
            covers: all
            output: output.pdf
```

Generating the PDF

After configuration, generate the PDF by running:

```
mkdocs build
```

For the configured aggregation, the combined PDF will be located at `site/output.pdf`. Individual page PDFs are also generated and placed in the same `site/` directory alongside the static HTML files.

2026-07-09 GitHub 

Implementing a Dark/Light Mode Toggle

This page describes how to add a dark and light mode toggle to Material for MkDocs documentation.

Color Scheme Preferences

Material for MkDocs supports multiple color schemes, allowing users to choose between light and dark modes based on their preference. The schemes can be configured to automatically use the users system preference while providing a manual toggle to switch between them.

Setting up the Color Scheme Toggle

To add a dark and light mode toggle to the site, configure the `palette` section in `mkdocs.yml` file. This section defines the available color schemes and how users can switch between them.

Basic Configuration

Add the palette configuration to the `theme` section in `mkdocs.yml`:

```yaml
theme:
  name: material

  palette:
    - media: "(prefers-color-scheme: light)"
      scheme: default
      primary: white
      accent: blue
      toggle:
        icon: material/weather-night
        name: Dark mode
    - media: "(prefers-color-scheme: dark)"
      scheme: slate
      primary: black
      accent: blue
      toggle:
        icon: material/weather-sunny
        name: Light mode
```

2026-07-09 GitHub 

Show information about a page

This page shows how to add page information to MkDocs site with two plugins: one for showing who edited a page and one for showing when it was last changed.

Show who edited the page

Use mkdocs-git-committers-plugin-2 to display the contributors who edited a page. The infomration will be displayed in the footer of the page.

Add the plugin to `mkdocs.yml` like this:

```
plugins:
  - git-committers:
      repository: khde/diataxis-material-mkdocs-guide
      branch: master
      cache_dir: .cache/plugin/git-committers/
```

The plugin simply reads the Git history and shows the people who contributed to the page.

Show when the page was last edited

Use mkdocs-git-revision-date-localized-plugin to show the last edit date on each page.

Add it to `mkdocs.yml` like this:

```
plugins:
  - git-revision-date-localized:
      locale: en
      timezone: Europe/Amsterdam
      type: timeago
```

2026-07-09 GitHub 

Reference

- Commands
- YAML Configuration
- Glossary
- References

2026-07-09 GitHub

Command reference

This page provides a quick reference to the main MkDocs commands. For the complete set of options, refer to your local help output: `mkdocs --help` and `mkdocs <command> --help`.

`mkdocs new`

Creates a new project directory with a minimal `mkdocs.yml` and a starter docs page.

```
mkdocs new my-project
```

`mkdocs serve`

Runs a local development server and rebuilds the site on file change.

```
mkdocs serve
```

`mkdocs build`

Builds the static site output (defaults to the `site/` directory).

```
mkdocs build
```

`mkdocs gh-deploy`

Builds the site and publishes it to a Git branch, intended for GitHub Pages.

```
mkdocs gh-deploy
```

For a detailed how-to on publishing MkDocs sites to GitHub Pages, refer to this guide here.

2026-07-09 GitHub 

YAML configuration

Reference configuration

```yaml
site_name: Your Project Docs # (1)
site_description: My description here # (2)
site_url: https://<your-user>.github.io/<your-repo>/ # (3)
repo_url: https://github.com/<your-user>/<your-repo>/ # (4)
edit_uri: edit/main/docs/ # (5)

docs_dir: docs # (6)

nav:
  - Home: index.md # (7)
  - Getting started:
      - getting-started/index.md # (8)
      - Installation: getting-started/installation.md
      - Usage: getting-started/usage.md
  - About: about.md

theme:
  name: material # (9)
  icon:
    repo: fontawesome/brands/github # (10)
    logo: material/book-open-page-variant # (11)

