A Markdown Interpreter for TEX

Vít Novotný
witiko@mail.muni.cz

Version 2.15.0-0-g9296cf1
2022/03/31

Contents

List of Figures

1 Introduction

The Markdown package[1] converts markdown[2] markup to TEX commands. The functionality is provided both as a Lua module and as plain TEX, LATEX, and ConTEXt macro packages that can be used to directly typeset TEX documents containing markdown markup. Unlike other convertors, the Markdown package does not require any external programs, and makes it easy to redefine how each and every markdown element is rendered. Creative abuse of the markdown syntax is encouraged. 😉

This document is a technical documentation for the Markdown package. It consists of three sections. This section introduces the package and outlines its prerequisites. Section 2 describes the interfaces exposed by the package. Section 3 describes the

[1] See https://ctan.org/pkg/markdown.
[2] See https://daringfireball.net/projects/markdown/basics.

implementation of the package. The technical documentation contains only a limited number of tutorials and code examples. You can find more of these in the user manual.[3]

```
local metadata = {
    version   = "(((VERSION)))",
    comment   = "A module for the conversion from markdown to plain TeX",
    author    = "John MacFarlane, Hans Hagen, Vít Novotný",
    copyright = {"2009-2016 John MacFarlane, Hans Hagen",
                 "2016-2022 Vít Novotný"},
    license   = "LPPL 1.3c"
}

if not modules then modules = { } end
modules['markdown'] = metadata
```

1.1 Requirements

This section gives an overview of all resources required by the package.

1.1.1 Lua Requirements

The Lua part of the package requires that the following Lua modules are available from within the LuaTEX engine:

LPeg $\geqslant$ 0.10 A pattern-matching library for the writing of recursive descent parsers via the Parsing Expression Grammars (PEGs). It is used by the Lunamark library to parse the markdown input. LPeg $\geqslant$ 0.10 is included in LuaTEX $\geqslant$ 0.72.0 (TEXLive $\geqslant$ 2013).

```
local lpeg = require("lpeg")
```

Selene Unicode A library that provides support for the processing of wide strings. It is used by the Lunamark library to cast image, link, and footnote tags to the lower case. Selene Unicode is included in all releases of LuaTEX (TEXLive $\geqslant$ 2008).

```
local ran_ok, unicode = pcall(require, "unicode")
```

If the Selene Unicode library is unavailable and we are using Lua $\geqslant$ 5.3, we will use the built-in support for Unicode.

```
if not ran_ok then
  unicode = {["utf8"]={char=utf8.char}}
end
```

[3] See http://mirrors.ctan.org/macros/generic/markdown/markdown.html.

MD5 A library that provides MD5 crypto functions. It is used by the Lunamark library to compute the digest of the input for caching purposes. MD5 is included in all releases of LuaTeX (TeXLive ⩾ 2008).

```
local md5 = require("md5")
```

All the abovelisted modules are statically linked into the current version of the LuaTeX engine [1, Section 3.3]. Beside these, we also carry the following third-party Lua libraries:

api7/lua-tinyyaml A library that provides a regex-based recursive descent YAML (subset) parser that is used to read YAML metadata when the `jekyllData` option is enabled.

1.1.2 Plain TeX Requirements

The plain TeX part of the package requires that the plain TeX format (or its superset) is loaded, all the Lua prerequisites (see Section 1.1.1), and the following package:

expl3 A package that enables the expl3 language from the LaTeX3 kernel in TeX Live ⩽ 2019. It is not used for anything... yet.

```
\ifx\ExplSyntaxOn\undefined
  \input expl3-generic\relax
\fi
```

The plain TeX part of the package also requires the following Lua module:

Lua File System A library that provides access to the filesystem via OS-specific syscalls. It is used by the plain TeX code to create the cache directory specified by the `\markdownOptionCacheDir` macro before interfacing with the Lunamark library. Lua File System is included in all releases of LuaTeX (TeXLive ⩾ 2008).

The plain TeX code makes use of the `isdir` method that was added to the Lua File System library by the LuaTeX engine developers [1, Section 3.2].

The Lua File System module is statically linked into the LuaTeX engine [1, Section 3.3].

Unless you convert markdown documents to TeX manually using the Lua command-line interface (see Section 2.1.5), the plain TeX part of the package will require that either the LuaTeX `\directlua` primitive or the shell access file stream 18 is available in your TeX engine. If only the shell access file stream is available in your TeX engine (as is the case with pdfTeX and XeTeX) or if you enforce the use of shell using the `\markdownMode` macro, then unless your TeX engine is globally configured to enable shell access, you will need to provide the `-shell-escape` parameter to your engine when typesetting a document.

1.1.3 LaTeX Requirements

The LaTeX part of the package requires that the LaTeX 2_ε format is loaded,

```
\NeedsTeXFormat{LaTeX2e}%
```

a TeX engine that extends ε-TeX, all the plain TeX prerequisites (see Section 1.1.2), and the following LaTeX 2_ε packages:

keyval A package that enables the creation of parameter sets. This package is used to provide the `\markdownSetup` macro, the package options processing, as well as the parameters of the `markdown*` LaTeX environment.

```
\RequirePackage{keyval}
```

xstring A package that provides useful macros for manipulating strings of tokens.

```
\RequirePackage{xstring}
```

The following packages are soft prerequisites. They are only used to provide default token renderer prototypes (see sections 2.2.4 and 3.3.4) or LaTeX themes (see Section 2.3.2.2) and will not be loaded if the `plain` package option has been enabled (see Section 2.3.2.1):

url A package that provides the `\url` macro for the typesetting of links.

graphicx A package that provides the `\includegraphics` macro for the typesetting of images.

paralist A package that provides the `compactitem`, `compactenum`, and `compactdesc` macros for the typesetting of tight bulleted lists, ordered lists, and definition lists.

ifthen A package that provides a concise syntax for the inspection of macro values. It is used in the `witiko/dot` LaTeX theme (see Section 2.3.2.2), and to provide default token renderer prototypes.

fancyvrb A package that provides the `\VerbatimInput` macros for the verbatim inclusion of files containing code.

csvsimple A package that provides the `\csvautotabular` macro for typesetting CSV files in the default renderer prototypes for iA Writer content blocks.

gobble A package that provides the `\@gobblethree` TeX command that is used in the default renderer prototype for citations. The package is included in TeXLive $\geqslant$ 2016.

amsmath and amssymb Packages that provide symbols used for drawing ticked and unticked boxes.

catchfile A package that catches the contents of a file and puts it in a macro. It is used in the `witiko/graphicx/http` LaTeX theme, see Section 2.3.2.2.

grffile A package that extends the name processing of package graphics to support a larger range of file names in 2006 ⩽ TeX Live ⩽ 2019. Since TeX Live ⩾ 2020, the functionality of the package has been integrated in the LaTeX 2_ε kernel. It is used in the `witiko/dot` and `witiko/graphicx/http` LaTeX themes, see Section 2.3.2.2.

etoolbox A package that is used to polyfill the general hook management system in the default renderer prototypes for YAML metadata, see Section 3.3.4.6, and also in the default renderer prototype for attribute identifiers.

expl3 A package that enables the expl3 language from the LaTeX3 kernel in TeX Live ⩽ 2019. It is used in the default renderer prototypes for links (see Section **??**), YAML metadata (see Section 3.3.4.6), and in the implementation of LaTeX themes (see Section 3.3.2.1).

```
\RequirePackage{expl3}
```

1.1.4 ConTeXt Prerequisites

The ConTeXt part of the package requires that either the Mark II or the Mark IV format is loaded, all the plain TeX prerequisites (see Section 1.1.2), and the following ConTeXt modules:

m-database A module that provides the default token renderer prototype for iA Writer content blocks with the CSV filename extension (see Section 2.2.4).

1.2 Feedback

Please use the Markdown project page on GitHub[4] to report bugs and submit feature requests. If you do not want to report a bug or request a feature but are simply in need of assistance, you might want to consider posting your question to the TeX-LaTeX Stack Exchange.[5] community question answering web site under the `markdown` tag.

1.3 Acknowledgements

The Lunamark Lua module provides speedy markdown parsing for the package. I would like to thank John Macfarlane, the creator of Lunamark, for releasing Lunamark under a permissive license, which enabled its use in the Markdown package.

[4] See `https://github.com/witiko/markdown/issues`.
[5] See `https://tex.stackexchange.com`.

Extensive user documentation for the Markdown package was kindly written by Lian Tze Lim and published by Overleaf.

Funding by the the Faculty of Informatics at the Masaryk University in Brno [2] is gratefully acknowledged.

Support for content slicing (Lua options `shiftHeadings` and `slice`) and pipe tables (Lua options `pipeTables` and `tableCaptions`) was graciously sponsored by David Vins and Omedym.

The TEX implementation of the package draws inspiration from several sources including the source code of LATEX 2_ε, the minted package by Geoffrey M. Poore, which likewise tackles the issue of interfacing with an external interpreter from TEX, the filecontents package by Scott Pakin and others.

2 Interfaces

This part of the documentation describes the interfaces exposed by the package along with usage notes and examples. It is aimed at the user of the package.

Since neither TEX nor Lua provide interfaces as a language construct, the separation to interfaces and implementations is a *gentlemen's agreement*. It serves as a means of structuring this documentation and as a promise to the user that if they only access the package through the interface, the future minor versions of the package should remain backwards compatible.

Figure 1 shows the high-level structure of the Markdown package: The translation from markdown to TEX *token renderers* is exposed by the Lua layer. The plain TEX layer exposes the conversion capabilities of Lua as TEX macros. The LATEX and ConTEXt layers provide syntactic sugar on top of plain TEX macros. The user can interface with any and all layers.

2.1 Lua Interface

The Lua interface provides the conversion from UTF-8 encoded markdown to plain TEX. This interface is used by the plain TEX implementation (see Section 3.2) and will be of interest to the developers of other packages and Lua modules.

The Lua interface is implemented by the `markdown` Lua module.

```
25 local M = {metadata = metadata}
```

2.1.1 Conversion from Markdown to Plain TEX

The Lua interface exposes the `new(options)` method. This method creates converter functions that perform the conversion from markdown to plain TEX according to the table `options` that contains options recognized by the Lua interface. (see Section 2.1.2). The `options` parameter is optional; when unspecified, the behaviour will be the same as if `options` were an empty table.

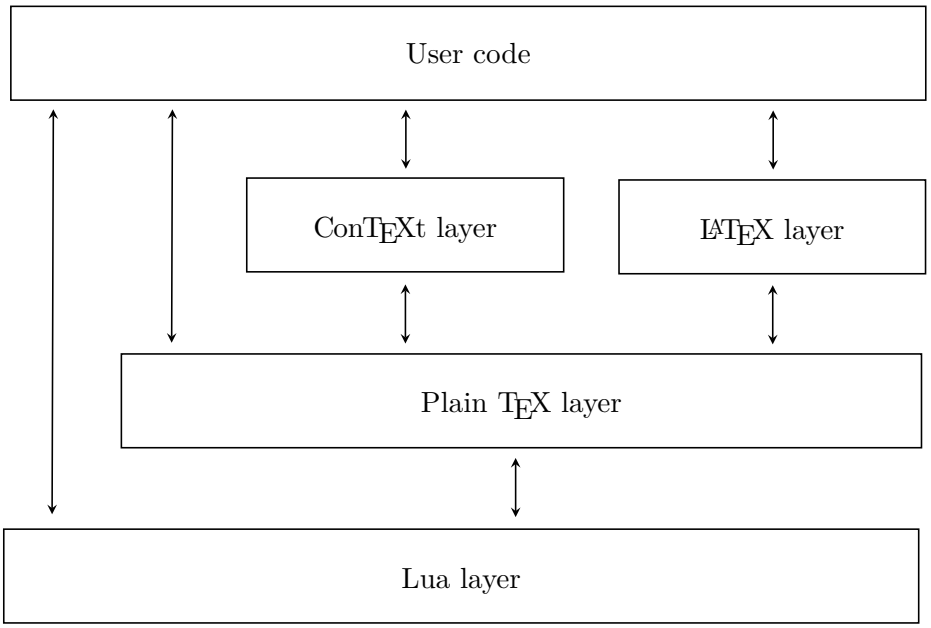

Figure 1: A block diagram of the Markdown package

The following example Lua code converts the markdown string `Hello *world*!` to a TEX output using the default options and prints the TEX output:

```lua
local md = require("markdown")
local convert = md.new()
print(convert("Hello *world*!"))
```

2.1.2 Options

The Lua interface recognizes the following options. When unspecified, the value of a key is taken from the `defaultOptions` table.

```
local defaultOptions = {}
```

To enable the enumeration of Lua options, we will maintain the `\g_@@_lua_options_seq` sequence.

```
\ExplSyntaxOn
\seq_new:N \g_@@_lua_options_seq
```

To enable the reflection of default Lua options and their types, we will maintain the `\g_@@_default_lua_options_prop` and `\g_@@_lua_option_types_prop` property lists, respectively.

```
\prop_new:N \g_@@_lua_option_types_prop
\prop_new:N \g_@@_default_lua_options_prop
```

2.1.3 File and Directory Names

`cacheDir`=⟨*path*⟩ default: `.`

A path to the directory containing auxiliary cache files. If the last segment of the path does not exist, it will be created by the Lua command-line and plain TEX implementations. The Lua implementation expects that the entire path already exists.

When iteratively writing and typesetting a markdown document, the cache files are going to accumulate over time. You are advised to clean the cache directory every now and then, or to set it to a temporary filesystem (such as `/tmp` on UN*X systems), which gets periodically emptied.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { cacheDir }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { cacheDir }
  { string }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { cacheDir }
  { . }
```

```
defaultOptions.cacheDir = "."
```

`frozenCacheFileName`=⟨*path*⟩ default: `frozenCache.tex`

A path to an output file (frozen cache) that will be created when the `finalizeCache` option is enabled and will contain a mapping between an enumeration of markdown documents and their auxiliary cache files.

The frozen cache makes it possible to later typeset a plain TEX document that contains markdown documents without invoking Lua using the `\markdownOptionFrozenCache` plain TEX option. As a result, the plain TEX document becomes more portable, but further changes in the order and the content of markdown documents will not be reflected.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { frozenCacheFileName }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { frozenCacheFileName }
  { string }
```

```
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { frozenCacheFileName }
  { frozenCache.tex }
defaultOptions.frozenCacheFileName = "frozenCache.tex"
```

2.1.4 Parser Options

`blankBeforeBlockquote`=`true`, `false` default: `false`

`true` Require a blank line between a paragraph and the following blockquote.

`false` Do not require a blank line between a paragraph and the following blockquote.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { blankBeforeBlockquote }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { blankBeforeBlockquote }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { blankBeforeBlockquote }
  { false }
defaultOptions.blankBeforeBlockquote = false
```

`blankBeforeCodeFence`=`true`, `false` default: `false`

`true` Require a blank line between a paragraph and the following fenced code block.

`false` Do not require a blank line between a paragraph and the following fenced code block.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { blankBeforeCodeFence }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { blankBeforeCodeFence }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { blankBeforeCodeFence }
  { false }
```

```
defaultOptions.blankBeforeCodeFence = false
```

`blankBeforeHeading`=`true`, `false` default: `false`

`true`	Require a blank line between a paragraph and the following header.
`false`	Do not require a blank line between a paragraph and the following header.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { blankBeforeHeading }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { blankBeforeHeading }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { blankBeforeHeading }
  { false }
```

```
defaultOptions.blankBeforeHeading = false
```

`breakableBlockquotes`=`true`, `false` default: `false`

`true`	A blank line separates block quotes.
`false`	Blank lines in the middle of a block quote are ignored.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { breakableBlockquotes }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { breakableBlockquotes }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { breakableBlockquotes }
  { false }
```

```
defaultOptions.breakableBlockquotes = false
```

`citationNbsps`=true, false default: `false`

`true` Replace regular spaces with non-breaking spaces inside the prenotes and postnotes of citations produced via the pandoc citation syntax extension.

`false` Do not replace regular spaces with non-breaking spaces inside the prenotes and postnotes of citations produced via the pandoc citation syntax extension.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { citationNbsps }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { citationNbsps }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { citationNbsps }
  { true }
defaultOptions.citationNbsps = true
```

`citations`=true, false default: `false`

`true` Enable the pandoc citation syntax extension:

```
Here is a simple parenthetical citation [@doe99] and here
is a string of several [see @doe99, pp. 33-35; also
@smith04, chap. 1].

A parenthetical citation can have a [prenote @doe99] and
a [@smith04 postnote]. The name of the author can be
suppressed by inserting a dash before the name of an
author as follows [-@smith04].

Here is a simple text citation @doe99 and here is
a string of several @doe99 [pp. 33-35; also @smith04,
chap. 1]. Here is one with the name of the author
suppressed -@doe99.
```

`false` Disable the pandoc citation syntax extension.

```
\seq_put_right:Nn
```

```
    \g_@@_lua_options_seq
    { citations }
\prop_put:Nnn
    \g_@@_lua_option_types_prop
    { citations }
    { boolean }
\prop_put:Nnn
    \g_@@_default_lua_options_prop
    { citations }
    { false }
defaultOptions.citations = false
```

`codeSpans`=`true`, `false` default: `true`

`true` Enable the code span syntax:

```
Use the `printf()` function.
``There is a literal backtick (`) here.``
```

`false` Disable the code span syntax. This allows you to easily use the quotation mark ligatures in texts that do not contain code spans:

```
``This is a quote.''
```

```
\seq_put_right:Nn
    \g_@@_lua_options_seq
    { codeSpans }
\prop_put:Nnn
    \g_@@_lua_option_types_prop
    { codeSpans }
    { boolean }
\prop_put:Nnn
    \g_@@_default_lua_options_prop
    { codeSpans }
    { true }
defaultOptions.codeSpans = true
```

`contentBlocks`=`true`, `false` default: `false`

`true` Enable the iA Writer content blocks syntax extension [3]:

```
http://example.com/minard.jpg (Napoleon's
  disastrous Russian campaign of 1812)
/Flowchart.png "Engineering Flowchart"
```

```
/Savings Account.csv 'Recent Transactions'
/Example.swift
/Lorem Ipsum.txt
```

`false` Disable the iA Writer content blocks syntax extension.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { contentBlocks }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { contentBlocks }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { contentBlocks }
  { false }
defaultOptions.contentBlocks = false
```

`contentBlocksLanguageMap`=⟨*filename*⟩
default: `markdown-languages.json`

The filename of the JSON file that maps filename extensions to programming language names in the iA Writer content blocks. See Section 2.2.3.11 for more information.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { contentBlocksLanguageMap }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { contentBlocksLanguageMap }
  { string }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { contentBlocksLanguageMap }
  { markdown-languages.json }
defaultOptions.contentBlocksLanguageMap = "markdown-languages.json"
```

`definitionLists`=`true`, `false` default: `false`

`true` Enable the pandoc definition list syntax extension:

```
Term 1

```

```
:   Definition 1

Term 2 with *inline markup*

:   Definition 2

        { some code, part of Definition 2 }

    Third paragraph of definition 2.
```

`false` Disable the pandoc definition list syntax extension.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { definitionLists }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { definitionLists }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { definitionLists }
  { false }
defaultOptions.definitionLists = false
```

`eagerCache`=`true`, `false` default: `true`

`true` Converted markdown documents will be cached in `cacheDir`. This can be useful for post-processing the converted documents and for recovering historical versions of the documents from the cache. However, it also produces a large number of auxiliary files on the disk and obscures the output of the Lua command-line interface when it is used for plumbing.

This behavior will always be used if the `finalizeCache` option is enabled.

`false` Converted markdown documents will not be cached. This decreases the number of auxiliary files that we produce and makes it easier to use the Lua command-line interface for plumbing.

This behavior will only be used when the `finalizeCache` option is disabled. Furthermore, this behavior is planned to be the new default in the next major release of the Markdown package.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { eagerCache }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { eagerCache }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { eagerCache }
  { true }
defaultOptions.eagerCache = true
```

`expectJekyllData`=`true`, `false` default: `false`

`false` When the `jekyllData` option is enabled, then a markdown document may begin with YAML metadata if and only if the metadata begin with the end-of-directives marker (`---`) and they end with either the end-of-directives or the end-of-document marker (`...`):

```
\documentclass{article}
\usepackage[jekyllData]{markdown}
\begin{document}
\begin{markdown}
---
- this
- is
- YAML
...
- followed
- by
- Markdown
\end{markdown}
\begin{markdown}
- this
- is
- Markdown
\end{markdown}
\end{document}
```

`true` When the `jekyllData` option is enabled, then a markdown document may begin directly with YAML metadata and may contain nothing but YAML metadata.

