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