Markdown 2.10.0

\LaTeX Themes & Snippets,
Two Flavors of Comments,
and LuaMeta\LaTeX

Vít Novotný, Grandma Jane, and the Wolf

Faculty of Informatics, Masaryk University
Brno, Czech Republic

August 7, 2021
Chapter 1

Introduction

1.1 Section

1.1.1 Subsection

Hello *Markdown*!

<table>
<thead>
<tr>
<th>Right</th>
<th>Left</th>
<th>Default</th>
<th>Center</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

: Table

<table>
<thead>
<tr>
<th>Right</th>
<th>Left</th>
<th>Default</th>
<th>Center</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1.1: Table
Contents

\LaTeX{} themes & setup snippets
  Built-in themes
  Creating your own theme
  Setup snippets
Two flavors of comments
  Semantic HTML comments
  Lexical \TeX{} comments
Support for LuaMeta\TeX{}
What’s next and how do I contribute?
  Actionable HTML attributes
  Jekyll front matter
  Online images using Luasocket
  Integration with Pandoc
  Direct mapping of elements
**LaTeX themes & setup snippets**

The goal of the Markdown package is to bring fire 🔥 to the users of \TeX, so that they can playfully incinerate each and every element of their markdown documents. 😊

Unlike in the cathedral 🏛️ of Con\TeX t, where packages are few, an extraordinary bazaar 🎉 of action, ferment, and innovation has sprung up in the wake of the \LaTeX\,2ε kernel.

- Markdown’s *themes* are \LaTeX packages that specify what markdown elements *do*.
- Themes apply *globally*, but may define *setup snippets* that apply *locally*.
- We will discuss *built-in themes* and creating your own themes and setup snippets.


Built-in themes I

The witiko/tilde theme

```latex
\documentclass{article}
\usepackage[theme=witiko/tilde]{markdown}
\begin{document}
\begin{markdown}
Bartel~Leendert van~der~Waerden
\end{markdown}
\end{document}
```

Bartel·Leendert van·der·Waerden, where a middot (·) represents a non-breaking space.
\documentclass{article}
\usepackage[theme=witiko/dot]{markdown}
\begin{document}
\begin{markdown}
```dot
A parse tree of “Let's eat grandma!”

digraph tree {
  graph [margin = 0]; node [shape = none]
  edge [arrowhead = none]
  {rank=same; VP1[label = VP]}
  {rank=same; Let; NP1[label = NP];
    VP2[label = VP]}
  {rank=same; us; eat; NP2[label = NP]}
  {rank=same; S};
S -> VP1; VP1 -> Let; VP1 -> NP1
  VP1 -> VP2; NP1 -> us; VP2 -> eat
  VP2 -> NP2; NP2 -> grandma }
\end{markdown}
\end{document}

Figure: A parse tree of “Let’s eat grandma!”
Built-in themes III

The witiko/graphicx/http theme