```
\documentclass{article}
\usepackage[jekyllData, expectJekyllData]{markdown}
\begin{document}
\begin{markdown}
- this
- is
- YAML
...
- followed
- by
- Markdown
\end{markdown}
\begin{markdown}
- this
- is
- YAML
\end{markdown}
\end{document}
```

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { expectJekyllData }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { expectJekyllData }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { expectJekyllData }
  { false }
```

```
defaultOptions.expectJekyllData = false
```

`fencedCode`=`true`, `false` default: `false`

`true` Enable the commonmark fenced code block extension:

```
~~~ js
if (a > 3) {
    moveShip(5 * gravity, DOWN);
}
~~~~~~
```

```` ``` html
<pre>
  <code>
    // Some comments
    line 1 of code
    line 2 of code
    line 3 of code
  </code>
</pre>
```
````

`false` Disable the commonmark fenced code block extension.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { fencedCode }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { fencedCode }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { fencedCode }
  { false }
defaultOptions.fencedCode = false
```

`finalizeCache`=`true`, `false` default: `false`

Whether an output file specified with the `frozenCacheFileName` option (frozen cache) that contains a mapping between an enumeration of markdown documents and their auxiliary cache files will be created.

The frozen cache makes it possible to later typeset a plain TEX document that contains markdown documents without invoking Lua using the `\markdownOptionFrozenCache` plain TEX option. As a result, the plain TEX document becomes more portable, but further changes in the order and the content of markdown documents will not be reflected.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { finalizeCache }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { finalizeCache }
  { boolean }
```
```

```
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { finalizeCache }
  { false }
defaultOptions.finalizeCache = false
```

`footnotes`=`true`, `false` default: `false`

`true` Enable the pandoc footnote syntax extension:

```
Here is a footnote reference,[^1] and another.[^longnote]

[^1]: Here is the footnote.

[^longnote]: Here's one with multiple blocks.

    Subsequent paragraphs are indented to show that they
belong to the previous footnote.

        { some.code }

    The whole paragraph can be indented, or just the
    first line.  In this way, multi-paragraph footnotes
    work like multi-paragraph list items.

This paragraph won't be part of the note, because it
isn't indented.
```

`false` Disable the pandoc footnote syntax extension.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { footnotes }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { footnotes }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { footnotes }
  { false }
defaultOptions.footnotes = false
```

`frozenCacheCounter`=⟨*number*⟩ default: `0`

The number of the current markdown document that will be stored in an output file (frozen cache) when the `finalizeCache` is enabled. When the document number is 0, then a new frozen cache will be created. Otherwise, the frozen cache will be appended.

Each frozen cache entry will define a TEX macro `\markdownFrozenCache`⟨*number*⟩ that will typeset markdown document number ⟨*number*⟩.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { frozenCacheCounter }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { frozenCacheCounter }
  { counter }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { frozenCacheCounter }
  { 0 }
```

```
defaultOptions.frozenCacheCounter = 0
```

`hardLineBreaks`=`true`, `false` default: `false`

`true` Interpret all newlines within a paragraph as hard line breaks instead of spaces.

`false` Interpret all newlines within a paragraph as spaces.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { hardLineBreaks }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { hardLineBreaks }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { hardLineBreaks }
  { false }
```

```
defaultOptions.hardLineBreaks = false
```

`hashEnumerators`=`true`, `false` default: `false`

`true` Enable the use of hash symbols (`#`) as ordered item list markers:

```
#. Bird
#. McHale
#. Parish
```

`false` Disable the use of hash symbols (`#`) as ordered item list markers.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { hashEnumerators }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { hashEnumerators }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { hashEnumerators }
  { false }
```

```
defaultOptions.hashEnumerators = false
```

`headerAttributes`=`true`, `false` default: `false`

`true` Enable the assignment of HTML attributes to headings:

```
# My first heading {#foo}

## My second heading ##    {#bar .baz}

Yet another heading   {key=value}
==================
```

These HTML attributes have currently no effect other than enabling content slicing, see the `slice` option.

`false` Disable the assignment of HTML attributes to headings.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { headerAttributes }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { headerAttributes }
```

```
    { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { headerAttributes }
  { false }
defaultOptions.headerAttributes = false
```

`html`=`true`, `false` default: `false`

`true` Enable the recognition of inline HTML tags, block HTML elements, HTML comments, HTML instructions, and entities in the input. Inline HTML tags, block HTML elements and HTML comments will be rendered, HTML instructions will be ignored, and HTML entities will be replaced with the corresponding Unicode codepoints.

`false` Disable the recognition of HTML markup. Any HTML markup in the input will be rendered as plain text.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { html }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { html }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { html }
  { false }
defaultOptions.html = false
```

`hybrid`=`true`, `false` default: `false`

`true` Disable the escaping of special plain TEX characters, which makes it possible to intersperse your markdown markup with TEX code. The intended usage is in documents prepared manually by a human author. In such documents, it can often be desirable to mix TEX and markdown markup freely.

`false` Enable the escaping of special plain TEX characters outside verbatim environments, so that they are not interpretted by TEX. This is encouraged when typesetting automatically generated content or markdown documents that were not prepared with this package in mind.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { hybrid }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { hybrid }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { hybrid }
  { false }
```

```
defaultOptions.hybrid = false
```

`inlineFootnotes`=true, false default: `false`

`true` Enable the pandoc inline footnote syntax extension:

```
Here is an inline note.^[Inlines notes are easier to
write, since you don't have to pick an identifier and
move down to type the note.]
```

`false` Disable the pandoc inline footnote syntax extension.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { inlineFootnotes }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { inlineFootnotes }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { inlineFootnotes }
  { false }
```

```
defaultOptions.inlineFootnotes = false
```

`jekyllData`=true, false default: `false`

`true` Enable the Pandoc `yaml_metadata_block` syntax extension for entering metadata in YAML:

```
---
title:  'This is the title: it contains a colon'
author:
```

```
- Author One
- Author Two
keywords: [nothing, nothingness]
abstract: |
  This is the abstract.

  It consists of two paragraphs.
---
```

`false` Disable the Pandoc `yaml_metadata_block` syntax extension for entering metadata in YAML.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { jekyllData }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { jekyllData }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { jekyllData }
  { false }
defaultOptions.jekyllData = false
```

`pipeTables`=`true`, `false` default: `false`

`true` Enable the PHP Markdown table syntax extension:

```
| Right | Left | Default | Center |
|------:|:-----|---------|:------:|
|   12  |  12  |    12   |    12  |
|  123  |  123 |   123   |   123  |
|    1  |    1 |     1   |     1  |
```

`false` Disable the PHP Markdown table syntax extension.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { pipeTables }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { pipeTables }
  { boolean }
```

```
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { pipeTables }
  { false }
defaultOptions.pipeTables = false
```

`preserveTabs`=true, false default: `false`

`true` Preserve tabs in code block and fenced code blocks.

`false` Convert any tabs in the input to spaces.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { preserveTabs }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { preserveTabs }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { preserveTabs }
  { false }
defaultOptions.preserveTabs = false
```

`relativeReferences`=true, false default: `false`

`true` Enable relative references[6] in autolinks:

```
I conclude in Section <#conclusion>.

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

`false` Disable relative references in autolinks.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { relativeReferences }
\prop_put:Nnn
```

[6] See https://datatracker.ietf.org/doc/html/rfc3986#section-4.2.

```
  \g_@@_lua_option_types_prop
  { relativeReferences }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { relativeReferences }
  { false }
defaultOptions.relativeReferences = false
```

`shiftHeadings`=⟨*shift amount*⟩ default: `0`

All headings will be shifted by ⟨*shift amount*⟩, which can be both positive and negative. Headings will not be shifted beyond level 6 or below level 1. Instead, those headings will be shifted to level 6, when ⟨*shift amount*⟩ is positive, and to level 1, when ⟨*shift amount*⟩ is negative.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { shiftHeadings }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { shiftHeadings }
  { number }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { shiftHeadings }
  { 0 }
defaultOptions.shiftHeadings = 0
```

`slice`=⟨*the beginning and the end of a slice*⟩ default: `^ $`

Two space-separated selectors that specify the slice of a document that will be processed, whereas the remainder of the document will be ignored. The following selectors are recognized:

- The circumflex (`^`) selects the beginning of a document.
- The dollar sign (`$`) selects the end of a document.
- `^`⟨*identifier*⟩ selects the beginning of a section with the HTML attribute `#`⟨*identifier*⟩ (see the `headerAttributes` option).
- `$`⟨*identifier*⟩ selects the end of a section with the HTML attribute `#`⟨*identifier*⟩.
- ⟨*identifier*⟩ corresponds to `^`⟨*identifier*⟩ for the first selector and to `$`⟨*identifier*⟩ for the second selector.

Specifying only a single selector, ⟨*identifier*⟩, is equivalent to specifying the two selectors ⟨*identifier*⟩ ⟨*identifier*⟩, which is equivalent to `^`⟨*identifier*⟩ `$`⟨*identifier*⟩, i.e. the entire section with the HTML attribute `#`⟨*identifier*⟩ will be selected.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { slice }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { slice }
  { string }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { slice }
  { ^~$ }
```

```
defaultOptions.slice = "^ $"
```

`smartEllipses`=`true`, `false` default: `false`

`true` Convert any ellipses in the input to the `\markdownRendererEllipsis` TEX macro.

`false` Preserve all ellipses in the input.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { smartEllipses }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { smartEllipses }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { smartEllipses }
  { false }
```

```
defaultOptions.smartEllipses = false
```

`startNumber`=`true`, `false` default: `true`

`true` Make the number in the first item of an ordered lists significant. The item numbers will be passed to the `\markdownRendererOlItemWithNumber` TEX macro.

`false` Ignore the numbers in the ordered list items. Each item will only produce a `\markdownRendererOlItem` TEX macro.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { startNumber }
```

```
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { startNumber }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { startNumber }
  { true }
defaultOptions.startNumber = true
```

`stripIndent`=true, `false` default: `false`

`true` Strip the minimal indentation of non-blank lines from all lines in a markdown document. Requires that the `preserveTabs` Lua option is `false`:

```
\documentclass{article}
\usepackage[stripIndent]{markdown}
\begin{document}
    \begin{markdown}
        Hello *world*!
    \end{markdown}
\end{document}
```

`false` Do not strip any indentation from the lines in a markdown document.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { stripIndent }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { stripIndent }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { stripIndent }
  { false }
defaultOptions.stripIndent = false
```

`tableCaptions`=true, `false` default: `false`

`true` Enable the Pandoc `table_captions` syntax extension for pipe tables (see the `pipeTables` option).

```
| Right | Left | Default | Center |
|------:|:-----|---------|:------:|
|   12  |  12  |    12   |    12  |
|  123  |  123 |   123   |   123  |
|    1  |    1 |     1   |     1  |

  : Demonstration of pipe table syntax.
```

`false` Disable the Pandoc `table_captions` syntax extension.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { tableCaptions }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { tableCaptions }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { tableCaptions }
  { false }
defaultOptions.tableCaptions = false
```

`taskLists`=`true`, `false` default: `false`

`true` Enable the Pandoc `task_lists` syntax extension.

```
- [ ] an unticked task list item
- [/] a half-checked task list item
- [X] a ticked task list item
```

`false` Disable the Pandoc `task_lists` syntax extension.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { taskLists }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { taskLists }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { taskLists }
  { false }
defaultOptions.taskLists = false
```

`texComments`=`true`, `false` default: `false`

`true` Strip TEX-style comments.

```
\documentclass{article}
\usepackage[texComments]{markdown}
\begin{document}
\begin{markdown}
Hello *world*!
\end{markdown}
\end{document}
```

Always enabled when `hybrid` is enabled.

`false` Do not strip TEX-style comments.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { texComments }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { texComments }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { texComments }
  { false }
defaultOptions.texComments = false
```

`tightLists`=`true`, `false` default: `true`

`true` Unordered and ordered Lists whose items do not consist of multiple paragraphs will be considered *tight*. Tight lists will produce tight renderers that may produce different output than lists that are not tight:

```
- This is
- a tight
- unordered list.

- This is

  not a tight

- unordered list.
```

`false` Unordered and ordered lists whose items consist of multiple paragraphs will be treated the same way as lists that consist of multiple paragraphs.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { tightLists }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { tightLists }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { tightLists }
  { true }
```

```
defaultOptions.tightLists = true
```

`underscores`=`true`, `false` default: `true`

`true` Both underscores and asterisks can be used to denote emphasis and strong emphasis:

```
*single asterisks*
_single underscores_
**double asterisks**
__double underscores__
```

`false` Only asterisks can be used to denote emphasis and strong emphasis. This makes it easy to write math with the `hybrid` option without the need to constantly escape subscripts.

```
\seq_put_right:Nn
  \g_@@_lua_options_seq
  { underscores }
\prop_put:Nnn
  \g_@@_lua_option_types_prop
  { underscores }
  { boolean }
\prop_put:Nnn
  \g_@@_default_lua_options_prop
  { underscores }
  { true }
\ExplSyntaxOff
```

```
defaultOptions.underscores = true
```

2.1.5 Command-Line Interface

The high-level operation of the Markdown package involves the communication between several programming layers: the plain TEX layer hands markdown documents to the Lua layer. Lua converts the documents to TEX, and hands the converted documents back to plain TEX layer for typesetting, see Figure 2.

This procedure has the advantage of being fully automated. However, it also has several important disadvantages: The converted TEX documents are cached on the file system, taking up increasing amount of space. Unless the TEX engine includes a Lua interpreter, the package also requires shell access, which opens the door for a malicious actor to access the system. Last, but not least, the complexity of the procedure impedes debugging.

A solution to the above problems is to decouple the conversion from the typesetting. For this reason, a command-line Lua interface for converting a markdown document to TEX is also provided, see Figure 3.

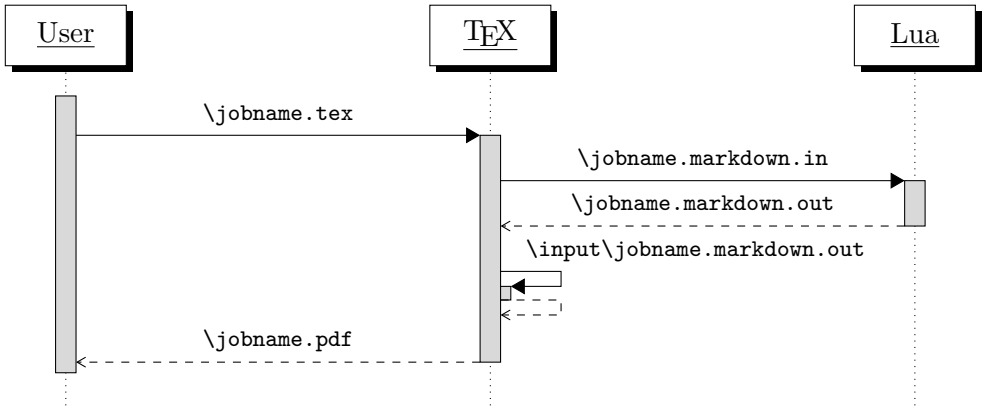

Figure 2: A sequence diagram of the Markdown package typesetting a markdown document using the TEX interface

```
488
489 HELP_STRING = [[
490 Usage: texlua ]] .. arg[0] .. [[ [OPTIONS] -- [INPUT_FILE] [OUTPUT_FILE]
491 where OPTIONS are documented in the Lua interface section of the
492 technical Markdown package documentation.
493
494 When OUTPUT_FILE is unspecified, the result of the conversion will be
495 written to the standard output. When INPUT_FILE is also unspecified, the
496 result of the conversion will be read from the standard input.
497
498 Report bugs to: witiko@mail.muni.cz
499 Markdown package home page: <https://github.com/witiko/markdown>]]
500
```

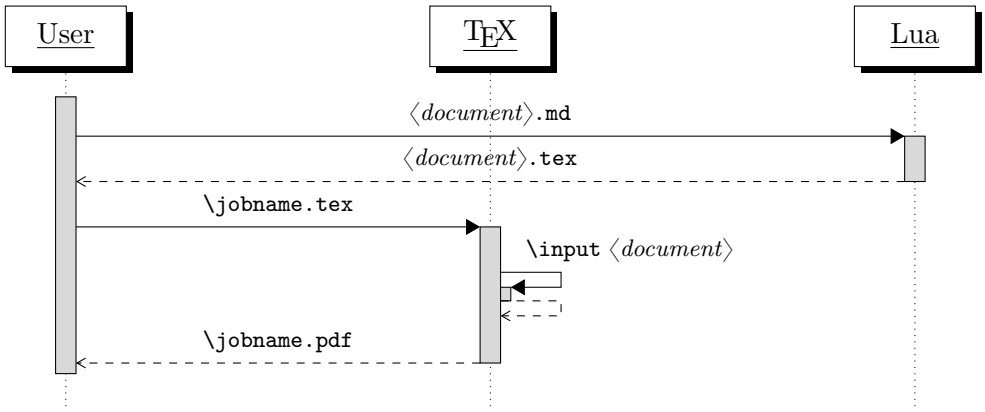

Figure 3: A sequence diagram of the Markdown package typesetting a markdown document using the Lua command-line interface

```
VERSION_STRING = [[
markdown-cli.lua (Markdown) ]] .. metadata.version .. [[

Copyright (C) ]] .. table.concat(metadata.copyright,
                                 "\nCopyright (C) ") .. [[

License: ]] .. metadata.license

local function warn(s)
  io.stderr:write("Warning: " .. s .. "\n") end

local function error(s)
  io.stderr:write("Error: " .. s .. "\n")
  os.exit(1) end

local process_options = true
local options = {}
local input_filename
local output_filename
for i = 1, #arg do
  if process_options then
```

After the optional `--` argument has been specified, the remaining arguments are assumed to be input and output filenames. This argument is optional, but encouraged, because it helps resolve ambiguities when deciding whether an option or a filename has been specified.

```
    if arg[i] == "--" then
      process_options = false
      goto continue
```

Unless the `--` argument has been specified before, an argument containing the equals sign (`=`) is assumed to be an option specification in a ⟨*key*⟩`=`⟨*value*⟩ format. The available options are listed in Section 2.1.2.

```
    elseif arg[i]:match("=") then
      key, value = arg[i]:match("(.-)=(.*)")
```

The `defaultOptions` table is consulted to identify whether ⟨*value*⟩ should be parsed as a string or as a boolean.

```
      default_type = type(defaultOptions[key])
      if default_type == "boolean" then
        options[key] = (value == "true")
      elseif default_type == "number" then
        options[key] = tonumber(value)
      else
        if default_type ~= "string" then
          if default_type == "nil" then
            warn('Option "' .. key .. '" not recognized.')
          else
            warn('Option "' .. key .. '" type not recognized, please file ' ..
                 'a report to the package maintainer.')
          end
          warn('Parsing the ' .. 'value "' .. value ..'" of option "' ..
               key .. '" as a string.')
        end
        options[key] = value
      end
      goto continue
```

Unless the `--` argument has been specified before, an argument `--help`, or `-h` causes a brief documentation for how to invoke the program to be printed to the standard output.

```
    elseif arg[i] == "--help" or arg[i] == "-h" then
      print(HELP_STRING)
      os.exit()
```

Unless the `--` argument has been specified before, an argument `--version`, or `-v` causes the program to print information about its name, version, origin and legal status, all on standard output.

```
    elseif arg[i] == "--version" or arg[i] == "-v" then
      print(VERSION_STRING)
      os.exit()
    end
  end
```

The first argument that matches none of the above patters is assumed to be the input filename. The input filename should correspond to the Markdown document that is going to be converted to a TEX document.

```
554   if input_filename == nil then
555     input_filename = arg[i]
```

The first argument that matches none of the above patters is assumed to be the output filename. The output filename should correspond to the TEX document that will result from the conversion.

```
556   elseif output_filename == nil then
557     output_filename = arg[i]
558   else
559     error('Unexpected argument: "' .. arg[i] .. '".')
560   end
561   ::continue::
562 end
```

The command-line Lua interface is implemented by the `markdown-cli.lua` file that can be invoked from the command line as follows:

```
texlua /path/to/markdown-cli.lua cacheDir=. -- hello.md hello.tex
```

to convert the Markdown document `hello.md` to a TEX document `hello.tex`. After the Markdown package for our TEX format has been loaded, the converted document can be typeset as follows:

```
\input hello
```

2.2 Plain TEX Interface

The plain TEX interface provides macros for the typesetting of markdown input from within plain TEX, for setting the Lua interface options (see Section 2.1.2) used during the conversion from markdown to plain TEX and for changing the way markdown the tokens are rendered.

```
563 \def\markdownLastModified{(((LASTMODIFIED)))}%
564 \def\markdownVersion{(((VERSION)))}%
```

The plain TEX interface is implemented by the `markdown.tex` file that can be loaded as follows:

```
\input markdown
```

It is expected that the special plain TEX characters have the expected category codes, when `\input`ting the file.

2.2.1 Typesetting Markdown

The interface exposes the \markdownBegin, \markdownEnd, and \markdownInput macros.

The \markdownBegin macro marks the beginning of a markdown document fragment and the \markdownEnd macro marks its end.

```
565 \let\markdownBegin\relax
566 \let\markdownEnd\relax
```

You may prepend your own code to the \markdownBegin macro and redefine the \markdownEnd macro to produce special effects before and after the markdown block.

There are several limitations to the macros you need to be aware of. The first limitation concerns the \markdownEnd macro, which must be visible directly from the input line buffer (it may not be produced as a result of input expansion). Otherwise, it will not be recognized as the end of the markdown string. As a corrolary, the \markdownEnd string may not appear anywhere inside the markdown input.

