Markdown 2.17.1: What's new, what's next?

Vít Novotný

Abstract

In this article, we introduce new features developed for the Markdown package and ideas for its future.

The article is divided into four sections. In the first three sections, we introduce the new features to three different audiences of the Markdown package:

1. the writers, who type content in Markdown;
2. the coders, who prepare templates and solutions;
3. the developers, who make the package better.

In Section 4, we outline the roadmap for the next major version of the Markdown package.

1 Writer's newsletter

In this section, we introduce four new Markdown tags, which you can use to format your manuscripts.

1.1 Superscripts and subscripts

Use superscripts and subscripts to write ordinal indicators, exponents, or atomic valencies. Since version 2.16.0, the Markdown package has supported the `superscripts` and `subscripts` options:[1]

```
\documentclass{article}
\usepackage[superscripts, subscripts]
           {markdown}
\begin{document}
\begin{markdown}
2^10^ is 1024.
H~2~O is a liquid.
\end{markdown}
\end{document}
```

Output:

2^{10} is 1024.
H_2O is a liquid.

1.2 Strike-throughs

Use strike-throughs to denote information that is no longer accurate. Since version 2.16.0, the Markdown package has supported the `strikeThrough` option:[2]

```
\documentclass{article}
\usepackage[strikeThrough]{markdown}
\begin{document}
\begin{markdown}
Under his pillow P'raps found
~~A cake that weighed a half a pound.~~
A plenty of space to roll around.
\end{markdown}
\end{document}
```

Output:

Under his pillow P'raps found
~~A cake that weighed a half a pound.~~
A plenty of space to roll around.

1.3 Fancy lists

In lists, it can be important to display item labels exactly as you wrote them. Since version 2.16.0, the Markdown has supported the `fancyLists` option:[3]

```
\documentclass{article}
\usepackage[fancyLists]{markdown}
\begin{document}
\begin{markdown}
You are:
a) awesome
b) brilliant
c) charming
\end{markdown}
\end{document}
```

Output:

You are:
a) awesome
b) brilliant
c) charming

2 Coder's newsletter

In this section, we introduce a new API for reacting to YAML metadata and user-defined syntax extensions.

2.1 Building better APIs with YAML

In our previous article, [3, Section 2.1] we showed how we can react to YAML metadata in Markdown documents. However, our approach used a low-level API that required use of the expl3 programming language. Since Markdown 2.16.0, the `\markdownSetup` LaTeX command has supported the `jekyllDataRenderers` key, which provides a high-level API for reacting to YAML metadata without the need to use expl3:[4]

```
\documentclass{article}
\usepackage[jekyllData]{markdown}
\newtoks\abstract \newtoks\authors
\markdownSetup {
    jekyllDataRenderers = {
        abstract = {\abstract={#1}},
        title = {\global\title{#1}},
        /authors/* = {%
            \authors=\expandafter{%
                \the\authors \and #1}%
        }, year = {%
            \global\date{%
                One year after
                \the\numexpr(#1-1)\relax}%
        },
    }, renderers = {
        jekyllDataEnd = {
            \global\author{\the\authors}%
            \maketitle \section*{Abstract}
                       \the\abstract
        },
    },
}
```

[1] See https://github.com/witiko/markdown/pull/162
[2] See https://github.com/witiko/markdown/pull/160
[3] See https://github.com/witiko/markdown/pull/168
[4] See https://github.com/witiko/markdown/pull/175

doi.org/10.47397/tb/43-3/tb135novotny-markdown

```
\begin{document}
\begin{markdown*}{expectJekyllData}
title: 'This is a title: with a colon'
authors: [Jane Doe, John Doe]
year: 2022
abstract: |
 This is the
 abstract

 It contains
 two paragraphs.
\end{markdown*}
\end{document}
```

Output:

This is a title: with a colon

Jane Doe John Doe

One year after 2021

Abstract

This is the abstract
It contains two paragraphs.

