MarkdownSubscript Documentation

Release 2.1.1

Andrew Pinkham

Dec 02, 2018

Contents:

1 Installation **3**
- 1.1 Stable release . . . 3
- 1.2 From source . . . 3

2 Usage **5**
- 2.1 Python . . . 5
- 2.2 Command Line . . . 5

3 Contributing **7**
- 3.1 Types of Contributions . . . 7
- 3.2 Get Started! . . . 8
- 3.3 Pull Request Guidelines . . . 9

4 Package Release Guidelines **11**

5 Credits **13**
- 5.1 Development Lead . . . 13
- 5.2 Contributors . . . 13

6 History **15**
- 6.1 2.1.1 (2018-10-09) . . . 15
- 6.2 2.1.0 (2018-10-08) . . . 15
- 6.3 2.0.0 (2017-04-17) . . . 15
- 6.4 1.0.1 (2014-10-17) . . . 16
- 6.5 1.0.0 (2014-07-29) . . . 16

This Python package is an extension to the Python Markdown project which adds the ability to subscript text. To do so, the character ~ becomes a Markdown tag for text meant to be subscripted, and is replaced with the HTML `sub` tag.

For example, the extension transforms the text directly below into the HTML shown after it.

```
Water is H~2~O.
```

```
<p>Water is H<sub>2</sub>O.</p>
```

The code is Simplified (2 Clause) BSD license and is available on Github.

CHAPTER 1

Installation

1.1 Stable release

The easiest way to install Markdown Subscript is to use pip.

```
$ python -m pip install MarkdownSubscript
```

This will install the latest stable version. If you need an older version, you may pin or limit the requirements.

```
$ python -m pip install 'MarkdownSubscript==2.1.0'
$ python -m pip install 'MarkdownSubscript>=2.0.0,<3'
```

If you don't have pip installed, this Python installation guide can guide you through the process.

1.2 From source

The source files for Markdown Subscript can be downloaded from the Github repo.

You may use pip to install the latest version:

```
$ python -m pip install git+git://github.com/jambonrose/markdown_subscript_extension.
↪git
```

Alternatively, you can clone the public repository:

```
$ git clone git://github.com/jambonrose/markdown_subscript_extension
```

Or download the tarball:

```
$ curl -OL https://github.com/jambonrose/markdown_subscript_extension/tarball/
↪development
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 2

Usage

- *Python*
- *Command Line*

2.1 Python

```
>>> from markdown import markdown
>>> from mdx_subscript import SubscriptExtension
>>> text = "The molecular composition of water is H~2~O."
>>> markdown(text, extensions=[SubscriptExtension()])
'<p>The molecular composition of water is H<sub>2</sub>O.</p>'
```

You may also refer to the extension by module name or short module name.

```
>>> markdown(text, extensions=['mdx_subscript'])
>>> markdown(text, extensions=['subscript'])
```

Note: In older versions of Markdown, you will need to refer to the module without the `mdx` prefix (the second line of code above).

2.2 Command Line

```
$ echo 'The molecular composition of water is H~2~O.' > text.md
$ python -m markdown -o html -x 'mdx_subscript' -f text.html text.md
```

(continues on next page)

(continued from previous page)

```
$ cat text.html
<p>The molecular composition of water is H<sub>2</sub>O.</p>
```

CHAPTER 3

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs on Github issues.

If you are reporting a bug, please follow the guidelines shown to you while reporting the bug

3.1.2 Fix Bugs and Support Others

Look through the open GitHub issues. Anything tagged with "help wanted" is open to whoever wants to implement it. Anything tagged with "support" means someone needs help, and you may be the person to help them!

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

3.1.4 Write Documentation

Markdown Subscript could always use more documentation, whether as part of the official Markdown Subscript docs, in docstrings, or even on the web in blog posts, articles, and such.

3.1.5 Submit Feedback

The best way to send feedback is to file an issue on Github.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.2 Get Started!

Ready to contribute? Here's how to set up `markdown_subscript_extension` for local development.

1. Fork the `markdown_subscript_extension` repository on GitHub.
2. Create a directory for your project, and then create a virtual environment. If you new to virtual environments, RealPython's Virtualenv Primer is for you! The code below demonstrates how to create a virtual project with Virtualenvwrapper

```
$ mkproject markdown_subscript_extension
```

3. Clone your fork locally into your new project.