Another limitation concerns spaces at the right end of an input line. In markdown, these are used to produce a forced line break. However, any such spaces are removed before the lines enter the input buffer of TEX [4, p. 46]. As a corrolary, the \markdownBegin macro also ignores them.

The \markdownBegin and \markdownEnd macros will also consume the rest of the lines at which they appear. In the following example plain TEX code, the characters c, e, and f will not appear in the output.

```
\input markdown
a
b \markdownBegin c
d
e \markdownEnd   f
g
\bye
```

Note that you may also not nest the \markdownBegin and \markdownEnd macros.

The following example plain TEX code showcases the usage of the \markdownBegin and \markdownEnd macros:

```
\input markdown
\markdownBegin
_Hello_ **world** ...
\markdownEnd
\bye
```

The `\markdownInput` macro accepts a single parameter containing the filename of a markdown document and expands to the result of the conversion of the input markdown document to plain TEX.

```
\let\markdownInput\relax
```

This macro is not subject to the abovelisted limitations of the `\markdownBegin` and `\markdownEnd` macros.

The following example plain TEX code showcases the usage of the `\markdownInput` macro:

```
\input markdown
\markdownInput{hello.md}
\bye
```

2.2.2 Options

The plain TEX options are represented by TEX commands. Some of them map directly to the options recognized by the Lua interface (see Section 2.1.2), while some of them are specific to the plain TEX interface.

2.2.2.1 Finalizing and Freezing the Cache

The `\markdownOptionFinalizeCache` option corresponds to the Lua interface `finalizeCache` option, which creates an output file `\markdownOptionFrozenCacheFileName` (frozen cache) that contains a mapping between an enumeration of the markdown documents in the plain TEX document and their auxiliary files cached in the `cacheDir` directory.

```
\let\markdownOptionFinalizeCache\undefined
```

The `\markdownOptionFrozenCache` option uses the mapping previously created by the `\markdownOptionFinalizeCache` option, and uses it to typeset the plain TEX document without invoking Lua. As a result, the plain TEX document becomes more portable, but further changes in the order and the content of markdown documents will not be reflected. It defaults to `false`.

The standard usage of the above two options is as follows:

1. Remove the `cacheDir` cache directory with stale auxiliary cache files.
2. Enable the `\markdownOptionFinalizeCache` option.
4. Typeset the plain TEX document to populate and finalize the cache.
5. Enable the `\markdownOptionFrozenCache` option.
6. Publish the source code of the plain TEX document and the `cacheDir` directory.

2.2.2.2 File and Directory Names

The `\markdownOptionHelperScriptFileName` macro sets the filename of the helper Lua script file that is created during the conversion from markdown to plain TEX in TEX engines without the `\directlua`

primitive. It defaults to `\jobname.markdown.lua`, where `\jobname` is the base name of the document being typeset.

The expansion of this macro must not contain quotation marks (`"`) or backslash symbols (`\`). Mind that TeX engines tend to put quotation marks around `\jobname`, when it contains spaces.

```
\def\markdownOptionHelperScriptFileName{\jobname.markdown.lua}%
```

The `\markdownOptionInputTempFileName` macro sets the filename of the temporary input file that is created during the conversion from markdown to plain TeX in `\markdownMode` other than `2`. It defaults to `\jobname.markdown.out`. The same limitations as in the case of the `\markdownOptionHelperScriptFileName` macro apply here.

```
\def\markdownOptionInputTempFileName{\jobname.markdown.in}%
```

The `\markdownOptionOutputTempFileName` macro sets the filename of the temporary output file that is created during the conversion from markdown to plain TeX in `\markdownMode` other than `2`. It defaults to `\jobname.markdown.out`. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
\def\markdownOptionOutputTempFileName{\jobname.markdown.out}%
```

The `\markdownOptionErrorTempFileName` macro sets the filename of the temporary output file that is created when a Lua error is encountered during the conversion from markdown to plain TeX in `\markdownMode` other than `2`. It defaults to `\jobname.markdown.err`. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
\def\markdownOptionErrorTempFileName{\jobname.markdown.err}%
```

The `\markdownOptionOutputDir` macro sets the path to the directory that will contain the auxiliary cache files produced by the Lua implementation and also the auxiliary files produced by the plain TeX implementation. The option defaults to `.`.

The path must be set to the same value as the `-output-directory` option of your TeX engine for the package to function correctly. We need this macro to make the Lua implementation aware where it should store the helper files. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
\def\markdownOptionOutputDir{.}%
```

The `\markdownOptionCacheDir` macro corresponds to the Lua interface `cacheDir` option that sets the path to the directory that will contain the produced cache files. The option defaults to `_markdown_\jobname`, which is a similar naming scheme to the one used by the minted LaTeX package. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
\def\markdownOptionCacheDir{\markdownOptionOutputDir/_markdown_\jobname}%
```

The `\markdownOptionFrozenCacheFileName` macro corresponds to the Lua interface `frozenCacheFileName` option that sets the path to an output file (frozen

cache) that will contain a mapping between an enumeration of the markdown documents in the plain TEX document and their auxiliary cache files. The option defaults to `frozenCache.tex`. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
\def\markdownOptionFrozenCacheFileName{\markdownOptionCacheDir/frozenCache.tex}
```

2.2.2.3 Lua Interface Options

The following macros map directly to the options recognized by the Lua interface (see Section 2.1.2) and are not processed by the plain TEX implementation, only passed along to Lua. They are undefined, which makes them fall back to the default values provided by the Lua interface.

For the macros that correspond to the non-boolean options recognized by the Lua interface, the same limitations apply here in the case of the `\markdownOptionHelperScriptFileName` macro.

```
\let\markdownOptionBlankBeforeBlockquote\undefined
\let\markdownOptionBlankBeforeCodeFence\undefined
\let\markdownOptionBlankBeforeHeading\undefined
\let\markdownOptionBreakableBlockquotes\undefined
\let\markdownOptionCitations\undefined
\let\markdownOptionCitationNbsps\undefined
\let\markdownOptionContentBlocks\undefined
\let\markdownOptionContentBlocksLanguageMap\undefined
\let\markdownOptionDefinitionLists\undefined
\let\markdownOptionEagerCache\undefined
\let\markdownOptionFootnotes\undefined
\let\markdownOptionFencedCode\undefined
\let\markdownOptionHardLineBreaks\undefined
\let\markdownOptionHashEnumerators\undefined
\let\markdownOptionHeaderAttributes\undefined
\let\markdownOptionHtml\undefined
\let\markdownOptionHybrid\undefined
\let\markdownOptionInlineFootnotes\undefined
\let\markdownOptionJekyllData\undefined
\let\markdownOptionPipeTables\undefined
\let\markdownOptionPreserveTabs\undefined
\let\markdownOptionRelativeReferences\undefined
\let\markdownOptionShiftHeadings\undefined
\let\markdownOptionSlice\undefined
\let\markdownOptionSmartEllipses\undefined
\let\markdownOptionStartNumber\undefined
\let\markdownOptionStripIndent\undefined
\let\markdownOptionTableCaptions\undefined
\let\markdownOptionTaskLists\undefined
\let\markdownOptionTexComments\undefined
\let\markdownOptionTightLists\undefined
```

2.2.2.4 Miscellaneous Options

The `\markdownOptionStripPercentSigns` macro controls whether a percent sign (`%`) at the beginning of a line will be discarded when buffering Markdown input (see Section 3.2.5) or not. Notably, this enables the use of markdown when writing TEX package documentation using the Doc LATEX package [5] or similar. The recognized values of the macro are `true` (discard) and `false` (retain). It defaults to `false`.

```
\def\markdownOptionStripPercentSigns{false}%
```

2.2.3 Token Renderers

The following TEX macros may occur inside the output of the converter functions exposed by the Lua interface (see Section 2.1.1) and represent the parsed markdown tokens. These macros are intended to be redefined by the user who is typesetting a document. By default, they point to the corresponding prototypes (see Section 2.2.4).

To enable the enumeration of token renderers, we will maintain the `\g_@@_renderers_seq` sequence.

```
\ExplSyntaxOn
\seq_new:N \g_@@_renderers_seq
```

To enable the reflection of token renderers and their parameters, we will maintain the `\g_@@_renderer_arities_prop` property list.

```
\prop_new:N \g_@@_renderer_arities_prop
\ExplSyntaxOff
```

2.2.3.1 Tickbox Renderers

The macros named `\markdownRendererTickedBox`, `\markdownRendererHalfTickedBox`, and `\markdownRendererUntickedBox` represent ticked and unticked boxes, respectively. These macros will either be produced, when the `taskLists` option is enabled, or when the Ballot Box with X (☒, U+2612), Hourglass (⊡, U+231B) or Ballot Box (▫, U+2610) Unicode characters are encountered in the markdown input, respectively.

```
\def\markdownRendererTickedBox{%
  \markdownRendererTickedBoxPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { tickedBox }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { tickedBox }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererHalfTickedBox{%
  \markdownRendererHalfTickedBoxPrototype}%
\ExplSyntaxOn
```

```
\seq_put_right:Nn
  \g_@@_renderers_seq
  { halfTickedBox }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { halfTickedBox }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererUntickedBox{%
  \markdownRendererUntickedBoxPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { untickedBox }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { untickedBox }
  { 0 }
\ExplSyntaxOff
```

2.2.3.2 Markdown Document Renderers

The `\markdownRendererDocumentBegin` and `\markdownRendererDocumentEnd` macros represent the beginning and the end of a *markdown* document. The macros receive no arguments.

A TEX document may contain any number of markdown documents. Additionally, markdown documents may appear not only in a sequence, but several markdown documents may also be *nested*. Redefinitions of the macros should take this into account.

```
\def\markdownRendererDocumentBegin{%
  \markdownRendererDocumentBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { documentBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { documentBegin }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererDocumentEnd{%
  \markdownRendererDocumentEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { documentEnd }
\prop_put:Nnn
```

```
  \g_@@_renderer_arities_prop
  { documentEnd }
  { 0 }
\ExplSyntaxOff
```

2.2.3.3 Interblock Separator Renderer

The `\markdownRendererInterblockSeparator` macro represents a separator between two markdown block elements. The macro receives no arguments.

```
\def\markdownRendererInterblockSeparator{%
  \markdownRendererInterblockSeparatorPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { interblockSeparator }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { interblockSeparator }
  { 0 }
\ExplSyntaxOff
```

2.2.3.4 Line Break Renderer

The `\markdownRendererLineBreak` macro represents a forced line break. The macro receives no arguments.

```
\def\markdownRendererLineBreak{%
  \markdownRendererLineBreakPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { lineBreak }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { lineBreak }
  { 0 }
\ExplSyntaxOff
```

2.2.3.5 Ellipsis Renderer

The `\markdownRendererEllipsis` macro replaces any occurance of ASCII ellipses in the input text. This macro will only be produced, when the `smartEllipses` option is enabled. The macro receives no arguments.

```
\def\markdownRendererEllipsis{%
  \markdownRendererEllipsisPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { ellipsis }
\prop_put:Nnn
```

```
  \g_@@_renderer_arities_prop
  { ellipsis }
  { 0 }
\ExplSyntaxOff
```

2.2.3.6 Non-Breaking Space Renderer

The `\markdownRendererNbsp` macro represents a non-breaking space.

```
\def\markdownRendererNbsp{%
  \markdownRendererNbspPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { nbsp }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { nbsp }
  { 0 }
\ExplSyntaxOff
```

2.2.3.7 Special Character Renderers

The following macros replace any special plain TEX characters, including the active pipe character (`|`) of ConTEXt, in the input text. These macros will only be produced, when the `hybrid` option is `false`.

```
\def\markdownRendererLeftBrace{%
  \markdownRendererLeftBracePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { leftBrace }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { leftBrace }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererRightBrace{%
  \markdownRendererRightBracePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { rightBrace }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { rightBrace }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererDollarSign{%
```

```
  \markdownRendererDollarSignPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { dollarSign }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { dollarSign }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererPercentSign{%
  \markdownRendererPercentSignPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { percentSign }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { percentSign }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererAmpersand{%
  \markdownRendererAmpersandPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { ampersand }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { ampersand }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererUnderscore{%
  \markdownRendererUnderscorePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { underscore }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { underscore }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererHash{%
  \markdownRendererHashPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
```

```
  \g_@@_renderers_seq
  { hash }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { hash }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererCircumflex{%
  \markdownRendererCircumflexPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { circumflex }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { circumflex }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererBackslash{%
  \markdownRendererBackslashPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { backslash }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { backslash }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererTilde{%
  \markdownRendererTildePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { tilde }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { tilde }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererPipe{%
  \markdownRendererPipePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { pipe }
\prop_put:Nnn
```

```
  \g_@@_renderer_arities_prop
  { pipe }
  { 0 }
\ExplSyntaxOff
```

2.2.3.8 Code Span Renderer

The `\markdownRendererCodeSpan` macro represents inlined code span in the input text. It receives a single argument that corresponds to the inlined code span.

```
\def\markdownRendererCodeSpan{%
  \markdownRendererCodeSpanPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { codeSpan }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { codeSpan }
  { 1 }
\ExplSyntaxOff
```

2.2.3.9 Link Renderer

The `\markdownRendererLink` macro represents a hyperlink. It receives four arguments: the label, the fully escaped URI that can be directly typeset, the raw URI that can be used outside typesetting, and the title of the link.

```
\def\markdownRendererLink{%
  \markdownRendererLinkPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { link }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { link }
  { 4 }
\ExplSyntaxOff
```

2.2.3.10 Image Renderer

The `\markdownRendererImage` macro represents an image. It receives four arguments: the label, the fully escaped URI that can be directly typeset, the raw URI that can be used outside typesetting, and the title of the link.

```
\def\markdownRendererImage{%
  \markdownRendererImagePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
```

```
  { image }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { image }
  { 4 }
\ExplSyntaxOff
```

2.2.3.11 Content Block Rendere

The `\markdownRendererContentBlock` macro represents an iA Writer content block. It receives four arguments: the local file or online image filename extension cast to the lower case, the fully escaped URI that can be directly typeset, the raw URI that can be used outside typesetting, and the title of the content block.

```
\def\markdownRendererContentBlock{%
  \markdownRendererContentBlockPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { contentBlock }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { contentBlock }
  { 4 }
\ExplSyntaxOff
```

The `\markdownRendererContentBlockOnlineImage` macro represents an iA Writer online image content block. The macro receives the same arguments as `\markdownRendererContentBlock`.

```
\def\markdownRendererContentBlockOnlineImage{%
  \markdownRendererContentBlockOnlineImagePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { contentBlockOnlineImage }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { contentBlockOnlineImage }
  { 4 }
\ExplSyntaxOff
```

The `\markdownRendererContentBlockCode` macro represents an iA Writer content block that was recognized as a file in a known programming language by its filename extension s. If any `markdown-languages.json` file found by kpathsea[7] contains a record (k, v), then a non-online-image content block with the filename

[7] Local files take precedence. Filenames other than `markdown-languages.json` may be specified using the `contentBlocksLanguageMap` Lua option.

extension s, `s:lower()` $= k$ is considered to be in a known programming language v. The macro receives five arguments: the local file name extension s cast to the lower case, the language v, the fully escaped URI that can be directly typeset, the raw URI that can be used outside typesetting, and the title of the content block.

Note that you will need to place place a `markdown-languages.json` file inside your working directory or inside your local TEX directory structure. In this file, you will define a mapping between filename extensions and the language names recognized by your favorite syntax highlighter; there may exist other creative uses beside syntax highlighting. The `Languages.json` file provided by Sotkov [3] is a good starting point.

```
\def\markdownRendererContentBlockCode{%
  \markdownRendererContentBlockCodePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { contentBlockCode }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { contentBlockCode }
  { 5 }
\ExplSyntaxOff
```

2.2.3.12 Bullet List Renderers

The `\markdownRendererUlBegin` macro represents the beginning of a bulleted list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
\def\markdownRendererUlBegin{%
  \markdownRendererUlBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { ulBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { ulBegin }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererUlBeginTight` macro represents the beginning of a bulleted list that contains no item with several paragraphs of text (the list is tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
\def\markdownRendererUlBeginTight{%
  \markdownRendererUlBeginTightPrototype}%
\ExplSyntaxOn
```

```
\seq_put_right:Nn
  \g_@@_renderers_seq
  { ulBeginTight }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { ulBeginTight }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererUlItem` macro represents an item in a bulleted list. The macro receives no arguments.

```
\def\markdownRendererUlItem{%
  \markdownRendererUlItemPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { ulItem }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { ulItem }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererUlItemEnd` macro represents the end of an item in a bulleted list. The macro receives no arguments.

```
\def\markdownRendererUlItemEnd{%
  \markdownRendererUlItemEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { ulItemEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { ulItemEnd }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererUlEnd` macro represents the end of a bulleted list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
\def\markdownRendererUlEnd{%
  \markdownRendererUlEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { ulEnd }
```

```
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { ulEnd }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererUlEndTight` macro represents the end of a bulleted list that contains no item with several paragraphs of text (the list is tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
\def\markdownRendererUlEndTight{%
  \markdownRendererUlEndTightPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { ulEndTight }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { ulEndTight }
  { 0 }
\ExplSyntaxOff
```

2.2.3.13 Ordered List Renderers

The `\markdownRendererOlBegin` macro represents the beginning of an ordered list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
\def\markdownRendererOlBegin{%
  \markdownRendererOlBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { olBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { olBegin }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererOlBeginTight` macro represents the beginning of an ordered list that contains no item with several paragraphs of text (the list is tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
\def\markdownRendererOlBeginTight{%
  \markdownRendererOlBeginTightPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
```

```
  \g_@@_renderers_seq
  { olBeginTight }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { olBeginTight }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererOlItem` macro represents an item in an ordered list. This macro will only be produced, when the `startNumber` option is `false`. The macro receives no arguments.

```
\def\markdownRendererOlItem{%
  \markdownRendererOlItemPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { olItem }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { olItem }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererOlItemEnd` macro represents the end of an item in an ordered list. The macro receives no arguments.

```
\def\markdownRendererOlItemEnd{%
  \markdownRendererOlItemEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { olItemEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { olItemEnd }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererOlItemWithNumber` macro represents an item in an ordered list. This macro will only be produced, when the `startNumber` option is enabled. The macro receives a single numeric argument that corresponds to the item number.

```
\def\markdownRendererOlItemWithNumber{%
  \markdownRendererOlItemWithNumberPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
```

```
  { olItemWithNumber }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { olItemWithNumber }
  { 1 }
\ExplSyntaxOff
```

The `\markdownRendererOlEnd` macro represents the end of an ordered list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
\def\markdownRendererOlEnd{%
  \markdownRendererOlEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { olEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { olEnd }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererOlEndTight` macro represents the end of an ordered list that contains no item with several paragraphs of text (the list is tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
\def\markdownRendererOlEndTight{%
  \markdownRendererOlEndTightPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { olEndTight }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { olEndTight }
  { 0 }
\ExplSyntaxOff
```

2.2.3.14 Definition List Renderers

The following macros are only produced, when the `definitionLists` option is enabled.

The `\markdownRendererDlBegin` macro represents the beginning of a definition list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
\def\markdownRendererDlBegin{%
  \markdownRendererDlBeginPrototype}%
```

```
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { dlBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { dlBegin }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererDlBeginTight` macro represents the beginning of a definition list that contains an item with several paragraphs of text (the list is not tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
\def\markdownRendererDlBeginTight{%
  \markdownRendererDlBeginTightPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { dlBeginTight }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { dlBeginTight }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererDlItem` macro represents a term in a definition list. The macro receives a single argument that corresponds to the term being defined.

```
\def\markdownRendererDlItem{%
  \markdownRendererDlItemPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { dlItem }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { dlItem }
  { 1 }
\ExplSyntaxOff
```

The `\markdownRendererDlItemEnd` macro represents the end of a list of definitions for a single term.

```
\def\markdownRendererDlItemEnd{%
  \markdownRendererDlItemEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
```

```
  \g_@@_renderers_seq
  { dlItemEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { dlItemEnd }
  { 0 }
\ExplSyntaxOff
```

The \markdownRendererDlDefinitionBegin macro represents the beginning of a definition in a definition list. There can be several definitions for a single term.

```
\def\markdownRendererDlDefinitionBegin{%
  \markdownRendererDlDefinitionBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { dlDefinitionBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { dlDefinitionBegin }
  { 0 }
\ExplSyntaxOff
```

The \markdownRendererDlDefinitionEnd macro represents the end of a definition in a definition list. There can be several definitions for a single term.

