Markdown 2.15.0: What's new?

Vít Novotný, Dominik Rehák, Michal Hoftich, Tereza Vrabcová

Abstract

At TUG 2021, we celebrated the fifth birthday of markdown in TEX. In this article, we introduce new features developed in the five months since and ideas for the future development of the Markdown package.

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

1. the writers, who type content in markdown, and
2. the coders, who prepare templates and solutions.

In Section 3, we discuss ideas for the future to the third audience of Markdown: the developers, who alter and further improve the Markdown package.

1 Writer's newsletter

Michael Thompson from the pandoc-discuss mailing list characterized markdown as a perfectly minimalist markup language that only faces the writer with one question: what the next sentence should be. [4] However, in some types of documents, the few structural elements of markdown can be too few. The writers may enable the `hybrid` option and combine TEX and markdown markup, but this tends to reduce clarity, stability, and ease of reuse. To reduce the need for hybrid markup, we introduce new syntax extensions for Markdown in sections 1.1–1.4.

Since version 2.10.0 of the Markdown package, writers have been able to redesign their Markdown documents without programming using *LATEX themes*. [3] However, few LATEX themes have been publicly available until recently. In Section 1.5, we introduce LATEX themes, which self-publishers can use for typesetting books and publishing collaterals.

1.1 Task lists

To track progress on your goals, it can be useful to add checkboxes to list items. Since version 2.11.0, Markdown has supported the `taskLists` option:[1]

```
\documentclass{article}
\usepackage[taskLists]{markdown}
\begin{document}
\begin{markdown}
- Tasks:
    1. [x] Draft title.
    2. [.] Draft outline.
    3. [ ] Copy edit.
\end{markdown}
\end{document}
```

Output:

- Tasks:
  - ☒ Draft title.
  - ⊡ Draft outline.
  - ☐ Copy edit.

[1] See `http://github.com/witiko/markdown/issues/95`.

1.2 Emphatic line breaks

In poems and plays, line breaks carry a meaning and must be preserved. In markdown, you can write a line break by ending a line with two or more spaces:

```
Memory and desire, stirring␣␣
Dull roots with spring rain.
```

However, this can be tedious for longer texts. Furthermore, the Markdown package only supports line breaks in the `\markdownInput` command, because TEX strips trailing newlines from the input:[2]

```
\documentclass{article}
\usepackage{markdown}
\begin{document}
\begin{markdown}
Memory and desire, stirring␣␣
Dull roots with spring rain.
\end{markdown}
\end{document}
```

Output:

Memory and desire, stirring Dull roots with spring rain.

Since version 2.12.0, the Markdown package supports the `hardLineBreaks` option,[3] which makes every line break emphatic:

```
\documentclass{article}
\usepackage[hardLineBreaks]
           {markdown}
\begin{document}
\begin{markdown}
Memory and desire, stirring
Dull roots with spring rain.
\end{markdown}
\end{document}
```

Output:

Memory and desire, stirring
Dull roots with spring rain.

This makes it easier to typeset long poems and plays.

1.3 Cross-references

In technical and academic writing, cross-references between sections are common. Previously, writers would need to combine TEX and markdown markup:

```
\begin{markdown}
\documentclass{article}
\usepackage[hybrid]{markdown}
I conclude in Section \ref{sec:conclusion}.

Conclusion \label{sec:conclusion}
==========
In this paper, we have discovered that most
grandmas would rather eat dinner with their
grandchildren than get eaten. Begone, wolf!
\end{markdown}
\end{document}
```

[2] This limitation of TEX does not apply to ConTEXt MkIV, see also `http://github.com/Witiko/markdown/issues/101`.