```
$ git clone git@github.com:YOUR_USERNAME_HERE/markdown_subscript_extension.git .
```

4. Install the necessary dependencies using pip. If you don't have pip installed, the Python installation guide can guide you through the process.

```
$ pip install -r requirements/dev_requirements.txt
$ pip install -r requirements/lint_requirements.txt
$ pip install -r requirements/test_requirements.txt
$ python setup.py develop
```

5. Create a branch for local development and begin to make changes!

```
$ git checkout -b name-of-your-bugfix-or-feature
```

6. Make small, targeted changes. Commit often, and write clear messages! Remember that this is a volunteer-drive project. The clearer your intent and your changes, the easier it will be for us to review your work.

```
$ git add .
$ git commit -m "Your detailed description of your changes."
```

5. The project is configured to use pre-commit. If you'd like to have each commit checked as you make it, install it as a hook for your repository, as demonstrated below.

```
$ pre-commit install
```

6. When you're done making changes, check that your changes pass the test suite. If you're not using pre-commit, you will also need to use the linters and auto-formatters.

```
$ make test  # or python setup.py test
$ # linters below
$ flake8 mdx_subscript.py tests
```

(continues on next page)

(continued from previous page)

```
$ isort --verbose --check-only --diff mdx_subscript.py
$ black mdx_subscript.py tests
```

7. If you have Python 2.7, 3.4, 3.5, 3.6, and 3.7 installed, as well as both PyPy and PyPy3, you may use `tox` to run all of the tests in all of the supported environments.

```
$ tox
```

8. Once you've made the changes you want, push your branch to GitHub.

```
$ git push -u origin name-of-your-bugfix-or-feature
```

9. Submit a pull request from your new branch to the original repository through the GitHub website.
10. Thank you for taking the time to read this and for your potential contributions!

3.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. If a bug-fix or new feature, the pull request should include tests.
2. If the pull request adds functionality, the docs should be updated.
3. The pull request should work for all supported Python versions. If you are unable to run `tox` locally, the CI will run the test suite for you (but please consider running the suite beforehand).

CHAPTER 4

Package Release Guidelines

This document serves as a guide for the release manage of the project. The package is to be released on both Github and PyPI.

The project follows semantic versioning and uses the 'Squash and Rebase' git branch model.

If the version of the package has not been properly increased, please use `bumpversion` to increase the major, minor, or patch number as needed. Add change notes to the history document.

Once the branch for bumping the version has been tested and squashed, the **package release** should occur on the **development branch**. Remember to pull/fast-forward the development branch from Github!

To release to PyPI, first run the Makefile `dist` target. If all goes well, upload the release using the `release` target.

```
$ make dist
$ make release
```

Tag the latest commit with the new version (the version should be prefixed with a `v` for consistency), and then push to Github.

```
$ git tag v2.0.1
$ git push origin --tags
```

That's all! You've released on Github and PyPI!

CHAPTER 5

Credits

5.1 Development Lead

- Andrew Pinkham <http://andrewsforge.com/>

5.2 Contributors

- Adrian <https://github.com/adi->

CHAPTER 6

History

6.1 2.1.1 (2018-10-09)

- (Re-)Enable short-name module reference (specific to Markdown v3; PR #53)
- Improved setup.py metadata

6.2 2.1.0 (2018-10-08)

- Add support for Python-Markdown 3.0
- Add support for Python 3.7 and PyPy3
- Switch tests to use PyTest
- Tox now checks package distribution
- Integrate PyUp
- Write documentation in Sphinx
- Integrate Read The Docs
- Rework existing restructured text documents
- Fix badges in Read Me

6.3 2.0.0 (2017-04-17)

- Add test/support for Python-Markdown 2.6
- **Drop Support For:**
  - Python 2.6 (EOL 2013)

  - Python 3.2 (EOL 2016)
  - Python-Markdown 2.4 (Superseded in 2014)
- Fix Issue #2 - Setup.py errors; Description.rst missing

6.4 1.0.1 (2014-10-17)

- Update for Python-Markdown 2.5

6.5 1.0.0 (2014-07-29)

- Initial Release of Markdown Subscript Extension
- **Compatible with:**
  - Python-Markdown 2.4
  - Python 2.6, 2.7
  - Python 3.2, 3.3, 3.4