Advanced. Theme Material

On line links

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

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" }
```

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:
```

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?

Generic grid

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

=== "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

``` title="Content tabs"
=== "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
```

</div>
```

Unordered list **Ordered list**

- Sed sagittis eleifend rutrum
- Donec vitae suscipit est

```
Content tabs

=== "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
```

vitae suscipit est

3. Nulla tempor lobortis orci

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;
}
```

```
#include
<iostream>

int
main(void) {
  std::cout <<
"Hello
```

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 Ordered list

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

1. Sed sagittis eleifend rutrum

2. Donec vitae suscipit est

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 **Ordered List**

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

1. Sed

   eleifend rutrum

2. Donec vitae

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 **Tab 2**

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

1. 🙋‍♂️ I'm an annotation!

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!

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 **Center** **Right**

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

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

sortable, you can add tablesort, which is natively integrated with
lso work with instant loading via additional JavaScript:

```
| Method      |
Description                          |
```

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

```
close:     Delete resource |
```

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

Method	Description
`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   Import data from Excel file   Import data from other file types

```
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/assets/files/data.csv') /}/}

Result from data.csv

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

```
pip install openpyxl

./docs/assets/files/data.xlsx

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

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

For document with sheets.

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

Result from data.xlsx

Column 1	Column 2	Colomn 3
AA	AB	AC

Column 1	Column 2	Colomn 3
BA	BB	BC
CA	CB	CC

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

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 **Disable**

```
# Code
block
content
```

```
# Code
block
content
```

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]
```
~~~

```python
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]
```

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<-- "./20_syntax/data.src"`
```
~~~

data.src

```

```

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

[Send :fontawesome-solid-paper-plane:](#){ .md-button }

Send