[3] See `http://github.com/witiko/markdown/issues/98`.

xnotdoi.org/10.47397/tb/0-0/output

Since version 2.14.0, Markdown has supported attributes on section headings and the `relativeLinks` option,[4] which enables cross-references in markdown:

```
\begin{markdown}
\documentclass{article}
\usepackage[headerAttributes, relativeLinks]
           {markdown}
I conclude in Section <#sec:conclusion>.

Conclusion {#sec:conclusion}
==========
In this paper, we have discovered that most
grandmas would rather eat dinner with their
grandchildren than get eaten. Begone, wolf!
\end{markdown}
\end{document}
```

1.4 Document metadata

Even though writers can prepare their documents in markdown, they previously needed to specify metadata for their documents (such as the title or the author's name) in TEX:

```
\begin{markdown}
\documentclass{article}
\usepackage{markdown}
\title{On \emph{Wolves} \& \emph{Grandmas}}
\author{Little Red Cap}
\begin{document}
\maketitle
\begin{markdown}
When Little Red Cap entered the woods
a wolf came up to her.
\end{markdown}
\end{document}
```

Since version 2.11.0, the Markdown package has supported the `jekyllData` option,[5] which allows us to write metadata in markdown:

```
\begin{markdown}
\documentclass{article}
\usepackage[jekyllData]{markdown}
\begin{document}
\begin{markdown}
---
title: Of *Wolves* & _Grandmas_
author: Little Red Cap
---
When Little Red Cap entered the woods
a wolf came up to her.
\end{markdown}
\end{document}
```

1.5 LaTeX themes for self-publishers

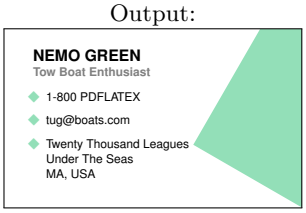
Writers who are unaccustomed to TEX may find their precious time slipping away, spent scouring online forums looking for a fix for that *one thing* that is messing up the whole layout. In cooperation with the Writersglen publishing house, we have created a set of LaTeX themes for typesetting books and publishing collaterals in markdown.[6]

Let's prove the ease of use of these templates with an example. Using the business card template, we might end up with a content file looking like this:

```
# Nemo Green
## Tow Boat Enthusiast

- 1-800 PDFLATEX
- tug@boats.com
- Twenty Thousand
  Leagues␣␣
  Under The Seas␣␣
  MA, USA
```

Output:



As you can see, easy as pie! So why not give it a try?

2 Coder's newsletter

In *Digital Typography*, [2] Knuth stresses the importance of stability in TEX and METAFONT, which ensures identical output across time and across different computer systems. Over the last forty years, this stability has allowed an ecosystem of third-party software to grow around TEX. To make it easier to develop complex software solutions, we show how coders can integrate Markdown with third-party software in sections 2.1–2.3.

In Section 1, we showed new syntax extensions for Markdown. However, syntax extensions are costly to implement, undermine the minimalism of markdown, and can never account for all components and concepts a writer may need. Therefore in Section 2.4, we present the concepts of *HTML attributes* and *attribute contexts*, which can be used to define domain-specific dialects of markdown in TEX without the need for new syntax extensions.

2.1 Building better APIs with YAML

In Section 1.4, we showed how authors can include metadata in their markdown documents using the YAML language. To react to the metadata, we can use a high-level key–value interface in the expl3 programming language:[7]

[4] See `http://github.com/witiko/markdown/issues/91`.
[5] See `http://github.com/witiko/markdown/issues/22`.
[6] See `http://github.com/xvrabcov/md-templates`.
[7] See `http://github.com/witiko/markdown/issues/22`.

```
\documentclass{article}
\usepackage[jekyllData]{markdown}
\ExplSyntaxOn
\tl_new:N  \g_abstract_tl
\seq_new:N \g_authors_seq
\keys_define:nn { markdown/jekyllData } {
   abstract .tl_gset:N = \g_abstract_tl,
   /authors/* .code:n = {
      \seq_put_right:Nn
         \g_authors_seq
         { #1 }
   }, title .code:n = {
      \global \title { #1 }
   }, year .code:n = {
      \global \date {
         One~year~after~
         \int_eval:n { #1 - 1 }
      }
   },
}
\markdownSetup {
   renderers = {
      jekyllDataEnd = {
         \exp_args:NNx
            \global \author {
               \seq_use:Nn
                  \g_authors_seq
                  { \and }
            }
         \maketitle
         \section*{Abstract}
         \g_abstract_tl
      },
   },
}
\ExplSyntaxOff
\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 Passing HTML through to TEX4ht

Using the TEX4ht system, we can convert TEX documents to HTML for publishing on the web. Since TEX4ht uses LATEX for the conversion, it supports the Markdown package out-of-box. However, it is still necessary to use correct command-line options depending on which TEX engine we use. To use LuaTEX, we can use the `--lua` option:

```
$ make4ht --lua document.tex
```

With other TEX engines, we must use the `--shell-escape` option, which enables shell access:

```
$ make4ht --shell-escape document.tex
```

Since version 2.3.0, Markdown has supported the `html` option, which allows us to use HTML tags in markdown documents. Since version 2.14.0, Markdown has also supported renderers for HTML tags.[8] Unless redefined by the user, these renderers will pass any HTML elements through to the output of TEX4ht, whereas they will be ignored in PDF output:

```
\usepackage[html]{markdown}
\begin{document}
\begin{markdown}
Hello <b>world</b>!
\end{markdown}
\end{document}
```

PDF output:

Hello world!

TEX4ht output:

Hello **world**!

2.3 Integration with Pandoc

Pandoc is a tool for converting between dozens of document formats. In our proof-of-concept,[9] we integrate Pandoc with the Markdown package, so that we can typeset and style any document format understood by Pandoc directly from TEX.

To give an example, we have prepared a manual page `wolf.1` in the roff language:

```
.SH NAME
wolf \- tool for befriending grandmas
.SH SYNOPSIS
.B wolf
[\fB-b\fR|\fB--befriend\fR]
[\fB-s\fR|\fB--scare\fR]
<\fIgrandma\fR>
```

Here is how we would typeset our manual page:

```
\documentclass{article}
\usepackage{pandoc-to-markdown, emoji}
\markdownSetup{renderers = {
 headingOne = {\section*{\emoji{wolf}#1}}}}
\begin{document}
\pandocInput[format=man]{wolf.1}
\end{document}
```

[8] See `http://github.com/witiko/markdown/issues/90`.
[9] See `http://github.com/drehak/pandoc-to-markdown`.

Output:

NAME

wolf - tool for befriending grandmas

SYNOPSIS

wolf [**-b**|**--befriend**] [**-s**|**--scare**] <*grandma*>

Our proof of concept consists of a Lua writer that produces TEX commands corresponding to the abstract syntax tree of Pandoc and a TEX package that maps these commands to the renderers of the Markdown package. A rewrite of our Lua writer in Haskell will be offered as a basis of the upcoming plain TEX writer for Pandoc.[10]

2.4 Actionable attributes and contexts

In Section 1.3, we showed how authors can add HTML attributes to section headings. We can react to the attibutes by redefining attribute renderers. Furthermore, the HTML attributes of a markdown element are surrounded by attribute contexts, which we can use to limit the effects of an attribute:[11]

```
\documentclass{article}
\usepackage[headerAttributes]{markdown}
\markdownSetup{
    renderers = {
        headerAttributeContextBegin =
            \begingroup,
        headerAttributeContextEnd =
            \endgroup,
        attributeClassName = {%
            \markdownIfSnippetExists{#1}{%
                \markdownSetup{snippet=#1}%
            }{}%
        },
    }
}
\markdownSetupSnippet{sans-serif}{
    code = {%
        \def\familydefault{\sfdefault}%
        \fontfamily{\familydefault}%
        \selectfont
    },
}
\begin{document}
\begin{markdown}
# A section
This section is typeset in a serif typeface.

# Another section {.sans-serif}
This section is typeset in sans-serif ...

## A subsection
... and so is
this subsection.

# Another section
This section is,
again, typeset
in serif.
\end{markdown}
\end{document}
```

Output:

1 A section

This section is typeset in a serif typeface.

2 Another section

This section is typeset in sans-serif ...

2.1 A subsection

... and so is this subsection.

3 Another section

This section is, again, typeset in serif.

In Section 3.2, we discuss our plans for other elements of markdown that may be able to receive HTML attributes in the future.

3 Developer's newsletter

In the following sections, we describe ideas for improving the Lua parser (3.1 and 3.2), LATEX interface (3.4 and 3.3), ConTEXt interface (3.5), and Docker images (3.6) of Markdown.

3.1 Smart backslashes and math support

Since Markdown does not detect math at parse time, it can be difficult to write math:

```
\documentclass{article}
\usepackage{mathtools}
\usepackage[hybrid]{markdown}
\begin{document}
\begin{markdown}
$$  x_i + y_j =
    \begin{dcases}
        a & b \\
        c & d
    \end{dcases} $$
\end{markdown}
\end{document}
```

Output:

$$xi + yj =$$
\begin{dcases}
a & b \\
c & d \\
\end{dcases}

Specifically, it is necessary to escape underscores and backslashes, and to be careful with indent:

```
\documentclass{article}
\usepackage{mathtools}
\usepackage[hybrid]{markdown}
\begin{document}
\begin{markdown}
$$ x\_i + y\_j =
\begin{dcases}
  a & b \\\\
  c & d
\end{dcases} $$
\end{markdown}
\end{document}
```

Output:

$$x_i + y_j = \begin{cases} a & b \\ c & d \end{cases}$$

In our previous article, [3, Figure 4] we have shown how we can construct a smart lexical preprocessor that only requires the escaping of backslashes

[10] See http://github.com/jgm/pandoc/issues/1541.
[11] See http://github.com/witiko/markdown/issues/91.

when they precede another escapable character. Furthermore, we can use well-defined heuristics such as dollar signs to detect math at parse time and disable underscores, code listings, and other elements in it:[12]

```
\documentclass{article}
\usepackage{mathtools}
\usepackage[smartBackslashes, mathDollars]
           {markdown}
\begin{document}
\begin{markdown}
$$  x_i + y_j =
    \begin{dcases}
        a & b \\
        c & d
    \end{dcases} $$
\end{markdown}
\end{document}
```

Desired output:

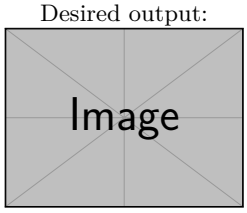
$$x_i + y_j = \begin{cases} a & b \\ c & d \end{cases}$$

3.2 Attributes on links and images

In Section 1.3, we showed how authors can add HTML attributes to headings in markdown. In order to define domain-specific dialects of markdown in TEX, it may be useful to suppport HTML attributes on various other elements of markdown, perhaps most importantly on links and images:[13]

```
\documentclass{article}
\usepackage[linkAttributes]{markdown}
\usepackage{graphicx}
\markdownSetup{
    renderers = {
        linkAttributeContextEnd =
            \endgroup,
        linkAttributeContextBegin = {%
            \begingroup
            \markdownSetup{
                renderers = {
                    attributeKeyValue = {%
                        \setkeys{Gin}{
                            {#1} = {#2},
                        }%
                    },
                },
            }%
        },
    },
}
\begin{document}
\begin{markdown}
  ![image](example-image){width=5cm}
\end{markdown}
\end{document}
```

Desired output:



[12] See http://github.com/witiko/markdown/issues/61.
[13] See http://github.com/witiko/markdown/issues/123.

3.3 Importing LaTeX setup snippets

In our previous article, [3, Section 1] we have introduced LaTeX themes and snippets, which can be used to build powerful abstractions in Markdown. Suppose the `jdoe/longpackagename/lists` LaTeX theme defines the `arabic`, `roman`, and `alpha` setup snippets. If we want to access these snippets by their short names, we must first load the theme and then assign names to the snippets:

```
\markdownSetup{
 theme=jdoe/longpackagename/lists}
\markdownSetupSnippet{arabic}{
 snippet=jdoe/longpackagename/lists/arabic}
\markdownSetupSnippet{roman}{
 snippet=jdoe/longpackagename/lists/roman}
\markdownSetupSnippet{alphabetic}{
 snippet=jdoe/longpackagename/lists/alpha}
```

In order to make the code easier to read and the intent clearer, it may be useful to have a dedicated syntax for importing setup snippets:[14]