```
\def\markdownRendererDlDefinitionEnd{%
  \markdownRendererDlDefinitionEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { dlDefinitionEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { dlDefinitionEnd }
  { 0 }
\ExplSyntaxOff
```

The \markdownRendererDlEnd macro represents the end of a definition list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
\def\markdownRendererDlEnd{%
  \markdownRendererDlEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { dlEnd }
\prop_put:Nnn
```

```
  \g_@@_renderer_arities_prop
  { dlEnd }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererDlEndTight` macro represents the end of a definition list that contains no item with several paragraphs of text (the list is tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
\def\markdownRendererDlEndTight{%
  \markdownRendererDlEndTightPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { dlEndTight }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { dlEndTight }
  { 0 }
\ExplSyntaxOff
```

2.2.3.15 Emphasis Renderers

The `\markdownRendererEmphasis` macro represents an emphasized span of text. The macro receives a single argument that corresponds to the emphasized span of text.

```
\def\markdownRendererEmphasis{%
  \markdownRendererEmphasisPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { emphasis }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { emphasis }
  { 1 }
\ExplSyntaxOff
```

The `\markdownRendererStrongEmphasis` macro represents a strongly emphasized span of text. The macro receives a single argument that corresponds to the emphasized span of text.

```
\def\markdownRendererStrongEmphasis{%
  \markdownRendererStrongEmphasisPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { strongEmphasis }
```

```
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { strongEmphasis }
  { 1 }
\ExplSyntaxOff
```

2.2.3.16 Block Quote Renderers

The `\markdownRendererBlockQuoteBegin` macro represents the beginning of a block quote. The macro receives no arguments.

```
\def\markdownRendererBlockQuoteBegin{%
  \markdownRendererBlockQuoteBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { blockQuoteBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { blockQuoteBegin }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererBlockQuoteEnd` macro represents the end of a block quote. The macro receives no arguments.

```
\def\markdownRendererBlockQuoteEnd{%
  \markdownRendererBlockQuoteEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { blockQuoteEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { blockQuoteEnd }
  { 0 }
\ExplSyntaxOff
```

2.2.3.17 Code Block Renderers

The `\markdownRendererInputVerbatim` macro represents a code block. The macro receives a single argument that corresponds to the filename of a file contaning the code block contents.

```
\def\markdownRendererInputVerbatim{%
  \markdownRendererInputVerbatimPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { inputVerbatim }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
```

```
  { inputVerbatim }
  { 1 }
\ExplSyntaxOff
```

The `\markdownRendererInputFencedCode` macro represents a fenced code block. This macro will only be produced, when the `fencedCode` option is enabled. The macro receives two arguments that correspond to the filename of a file contaning the code block contents and to the code fence infostring.

```
\def\markdownRendererInputFencedCode{%
  \markdownRendererInputFencedCodePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { inputFencedCode }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { inputFencedCode }
  { 2 }
\ExplSyntaxOff
```

2.2.3.18 YAML Metadata Renderers

The `\markdownRendererJekyllDataBegin` macro represents the beginning of a YAML document. This macro will only be produced when the `jekyllData` option is enabled. The macro receives no arguments.

```
\def\markdownRendererJekyllDataBegin{%
  \markdownRendererJekyllDataBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { jekyllDataBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { jekyllDataBegin }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererJekyllDataEnd` macro represents the end of a YAML document. This macro will only be produced when the `jekyllData` option is enabled. The macro receives no arguments.

```
\def\markdownRendererJekyllDataEnd{%
  \markdownRendererJekyllDataEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { jekyllDataEnd }
\prop_put:Nnn
```

```
  \g_@@_renderer_arities_prop
  { jekyllDataEnd }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererJekyllDataMappingBegin` macro represents the beginning of a mapping in a YAML document. This macro will only be produced when the `jekyllData` option is enabled. The macro receives two arguments: the scalar key in the parent structure, cast to a string following YAML serialization rules, and the number of items in the mapping.

```
\def\markdownRendererJekyllDataMappingBegin{%
  \markdownRendererJekyllDataMappingBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { jekyllDataMappingBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { jekyllDataMappingBegin }
  { 2 }
\ExplSyntaxOff
```

The `\markdownRendererJekyllDataMappingEnd` macro represents the end of a mapping in a YAML document. This macro will only be produced when the `jekyllData` option is enabled. The macro receives no arguments.

```
\def\markdownRendererJekyllDataMappingEnd{%
  \markdownRendererJekyllDataMappingEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { jekyllDataMappingEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { jekyllDataMappingEnd }
  { 0 }
\ExplSyntaxOff
```

The `\markdownRendererJekyllDataSequenceBegin` macro represents the beginning of a sequence in a YAML document. This macro will only be produced when the `jekyllData` option is enabled. The macro receives two arguments: the scalar key in the parent structure, cast to a string following YAML serialization rules, and the number of items in the sequence.

```
\def\markdownRendererJekyllDataSequenceBegin{%
  \markdownRendererJekyllDataSequenceBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
```

```
  \g_@@_renderers_seq
  { jekyllDataSequenceBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { jekyllDataSequenceBegin }
  { 2 }
\ExplSyntaxOff
```

The \markdownRendererJekyllDataSequenceEnd macro represents the end of a sequence in a YAML document. This macro will only be produced when the jekyllData option is enabled. The macro receives no arguments.

```
\def\markdownRendererJekyllDataSequenceEnd{%
  \markdownRendererJekyllDataSequenceEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { jekyllDataSequenceEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { jekyllDataSequenceEnd }
  { 0 }
\ExplSyntaxOff
```

The \markdownRendererJekyllDataBoolean macro represents a boolean scalar value in a YAML document. This macro will only be produced when the jekyllData option is enabled. The macro receives two arguments: the scalar key in the parent structure, and the scalar value, both cast to a string following YAML serialization rules.

```
\def\markdownRendererJekyllDataBoolean{%
  \markdownRendererJekyllDataBooleanPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { jekyllDataBoolean }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { jekyllDataBoolean }
  { 2 }
\ExplSyntaxOff
```

The \markdownRendererJekyllDataNumber macro represents a numeric scalar value in a YAML document. This macro will only be produced when the jekyllData option is enabled. The macro receives two arguments: the scalar key in the parent structure, and the scalar value, both cast to a string following YAML serialization rules.

```
\def\markdownRendererJekyllDataNumber{%
```

```
  \markdownRendererJekyllDataNumberPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { jekyllDataNumber }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { jekyllDataNumber }
  { 2 }
\ExplSyntaxOff
```

The `\markdownRendererJekyllDataString` macro represents a string scalar value in a YAML document. This macro will only be produced when the `jekyllData` option is enabled. The macro receives two arguments: the scalar key in the parent structure, cast to a string following YAML serialization rules, and the scalar value.

```
\def\markdownRendererJekyllDataString{%
  \markdownRendererJekyllDataStringPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { jekyllDataString }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { jekyllDataString }
  { 2 }
\ExplSyntaxOff
```

The `\markdownRendererJekyllDataEmpty` macro represents an empty scalar value in a YAML document. This macro will only be produced when the `jekyllData` option is enabled. The macro receives one argument: the scalar key in the parent structure, cast to a string following YAML serialization rules.

See also Section 2.2.4.1 for the description of the high-level expl3 interface that you can also use to react to YAML metadata.

```
\def\markdownRendererJekyllDataEmpty{%
  \markdownRendererJekyllDataEmptyPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { jekyllDataEmpty }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { jekyllDataEmpty }
  { 1 }
\ExplSyntaxOff
```

2.2.3.19 Heading Renderers

The `\markdownRendererHeadingOne` macro represents a first level heading. The macro receives a single argument that corresponds to the heading text.

```
\def\markdownRendererHeadingOne{%
  \markdownRendererHeadingOnePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { headingOne }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { headingOne }
  { 1 }
\ExplSyntaxOff
```

The `\markdownRendererHeadingTwo` macro represents a second level heading. The macro receives a single argument that corresponds to the heading text.

```
\def\markdownRendererHeadingTwo{%
  \markdownRendererHeadingTwoPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { headingTwo }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { headingTwo }
  { 1 }
\ExplSyntaxOff
```

The `\markdownRendererHeadingThree` macro represents a third level heading. The macro receives a single argument that corresponds to the heading text.

```
\def\markdownRendererHeadingThree{%
  \markdownRendererHeadingThreePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { headingThree }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { headingThree }
  { 1 }
\ExplSyntaxOff
```

The `\markdownRendererHeadingFour` macro represents a fourth level heading. The macro receives a single argument that corresponds to the heading text.

```
\def\markdownRendererHeadingFour{%
```

```
  \markdownRendererHeadingFourPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { headingFour }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { headingFour }
  { 1 }
\ExplSyntaxOff
```

The `\markdownRendererHeadingFive` macro represents a fifth level heading. The macro receives a single argument that corresponds to the heading text.

```
\def\markdownRendererHeadingFive{%
  \markdownRendererHeadingFivePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { headingFive }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { headingFive }
  { 1 }
\ExplSyntaxOff
```

The `\markdownRendererHeadingSix` macro represents a sixth level heading. The macro receives a single argument that corresponds to the heading text.

```
\def\markdownRendererHeadingSix{%
  \markdownRendererHeadingSixPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { headingSix }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { headingSix }
  { 1 }
\ExplSyntaxOff
```

2.2.3.20 Horizontal Rule Renderer

The `\markdownRendererHorizontalRule` macro represents a horizontal rule. The macro receives no arguments.

```
\def\markdownRendererHorizontalRule{%
  \markdownRendererHorizontalRulePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
```

```
  { horizontalRule }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { horizontalRule }
  { 0 }
\ExplSyntaxOff
```

2.2.3.21 Footnote Renderer

The `\markdownRendererFootnote` macro represents a footnote. This macro will only be produced, when the `footnotes` option is enabled. The macro receives a single argument that corresponds to the footnote text.

```
\def\markdownRendererFootnote{%
  \markdownRendererFootnotePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { footnote }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { footnote }
  { 1 }
\ExplSyntaxOff
```

2.2.3.22 Parenthesized Citations Renderer

The `\markdownRendererCite` macro represents a string of one or more parenthetical citations. This macro will only be produced, when the `citations` option is enabled. The macro receives the parameter `{`⟨*number of citations*⟩`}` followed by ⟨*suppress author*⟩ `{`⟨*prenote*⟩`}{`⟨*postnote*⟩`}{`⟨*name*⟩`}` repeated ⟨*number of citations*⟩ times. The ⟨*suppress author*⟩ parameter is either the token `-`, when the author's name is to be suppressed, or `+` otherwise.

```
\def\markdownRendererCite{%
  \markdownRendererCitePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { cite }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { cite }
  { 1 }
\ExplSyntaxOff
```

2.2.3.23 Text Citations Renderer

The `\markdownRendererTextCite` macro represents a string of one or more text citations. This macro will only be produced,

when the `citations` option is enabled. The macro receives parameters in the same format as the `\markdownRendererCite` macro.

```
\def\markdownRendererTextCite{%
  \markdownRendererTextCitePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { textCite }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { textCite }
  { 1 }
\ExplSyntaxOff
```

2.2.3.24 Table Renderer

The `\markdownRendererTable` macro represents a table. This macro will only be produced, when the `pipeTables` option is enabled. The macro receives the parameters `{`⟨*caption*⟩`}{`⟨*number of rows*⟩`}{`⟨*number of columns*⟩`}` followed by `{`⟨*alignments*⟩`}` and then by `{`⟨*row*⟩`}` repeated ⟨*number of rows*⟩ times, where ⟨*row*⟩ is `{`⟨*column*⟩`}` repeated ⟨*number of columns*⟩ times, ⟨*alignments*⟩ is ⟨*alignment*⟩ repeated ⟨*number of columns*⟩ times, and ⟨*alignment*⟩ is one of the following:

- `d` – The corresponding column has an unspecified (default) alignment.
- `l` – The corresponding column is left-aligned.
- `c` – The corresponding column is centered.
- `r` – The corresponding column is right-aligned.

```
\def\markdownRendererTable{%
  \markdownRendererTablePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { table }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { table }
  { 3 }
\ExplSyntaxOff
```

2.2.3.25 HTML Comment Renderers

The `\markdownRendererInlineHtmlComment` macro represents the contents of an inline HTML comment. This macro will only be produced, when the `html` option is enabled. The macro receives a single argument that corresponds to the contents of the HTML comment.

The \markdownRendererBlockHtmlCommentBegin and \markdownRendererBlockHtmlCommentEn macros represent the beginning and the end of a block HTML comment. The macros receive no arguments.

```
\def\markdownRendererInlineHtmlComment{%
  \markdownRendererInlineHtmlCommentPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { inlineHtmlComment }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { inlineHtmlComment }
  { 1 }
\ExplSyntaxOff
\def\markdownRendererBlockHtmlCommentBegin{%
  \markdownRendererBlockHtmlCommentBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { blockHtmlCommentBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { blockHtmlCommentBegin }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererBlockHtmlCommentEnd{%
  \markdownRendererBlockHtmlCommentEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { blockHtmlCommentEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { blockHtmlCommentEnd }
  { 0 }
\ExplSyntaxOff
```

2.2.3.26 HTML Tag and Element Renderers

The \markdownRendererInlineHtmlTag macro represents an opening, closing, or empty inline HTML tag. This macro will only be produced, when the html option is enabled. The macro receives a single argument that corresponds to the contents of the HTML tag.

The \markdownRendererInputBlockHtmlElement macro represents a block HTML element. This macro will only be produced, when the html option is enabled. The macro receives a single argument that filename of a file containing the contents of the HTML element.

```
\def\markdownRendererInlineHtmlTag{%
  \markdownRendererInlineHtmlTagPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { inlineHtmlTag }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { inlineHtmlTag }
  { 1 }
\ExplSyntaxOff
\def\markdownRendererInputBlockHtmlElement{%
  \markdownRendererInputBlockHtmlElementPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { inputBlockHtmlElement }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { inputBlockHtmlElement }
  { 1 }
\ExplSyntaxOff
```

2.2.3.27 Attribute Renderers

The following macros are only produced, when the `headerAttributes` option is enabled.

`\markdownRendererAttributeIdentifier` represents the ⟨*identifier*⟩ of a markdown element (`id="`⟨*identifier*⟩`"` in HTML and `#`⟨*identifier*⟩ in Markdown's `headerAttributes` syntax extension). The macro receives a single attribute that corresponds to the ⟨*identifier*⟩.

`\markdownRendererAttributeClassName` represents the ⟨*class name*⟩ of a markdown element (`class="`⟨*class name*⟩ ...`"` in HTML and `.`⟨*class name*⟩ in Markdown's `headerAttributes` syntax extension). The macro receives a single attribute that corresponds to the ⟨*class name*⟩.

`\markdownRendererAttributeKeyValue` represents a HTML attribute in the form ⟨*key*⟩`=`⟨*value*⟩ that is neither an identifier nor a class name. The macro receives two attributes that correspond to the ⟨*key*⟩ and the ⟨*value*⟩, respectively.

```
\def\markdownRendererAttributeIdentifier{%
  \markdownRendererAttributeIdentifierPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { attributeIdentifier }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { attributeIdentifier }
```

```
  { 1 }
\ExplSyntaxOff
\def\markdownRendererAttributeClassName{%
  \markdownRendererAttributeClassNamePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { attributeClassName }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { attributeClassName }
  { 1 }
\ExplSyntaxOff
\def\markdownRendererAttributeKeyValue{%
  \markdownRendererAttributeKeyValuePrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { attributeKeyValue }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { attributeKeyValue }
  { 2 }
\ExplSyntaxOff
```

2.2.3.28 Header Attribute Context Renderers

The following macros are only produced, when the `headerAttributes` option is enabled.

The `\markdownRendererHeaderAttributeContextBegin` and `\markdownRendererHeaderAttril` macros represent the beginning and the end of a section in which the attributes of a heading apply. The macros receive no arguments.

```
\def\markdownRendererHeaderAttributeContextBegin{%
  \markdownRendererHeaderAttributeContextBeginPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
  { headerAttributeContextBegin }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { headerAttributeContextBegin }
  { 0 }
\ExplSyntaxOff
\def\markdownRendererHeaderAttributeContextEnd{%
  \markdownRendererHeaderAttributeContextEndPrototype}%
\ExplSyntaxOn
\seq_put_right:Nn
  \g_@@_renderers_seq
```

```
  { headerAttributeContextEnd }
\prop_put:Nnn
  \g_@@_renderer_arities_prop
  { headerAttributeContextEnd }
  { 0 }
\ExplSyntaxOff
```

2.2.4 Token Renderer Prototypes

2.2.4.1 YAML Metadata Renderer Prototypes

By default, the renderer prototypes for YAML metadata provide a high-level interface that can be programmed using the `markdown/jekyllData` key–values from the l3keys module of the LaTeX3 kernel.

```
\ExplSyntaxOn
\keys_define:nn
  { markdown/jekyllData }
  { }
\ExplSyntaxOff
```

The following TeX macros provide definitions for the token renderers (see Section 2.2.3) that have not been redefined by the user. These macros are intended to be redefined by macro package authors who wish to provide sensible default token renderers. They are also redefined by the LaTeX and ConTeXt implementations (see sections 3.3 and 3.4).

```
\def\markdownRendererAttributeIdentifierPrototype#1{}%
\def\markdownRendererAttributeClassNamePrototype#1{}%
\def\markdownRendererAttributeKeyValuePrototype#1#2{}%
\def\markdownRendererDocumentBeginPrototype{}%
\def\markdownRendererDocumentEndPrototype{}%
\def\markdownRendererInterblockSeparatorPrototype{}%
\def\markdownRendererLineBreakPrototype{}%
\def\markdownRendererEllipsisPrototype{}%
\def\markdownRendererHeaderAttributeContextBeginPrototype{}%
\def\markdownRendererHeaderAttributeContextEndPrototype{}%
\def\markdownRendererNbspPrototype{}%
\def\markdownRendererLeftBracePrototype{}%
\def\markdownRendererRightBracePrototype{}%
\def\markdownRendererDollarSignPrototype{}%
\def\markdownRendererPercentSignPrototype{}%
\def\markdownRendererAmpersandPrototype{}%
\def\markdownRendererUnderscorePrototype{}%
\def\markdownRendererHashPrototype{}%
\def\markdownRendererCircumflexPrototype{}%
\def\markdownRendererBackslashPrototype{}%
\def\markdownRendererTildePrototype{}%
\def\markdownRendererPipePrototype{}%
\def\markdownRendererCodeSpanPrototype#1{}%
```

```
\def\markdownRendererLinkPrototype#1#2#3#4{}%
\def\markdownRendererImagePrototype#1#2#3#4{}%
\def\markdownRendererContentBlockPrototype#1#2#3#4{}%
\def\markdownRendererContentBlockOnlineImagePrototype#1#2#3#4{}%
\def\markdownRendererContentBlockCodePrototype#1#2#3#4#5{}%
\def\markdownRendererUlBeginPrototype{}%
\def\markdownRendererUlBeginTightPrototype{}%
\def\markdownRendererUlItemPrototype{}%
\def\markdownRendererUlItemEndPrototype{}%
\def\markdownRendererUlEndPrototype{}%
\def\markdownRendererUlEndTightPrototype{}%
\def\markdownRendererOlBeginPrototype{}%
\def\markdownRendererOlBeginTightPrototype{}%
\def\markdownRendererOlItemPrototype{}%
\def\markdownRendererOlItemWithNumberPrototype#1{}%
\def\markdownRendererOlItemEndPrototype{}%
\def\markdownRendererOlEndPrototype{}%
\def\markdownRendererOlEndTightPrototype{}%
\def\markdownRendererDlBeginPrototype{}%
\def\markdownRendererDlBeginTightPrototype{}%
\def\markdownRendererDlItemPrototype#1{}%
\def\markdownRendererDlItemEndPrototype{}%
\def\markdownRendererDlDefinitionBeginPrototype{}%
\def\markdownRendererDlDefinitionEndPrototype{}%
\def\markdownRendererDlEndPrototype{}%
\def\markdownRendererDlEndTightPrototype{}%
\def\markdownRendererEmphasisPrototype#1{}%
\def\markdownRendererStrongEmphasisPrototype#1{}%
\def\markdownRendererBlockQuoteBeginPrototype{}%
\def\markdownRendererBlockQuoteEndPrototype{}%
\def\markdownRendererInputVerbatimPrototype#1{}%
\def\markdownRendererInputFencedCodePrototype#1#2{}%
\def\markdownRendererJekyllDataBeginPrototype{}%
\def\markdownRendererJekyllDataEndPrototype{}%
\def\markdownRendererHeadingOnePrototype#1{}%
\def\markdownRendererHeadingTwoPrototype#1{}%
\def\markdownRendererHeadingThreePrototype#1{}%
\def\markdownRendererHeadingFourPrototype#1{}%
\def\markdownRendererHeadingFivePrototype#1{}%
\def\markdownRendererHeadingSixPrototype#1{}%
\def\markdownRendererHorizontalRulePrototype{}%
\def\markdownRendererFootnotePrototype#1{}%
\def\markdownRendererCitePrototype#1{}%
\def\markdownRendererTextCitePrototype#1{}%
\def\markdownRendererTablePrototype#1#2#3{}%
\def\markdownRendererInlineHtmlCommentPrototype#1{}%
\let\markdownRendererBlockHtmlCommentBeginPrototype=\iffalse
```

```
\let\markdownRendererBlockHtmlCommentBegin=\iffalse
\let\markdownRendererBlockHtmlCommentEndPrototype=\fi
\let\markdownRendererBlockHtmlCommentEnd=\fi
\def\markdownRendererInlineHtmlTagPrototype#1{}%
\def\markdownRendererInputBlockHtmlElementPrototype#1{}%
\def\markdownRendererTickedBoxPrototype{}%
\def\markdownRendererHalfTickedBoxPrototype{}%
\def\markdownRendererUntickedBoxPrototype{}%
```

2.2.5 Logging Facilities

The `\markdownInfo`, `\markdownWarning`, and `\markdownError` macros perform logging for the Markdown package. Their first argument specifies the text of the info, warning, or error message. The `\markdownError` macro receives a second argument that provides a help text. You may redefine these macros to redirect and process the info, warning, and error messages.

2.2.6 Miscellanea

The `\markdownMakeOther` macro is used by the package, when a TEX engine that does not support direct Lua access is starting to buffer a text. The plain TEX implementation changes the category code of plain TEX special characters to other, but there may be other active characters that may break the output. This macro should temporarily change the category of these to *other*.