2.2 User-defined syntax extensions

Since version 2.17.0, the Markdown package has supported user-defined syntax extensions, which you can use to customize Markdown to your tastes:[5]

```
\documentclass{article}
\usepackage{soul}
\begin{filecontents}
   [nosearch, noheader, overwrite]
   {strike-through.lua}
local strike_through = {
   api_version = 2,
   grammar_version = 1,
   finalize_grammar = function(reader)
      local nonspace, doubleslash
      nonspace = lpeg.P(1) - lpeg.S("\t ")
      doubleslash = lpeg.P("//")

      local function between(p, sep)
         ender = lpeg.B(nonspace) * sep
         return (sep * #nonspace
                * lpeg.Ct(p * (p - sep)^0)
                * sep)
      end

      local read_strike_through = between(
         lpeg.V("Inline"), doubleslash
      ) / function(s)
         return {"\\st{", s, "}"}
      end

      reader.insert_pattern(
         "Inline after Emph",
         read_strike_through)
      reader.add_special_character("/")
   end
}
return strike_through
\end{filecontents}
\usepackage{markdown}
\begin{document}
\begin{markdown*}
      {extension = strike-through.lua}
Under his pillow P'raps found
//A cake that weighed a half a pound.//
A plenty of space to roll around.
\end{markdown*}
\end{document}
```

Output same as in Section 1.2

For more information about syntax extensions, see the technical documentation of the Markdown package [2, Section 2.1.2] and the article about parsing complex data formats in Lua by Henri Menke. [1]

3 Developer's newsletter

In this section, we introduce new reflection capabilities and discuss a recent code clean-up.

3.1 Reflection of options and renderers

In versions 2.15.0 and 2.15.3, the Markdown package has received reflection capabilities that allowed it to take a look in a mirror and inspect itself.[6]

Using reflection, we have automated parts of the code that were previously hand-written. These include parts responsible for type-checking options, passing options from plain TEX to Lua, and defining high-level interfaces for LATEX and ConTEXt.

3.2 Refactoring TEX and Lua code

In patch versions 2.15.1 through 2.15.4, we focused on cleaning up the code of the Markdown package. In the following, we discuss the major changes.

In version 2.15.3, we separated a part of the Markdown package into its own separate package

[5] See https://github.com/witiko/markdown/pull/182

[6] See https://github.com/witiko/markdown/pull/137

called lt3luabridge.[7] With lt3luabridge, you can execute Lua code in TEX engines other than LuaTEX.

Also in version 2.15.3, we separated built-in syntax extensions such as subscripts, superscripts, and strike-throughs from the base grammar of markdown.[8] This change cut the development time of new syntax extensions in half and paved the way for the introduction of user-defined syntax extensions in Markdown 2.17.0 (see Section 2.2).

In version 2.15.4, we replaced all calls to the xstring and keyval packages with built-in functions from the expl3 programming language.[9]

4 Roadmap for Markdown 3.0.0

The next major version of Markdown will be 3.0.0. Markdown 3.0.0 will remove features that have been deprecated in Markdown 2.X.Y, such as the on-disk caching of conversion outputs and the leftover interfaces for what is now the lt3luabridge package (see Section 3.2). Furthermore, Markdown 3.0.0 should also make the base grammar of markdown compliant with the CommonMark standard and freeze it, so that authors of user-defined syntax extensions (see Section 2.2) do not have to aim at a moving target.

Before Markdown 3.0.0, all syntax extensions that have been implemented to the upstream lunamark library should be ported to the Markdown package as well.[10] Furthermore, all improvements to the high-level interface for LATEX that we have discussed in our previous article [3, sections 3.3 and 3.4] should also be implemented.[11] Finally, the user manual of Markdown should be typeset using the Markdown package and TEX4ht rather than Pandoc, which will allow us to automatically generate parts of the user manual using reflection (see Section 3.1).[12]

[7] See https://ctan.org/pkg/lt3luabridge

[8] See https://github.com/witiko/markdown/pull/143

[9] See https://github.com/witiko/markdown/issues/96

[10] See https://github.com/witiko/markdown/issues/123, https://github.com/witiko/markdown/issues/126 and https://github.com/witiko/markdown/issues/173

[11] See https://github.com/witiko/markdown/issues/107 and https://github.com/witiko/markdown/issues/121

[12] See https://github.com/witiko/markdown/issues/135 and https://github.com/witiko/markdown/issues/184

References

[1] H. Menke. Parsing complex data formats in LuaTEX with LPEG. *TUGboat* 40(2):129–135, 2019. tug.org/TUGboat/tb40-2/tb125menke-lpeg.pdf

[2] V. Novotný. A Markdown interpreter for TEX. Version 2.17.1-35-g2848cb5 (2022-10-15). mirrors.ctan.org/macros/generic/markdown/markdown.pdf

[3] V. Novotný, D. Rehák, et al. Markdown 2.15.0: What's new? *TUGboat* 43(1):10–15, 2022. tug.org/TUGboat/tb43-1/tb133novotny-markdown.pdf

◇ Vít Novotný
Studená 453/15
Brno, 638 00
Czech Republic
witiko (at) mail dot muni dot cz
github.com/witiko