```
\markdownSetup{
    importSnippets = {
        jdoe/longpackagename/lists = {
            arabic,
            roman,
            alpha as alphabetic,
        },
    },
}
```

3.4 Advanced renderer definitions in LaTeX

At the moment, the `\markdownSetup` LaTeX command only allows the redefinition of one renderer or renderer prototype at a time, which makes it difficult to redefine several renderers or renderer prototypes at once:

```
\markdownSetup{
  rendererPrototypes = {
    headingOne = {\chapter{🦊 #1}},
    headingTwo = {\section{🦊 #1}},
    headingThree = {\subsection{🦊 #1}},
    headingFour = {\subsubsection{🦊 #1}},
    headingFive = {\paragraph{🦊 #1}},
    headingSix = {\subparagraph{🦊 #1}},
  },
}
```

Furthermore, it is difficult to keep some parts of previous definitions without using low-level code:

```
\usepackage{etoolbox}
\xpatchcmd
    \markdownRendererHeadingOnePrototype
    {#1}{🦊 #1}{}{}
```

[14] See http://github.com/witiko/markdown/issues/107.

In order to make it easier to redefine renderers and renderer prototypes partially and in bulk, it may be useful to extend the syntax of `\markdownSetup`:[15]

```
\markdownSetup{
    rendererPrototypes = {
        heading* {#1} = {🐺 #1},
    },
}
```

3.5 Idiomatic ConTEXt setup

Unlike LaTeX, which has high-level syntax for setting up Markdown, ConTEXt has only few additions over the plain TEX interface for Markdown. Since version 2.15.0, there has been a concerted effort to extend Markdown, so that it can enumerate and examine its own options, renderers, and renderer prototypes.[16] This will make it easier to create and maintain new high-level interfaces for formats other than LaTeX, such as ConTEXt.[17]

3.6 Additional binary platforms in Docker

Since version 2.10.0, Markdown has been available as the `witiko/markdown` Docker image.[18] In version 2.15.0, images for TEX Live 2019–2021 are available, which makes it easy to use Markdown for continuous integration with services such as GitHub Actions:

```
name: Typeset a document
on: {push: ~}
jobs:
  typeset:
    runs-on: ubuntu-latest
    container:
     image: witiko/markdown:TL2019-historic
    steps:
      - uses: actions/checkout@v2
      - run: latexmk -lualatex document.tex
```

The `witiko/markdown` Docker image is based on the `texlive/texlive` Docker image from the Island of TEX, [1] which is only available for the linux/amd64 platform. This is sufficient for most continuous integration services. However, to allow interactive use of `witiko/markdown`, it may be useful to add support for multi-arch builds to the `texlive/texlive`.[19]

References

[1] Island of TEX. The island of TEX: Developing abroad, your next destination. *TUGboat* 41(2):182–184, 2020.

[2] D. Knuth. *Digital Typography*. No. 78 in CSLI Lecture Notes. Center for the Study of Language and Information (CSLI), 1999. The second printing (2012) contains numerous corrections.

[3] V. Novotný. Markdown 2.10.0: LaTeX themes & snippets, two flavors of comments, and luametaTEX. *TUGboat* 42(2):186–193, 2021.

[4] M. Thompson. Re: Error in "cabal install pandoc", 2010. `https://groups.google.com/g/pandoc-discuss/c/tKB4E7y6H2E/m/OiieKAuWsl4J`

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

◇ Dominik Rehák
Legionárska 71
Trenčín, 911 04
Slovak Republic
`drehak (at) firemail dot cc`
`github.com/drehak`

◇ Michal Hoftich
Magdalény Rettigové 4
Praha, 116 39
Czech Republic
`michal dot h21 (at) gmail dot com`
`github.com/michal-h21`

◇ Tereza Vrabcová
V Aleji 130/30
Děčín, 405 02
Czech Republic
`vrabcova dot tereza (at) email dot cz`
`github.com/xvrabcov`

[15] See `http://github.com/witiko/markdown/issues/121`.
[16] See `http://github.com/witiko/markdown/issues/119`.
[17] See `http://github.com/witiko/markdown/issues/17`.
[18] See `http://hub.docker.com/r/witiko/markdown`.
[19] See `http://gitlab.com/islandoftex/images/texlive/-/issues/15`.