```
\let\markdownMakeOther\relax
```

The `\markdownReadAndConvert` macro implements the `\markdownBegin` macro. The first argument specifies the token sequence that will terminate the markdown input (`\markdownEnd` in the instance of the `\markdownBegin` macro) when the plain TEX special characters have had their category changed to *other*. The second argument specifies the token sequence that will actually be inserted into the document, when the ending token sequence has been found.

```
\let\markdownReadAndConvert\relax
\begingroup
```

Locally swap the category code of the backslash symbol (`\`) with the pipe symbol (`|`). This is required in order that all the special symbols in the first argument of the `markdownReadAndConvert` macro have the category code *other*.

```
  \catcode`\|=0\catcode`\\=12%
  |gdef|markdownBegin{%
    |markdownReadAndConvert{\markdownEnd}%
                           {|markdownEnd}}%
|endgroup
```

The macro is exposed in the interface, so that the user can create their own markdown environments. Due to the way the arguments are passed to Lua (see Section 3.2.7),

the first argument may not contain the string `]]` (regardless of the category code of the bracket symbol (`]`)).

The `\markdownMode` macro specifies how the plain TEX implementation interfaces with the Lua interface. The valid values and their meaning are as follows:

- `0` – Shell escape via the 18 output file stream
- `1` – Shell escape via the Lua `os.execute` method
- `2` – Direct Lua access

By defining the macro, the user can coerce the package to use a specific mode. If the user does not define the macro prior to loading the plain TEX implementation, the correct value will be automatically detected. The outcome of changing the value of `\markdownMode` after the implementation has been loaded is undefined.

```
\ifx\markdownMode\undefined
  \ifx\directlua\undefined
    \def\markdownMode{0}%
  \else
    \def\markdownMode{2}%
  \fi
\fi
```

The following macros are no longer a part of the plain TEX interface and are only defined for backwards compatibility:

```
\def\markdownLuaRegisterIBCallback#1{\relax}%
\def\markdownLuaUnregisterIBCallback#1{\relax}%
```

2.3 LaTeX Interface

The LaTeX interface provides LaTeX environments for the typesetting of markdown input from within LaTeX, facilities for setting Lua interface options (see Section 2.1.2) used during the conversion from markdown to plain TEX, and facilities for changing the way markdown tokens are rendered. The rest of the interface is inherited from the plain TEX interface (see Section 2.2).

The LaTeX interface is implemented by the `markdown.sty` file, which can be loaded from the LaTeX document preamble as follows:

```
\usepackage[⟨options⟩]{markdown}
```

where ⟨*options*⟩ are the LaTeX interface options (see Section 2.3.2). Note that ⟨*options*⟩ inside the `\usepackage` macro may not set the `markdownRenderers` (see Section 2.3.2.5) and `markdownRendererPrototypes` (see Section 2.3.2.6) keys. This limitation is due to the way LaTeX 2ε parses package options.

2.3.1 Typesetting Markdown

The interface exposes the `markdown` and `markdown*` LaTeX environments, and redefines the `\markdownInput` command.

The `markdown` and `markdown*` LaTeX environments are used to typeset markdown document fragments. The starred version of the `markdown` environment accepts LaTeX interface options (see Section 2.3.2) as its only argument. These options will only influence this markdown document fragment.

```
\newenvironment{markdown}\relax\relax
\newenvironment{markdown*}[1]\relax\relax
```

You may prepend your own code to the `\markdown` macro and append your own code to the `\endmarkdown` macro to produce special effects before and after the `markdown` LaTeX environment (and likewise for the starred version).

Note that the `markdown` and `markdown*` LaTeX environments are subject to the same limitations as the `\markdownBegin` and `\markdownEnd` macros exposed by the plain TeX interface.

The following example LaTeX code showcases the usage of the `markdown` and `markdown*` environments:

```
\documentclass{article}
\usepackage{markdown}
\begin{document}
% ...
\begin{markdown}
_Hello_ **world** ...
\end{markdown}
% ...
\end{document}
```

```
\documentclass{article}
\usepackage{markdown}
\begin{document}
% ...
\begin{markdown*}{smartEllipses}
_Hello_ **world** ...
\end{markdown*}
% ...
\end{document}
```

The `\markdownInput` macro accepts a single mandatory parameter containing the filename of a markdown document and expands to the result of the conversion of the input markdown document to plain TeX. Unlike the `\markdownInput` macro provided by the plain TeX interface, this macro also accepts LaTeX interface options (see Section 2.3.2) as its optional argument. These options will only influnce this markdown document.

The following example LaTeX code showcases the usage of the `\markdownInput` macro:

```
\documentclass{article}
\usepackage{markdown}
\begin{document}
\markdownInput[smartEllipses]{hello.md}
\end{document}
```

2.3.2 Options

The LaTeX options are represented by a comma-delimited list of ⟨*key*⟩=⟨*value*⟩ pairs. For boolean options, the =⟨*value*⟩ part is optional, and ⟨*key*⟩ will be interpreted as ⟨*key*⟩=`true` if the =⟨*value*⟩ part has been omitted.

Except for the `plain` option described in Section 2.3.2.1, and the LaTeX themes described in Section 2.3.2.2, and the LaTeX setup snippets described in Section 2.3.2.3, LaTeX options map directly to the options recognized by the plain TeX interface (see Section 2.2.2) and to the markdown token renderers and their prototypes recognized by the plain TeX interface (see Sections 2.2.3 and 2.2.4).

The LaTeX options may be specified when loading the LaTeX package, when using the `markdown*` LaTeX environment or the `\markdownInput` macro (see Section 2.3), or via the `\markdownSetup` macro. The `\markdownSetup` macro receives the options to set up as its only argument:

```
\newcommand\markdownSetup[1]{%
  \setkeys{markdownOptions}{#1}%
```

We may also store LaTeX options as *setup snippets* and invoke them later using the `\markdownSetupSnippet` macro. The `\markdownSetupSnippet` macro receives two arguments: the name of the setup snippet and the options to store:

```
\newcommand\markdownSetupSnippet[2]{%
  \markdownIfSnippetExists{#1}%
    {%
      \markdownWarning
        {Redefined setup snippet \markdownLaTeXThemeName#1}%
      \csname markdownLaTeXSetupSnippet%
        \markdownLaTeXThemeName#1\endcsname={#2}%
    }{%
      \newtoks\next
        \next={#2}%
      \expandafter\let\csname markdownLaTeXSetupSnippet%
        \markdownLaTeXThemeName#1\endcsname=\next
    }}%
```

To decide whether a setup snippet exists, we can use the `\markdownIfSnippetExists` macro:

```
\newcommand\markdownIfSnippetExists[3]{%
  \@ifundefined
    {markdownLaTeXSetupSnippet\markdownLaTeXThemeName#1}%
    {#3}{#2}}%
```

See Section 2.3.2.2 for information on interactions between setup snippets and LaTeX themes. See Section 2.3.2.3 for information about invoking the stored setup snippets.

2.3.2.1 No default token renderer prototypes

Default token renderer prototypes require LaTeX packages that may clash with other packages used in a document.

Additionally, if we redefine token renderers and renderer prototypes ourselves, the default definitions will bring no benefit to us. Using the `plain` package option, we can keep the default definitions from the plain TEX implementation (see Section 3.2.3) and prevent the soft LATEX prerequisites in Section 1.1.3 from being loaded:

```
\usepackage[plain]{markdown}
```

```
\newif\ifmarkdownLaTeXPlain
  \markdownLaTeXPlainfalse
\define@key{markdownOptions}{plain}[true]{%
  \ifmarkdownLaTeXLoaded
    \markdownWarning
      {The plain option must be specified when loading the package}%
  \else
    \markdownLaTeXPlaintrue
  \fi}
```

2.3.2.2 LATEX themes

User-contributed LATEX themes for the Markdown package provide a domain-specific interpretation of some Markdown tokens. Similarly to LATEX packages, themes allow the authors to achieve a specific look and other high-level goals without low-level programming.

The LATEX option with key `theme` loads a LATEX package (further referred to as *a theme*) named `markdowntheme`⟨*munged theme name*⟩`.sty`, where the *munged theme name* is the *theme name* after a substitution of all forward slashes (`/`) for an underscore (`_`), the theme name is a value that is *qualified* and contains no underscores, and a value is qualified if and only if it contains at least one forward slash. Themes are inspired by the Beamer LATEX package, which provides similar functionality with its `\usetheme` macro [6, Section 15.1].

Theme names must be qualified to minimize naming conflicts between different themes intended for a single LATEX document class or for a single LATEX package. The preferred format of a theme name is ⟨*theme author*⟩`/`⟨*target LATEX document class or package*⟩`/`⟨*private naming scheme*⟩, where the *private naming scheme* may contain additional forward slashes. For example, a theme by a user `witiko` for the MU theme of the Beamer document class may have the name `witiko/beamer/MU`.

Theme names are munged, because LATEX packages are identified only by their filenames, not by their pathnames. [7] Therefore, we can't store the qualified theme names directly using directories, but we must encode the individual segments of the qualified theme in the filename. For example, loading a theme named `witiko/beamer/MU` would load a LATEX package named `markdownthemewitiko_beamer_MU.sty`.

If the LATEX option with key `theme` is (repeatedly) specified in the `\usepackage` macro, the loading of the theme(s) will be postponed in first-in-first-out order until

after the Markdown LaTeX package has been loaded. Otherwise, the theme(s) will be loaded immediately. For example, there is a theme named `witiko/dot`, which typesets fenced code blocks with the `dot` infostring as images of directed graphs rendered by the Graphviz tools. The following code would first load the Markdown package, then the `markdownthemewitiko_beamer_MU.sty` LaTeX package, and finally the `markdownthemewitiko_dot.sty` LaTeX package:

```
\usepackage[
  theme = witiko/beamer/MU,
  theme = witiko/dot,
]{markdown}
```

```
\newif\ifmarkdownLaTeXLoaded
  \markdownLaTeXLoadedfalse
\AtEndOfPackage{\markdownLaTeXLoadedtrue}
\define@key{markdownOptions}{theme}{%
  \IfSubStr{#1}{/}{}{%
    \markdownError
      {Won't load theme with unqualified name #1}%
      {Theme names must contain at least one forward slash}}%
  \StrSubstitute{#1}{/}{_}[\markdownLaTeXThemePackageName]%
  \edef\markdownLaTeXThemePackageName{%
    markdowntheme\markdownLaTeXThemePackageName}%
  \expandafter\markdownLaTeXThemeLoad\expandafter{%
    \markdownLaTeXThemePackageName}{#1/}}%
```

The LaTeX themes have a useful synergy with the setup snippets (see Section 2.3.2): To make it less likely that different themes will define setup snippets with the same name, we will prepend ⟨*theme name*⟩`/` before the snippet name and use the result as the snippet name. For example, if the `witiko/dot` theme defines the `product` setup snippet, the setup snippet will be available under the name `witiko/dot/product`.Due to limitations of LaTeX, themes may not be loaded after the beginning of a LaTeX document.

```
\@onlypreamble\KV@markdownOptions@theme
```

Example themes provided with the Markdown package include:

witiko/dot A theme that typesets fenced code blocks with the `dot …` infostring as images of directed graphs rendered by the Graphviz tools. The right tail of the infostring is used as the image title.

```
\documentclass{article}
\usepackage[theme=witiko/dot]{markdown}
\setkeys{Gin}{
  width = \columnwidth,
```

```
  height = 0.65\paperheight,
  keepaspectratio}
\begin{document}
\begin{markdown}
``` dot Various formats of mathemathical formulae
digraph tree {
  margin = 0;
  rankdir = "LR";

  latex -> pmml;
  latex -> cmml;
  pmml -> slt;
  cmml -> opt;
  cmml -> prefix;
  cmml -> infix;
  pmml -> mterms [style=dashed];
  cmml -> mterms;

  latex [label = "LaTeX"];
  pmml [label = "Presentation MathML"];
  cmml [label = "Content MathML"];
  slt [label = "Symbol Layout Tree"];
  opt [label = "Operator Tree"];
  prefix [label = "Prefix"];
  infix [label = "Infix"];
  mterms [label = "M-Terms"];
}
```
\end{markdown}
\end{document}
```

Typesetting the above document produces the output shown in Figure 4.

The theme requires a Unix-like operating system with GNU Diffutils and Graphviz installed. The theme also requires shell access unless the `\markdownOptionFrozenCache` plain TEX option is enabled.

```
\ProvidesPackage{markdownthemewitiko_dot}[2021/03/09]%
```

witiko/graphicx/http A theme that adds support for downloading images whose URL has the http or https protocol.

```
\documentclass{article}
\usepackage[theme=witiko/graphicx/http]{markdown}
```

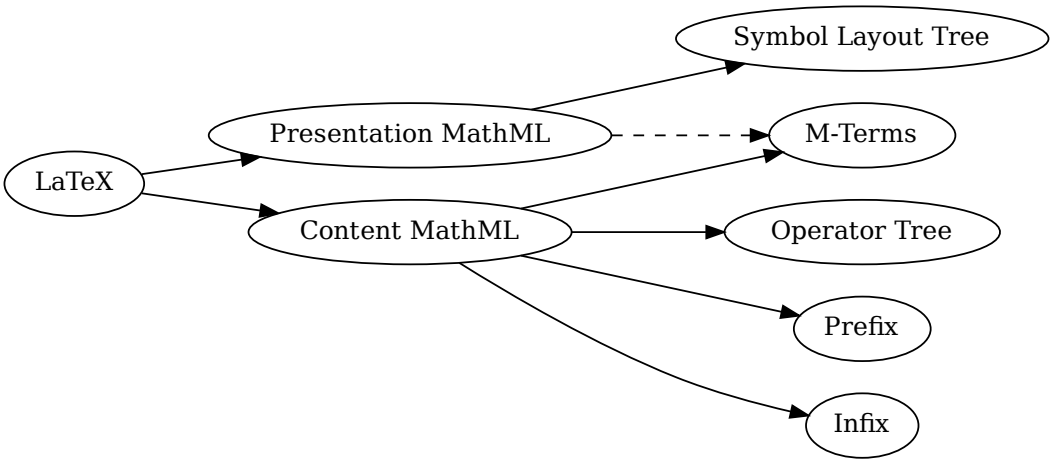

Figure 4: Various formats of mathemathical formulae

```
\begin{document}
\begin{markdown}
![img](https://github.com/witiko/markdown/raw/main/markdown.png
       "The banner of the Markdown package")
\end{markdown}
\end{document}
```

Typesetting the above document produces the output shown in Figure 5. The theme requires the catchfile LaTeX package and a Unix-like operating system with GNU Coreutils `md5sum` and either GNU Wget or cURL installed. The theme also requires shell access unless the `\markdownOptionFrozenCache` plain TeX option is enabled.

```
\ProvidesPackage{markdownthemewitiko_graphicx_http}[2021/03/22]%
```

witiko/tilde A theme that makes tilde (`~`) always typeset the non-breaking space even when the `hybrid` Lua option is `false`.

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

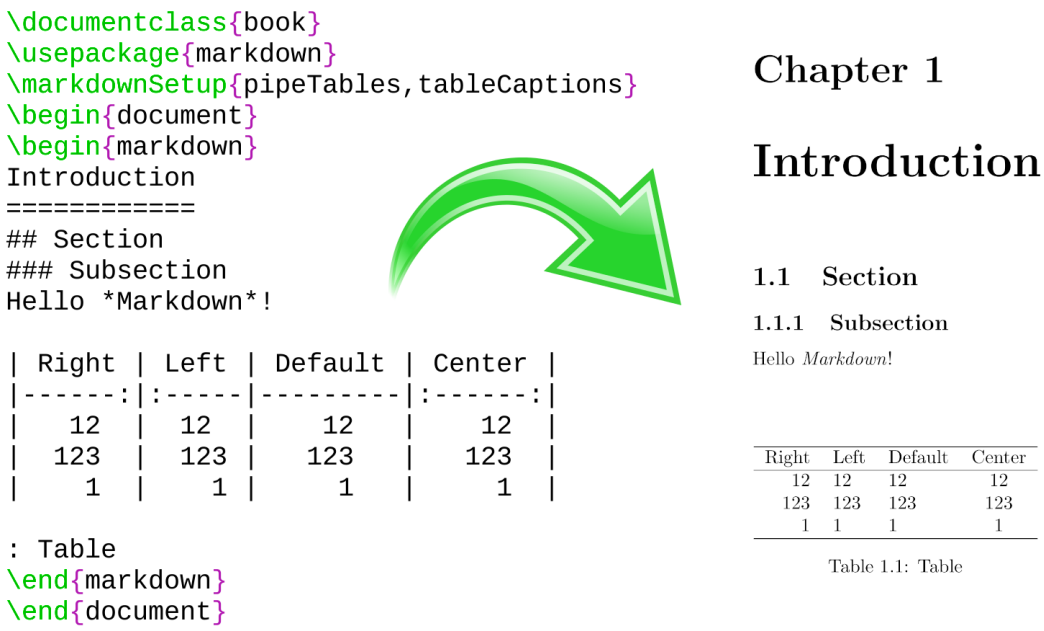

Figure 5: The banner of the Markdown package

Typesetting the above document produces the following text: “Bartel Leendert van der Waerden”.

```
\ProvidesPackage{markdownthemewitiko_tilde}[2021/03/22]%
```

Please, see Section 3.3.2.1 for implementation details of the example themes.

2.3.2.3 LaTeX setup snippets

The LaTeX option with key `snippet` invokes a snippet named ⟨*value*⟩:

```
\define@key{markdownOptions}{snippet}{%
  \markdownIfSnippetExists{#1}%
    {%
      \expandafter\markdownSetup\expandafter{%
        \the\csname markdownLaTeXSetupSnippet%
        \markdownLaTeXThemeName#1\endcsname}%
    }{%
      \markdownError
        {Can't invoke setup snippet #1}%
        {The setup snippet is undefined}%
    }%
  }%
```

Here is how we can use setup snippets to store options and invoke them later:

```
\markdownSetupSnippet{romanNumerals}{
  renderers = {
      olItemWithNumber = {\item[\romannumeral#1\relax.]},
  },
}
\begin{markdown}

The following ordered list will be preceded by arabic numerals:

1. wahid
2. aithnayn

\end{markdown}
\begin{markdown*}{snippet=romanNumerals}

The following ordered list will be preceded by roman numerals:

3. tres
4. quattuor

\end{markdown*}
```

2.3.2.4 Plain TEX Interface Options

The following options map directly to the option macros exposed by the plain TEX interface (see Section 2.2.2).