  features:
    - content.action.view # (12)
    - content.code.copy # (13)
    - content.code.annotate # (14)
    - navigation.footer # (15)
    - navigation.top # (16)
    - navigation.indexes # (17)
    - navigation.tabs # (18)
    - navigation.tabs.sticky # (19)
    - search.suggest # (20)
    - search.highlight # (21)
    - search.share # (22)

  palette:
    - media: "(prefers-color-scheme: light)" # (23)
      scheme: default
      primary: white
      accent: blue
      toggle:
        icon: material/weather-night
        name: Dark mode
    - media: "(prefers-color-scheme: dark)" # (24)
      scheme: slate
      primary: black
      accent: blue
      toggle:
        icon: material/weather-sunny
        name: Light mode

  font:
    text: Inter # (25)
```

```
    code: Google Sans Code # (26)markdown_extensions:  - admonition # (27)  -
attr_list # (28)  - md_in_html # (29)  - pymdownx.details # (30)  -
pymdownx.superfences # (31)plugins:  - search # (32)extra:  generator: false #
(33)
```

1. `site_name`: Sets the site's main title, appearing in the header and browser tab.
2. `site_description`: Provides a brief summary for search engine results and metadata.
3. `site_url`: Defines the canonical public URL for the site.
4. `repo_url`: Links to the source code repository, often displayed as an icon.
5. `edit_uri`: Constructs the "Edit this page" link for each page.
6. `docs_dir`: Specifies the directory containing the Markdown source files.
7. `nav`: Defines the site's navigation structure and page order.
8. Section `index.md` pages serve as stable, linkable landing pages for navigation sections.
9. `theme.name`: Specifies the theme to use for the site's appearance.
10. `theme.icon.repo`: Sets the icon for the repository link.
11. `theme.icon.logo`: Sets the site's logo icon, displayed in the header.
12. `features.content.action.view`: Adds a button to view the source of the current page.
13. `features.content.code.copy`: Adds a copy button to all code blocks.
14. `features.content.code.annotate`: Enables code annotations for explaining code snippets.
15. `features.navigation.footer`: Adds previous/next page links in the footer.
16. `features.navigation.top`: Adds a "back to top" button.
17. `features.navigation.indexes`: Makes section index pages act as landing pages in the navigation.
18. `features.navigation.tabs`: Renders top-level navigation items as tabs.
19. `features.navigation.tabs.sticky`: Makes the navigation tabs stick to the top of the viewport.
20. `features.search.suggest`: Provides search query auto-completion.
21. `features.search.highlight`: Highlights search terms on result pages.
22. `features.search.share`: Adds a button to share a search query.
23. `palette` (light): Defines the color scheme for light mode.
24. `palette` (dark): Defines the color scheme for dark mode.
25. `font.text`: Sets the main font for body text.

26. `font.code`: Sets the font for code blocks.
27. `markdown_extensions.admonition`: Enables styled blocks for notes, warnings, and tips.
28. `markdown_extensions.attr_list`: Allows adding HTML attributes to Markdown elements.
29. `markdown_extensions.md_in_html`: Allows processing Markdown syntax inside HTML blocks.
30. `markdown_extensions.pymdownx.details`: Enables collapsible content blocks.
31. `markdown_extensions.pymdownx.superfences`: Enables nesting code blocks and other advanced features.
32. `plugins.search`: Enables the built-in search functionality.
33. `extra.generator`: Hides the "Made with Material for MkDocs" notice in the footer.

2026-07-09 GitHub 

Glossary

Admonition

: A styled call-out block used for notes, tips, warnings, and other highlighted information.

Diátaxis

: A documentation framework that organises content into tutorials, how-to guides, reference, and explanations.

GitHub Actions

: GitHub's automation service for running workflows such as building and deploying documentation.

GitHub Pages

: A static hosting service for publishing a website directly from a GitHub repository.

Markdown

: A lightweight markup language used to write readable plain-text documentation.

Markdown extension

: An add-on that expands Markdown with extra features such as admonitions, tabs, or diagrams.

MkDocs

: A static site generator for project documentation, written in Python.

