
Flask-Misaka Documentation

Release 1.0.0

David Baumgold

Jan 17, 2019

Contents

1	Installation	3
2	Usage	5
3	API	7
4	Options	9
	Python Module Index	11

Flask-Misaka provides a pleasant interface between the [Flask](#) web framework and the [Misaka Markdown](#) parser.¹

¹ (Technically, [Misaka](#) is just a Python binding to the [Hoedown](#) library, which is written in C.)

CHAPTER 1

Installation

Install the extension with:

```
$ pip install Flask-Misaka
```


CHAPTER 2

Usage

Just import the `markdown()` function and use it!

```
>>> from flask_misaka import markdown
>>> markdown("A simple example.")
Markup(u'<p>A <em>simple</em> example.</p>\n')
```

To use Markdown in your templates, you just need to import the `Misaka` class and wrap your Flask instance with it:

```
from flask import Flask
from flask_misaka import Misaka

app = Flask(__name__)
Misaka(app)
```

or use the application factory pattern:

```
md = Misaka()
app = Flask(__name__)
md.init_app(app)
```

And then the markdown filter will be available in your Jinja2 templates. You can pass variables in your template through it:

```
{{ text|markdown }}
```

Or, you can use the `filter` tag to write your template directly in Markdown and have Jinja dynamically interpret it for you!

```
{% filter markdown %}
I'm writing my templates in *Markdown!*
{% endfilter %}
```


`flask_misaka.markdown(text, renderer=None, **options)`

Parses the provided Markdown-formatted text into valid HTML, and returns it as a `flask.Markup` instance.

Parameters

- **text** – Markdown-formatted text to be rendered into HTML
- **renderer** – A custom misaka renderer to be used instead of the default one
- **options** – Additional options for customizing the default renderer

Returns A `flask.Markup` instance representing the rendered text

class `flask_misaka.Misaka(app=None, renderer=None, **defaults)`

`__init__(app=None, renderer=None, **defaults)`

Set the default options for the `render()` method. If you want the `markdown` template filter to use options, set them here.

A custom misaka renderer can be specified to be used instead of the default one.

`init_app(app)`

Registers the rendering method as template filter.

Parameters `app` – a `flask.Flask` instance.

render (`text`, `**overrides`)

It delegates to the `markdown()` function, passing any default options or renderer set in the `__init__()` method.

The `markdown` template filter calls this method.

Parameters

- **text** – Markdown-formatted text to be rendered to HTML
- **overrides** – Additional options which may override the defaults

Returns A `flask.Markup` instance representing the rendered text

CHAPTER 4

Options

Misaka is very customizable, and [supports many Markdown extensions](#). Flask-Misaka provides a nicer API for these extensions. All functions in the public API (except `Misaka.init_app()`) accept the following boolean arguments, all of which default to False:

Table 1: Flask-Misaka options

Option Name	Description
<code>autolink</code>	Parse links even when they are not enclosed in <code><></code> characters. Autolinks for the <code>http</code> , <code>https</code> and <code>ftp</code> protocols will be automatically detected. Email addresses are also handled, and <code>http</code> links without protocol, but starting with <code>www</code> .
<code>fenced_code</code>	Blocks delimited with 3 or more <code>~</code> or backticks will be considered as code, without the need to be indented. An optional language name may be added at the end of the opening fence for the code block.
<code>underline</code>	Treat text surrounded by underscores (like <code>_this_</code>) as underlined, rather than emphasized.
<code>highlight</code>	Treat text surrounded by double equal signs (like <code>==this==</code>) as highlighted.
<code>quote</code>	Parse inline quotes (like <code>"this"</code>). This allows the renderer to control how they are rendered.
<code>math</code>	Parse inline LaTeX-style math blocks (like <code>\$\$this\$\$</code>).
<code>math_explicit</code>	Parse inline LaTeX-style math blocks with a single dollar, e.g. <code>\$x + y = 3\$</code>
<code>disable_indented_code</code> or <code>no_indented_code</code>	Ignore indented code blocks
<code>no_intra_emphasis</code>	Do not parse emphasis inside of words. Strings such as <code>foo_bar_baz</code> will not generate <code></code> tags.
<code>space_headers</code>	A space is always required between the hash at the beginning of a header and its name, e.g. <code>#this is my header</code> would not be a valid header.
<code>strikethrough</code>	Two <code>~</code> characters mark the start of a strikethrough, e.g. <code>this is ~~good~~ bad</code> .
<code>superscript</code>	Parse superscripts after the <code>^</code> character; contiguous superscripts are nested together, and complex values can be enclosed in parenthesis, e.g. <code>this is the 2^(nd) time</code> .
<code>tables</code>	Parse PHP-Markdown tables .
<code>hard_wrap</code> or <code>wrap</code>	Insert HTML <code>
</code> tags inside on paragraphs where the origin Markdown document had newlines (by default, Markdown ignores these newlines).
<code>footnotes</code>	Parse Markdown footnotes.
<code>escape</code>	Escape all HTML tags, regardless of what they are.
<code>skip_html</code> or <code>no_html</code>	Do not allow any user-inputted HTML in the output.
<code>use_xhtml</code> or <code>xhtml</code>	Output XHTML-conformant tags.
<code>smartypants</code>	Post-process rendered markdown text with SmartyPants .

Any option that starts with `no_` can also be passed as its inverse set to `False`. For example, `no_html=True` and `html=False` have exactly the same effect, just as `no_intra_emphasis=True` and `intra_emphasis=False` have exactly the same effect.

Note: To override an option, you must use exactly the same option name as you used to originally set the option. If you set `html=False` as a default, you must override it with `html=True`: using `no_html=False` or `skip_html=False` will not work, even though they all refer to the same thing.

f

flask_misaka, ??

Symbols

`__init__()` (*flask_misaka.Misaka method*), [7](#)

F

`flask_misaka` (*module*), [1](#)

I

`init_app()` (*flask_misaka.Misaka method*), [7](#)

M

`markdown()` (*in module flask_misaka*), [7](#)

`Misaka` (*class in flask_misaka*), [7](#)

R

`render()` (*flask_misaka.Misaka method*), [7](#)