```
\define@key{markdownOptions}{helperScriptFileName}{%
  \def\markdownOptionHelperScriptFileName{#1}}%
\define@key{markdownOptions}{inputTempFileName}{%
  \def\markdownOptionInputTempFileName{#1}}%
\define@key{markdownOptions}{outputTempFileName}{%
  \def\markdownOptionOutputTempFileName{#1}}%
\define@key{markdownOptions}{errorTempFileName}{%
  \def\markdownOptionErrorTempFileName{#1}}%
\define@key{markdownOptions}{cacheDir}{%
  \def\markdownOptionCacheDir{#1}}%
\define@key{markdownOptions}{outputDir}{%
  \def\markdownOptionOutputDir{#1}}%
\define@key{markdownOptions}{blankBeforeBlockquote}[true]{%
  \def\markdownOptionBlankBeforeBlockquote{#1}}%
\define@key{markdownOptions}{blankBeforeCodeFence}[true]{%
  \def\markdownOptionBlankBeforeCodeFence{#1}}%
\define@key{markdownOptions}{blankBeforeHeading}[true]{%
  \def\markdownOptionBlankBeforeHeading{#1}}%
```

```
\define@key{markdownOptions}{breakableBlockquotes}[true]{%
  \def\markdownOptionBreakableBlockquotes{#1}}%
\define@key{markdownOptions}{citations}[true]{%
  \def\markdownOptionCitations{#1}}%
\define@key{markdownOptions}{citationNbsps}[true]{%
  \def\markdownOptionCitationNbsps{#1}}%
\define@key{markdownOptions}{contentBlocks}[true]{%
  \def\markdownOptionContentBlocks{#1}}%
\define@key{markdownOptions}{codeSpans}[true]{%
  \def\markdownOptionCodeSpans{#1}}%
\define@key{markdownOptions}{contentBlocksLanguageMap}{%
  \def\markdownOptionContentBlocksLanguageMap{#1}}%
\define@key{markdownOptions}{definitionLists}[true]{%
  \def\markdownOptionDefinitionLists{#1}}%
\define@key{markdownOptions}{eagerCache}[true]{%
  \def\markdownOptionEagerCache{#1}}%
\define@key{markdownOptions}{expectJekyllData}[true]{%
  \def\markdownOptionExpectJekyllData{#1}}%
\define@key{markdownOptions}{footnotes}[true]{%
  \def\markdownOptionFootnotes{#1}}%
\define@key{markdownOptions}{fencedCode}[true]{%
  \def\markdownOptionFencedCode{#1}}%
\define@key{markdownOptions}{jekyllData}[true]{%
  \def\markdownOptionJekyllData{#1}}%
\define@key{markdownOptions}{hardLineBreaks}[true]{%
  \def\markdownOptionHardLineBreaks{#1}}%
\define@key{markdownOptions}{hashEnumerators}[true]{%
  \def\markdownOptionHashEnumerators{#1}}%
\define@key{markdownOptions}{headerAttributes}[true]{%
  \def\markdownOptionHeaderAttributes{#1}}%
\define@key{markdownOptions}{html}[true]{%
  \def\markdownOptionHtml{#1}}%
\define@key{markdownOptions}{hybrid}[true]{%
  \def\markdownOptionHybrid{#1}}%
\define@key{markdownOptions}{inlineFootnotes}[true]{%
  \def\markdownOptionInlineFootnotes{#1}}%
\define@key{markdownOptions}{pipeTables}[true]{%
  \def\markdownOptionPipeTables{#1}}%
\define@key{markdownOptions}{preserveTabs}[true]{%
  \def\markdownOptionPreserveTabs{#1}}%
\define@key{markdownOptions}{relativeReferences}[true]{%
  \def\markdownOptionRelativeReferences{#1}}%
\define@key{markdownOptions}{smartEllipses}[true]{%
  \def\markdownOptionSmartEllipses{#1}}%
\define@key{markdownOptions}{shiftHeadings}{%
  \def\markdownOptionShiftHeadings{#1}}%
\define@key{markdownOptions}{slice}{%
```

```
  \def\markdownOptionSlice{#1}%
\define@key{markdownOptions}{startNumber}[true]{%
  \def\markdownOptionStartNumber{#1}%
\define@key{markdownOptions}{stripIndent}[true]{%
  \def\markdownOptionStripIndent{#1}%
\define@key{markdownOptions}{tableCaptions}[true]{%
  \def\markdownOptionTableCaptions{#1}%
\define@key{markdownOptions}{taskLists}[true]{%
  \def\markdownOptionTaskLists{#1}%
\define@key{markdownOptions}{texComments}[true]{%
  \def\markdownOptionTexComments{#1}%
\define@key{markdownOptions}{tightLists}[true]{%
  \def\markdownOptionTightLists{#1}%
\define@key{markdownOptions}{underscores}[true]{%
  \def\markdownOptionUnderscores{#1}%
\define@key{markdownOptions}{stripPercentSigns}[true]{%
  \def\markdownOptionStripPercentSigns{#1}%
```

The `\markdownOptionFinalizeCache` and `\markdownOptionFrozenCache` plain TEX options are exposed through LATEX options with keys `finalizeCache` and `frozenCache`.

To ensure compatibility with the `minted` package [8, Section 5.1], which supports the `finalizecache` and `frozencache` package options with similar semantics, the Markdown package also recognizes these as aliases and recognizes them as document class options. By passing `finalizecache` and `frozencache` as document class options, you may conveniently control the behavior of both packages at once:

```
\documentclass[frozencache]{article}
\usepackage{markdown,minted}
\begin{document}
\end{document}
```

We hope that other packages will support the `finalizecache` and `frozencache` package options in the future, so that they can become a standard interface for preparing LATEX document sources for distribution.

```
\define@key{markdownOptions}{finalizeCache}[true]{%
  \def\markdownOptionFinalizeCache{#1}%
\DeclareOption{finalizecache}{\markdownSetup{finalizeCache}}
\define@key{markdownOptions}{frozenCache}[true]{%
  \def\markdownOptionFrozenCache{#1}%
\DeclareOption{frozencache}{\markdownSetup{frozenCache}}
\define@key{markdownOptions}{frozenCacheFileName}{%
  \def\markdownOptionFrozenCacheFileName{#1}%
```

The following example LaTeX code showcases a possible configuration of plain TeX interface options `\markdownOptionHybrid`, `\markdownOptionSmartEllipses`, and `\markdownOptionCacheDir`.

```
\markdownSetup{
  hybrid,
  smartEllipses,
  cacheDir = /tmp,
}
```

2.3.2.5 Plain TeX Markdown Token Renderers

The LaTeX interface recognizes an option with the `renderers` key, whose value must be a list of options that map directly to the markdown token renderer macros exposed by the plain TeX interface (see Section 2.2.3).

```
\ExplSyntaxOn
\cs_new:Nn \@@_latex_define_renderers:
  {
    \seq_map_function:NN
      \g_@@_renderers_seq
      \@@_latex_define_renderer:n
  }
\cs_new:Nn \@@_latex_define_renderer:n
  {
    \tl_set:Nn
      \l_tmpb_tl
%     TODO: Replace with \str_uppercase:n in TeX Live 2020.
      { \str_upper_case:n { #1 } }
    \tl_set:Nx
      \l_tmpa_tl
      {
        markdownRenderer
        \tl_head:f { \l_tmpb_tl }
        \tl_tail:n { #1 }
      }
    \prop_get:NnN
      \g_@@_renderer_arities_prop
      { #1 }
      \l_tmpb_tl
    \@@_latex_define_renderer:ncV
      { #1 }
      { \l_tmpa_tl }
      \l_tmpb_tl
  }
\cs_new:Nn \@@_latex_define_renderer:nNn
  {
```

```
    \define@key
      { markdownRenderers }
      { #1 }
      {
        \cs_generate_from_arg_count:NNnn
          #2
          \cs_set:Npn
          { #3 }
          { ##1 }
      }
  }
\cs_generate_variant:Nn
  \@@_latex_define_renderer:nNn
  { ncV }
\ExplSyntaxOff
```

The following example LaTeX code showcases a possible configuration of the `\markdownRendererLink` and `\markdownRendererEmphasis` markdown token renderers.

```
\markdownSetup{
  renderers = {
    link = {#4},                    % Render links as the link title.
    emphasis = {\emph{#1}},     % Render emphasized text via `\emph`.
  }
}
```

2.3.2.6 Plain TeX Markdown Token Renderer Prototypes

The LaTeX interface recognizes an option with the `rendererPrototypes` key, whose value must be a list of options that map directly to the markdown token renderer prototype macros exposed by the plain TeX interface (see Section 2.2.4).

```
\ExplSyntaxOn
\cs_new:Nn \@@_latex_define_renderer_prototypes:
  {
    \seq_map_function:NN
      \g_@@_renderers_seq
      \@@_latex_define_renderer_prototype:n
  }
\cs_new:Nn \@@_latex_define_renderer_prototype:n
  {
    \tl_set:Nn
      \l_tmpb_tl
%     TODO: Replace with \str_uppercase:n in TeX Live 2020.
      { \str_upper_case:n { #1 } }
    \tl_set:Nx
```

```
      \l_tmpa_tl
      {
        markdownRenderer
        \tl_head:f { \l_tmpb_tl }
        \tl_tail:n { #1 }
        Prototype
      }
    \prop_get:NnN
      \g_@@_renderer_arities_prop
      { #1 }
      \l_tmpb_tl
    \@@_latex_define_renderer_prototype:ncV
      { #1 }
      { \l_tmpa_tl }
      \l_tmpb_tl
  }
\cs_new:Nn \@@_latex_define_renderer_prototype:nNn
  {
    \define@key
      { markdownRendererPrototypes }
      { #1 }
      {
        \cs_generate_from_arg_count:NNnn
          #2
          \cs_set:Npn
          { #3 }
          { ##1 }
      }
  }
\cs_generate_variant:Nn
  \@@_latex_define_renderer_prototype:nNn
  { ncV }
\ExplSyntaxOff
```

The following example LaTeX code showcases a possible configuration of the `\markdownRendererImagePrototype` and `\markdownRendererCodeSpanPrototype` markdown token renderer prototypes.

```
\markdownSetup{
  rendererPrototypes = {
    image = {\includegraphics{#2}},
    codeSpan = {\texttt{#1}},    % Render inline code via `\texttt`.
  }
}
```

2.4 ConTEXt Interface

The ConTEXt interface provides a start-stop macro pair for the typesetting of markdown input from within ConTEXt. The rest of the interface is inherited from the plain TEX interface (see Section 2.2).

```
\writestatus{loading}{ConTeXt User Module / markdown}%
\startmodule[markdown]
\unprotect
```

The ConTEXt interface is implemented by the `t-markdown.tex` ConTEXt module file that can be loaded as follows:

```
\usemodule[t][markdown]
```

It is expected that the special plain TEX characters have the expected category codes, when `\input`ting the file.

2.4.1 Typesetting Markdown

The interface exposes the `\startmarkdown` and `\stopmarkdown` macro pair for the typesetting of a markdown document fragment.

```
\let\startmarkdown\relax
\let\stopmarkdown\relax
```

You may prepend your own code to the `\startmarkdown` macro and redefine the `\stopmarkdown` macro to produce special effects before and after the markdown block.

Note that the `\startmarkdown` and `\stopmarkdown` macros are subject to the same limitations as the `\markdownBegin` and `\markdownEnd` macros exposed by the plain TEX interface.

The following example ConTEXt code showcases the usage of the `\startmarkdown` and `\stopmarkdown` macros:

```
\usemodule[t][markdown]
\starttext
\startmarkdown
_Hello_ **world** ...
\stopmarkdown
\stoptext
```

3 Implementation

This part of the documentation describes the implementation of the interfaces exposed by the package (see Section 2) and is aimed at the developers of the package, as well as the curious users.

Figure 1 shows the high-level structure of the Markdown package: The translation from markdown to TEX *token renderers* is performed by the Lua layer. The plain TEX layer provides default definitions for the token renderers. The LATEX and ConTEXt layers correct idiosyncrasies of the respective TEX formats, and provide format-specific default definitions for the token renderers.

3.1 Lua Implementation

The Lua implementation implements `writer` and `reader` objects that provide the conversion from markdown to plain TEX.

The Lunamark Lua module implements writers for the conversion to various other formats, such as DocBook, Groff, or HTML. These were stripped from the module and the remaining markdown reader and plain TEX writer were hidden behind the converter functions exposed by the Lua interface (see Section 2.1).

```
local upper, gsub, format, length =
  string.upper, string.gsub, string.format, string.len
local concat = table.concat
local P, R, S, V, C, Cg, Cb, Cmt, Cc, Ct, B, Cs, any =
  lpeg.P, lpeg.R, lpeg.S, lpeg.V, lpeg.C, lpeg.Cg, lpeg.Cb,
  lpeg.Cmt, lpeg.Cc, lpeg.Ct, lpeg.B, lpeg.Cs, lpeg.P(1)
```

3.1.1 Utility Functions

This section documents the utility functions used by the plain TEX writer and the markdown reader. These functions are encapsulated in the `util` object. The functions were originally located in the `lunamark/util.lua` file in the Lunamark Lua module.

```
local util = {}
```

The `util.err` method prints an error message `msg` and exits. If `exit_code` is provided, it specifies the exit code. Otherwise, the exit code will be 1.

```
function util.err(msg, exit_code)
  io.stderr:write("markdown.lua: " .. msg .. "\n")
  os.exit(exit_code or 1)
end
```

The `util.cache` method computes the digest of `string` and `salt`, adds the `suffix` and looks into the directory `dir`, whether a file with such a name exists. If it does not, it gets created with `transform(string)` as its content. The filename is then returned.

```
function util.cache(dir, string, salt, transform, suffix)
  local digest = md5.sumhexa(string .. (salt or ""))
  local name = util.pathname(dir, digest .. suffix)
  local file = io.open(name, "r")
  if file == nil then -- If no cache entry exists, then create a new one.
```

```
    local file = assert(io.open(name, "w"),
      [[could not open file "]] .. name .. [[" for writing]])
    local result = string
    if transform ~= nil then
      result = transform(result)
    end
    assert(file:write(result))
    assert(file:close())
  end
  return name
end
```

The `util.table_copy` method creates a shallow copy of a table `t` and its metatable.

```
function util.table_copy(t)
  local u = { }
  for k, v in pairs(t) do u[k] = v end
  return setmetatable(u, getmetatable(t))
end
```

The `util.expand_tabs_in_line` expands tabs in string `s`. If `tabstop` is specified, it is used as the tab stop width. Otherwise, the tab stop width of 4 characters is used. The method is a copy of the tab expansion algorithm from Ierusalimschy [9, Chapter 21].

```
function util.expand_tabs_in_line(s, tabstop)
  local tab = tabstop or 4
  local corr = 0
  return (s:gsub("()\t", function(p)
            local sp = tab - (p - 1 + corr) % tab
            corr = corr - 1 + sp
            return string.rep(" ", sp)
          end))
end
```

The `util.walk` method walks a rope `t`, applying a function `f` to each leaf element in order. A rope is an array whose elements may be ropes, strings, numbers, or functions. If a leaf element is a function, call it and get the return value before proceeding.

```
function util.walk(t, f)
  local typ = type(t)
  if typ == "string" then
    f(t)
  elseif typ == "table" then
    local i = 1
    local n
    n = t[i]
    while n do
      util.walk(n, f)
      i = i + 1
```

```
      n = t[i]
    end
  elseif typ == "function" then
    local ok, val = pcall(t)
    if ok then
      util.walk(val,f)
    end
  else
    f(tostring(t))
  end
end
```

The `util.flatten` method flattens an array `ary` that does not contain cycles and returns the result.

```
function util.flatten(ary)
  local new = {}
  for _,v in ipairs(ary) do
    if type(v) == "table" then
      for _,w in ipairs(util.flatten(v)) do
        new[#new + 1] = w
      end
    else
      new[#new + 1] = v
    end
  end
  return new
end
```

The `util.rope_to_string` method converts a rope `rope` to a string and returns it. For the definition of a rope, see the definition of the `util.walk` method.

```
function util.rope_to_string(rope)
  local buffer = {}
  util.walk(rope, function(x) buffer[#buffer + 1] = x end)
  return table.concat(buffer)
end
```

The `util.rope_last` method retrieves the last item in a rope. For the definition of a rope, see the definition of the `util.walk` method.

```
function util.rope_last(rope)
  if #rope == 0 then
    return nil
  else
    local l = rope[#rope]
    if type(l) == "table" then
      return util.rope_last(l)
    else
      return l
    end
```

```
  end
end
```

Given an array `ary` and a string `x`, the `util.intersperse` method returns an array `new`, such that `ary[i] == new[2*(i-1)+1]` and `new[2*i] == x` for all $1 \leqslant$ `i` $\leqslant$ `#ary`.

```
function util.intersperse(ary, x)
  local new = {}
  local l = #ary
  for i,v in ipairs(ary) do
    local n = #new
    new[n + 1] = v
    if i ~= l then
      new[n + 2] = x
    end
  end
  return new
end
```

Given an array `ary` and a function `f`, the `util.map` method returns an array `new`, such that `new[i] == f(ary[i])` for all $1 \leqslant$ `i` $\leqslant$ `#ary`.

```
function util.map(ary, f)
  local new = {}
  for i,v in ipairs(ary) do
    new[i] = f(v)
  end
  return new
end
```

Given a table `char_escapes` mapping escapable characters to escaped strings and optionally a table `string_escapes` mapping escapable strings to escaped strings, the `util.escaper` method returns an escaper function that escapes all occurances of escapable strings and characters (in this order).

The method uses LPeg, which is faster than the Lua `string.gsub` built-in method.

```
function util.escaper(char_escapes, string_escapes)
```

Build a string of escapable characters.

```
  local char_escapes_list = ""
  for i,_ in pairs(char_escapes) do
    char_escapes_list = char_escapes_list .. i
  end
```

Create an LPeg capture `escapable` that produces the escaped string corresponding to the matched escapable character.

```
  local escapable = S(char_escapes_list) / char_escapes
```

If `string_escapes` is provided, turn `escapable` into the

$$\sum_{\texttt{(k,v)} \in \texttt{string\_escapes}} \texttt{P(k) / v} + \texttt{escapable}$$

capture that replaces any occurance of the string `k` with the string `v` for each $(k, v) \in$ `string_escapes`. Note that the pattern summation is not commutative and its operands are inspected in the summation order during the matching. As a corrolary, the strings always take precedence over the characters.

```
2001    if string_escapes then
2002      for k,v in pairs(string_escapes) do
2003        escapable = P(k) / v + escapable
2004      end
2005    end
```

Create an LPeg capture `escape_string` that captures anything `escapable` does and matches any other unmatched characters.

```
2006    local escape_string = Cs((escapable + any)^0)
```

Return a function that matches the input string `s` against the `escape_string` capture.

```
2007    return function(s)
2008      return lpeg.match(escape_string, s)
2009    end
2010  end
```

The `util.pathname` method produces a pathname out of a directory name `dir` and a filename `file` and returns it.

```
2011  function util.pathname(dir, file)
2012    if #dir == 0 then
2013      return file
2014    else
2015      return dir .. "/" .. file
2016    end
2017  end
```

3.1.2 HTML Entities

This section documents the HTML entities recognized by the markdown reader. These functions are encapsulated in the `entities` object. The functions were originally located in the `lunamark/entities.lua` file in the Lunamark Lua module.

```
2018  local entities = {}
2019
2020  local character_entities = {
2021    ["Tab"] = 9,
2022    ["NewLine"] = 10,
2023    ["excl"] = 33,
2024    ["quot"] = 34,
2025    ["QUOT"] = 34,
2026    ["num"] = 35,
2027    ["dollar"] = 36,
2028    ["percnt"] = 37,
```

```
2029  ["amp"] = 38,
2030  ["AMP"] = 38,
2031  ["apos"] = 39,
2032  ["lpar"] = 40,
2033  ["rpar"] = 41,
2034  ["ast"] = 42,
2035  ["midast"] = 42,
2036  ["plus"] = 43,
2037  ["comma"] = 44,
2038  ["period"] = 46,
2039  ["sol"] = 47,
2040  ["colon"] = 58,
2041  ["semi"] = 59,
2042  ["lt"] = 60,
2043  ["LT"] = 60,
2044  ["equals"] = 61,
2045  ["gt"] = 62,
2046  ["GT"] = 62,
2047  ["quest"] = 63,
2048  ["commat"] = 64,
2049  ["lsqb"] = 91,
2050  ["lbrack"] = 91,
2051  ["bsol"] = 92,
2052  ["rsqb"] = 93,
2053  ["rbrack"] = 93,
2054  ["Hat"] = 94,
2055  ["lowbar"] = 95,
2056  ["grave"] = 96,
2057  ["DiacriticalGrave"] = 96,
2058  ["lcub"] = 123,
2059  ["lbrace"] = 123,
2060  ["verbar"] = 124,
2061  ["vert"] = 124,
2062  ["VerticalLine"] = 124,
2063  ["rcub"] = 125,
2064  ["rbrace"] = 125,
2065  ["nbsp"] = 160,
2066  ["NonBreakingSpace"] = 160,
2067  ["iexcl"] = 161,
2068  ["cent"] = 162,
2069  ["pound"] = 163,
2070  ["curren"] = 164,
2071  ["yen"] = 165,
2072  ["brvbar"] = 166,
2073  ["sect"] = 167,
2074  ["Dot"] = 168,
2075  ["die"] = 168,
```

```
["DoubleDot"] = 168,
["uml"] = 168,
["copy"] = 169,
["COPY"] = 169,
["ordf"] = 170,
["laquo"] = 171,
["not"] = 172,
["shy"] = 173,
["reg"] = 174,
["circledR"] = 174,
["REG"] = 174,
["macr"] = 175,
["OverBar"] = 175,
["strns"] = 175,
["deg"] = 176,
["plusmn"] = 177,
["pm"] = 177,
["PlusMinus"] = 177,
["sup2"] = 178,
["sup3"] = 179,
["acute"] = 180,
["DiacriticalAcute"] = 180,
["micro"] = 181,
["para"] = 182,
["middot"] = 183,
["centerdot"] = 183,
["CenterDot"] = 183,
["cedil"] = 184,
["Cedilla"] = 184,
["sup1"] = 185,
["ordm"] = 186,
["raquo"] = 187,
["frac14"] = 188,
["frac12"] = 189,
["half"] = 189,
["frac34"] = 190,
["iquest"] = 191,
["Agrave"] = 192,
["Aacute"] = 193,
["Acirc"] = 194,
["Atilde"] = 195,
["Auml"] = 196,
["Aring"] = 197,
["AElig"] = 198,
["Ccedil"] = 199,
["Egrave"] = 200,
["Eacute"] = 201,
```