```latex
\documentclass{article}
\usepackage[texComments, contentBlocks, theme=witiko/graphicx/http]{markdown}
\begin{document}
\begin{markdown}
https://github.com/witiko\%/markdown/raw/master\%/banner.png
(Banner of the Markdown package)
\end{markdown}
\end{document}
```

Figure: Banner of the Markdown package
Creating your own theme

1. Decide on a name in the following form:

\langle theme author\rangle / \langle target package\rangle / \langle private naming\rangle

For example: jdoe/beamer/headings

2. Munge the theme name by substituting slashes (\/) with underscores (_).

For example: jdoe_beamer_headings

3. Create a text file named markdowntheme\langle munged theme name\rangle.sty.

For example markdownthemejdoe_beamer_headings.sty with the following:

\ProvidesPackage{markdownthemejdoe_beamer_headings}[2021/06/04]
\markdownSetup{
    rendererPrototypes = {
        headingOne = {\frametitle\{#1\}},
        headingTwo = {\framesubtitle\{#1\}}
    }
}
Creating your own theme II

\documentclass{beamer}
\usepackage[theme = witiko/dot, theme = jdoe/beamer/headings]{markdown}
\title{Dietary Assessment of Big Bad Wolf}
\author{Jane Doe}
\begin{document}
\maketitle
\begin{frame}[fragile]
\begin{markdown}
# What's on the Menu?
## Dietary Assessment
```
\begin{dot}
 digraph tree {
    Wolf -> Grandma; Wolf -> Hood
    Wolf [label = "Big Bad Wolf"]
    Hood [label = "Little Red Riding Hood"]
}\end{dot}
```
\end{markdown}
\end{frame}
\end{document}
Setup snippets I

\ProvidesPackage{markdownthemejdoe_lists_roman}
\[2021/06/04\]
\markdownSetup{
rendererPrototypes = {
olItemWithNumber = {
\item[\romannumeral#1\relax.]
}}}

Can be used only in preamble.

\documentclass{article}
\usepackage{markdown}
\begin{document}
\begin{markdown}
1. wahid
2. aithnayn
\end{markdown}
\begin{markdown*}% This won't work!
\{theme=jdoe/lists/roman\}
3. tres
4. quattuor
\end{markdown*}
\end{document}
\ProvidesPackage{markdownthemejdoe_lists}
[2021/06/04]
\markdownSetupSnippet{roman}{
    rendererPrototypes = {
        olItemWithNumber = {
            \item[\romannumeral#1\relax.]
        }
    }
}

1. wahid
2. aithnayn
3. tres
4. quattuor

\documentclass{article}
\usepackage[theme=jdoe/lists]{markdown}
\begin{document}
\begin{markdown}
1. wahid
2. aithnayn
\end{markdown}
\begin{markdown*}
\begin{snippet=jdoe/lists/roman}
3. tres
4. quattuor
\end{snippet=jdoe/lists/roman}
\end{markdown*}
\end{document}
Two flavors of comments

In \TeX{}, comments fulfil several \textit{distinct roles}:

1. Prevent the processing of code:
   \begin{verbatim}
   \author{Authors anonymized}
   \author{John Doe \and Jane Roe}
   \end{verbatim}

2. Mark up parallel documents:
   \begin{verbatim}
   \% \cs{foo} prints ``bar'':
   \% \begin{macrocode}
   \def\foo{bar}
   \% \end{macrocode}
   \end{verbatim}

3. Insert little side notes:
   \begin{verbatim}
   \% Aren't we missing a comma here? Let's eat grandma!
   \end{verbatim}

4. Consume newlines and leading spaces:

   My parents have first met in Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch.

The Markdown package supports two flavors of comments:

- \textbf{Semantic} May only appear in text. Markup is recognized. Enables roles 1, 2, and 3.
- \textbf{Lexical} May appear anywhere. Markup is ignored. Enables roles 1, 2, and 4.
Semantic HTML comments

\documentclass{article}
\usepackage{marginnote}
\usepackage[html]{markdown}
\markdownSetup{
  renderers = {
    inlineHtmlComment = {
      \marginnote{#1}
    }
  }
}
\begin{document}
\begin{markdown}
<!-- Aren't we missing
  *a comma* here? -->
Let's eat grandma!
\end{markdown}
\end{document}
**Lexical \TeX comments**

\documentclass{article}
\usepackage[texComments]{markdown}
\begin{document}
\begin{markdown}
[Here][1] is what the wolf has to say on the matter.

[1]: http://a.very.long.url/that/%should/enjoy\%20some#word-wrap
\end{markdown}
\end{document}

Here\(^1\) is what the wolf has to say on the matter.

\(^1\)http://a.very.long.url/that/should/enjoy\%20some#word-wrap
Support for LuaMetaTEX

The LuaMetaTEX engine is a fast minimalist development 🏷️ version of LuaTEX, which is used in the ConTEXt LMTX format. Since the Markdown package supports ConTEXt, the time is ripe 🍇 to make Markdown play nice with LuaMetaTEX as well.

- LuaMetaTEX lacks Selene Unicode, but uses Lua 5.4 with its built-in utf8 library.
- LuaMetaTEX lacks KPathSea, but provides an optional library interface.
- LuaMetaTEX provides Luasocket, but lacks high-level libraries such as socket.http.
What's next and how do I contribute?

The Markdown package would be well-served by encouraging users to lecture it 🎓 on ever-new things and release their lecture notes 📚 to the whole wide world. 🌍

There are many intriguing ideas for the future of Markdown:

- Some ideas are already *under development* by contributors.
  
  https://github.com/witiko/markdown/pulls

- Some ideas are only now beginning to be *discussed*:
  
  https://github.com/witiko/markdown/discussions & /issues

- Other ideas are yet to be discovered by you.
**What's next and how do I contribute?**

**Actionable HTML attributes**

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!

I conclude in Section \#sec:conclusion.

Conclusion \#sec:conclusion \some-snippet

\#sec:conclusion

In this paper, we have discovered that most grandmas would rather eat dinner with their grandchildren than get eaten. Begone, wolf!

- **Recommended reading:** Parsing complex data formats in \LaTeX{} with LPEG (Menke, 2019)
What's next and how do I contribute?  

Jekyll front matter

---

title: Of *Wolves* and _Grandmas_

author:  
- name: Little Red Riding Hood
- name: Big Bad Wolf

---

- Recommended reading: Parsing complex data formats in LuaTEX with LPEG (Menke, 2019)
- For more information, see /issues/22 and /pull/77.
Online images using Luasocket

Just use sockets. 📶
—Hans Hagen during a Q&A session at BachoTeX 2017

- The witiko/graphicx/http theme uses either GNU Wget or cURL.
- We could remove both prerequisites by using the Luasocket library.
- In /issues/82, I drafted an implementation:

```lua
local http = require("socket.http")
local result, status, headers = http.request(url)
```

The draft has several issues:

2. LuaMetaTeX lacks `socket.http`.
3. The \directlua command needs to be replaced with a shell escape for non-Lua TeX engines.

- Lua programmers familiar with Luasocket are encouraged to help tackle points 1 & 2.
Integration with Pandoc

- Pandoc is a Haskell library for converting between dozens of document formats.
- Pandoc uses an intermediate AST format, so that every document format only needs a conversion function from the format to the AST and back. \( N > 2 \implies 2N < N^2 \).
- If the Markdown package understood the AST, we could typeset any of the document formats of Pandoc while maintaining full control over the formatting:

  \[
  \textbf{\texttt{\usepackage[theme=jdoe/lists]{markdown}\begin{document}\pandocInput[snippet=jdoe/lists/roman]{of-wolves-and-grandmas.docx}\end{document}}}
  \]

- Recommended reading: Parsing complex data formats in \LaTeX{} with LPEG (Menke, 2019)
- For more information, see /issues/25, /issues/62, and drehak/lunamark.
What's next and how do I contribute?

Direct mapping of elements

\markdownSetup

\{ renderers = { emphasis = {\textbf{#1}}} }\}

Hello \markdownRendererEmphasis{Markdown}!

↓

Hello \markdownRendererEmphasis{Markdown}!

↓

Hello \textbf{Markdown}!

↓

Hello Markdown!
Direct mapping of elements II

\markdownSetup { 
    directMapping = { 
        headingOne = { 
            \begin{frame} 
            \frametitle{#1} 
        }, 
        horizontalRule = { 
            \end{frame} 
        }, 
    } 
} 

# What's on the Menu? 
Grandma 
*** 
↓
\begin{frame} 
\frametitle{What's on the Menu?} 
Grandma 
\end{frame}