Material for MkDocs

: A theme for MkDocs that provides a modern look and many features such as search, navigation, and extensions.

Mermaid

A diagram written in Mermaid syntax and rendered from text, useful for flowcharts, sequence diagrams, and more.

Python-Markdown

The Markdown parser used by MkDocs to convert Markdown content into HTML.

Static site generator

A tool that builds a website (HTML, CSS, JavaScript, etc.) from source files such as Markdown and configuration data.

2026-07-09 GitHub 

References

Material for MkDocs - squidfunk.github.io/mkdocs-material

MkDocs - mkdocs.org

Python-Markdown - python-markdown.github.io

Pymdown Extensions - facelessuser.github.io/pymdown-extensions

MkDocs Exporter plugin - adrienbrignon.github.io/mkdocs-exporter/getting-started

Git Committers plugin - github.com/ojacques/mkdocs-git-committers-plugin-2

Git Revision Date Localized plugin - timvink.github.io/mkdocs-git-revision-date-localized-plugin/index.html

2026-07-09 GitHub 

Explanation

Evaluation of MkDocs

Alternatives to MkDocs

2026-07-09 GitHub 

Evaluation of MkDocs

This page evaluates Material for MkDocs as a tool for creating documentations. It highlights major advantages and disadvantage to help considering to use it before adopting.

Pros

- Easy setup: MkDocs is straightforward to install and configure
- Markdown extensions support: Material for MkDocs adds many visual elements as theme features
- Extensive plugins: wide range of plugins for PDF export, metadata, etc.

Cons

- Maintenance concerns: MkDocs is no longer activly maintained, which is a significant red flag
- Core features rely on extensions and plugins: many common features are not built into the core framework and must be added through plugins

Conclusion

MkDocs can be a good choice for a simple documentation system with a clean appearance and a large plugin ecosystem. It is very practical for a fast setup and flexibility,

At the same time, it is a unmaintained project, so bugs and new features will not be going into this project anytime soon. While there is a MkDocs 2.0 in development, it is not a drop-in replacement for MkDocs. It has no backwards compatiblity. It also is not community driven.

In summary, MkDocs is a strong option for many documentation projects, but has also some bug issues.

For alternatives option of MkDocs, see Alternatives.

2026-07-09 GitHub 

Alternatives to MkDocs

Given that MkDocs is no longer actively maintained, concerns about plugin dependencies, or looking for different trade-offs, several alternatives exist for building static documentation sites. Some of them are:

Zensical

Zensical is a static site generator, developed by the creator of Material for MkDocs. While not being a plugin anymore, it is fully backwards compatible to the mkdocs.yml file and serves as a replacement for MkDocs.

Sphinx

Sphinx is a mature documentation tool widely used in the Python community.

Jekyll

Jekyll is a static site generator that powers GitHub Pages.

Docusaurus

Docusaurus is built by Meta and combines React and modern web standards for documentation. It is component based and builds single-page applications.

Quarkdown

Quarkdown is a new promosing typesetting system based on Markdown that also tries to support and simplify LateX. It uses its own Markdown flavour.

2026-07-09 GitHub 

Download

Download the complete documentation as a single PDF:

Download PDF

2026-07-09 GitHub

About this project

Overview

This project is a guide to **Material for MkDocs** documentation using the **Diátaxis framework**. It showcases how to organize content into the four pillars of Diátaxis: tutorials, how-to guides, reference, and explanations.

Purpose

This is a **study project** applying the Diátaxis framework to Material for MkDocs.

Credits

This documentation project is made possible by the contributions of several individuals and communities:

- **Diátaxis Framework**: Created by Daniele Procida
- **MkDocs**: Developed by the MkDocs community
- **Material for MkDocs**: Developed by Martin Donath

All original documentation, guidance, and frameworks are credited to their respective authors. This project serves as an educational implementation and study resource.

2026-07-09 GitHub