```
  ["Ecirc"] = 202,
  ["Euml"] = 203,
  ["Igrave"] = 204,
  ["Iacute"] = 205,
  ["Icirc"] = 206,
  ["Iuml"] = 207,
  ["ETH"] = 208,
  ["Ntilde"] = 209,
  ["Ograve"] = 210,
  ["Oacute"] = 211,
  ["Ocirc"] = 212,
  ["Otilde"] = 213,
  ["Ouml"] = 214,
  ["times"] = 215,
  ["Oslash"] = 216,
  ["Ugrave"] = 217,
  ["Uacute"] = 218,
  ["Ucirc"] = 219,
  ["Uuml"] = 220,
  ["Yacute"] = 221,
  ["THORN"] = 222,
  ["szlig"] = 223,
  ["agrave"] = 224,
  ["aacute"] = 225,
  ["acirc"] = 226,
  ["atilde"] = 227,
  ["auml"] = 228,
  ["aring"] = 229,
  ["aelig"] = 230,
  ["ccedil"] = 231,
  ["egrave"] = 232,
  ["eacute"] = 233,
  ["ecirc"] = 234,
  ["euml"] = 235,
  ["igrave"] = 236,
  ["iacute"] = 237,
  ["icirc"] = 238,
  ["iuml"] = 239,
  ["eth"] = 240,
  ["ntilde"] = 241,
  ["ograve"] = 242,
  ["oacute"] = 243,
  ["ocirc"] = 244,
  ["otilde"] = 245,
  ["ouml"] = 246,
  ["divide"] = 247,
  ["div"] = 247,
```

```
2170    ["oslash"] = 248,
2171    ["ugrave"] = 249,
2172    ["uacute"] = 250,
2173    ["ucirc"] = 251,
2174    ["uuml"] = 252,
2175    ["yacute"] = 253,
2176    ["thorn"] = 254,
2177    ["yuml"] = 255,
2178    ["Amacr"] = 256,
2179    ["amacr"] = 257,
2180    ["Abreve"] = 258,
2181    ["abreve"] = 259,
2182    ["Aogon"] = 260,
2183    ["aogon"] = 261,
2184    ["Cacute"] = 262,
2185    ["cacute"] = 263,
2186    ["Ccirc"] = 264,
2187    ["ccirc"] = 265,
2188    ["Cdot"] = 266,
2189    ["cdot"] = 267,
2190    ["Ccaron"] = 268,
2191    ["ccaron"] = 269,
2192    ["Dcaron"] = 270,
2193    ["dcaron"] = 271,
2194    ["Dstrok"] = 272,
2195    ["dstrok"] = 273,
2196    ["Emacr"] = 274,
2197    ["emacr"] = 275,
2198    ["Edot"] = 278,
2199    ["edot"] = 279,
2200    ["Eogon"] = 280,
2201    ["eogon"] = 281,
2202    ["Ecaron"] = 282,
2203    ["ecaron"] = 283,
2204    ["Gcirc"] = 284,
2205    ["gcirc"] = 285,
2206    ["Gbreve"] = 286,
2207    ["gbreve"] = 287,
2208    ["Gdot"] = 288,
2209    ["gdot"] = 289,
2210    ["Gcedil"] = 290,
2211    ["Hcirc"] = 292,
2212    ["hcirc"] = 293,
2213    ["Hstrok"] = 294,
2214    ["hstrok"] = 295,
2215    ["Itilde"] = 296,
2216    ["itilde"] = 297,
```

```
2217    ["Imacr"] = 298,
2218    ["imacr"] = 299,
2219    ["Iogon"] = 302,
2220    ["iogon"] = 303,
2221    ["Idot"] = 304,
2222    ["imath"] = 305,
2223    ["inodot"] = 305,
2224    ["IJlig"] = 306,
2225    ["ijlig"] = 307,
2226    ["Jcirc"] = 308,
2227    ["jcirc"] = 309,
2228    ["Kcedil"] = 310,
2229    ["kcedil"] = 311,
2230    ["kgreen"] = 312,
2231    ["Lacute"] = 313,
2232    ["lacute"] = 314,
2233    ["Lcedil"] = 315,
2234    ["lcedil"] = 316,
2235    ["Lcaron"] = 317,
2236    ["lcaron"] = 318,
2237    ["Lmidot"] = 319,
2238    ["lmidot"] = 320,
2239    ["Lstrok"] = 321,
2240    ["lstrok"] = 322,
2241    ["Nacute"] = 323,
2242    ["nacute"] = 324,
2243    ["Ncedil"] = 325,
2244    ["ncedil"] = 326,
2245    ["Ncaron"] = 327,
2246    ["ncaron"] = 328,
2247    ["napos"] = 329,
2248    ["ENG"] = 330,
2249    ["eng"] = 331,
2250    ["Omacr"] = 332,
2251    ["omacr"] = 333,
2252    ["Odblac"] = 336,
2253    ["odblac"] = 337,
2254    ["OElig"] = 338,
2255    ["oelig"] = 339,
2256    ["Racute"] = 340,
2257    ["racute"] = 341,
2258    ["Rcedil"] = 342,
2259    ["rcedil"] = 343,
2260    ["Rcaron"] = 344,
2261    ["rcaron"] = 345,
2262    ["Sacute"] = 346,
2263    ["sacute"] = 347,
```

```
  ["Scirc"] = 348,
  ["scirc"] = 349,
  ["Scedil"] = 350,
  ["scedil"] = 351,
  ["Scaron"] = 352,
  ["scaron"] = 353,
  ["Tcedil"] = 354,
  ["tcedil"] = 355,
  ["Tcaron"] = 356,
  ["tcaron"] = 357,
  ["Tstrok"] = 358,
  ["tstrok"] = 359,
  ["Utilde"] = 360,
  ["utilde"] = 361,
  ["Umacr"] = 362,
  ["umacr"] = 363,
  ["Ubreve"] = 364,
  ["ubreve"] = 365,
  ["Uring"] = 366,
  ["uring"] = 367,
  ["Udblac"] = 368,
  ["udblac"] = 369,
  ["Uogon"] = 370,
  ["uogon"] = 371,
  ["Wcirc"] = 372,
  ["wcirc"] = 373,
  ["Ycirc"] = 374,
  ["ycirc"] = 375,
  ["Yuml"] = 376,
  ["Zacute"] = 377,
  ["zacute"] = 378,
  ["Zdot"] = 379,
  ["zdot"] = 380,
  ["Zcaron"] = 381,
  ["zcaron"] = 382,
  ["fnof"] = 402,
  ["imped"] = 437,
  ["gacute"] = 501,
  ["jmath"] = 567,
  ["circ"] = 710,
  ["caron"] = 711,
  ["Hacek"] = 711,
  ["breve"] = 728,
  ["Breve"] = 728,
  ["dot"] = 729,
  ["DiacriticalDot"] = 729,
  ["ring"] = 730,
```

```
2311  ["ogon"] = 731,
2312  ["tilde"] = 732,
2313  ["DiacriticalTilde"] = 732,
2314  ["dblac"] = 733,
2315  ["DiacriticalDoubleAcute"] = 733,
2316  ["DownBreve"] = 785,
2317  ["UnderBar"] = 818,
2318  ["Alpha"] = 913,
2319  ["Beta"] = 914,
2320  ["Gamma"] = 915,
2321  ["Delta"] = 916,
2322  ["Epsilon"] = 917,
2323  ["Zeta"] = 918,
2324  ["Eta"] = 919,
2325  ["Theta"] = 920,
2326  ["Iota"] = 921,
2327  ["Kappa"] = 922,
2328  ["Lambda"] = 923,
2329  ["Mu"] = 924,
2330  ["Nu"] = 925,
2331  ["Xi"] = 926,
2332  ["Omicron"] = 927,
2333  ["Pi"] = 928,
2334  ["Rho"] = 929,
2335  ["Sigma"] = 931,
2336  ["Tau"] = 932,
2337  ["Upsilon"] = 933,
2338  ["Phi"] = 934,
2339  ["Chi"] = 935,
2340  ["Psi"] = 936,
2341  ["Omega"] = 937,
2342  ["alpha"] = 945,
2343  ["beta"] = 946,
2344  ["gamma"] = 947,
2345  ["delta"] = 948,
2346  ["epsiv"] = 949,
2347  ["varepsilon"] = 949,
2348  ["epsilon"] = 949,
2349  ["zeta"] = 950,
2350  ["eta"] = 951,
2351  ["theta"] = 952,
2352  ["iota"] = 953,
2353  ["kappa"] = 954,
2354  ["lambda"] = 955,
2355  ["mu"] = 956,
2356  ["nu"] = 957,
2357  ["xi"] = 958,
```

```
["omicron"] = 959,
["pi"] = 960,
["rho"] = 961,
["sigmav"] = 962,
["varsigma"] = 962,
["sigmaf"] = 962,
["sigma"] = 963,
["tau"] = 964,
["upsi"] = 965,
["upsilon"] = 965,
["phi"] = 966,
["phiv"] = 966,
["varphi"] = 966,
["chi"] = 967,
["psi"] = 968,
["omega"] = 969,
["thetav"] = 977,
["vartheta"] = 977,
["thetasym"] = 977,
["Upsi"] = 978,
["upsih"] = 978,
["straightphi"] = 981,
["piv"] = 982,
["varpi"] = 982,
["Gammad"] = 988,
["gammad"] = 989,
["digamma"] = 989,
["kappav"] = 1008,
["varkappa"] = 1008,
["rhov"] = 1009,
["varrho"] = 1009,
["epsi"] = 1013,
["straightepsilon"] = 1013,
["bepsi"] = 1014,
["backepsilon"] = 1014,
["IOcy"] = 1025,
["DJcy"] = 1026,
["GJcy"] = 1027,
["Jukcy"] = 1028,
["DScy"] = 1029,
["Iukcy"] = 1030,
["YIcy"] = 1031,
["Jsercy"] = 1032,
["LJcy"] = 1033,
["NJcy"] = 1034,
["TSHcy"] = 1035,
["KJcy"] = 1036,
```

```
  ["Ubrcy"] = 1038,
  ["DZcy"] = 1039,
  ["Acy"] = 1040,
  ["Bcy"] = 1041,
  ["Vcy"] = 1042,
  ["Gcy"] = 1043,
  ["Dcy"] = 1044,
  ["IEcy"] = 1045,
  ["ZHcy"] = 1046,
  ["Zcy"] = 1047,
  ["Icy"] = 1048,
  ["Jcy"] = 1049,
  ["Kcy"] = 1050,
  ["Lcy"] = 1051,
  ["Mcy"] = 1052,
  ["Ncy"] = 1053,
  ["Ocy"] = 1054,
  ["Pcy"] = 1055,
  ["Rcy"] = 1056,
  ["Scy"] = 1057,
  ["Tcy"] = 1058,
  ["Ucy"] = 1059,
  ["Fcy"] = 1060,
  ["KHcy"] = 1061,
  ["TScy"] = 1062,
  ["CHcy"] = 1063,
  ["SHcy"] = 1064,
  ["SHCHcy"] = 1065,
  ["HARDcy"] = 1066,
  ["Ycy"] = 1067,
  ["SOFTcy"] = 1068,
  ["Ecy"] = 1069,
  ["YUcy"] = 1070,
  ["YAcy"] = 1071,
  ["acy"] = 1072,
  ["bcy"] = 1073,
  ["vcy"] = 1074,
  ["gcy"] = 1075,
  ["dcy"] = 1076,
  ["iecy"] = 1077,
  ["zhcy"] = 1078,
  ["zcy"] = 1079,
  ["icy"] = 1080,
  ["jcy"] = 1081,
  ["kcy"] = 1082,
  ["lcy"] = 1083,
  ["mcy"] = 1084,
```

```
["ncy"] = 1085,
["ocy"] = 1086,
["pcy"] = 1087,
["rcy"] = 1088,
["scy"] = 1089,
["tcy"] = 1090,
["ucy"] = 1091,
["fcy"] = 1092,
["khcy"] = 1093,
["tscy"] = 1094,
["chcy"] = 1095,
["shcy"] = 1096,
["shchcy"] = 1097,
["hardcy"] = 1098,
["ycy"] = 1099,
["softcy"] = 1100,
["ecy"] = 1101,
["yucy"] = 1102,
["yacy"] = 1103,
["iocy"] = 1105,
["djcy"] = 1106,
["gjcy"] = 1107,
["jukcy"] = 1108,
["dscy"] = 1109,
["iukcy"] = 1110,
["yicy"] = 1111,
["jsercy"] = 1112,
["ljcy"] = 1113,
["njcy"] = 1114,
["tshcy"] = 1115,
["kjcy"] = 1116,
["ubrcy"] = 1118,
["dzcy"] = 1119,
["ensp"] = 8194,
["emsp"] = 8195,
["emsp13"] = 8196,
["emsp14"] = 8197,
["numsp"] = 8199,
["puncsp"] = 8200,
["thinsp"] = 8201,
["ThinSpace"] = 8201,
["hairsp"] = 8202,
["VeryThinSpace"] = 8202,
["ZeroWidthSpace"] = 8203,
["NegativeVeryThinSpace"] = 8203,
["NegativeThinSpace"] = 8203,
["NegativeMediumSpace"] = 8203,
```

```
2499  ["NegativeThickSpace"] = 8203,
2500  ["zwnj"] = 8204,
2501  ["zwj"] = 8205,
2502  ["lrm"] = 8206,
2503  ["rlm"] = 8207,
2504  ["hyphen"] = 8208,
2505  ["dash"] = 8208,
2506  ["ndash"] = 8211,
2507  ["mdash"] = 8212,
2508  ["horbar"] = 8213,
2509  ["Verbar"] = 8214,
2510  ["Vert"] = 8214,
2511  ["lsquo"] = 8216,
2512  ["OpenCurlyQuote"] = 8216,
2513  ["rsquo"] = 8217,
2514  ["rsquor"] = 8217,
2515  ["CloseCurlyQuote"] = 8217,
2516  ["lsquor"] = 8218,
2517  ["sbquo"] = 8218,
2518  ["ldquo"] = 8220,
2519  ["OpenCurlyDoubleQuote"] = 8220,
2520  ["rdquo"] = 8221,
2521  ["rdquor"] = 8221,
2522  ["CloseCurlyDoubleQuote"] = 8221,
2523  ["ldquor"] = 8222,
2524  ["bdquo"] = 8222,
2525  ["dagger"] = 8224,
2526  ["Dagger"] = 8225,
2527  ["ddagger"] = 8225,
2528  ["bull"] = 8226,
2529  ["bullet"] = 8226,
2530  ["nldr"] = 8229,
2531  ["hellip"] = 8230,
2532  ["mldr"] = 8230,
2533  ["permil"] = 8240,
2534  ["pertenk"] = 8241,
2535  ["prime"] = 8242,
2536  ["Prime"] = 8243,
2537  ["tprime"] = 8244,
2538  ["bprime"] = 8245,
2539  ["backprime"] = 8245,
2540  ["lsaquo"] = 8249,
2541  ["rsaquo"] = 8250,
2542  ["oline"] = 8254,
2543  ["caret"] = 8257,
2544  ["hybull"] = 8259,
2545  ["frasl"] = 8260,
```

```
["bsemi"] = 8271,
["qprime"] = 8279,
["MediumSpace"] = 8287,
["NoBreak"] = 8288,
["ApplyFunction"] = 8289,
["af"] = 8289,
["InvisibleTimes"] = 8290,
["it"] = 8290,
["InvisibleComma"] = 8291,
["ic"] = 8291,
["euro"] = 8364,
["tdot"] = 8411,
["TripleDot"] = 8411,
["DotDot"] = 8412,
["Copf"] = 8450,
["complexes"] = 8450,
["incare"] = 8453,
["gscr"] = 8458,
["hamilt"] = 8459,
["HilbertSpace"] = 8459,
["Hscr"] = 8459,
["Hfr"] = 8460,
["Poincareplane"] = 8460,
["quaternions"] = 8461,
["Hopf"] = 8461,
["planckh"] = 8462,
["planck"] = 8463,
["hbar"] = 8463,
["plankv"] = 8463,
["hslash"] = 8463,
["Iscr"] = 8464,
["imagline"] = 8464,
["image"] = 8465,
["Im"] = 8465,
["imagpart"] = 8465,
["Ifr"] = 8465,
["Lscr"] = 8466,
["lagran"] = 8466,
["Laplacetrf"] = 8466,
["ell"] = 8467,
["Nopf"] = 8469,
["naturals"] = 8469,
["numero"] = 8470,
["copysr"] = 8471,
["weierp"] = 8472,
["wp"] = 8472,
["Popf"] = 8473,
```

```
["primes"] = 8473,
["rationals"] = 8474,
["Qopf"] = 8474,
["Rscr"] = 8475,
["realine"] = 8475,
["real"] = 8476,
["Re"] = 8476,
["realpart"] = 8476,
["Rfr"] = 8476,
["reals"] = 8477,
["Ropf"] = 8477,
["rx"] = 8478,
["trade"] = 8482,
["TRADE"] = 8482,
["integers"] = 8484,
["Zopf"] = 8484,
["ohm"] = 8486,
["mho"] = 8487,
["Zfr"] = 8488,
["zeetrf"] = 8488,
["iiota"] = 8489,
["angst"] = 8491,
["bernou"] = 8492,
["Bernoullis"] = 8492,
["Bscr"] = 8492,
["Cfr"] = 8493,
["Cayleys"] = 8493,
["escr"] = 8495,
["Escr"] = 8496,
["expectation"] = 8496,
["Fscr"] = 8497,
["Fouriertrf"] = 8497,
["phmmat"] = 8499,
["Mellintrf"] = 8499,
["Mscr"] = 8499,
["order"] = 8500,
["orderof"] = 8500,
["oscr"] = 8500,
["alefsym"] = 8501,
["aleph"] = 8501,
["beth"] = 8502,
["gimel"] = 8503,
["daleth"] = 8504,
["CapitalDifferentialD"] = 8517,
["DD"] = 8517,
["DifferentialD"] = 8518,
["dd"] = 8518,
```

```
["ExponentialE"] = 8519,
["exponentiale"] = 8519,
["ee"] = 8519,
["ImaginaryI"] = 8520,
["ii"] = 8520,
["frac13"] = 8531,
["frac23"] = 8532,
["frac15"] = 8533,
["frac25"] = 8534,
["frac35"] = 8535,
["frac45"] = 8536,
["frac16"] = 8537,
["frac56"] = 8538,
["frac18"] = 8539,
["frac38"] = 8540,
["frac58"] = 8541,
["frac78"] = 8542,
["larr"] = 8592,
["leftarrow"] = 8592,
["LeftArrow"] = 8592,
["slarr"] = 8592,
["ShortLeftArrow"] = 8592,
["uarr"] = 8593,
["uparrow"] = 8593,
["UpArrow"] = 8593,
["ShortUpArrow"] = 8593,
["rarr"] = 8594,
["rightarrow"] = 8594,
["RightArrow"] = 8594,
["srarr"] = 8594,
["ShortRightArrow"] = 8594,
["darr"] = 8595,
["downarrow"] = 8595,
["DownArrow"] = 8595,
["ShortDownArrow"] = 8595,
["harr"] = 8596,
["leftrightarrow"] = 8596,
["LeftRightArrow"] = 8596,
["varr"] = 8597,
["updownarrow"] = 8597,
["UpDownArrow"] = 8597,
["nwarr"] = 8598,
["UpperLeftArrow"] = 8598,
["nwarrow"] = 8598,
["nearr"] = 8599,
["UpperRightArrow"] = 8599,
["nearrow"] = 8599,
```

```
["searr"] = 8600,
["searrow"] = 8600,
["LowerRightArrow"] = 8600,
["swarr"] = 8601,
["swarrow"] = 8601,
["LowerLeftArrow"] = 8601,
["nlarr"] = 8602,
["nleftarrow"] = 8602,
["nrarr"] = 8603,
["nrightarrow"] = 8603,
["rarrw"] = 8605,
["rightsquigarrow"] = 8605,
["Larr"] = 8606,
["twoheadleftarrow"] = 8606,
["Uarr"] = 8607,
["Rarr"] = 8608,
["twoheadrightarrow"] = 8608,
["Darr"] = 8609,
["larrtl"] = 8610,
["leftarrowtail"] = 8610,
["rarrtl"] = 8611,
["rightarrowtail"] = 8611,
["LeftTeeArrow"] = 8612,
["mapstoleft"] = 8612,
["UpTeeArrow"] = 8613,
["mapstoup"] = 8613,
["map"] = 8614,
["RightTeeArrow"] = 8614,
["mapsto"] = 8614,
["DownTeeArrow"] = 8615,
["mapstodown"] = 8615,
["larrhk"] = 8617,
["hookleftarrow"] = 8617,
["rarrhk"] = 8618,
["hookrightarrow"] = 8618,
["larrlp"] = 8619,
["looparrowleft"] = 8619,
["rarrlp"] = 8620,
["looparrowright"] = 8620,
["harrw"] = 8621,
["leftrightsquigarrow"] = 8621,
["nharr"] = 8622,
["nleftrightarrow"] = 8622,
["lsh"] = 8624,
["Lsh"] = 8624,
["rsh"] = 8625,
["Rsh"] = 8625,
```

```
2734  ["ldsh"] = 8626,
2735  ["rdsh"] = 8627,
2736  ["crarr"] = 8629,
2737  ["cularr"] = 8630,
2738  ["curvearrowleft"] = 8630,
2739  ["curarr"] = 8631,
2740  ["curvearrowright"] = 8631,
2741  ["olarr"] = 8634,
2742  ["circlearrowleft"] = 8634,
2743  ["orarr"] = 8635,
2744  ["circlearrowright"] = 8635,
2745  ["lharu"] = 8636,
2746  ["LeftVector"] = 8636,
2747  ["leftharpoonup"] = 8636,
2748  ["lhard"] = 8637,
2749  ["leftharpoondown"] = 8637,
2750  ["DownLeftVector"] = 8637,
2751  ["uharr"] = 8638,
2752  ["upharpoonright"] = 8638,
2753  ["RightUpVector"] = 8638,
2754  ["uharl"] = 8639,
2755  ["upharpoonleft"] = 8639,
2756  ["LeftUpVector"] = 8639,
2757  ["rharu"] = 8640,
2758  ["RightVector"] = 8640,
2759  ["rightharpoonup"] = 8640,
2760  ["rhard"] = 8641,
2761  ["rightharpoondown"] = 8641,
2762  ["DownRightVector"] = 8641,
2763  ["dharr"] = 8642,
2764  ["RightDownVector"] = 8642,
2765  ["downharpoonright"] = 8642,
2766  ["dharl"] = 8643,
2767  ["LeftDownVector"] = 8643,
2768  ["downharpoonleft"] = 8643,
2769  ["rlarr"] = 8644,
2770  ["rightleftarrows"] = 8644,
2771  ["RightArrowLeftArrow"] = 8644,
2772  ["udarr"] = 8645,
2773  ["UpArrowDownArrow"] = 8645,
2774  ["lrarr"] = 8646,
2775  ["leftrightarrows"] = 8646,
2776  ["LeftArrowRightArrow"] = 8646,
2777  ["llarr"] = 8647,
2778  ["leftleftarrows"] = 8647,
2779  ["uuarr"] = 8648,
2780  ["upuparrows"] = 8648,
```

```
2781    ["rrarr"] = 8649,
2782    ["rightrightarrows"] = 8649,
2783    ["ddarr"] = 8650,
2784    ["downdownarrows"] = 8650,
2785    ["lrhar"] = 8651,
2786    ["ReverseEquilibrium"] = 8651,
2787    ["leftrightharpoons"] = 8651,
2788    ["rlhar"] = 8652,
2789    ["rightleftharpoons"] = 8652,
2790    ["Equilibrium"] = 8652,
2791    ["nlArr"] = 8653,
2792    ["nLeftarrow"] = 8653,
2793    ["nhArr"] = 8654,
2794    ["nLeftrightarrow"] = 8654,
2795    ["nrArr"] = 8655,
2796    ["nRightarrow"] = 8655,
2797    ["lArr"] = 8656,
2798    ["Leftarrow"] = 8656,
2799    ["DoubleLeftArrow"] = 8656,
2800    ["uArr"] = 8657,
2801    ["Uparrow"] = 8657,
2802    ["DoubleUpArrow"] = 8657,
2803    ["rArr"] = 8658,
2804    ["Rightarrow"] = 8658,
2805    ["Implies"] = 8658,
2806    ["DoubleRightArrow"] = 8658,
2807    ["dArr"] = 8659,
2808    ["Downarrow"] = 8659,
2809    ["DoubleDownArrow"] = 8659,
2810    ["hArr"] = 8660,
2811    ["Leftrightarrow"] = 8660,
2812    ["DoubleLeftRightArrow"] = 8660,
2813    ["iff"] = 8660,
2814    ["vArr"] = 8661,
2815    ["Updownarrow"] = 8661,
2816    ["DoubleUpDownArrow"] = 8661,
2817    ["nwArr"] = 8662,
2818    ["neArr"] = 8663,
2819    ["seArr"] = 8664,
2820    ["swArr"] = 8665,
2821    ["lAarr"] = 8666,
2822    ["Lleftarrow"] = 8666,
2823    ["rAarr"] = 8667,
2824    ["Rrightarrow"] = 8667,
2825    ["zigrarr"] = 8669,
2826    ["larrb"] = 8676,
2827    ["LeftArrowBar"] = 8676,
```

```
["rarrb"] = 8677,
["RightArrowBar"] = 8677,
["duarr"] = 8693,
["DownArrowUpArrow"] = 8693,
["loarr"] = 8701,
["roarr"] = 8702,
["hoarr"] = 8703,
["forall"] = 8704,
["ForAll"] = 8704,
["comp"] = 8705,
["complement"] = 8705,
["part"] = 8706,
["PartialD"] = 8706,
["exist"] = 8707,
["Exists"] = 8707,
["nexist"] = 8708,
["NotExists"] = 8708,
["nexists"] = 8708,
["empty"] = 8709,
["emptyset"] = 8709,
["emptyv"] = 8709,
["varnothing"] = 8709,
["nabla"] = 8711,
["Del"] = 8711,
["isin"] = 8712,
["isinv"] = 8712,
["Element"] = 8712,
["in"] = 8712,
["notin"] = 8713,
["NotElement"] = 8713,
["notinva"] = 8713,
["niv"] = 8715,
["ReverseElement"] = 8715,
["ni"] = 8715,
["SuchThat"] = 8715,
["notni"] = 8716,
["notniva"] = 8716,
["NotReverseElement"] = 8716,
["prod"] = 8719,
["Product"] = 8719,
["coprod"] = 8720,
["Coproduct"] = 8720,
["sum"] = 8721,
["Sum"] = 8721,
["minus"] = 8722,
["mnplus"] = 8723,
["mp"] = 8723,
```

```
2875    ["MinusPlus"] = 8723,
2876    ["plusdo"] = 8724,
2877    ["dotplus"] = 8724,
2878    ["setmn"] = 8726,
2879    ["setminus"] = 8726,
2880    ["Backslash"] = 8726,
2881    ["ssetmn"] = 8726,
2882    ["smallsetminus"] = 8726,
2883    ["lowast"] = 8727,
2884    ["compfn"] = 8728,
2885    ["SmallCircle"] = 8728,
2886    ["radic"] = 8730,
2887    ["Sqrt"] = 8730,
2888    ["prop"] = 8733,
2889    ["propto"] = 8733,
2890    ["Proportional"] = 8733,
2891    ["vprop"] = 8733,
2892    ["varpropto"] = 8733,
2893    ["infin"] = 8734,
2894    ["angrt"] = 8735,
2895    ["ang"] = 8736,
2896    ["angle"] = 8736,
2897    ["angmsd"] = 8737,
2898    ["measuredangle"] = 8737,
2899    ["angsph"] = 8738,
2900    ["mid"] = 8739,
2901    ["VerticalBar"] = 8739,
2902    ["smid"] = 8739,
2903    ["shortmid"] = 8739,
2904    ["nmid"] = 8740,
2905    ["NotVerticalBar"] = 8740,
2906    ["nsmid"] = 8740,
2907    ["nshortmid"] = 8740,
2908    ["par"] = 8741,
2909    ["parallel"] = 8741,
2910    ["DoubleVerticalBar"] = 8741,
2911    ["spar"] = 8741,
2912    ["shortparallel"] = 8741,
2913    ["npar"] = 8742,
2914    ["nparallel"] = 8742,
2915    ["NotDoubleVerticalBar"] = 8742,
2916    ["nspar"] = 8742,
2917    ["nshortparallel"] = 8742,
2918    ["and"] = 8743,
2919    ["wedge"] = 8743,
2920    ["or"] = 8744,
2921    ["vee"] = 8744,
```

```
["cap"] = 8745,
["cup"] = 8746,
["int"] = 8747,
["Integral"] = 8747,
["Int"] = 8748,
["tint"] = 8749,
["iiint"] = 8749,
["conint"] = 8750,
["oint"] = 8750,
["ContourIntegral"] = 8750,
["Conint"] = 8751,
["DoubleContourIntegral"] = 8751,
["Cconint"] = 8752,
["cwint"] = 8753,
["cwconint"] = 8754,
["ClockwiseContourIntegral"] = 8754,
["awconint"] = 8755,
["CounterClockwiseContourIntegral"] = 8755,
["there4"] = 8756,
["therefore"] = 8756,
["Therefore"] = 8756,
["becaus"] = 8757,
["because"] = 8757,
["Because"] = 8757,
["ratio"] = 8758,
["Colon"] = 8759,
["Proportion"] = 8759,
["minusd"] = 8760,
["dotminus"] = 8760,
["mDDot"] = 8762,
["homtht"] = 8763,
["sim"] = 8764,
["Tilde"] = 8764,
["thksim"] = 8764,
["thicksim"] = 8764,
["bsim"] = 8765,
["backsim"] = 8765,
["ac"] = 8766,
["mstpos"] = 8766,
["acd"] = 8767,
["wreath"] = 8768,
["VerticalTilde"] = 8768,
["wr"] = 8768,
["nsim"] = 8769,
["NotTilde"] = 8769,
["esim"] = 8770,
["EqualTilde"] = 8770,
```

```
["eqsim"] = 8770,
["sime"] = 8771,
["TildeEqual"] = 8771,
["simeq"] = 8771,
["nsime"] = 8772,
["nsimeq"] = 8772,
["NotTildeEqual"] = 8772,
["cong"] = 8773,
["TildeFullEqual"] = 8773,
["simne"] = 8774,
["ncong"] = 8775,
["NotTildeFullEqual"] = 8775,
["asymp"] = 8776,
["ap"] = 8776,
["TildeTilde"] = 8776,
["approx"] = 8776,
["thkap"] = 8776,
["thickapprox"] = 8776,
["nap"] = 8777,
["NotTildeTilde"] = 8777,
["napprox"] = 8777,
["ape"] = 8778,
["approxeq"] = 8778,
["apid"] = 8779,
["bcong"] = 8780,
["backcong"] = 8780,
["asympeq"] = 8781,
["CupCap"] = 8781,
["bump"] = 8782,
["HumpDownHump"] = 8782,
["Bumpeq"] = 8782,
["bumpe"] = 8783,
["HumpEqual"] = 8783,
["bumpeq"] = 8783,
["esdot"] = 8784,
["DotEqual"] = 8784,
["doteq"] = 8784,
["eDot"] = 8785,
["doteqdot"] = 8785,
["efDot"] = 8786,
["fallingdotseq"] = 8786,
["erDot"] = 8787,
["risingdotseq"] = 8787,
["colone"] = 8788,
["coloneq"] = 8788,
["Assign"] = 8788,
["ecolon"] = 8789,
```

```
["eqcolon"] = 8789,
["ecir"] = 8790,
["eqcirc"] = 8790,
["cire"] = 8791,
["circeq"] = 8791,
["wedgeq"] = 8793,
["veeeq"] = 8794,
["trie"] = 8796,
["triangleq"] = 8796,
["equest"] = 8799,
["questeq"] = 8799,
["ne"] = 8800,
["NotEqual"] = 8800,
["equiv"] = 8801,
["Congruent"] = 8801,
["nequiv"] = 8802,
["NotCongruent"] = 8802,
["le"] = 8804,
["leq"] = 8804,
["ge"] = 8805,
["GreaterEqual"] = 8805,
["geq"] = 8805,
["lE"] = 8806,
["LessFullEqual"] = 8806,
["leqq"] = 8806,
["gE"] = 8807,
["GreaterFullEqual"] = 8807,
["geqq"] = 8807,
["lnE"] = 8808,
["lneqq"] = 8808,
["gnE"] = 8809,
["gneqq"] = 8809,
["Lt"] = 8810,
["NestedLessLess"] = 8810,
["ll"] = 8810,
["Gt"] = 8811,
["NestedGreaterGreater"] = 8811,
["gg"] = 8811,
["twixt"] = 8812,
["between"] = 8812,
["NotCupCap"] = 8813,
["nlt"] = 8814,
["NotLess"] = 8814,
["nless"] = 8814,
["ngt"] = 8815,
["NotGreater"] = 8815,
["ngtr"] = 8815,
```

```
["nle"] = 8816,
["NotLessEqual"] = 8816,
["nleq"] = 8816,
["nge"] = 8817,
["NotGreaterEqual"] = 8817,
["ngeq"] = 8817,
["lsim"] = 8818,
["LessTilde"] = 8818,
["lesssim"] = 8818,
["gsim"] = 8819,
["gtrsim"] = 8819,
["GreaterTilde"] = 8819,
["nlsim"] = 8820,
["NotLessTilde"] = 8820,
["ngsim"] = 8821,
["NotGreaterTilde"] = 8821,
["lg"] = 8822,
["lessgtr"] = 8822,
["LessGreater"] = 8822,
["gl"] = 8823,
["gtrless"] = 8823,
["GreaterLess"] = 8823,
["ntlg"] = 8824,
["NotLessGreater"] = 8824,
["ntgl"] = 8825,
["NotGreaterLess"] = 8825,
["pr"] = 8826,
["Precedes"] = 8826,
["prec"] = 8826,
["sc"] = 8827,
["Succeeds"] = 8827,
["succ"] = 8827,
["prcue"] = 8828,
["PrecedesSlantEqual"] = 8828,
["preccurlyeq"] = 8828,
["sccue"] = 8829,
["SucceedsSlantEqual"] = 8829,
["succcurlyeq"] = 8829,
["prsim"] = 8830,
["precsim"] = 8830,
["PrecedesTilde"] = 8830,
["scsim"] = 8831,
["succsim"] = 8831,
["SucceedsTilde"] = 8831,
["npr"] = 8832,
["nprec"] = 8832,
["NotPrecedes"] = 8832,
```

```
["nsc"] = 8833,
["nsucc"] = 8833,
["NotSucceeds"] = 8833,
["sub"] = 8834,
["subset"] = 8834,
["sup"] = 8835,
["supset"] = 8835,
["Superset"] = 8835,
["nsub"] = 8836,
["nsup"] = 8837,
["sube"] = 8838,
["SubsetEqual"] = 8838,
["subseteq"] = 8838,
["supe"] = 8839,
["supseteq"] = 8839,
["SupersetEqual"] = 8839,
["nsube"] = 8840,
["nsubseteq"] = 8840,
["NotSubsetEqual"] = 8840,
["nsupe"] = 8841,
["nsupseteq"] = 8841,
["NotSupersetEqual"] = 8841,
["subne"] = 8842,
["subsetneq"] = 8842,
["supne"] = 8843,
["supsetneq"] = 8843,
["cupdot"] = 8845,
["uplus"] = 8846,
["UnionPlus"] = 8846,
["sqsub"] = 8847,
["SquareSubset"] = 8847,
["sqsubset"] = 8847,
["sqsup"] = 8848,
["SquareSuperset"] = 8848,
["sqsupset"] = 8848,
["sqsube"] = 8849,
["SquareSubsetEqual"] = 8849,
["sqsubseteq"] = 8849,
["sqsupe"] = 8850,
["SquareSupersetEqual"] = 8850,
["sqsupseteq"] = 8850,
["sqcap"] = 8851,
["SquareIntersection"] = 8851,
["sqcup"] = 8852,
["SquareUnion"] = 8852,
["oplus"] = 8853,
["CirclePlus"] = 8853,
```

```
3157  ["ominus"] = 8854,
3158  ["CircleMinus"] = 8854,
3159  ["otimes"] = 8855,
3160  ["CircleTimes"] = 8855,
3161  ["osol"] = 8856,
3162  ["odot"] = 8857,
3163  ["CircleDot"] = 8857,
3164  ["ocir"] = 8858,
3165  ["circledcirc"] = 8858,
3166  ["oast"] = 8859,
3167  ["circledast"] = 8859,
3168  ["odash"] = 8861,
3169  ["circleddash"] = 8861,
3170  ["plusb"] = 8862,
3171  ["boxplus"] = 8862,
3172  ["minusb"] = 8863,
3173  ["boxminus"] = 8863,
3174  ["timesb"] = 8864,
3175  ["boxtimes"] = 8864,
3176  ["sdotb"] = 8865,
3177  ["dotsquare"] = 8865,
3178  ["vdash"] = 8866,
3179  ["RightTee"] = 8866,
3180  ["dashv"] = 8867,
3181  ["LeftTee"] = 8867,
3182  ["top"] = 8868,
3183  ["DownTee"] = 8868,
3184  ["bottom"] = 8869,
3185  ["bot"] = 8869,
3186  ["perp"] = 8869,
3187  ["UpTee"] = 8869,
3188  ["models"] = 8871,
3189  ["vDash"] = 8872,
3190  ["DoubleRightTee"] = 8872,
3191  ["Vdash"] = 8873,
3192  ["Vvdash"] = 8874,
3193  ["VDash"] = 8875,
3194  ["nvdash"] = 8876,
3195  ["nvDash"] = 8877,
3196  ["nVdash"] = 8878,
3197  ["nVDash"] = 8879,
3198  ["prurel"] = 8880,
3199  ["vltri"] = 8882,
3200  ["vartriangleleft"] = 8882,
3201  ["LeftTriangle"] = 8882,
3202  ["vrtri"] = 8883,
3203  ["vartriangleright"] = 8883,
```

```
["RightTriangle"] = 8883,
["ltrie"] = 8884,
["trianglelefteq"] = 8884,
["LeftTriangleEqual"] = 8884,
["rtrie"] = 8885,
["trianglerighteq"] = 8885,
["RightTriangleEqual"] = 8885,
["origof"] = 8886,
["imof"] = 8887,
["mumap"] = 8888,
["multimap"] = 8888,
["hercon"] = 8889,
["intcal"] = 8890,
["intercal"] = 8890,
["veebar"] = 8891,
["barvee"] = 8893,
["angrtvb"] = 8894,
["lrtri"] = 8895,
["xwedge"] = 8896,
["Wedge"] = 8896,
["bigwedge"] = 8896,
["xvee"] = 8897,
["Vee"] = 8897,
["bigvee"] = 8897,
["xcap"] = 8898,
["Intersection"] = 8898,
["bigcap"] = 8898,
["xcup"] = 8899,
["Union"] = 8899,
["bigcup"] = 8899,
["diam"] = 8900,
["diamond"] = 8900,
["Diamond"] = 8900,
["sdot"] = 8901,
["sstarf"] = 8902,
["Star"] = 8902,
["divonx"] = 8903,
["divideontimes"] = 8903,
["bowtie"] = 8904,
["ltimes"] = 8905,
["rtimes"] = 8906,
["lthree"] = 8907,
["leftthreetimes"] = 8907,
["rthree"] = 8908,
["rightthreetimes"] = 8908,
["bsime"] = 8909,
["backsimeq"] = 8909,
```

```
["cuvee"] = 8910,
["curlyvee"] = 8910,
["cuwed"] = 8911,
["curlywedge"] = 8911,
["Sub"] = 8912,
["Subset"] = 8912,
["Sup"] = 8913,
["Supset"] = 8913,
["Cap"] = 8914,
["Cup"] = 8915,
["fork"] = 8916,
["pitchfork"] = 8916,
["epar"] = 8917,
["ltdot"] = 8918,
["lessdot"] = 8918,
["gtdot"] = 8919,
["gtrdot"] = 8919,
["Ll"] = 8920,
["Gg"] = 8921,
["ggg"] = 8921,
["leg"] = 8922,
["LessEqualGreater"] = 8922,
["lesseqgtr"] = 8922,
["gel"] = 8923,
["gtreqless"] = 8923,
["GreaterEqualLess"] = 8923,
["cuepr"] = 8926,
["curlyeqprec"] = 8926,
["cuesc"] = 8927,
["curlyeqsucc"] = 8927,
["nprcue"] = 8928,
["NotPrecedesSlantEqual"] = 8928,
["nsccue"] = 8929,
["NotSucceedsSlantEqual"] = 8929,
["nsqsube"] = 8930,
["NotSquareSubsetEqual"] = 8930,
["nsqsupe"] = 8931,
["NotSquareSupersetEqual"] = 8931,
["lnsim"] = 8934,
["gnsim"] = 8935,
["prnsim"] = 8936,
["precnsim"] = 8936,
["scnsim"] = 8937,
["succnsim"] = 8937,
["nltri"] = 8938,
["ntriangleleft"] = 8938,
["NotLeftTriangle"] = 8938,
```

```
["nrtri"] = 8939,
["ntriangleright"] = 8939,
["NotRightTriangle"] = 8939,
["nltrie"] = 8940,
["ntrianglelefteq"] = 8940,
["NotLeftTriangleEqual"] = 8940,
["nrtrie"] = 8941,
["ntrianglerighteq"] = 8941,
["NotRightTriangleEqual"] = 8941,
["vellip"] = 8942,
["ctdot"] = 8943,
["utdot"] = 8944,
["dtdot"] = 8945,
["disin"] = 8946,
["isinsv"] = 8947,
["isins"] = 8948,
["isindot"] = 8949,
["notinvc"] = 8950,
["notinvb"] = 8951,
["isinE"] = 8953,
["nisd"] = 8954,
["xnis"] = 8955,
["nis"] = 8956,
["notnivc"] = 8957,
["notnivb"] = 8958,
["barwed"] = 8965,
["barwedge"] = 8965,
["Barwed"] = 8966,
["doublebarwedge"] = 8966,
["lceil"] = 8968,
["LeftCeiling"] = 8968,
["rceil"] = 8969,
["RightCeiling"] = 8969,
["lfloor"] = 8970,
["LeftFloor"] = 8970,
["rfloor"] = 8971,
["RightFloor"] = 8971,
["drcrop"] = 8972,
["dlcrop"] = 8973,
["urcrop"] = 8974,
["ulcrop"] = 8975,
["bnot"] = 8976,
["profline"] = 8978,
["profsurf"] = 8979,
["telrec"] = 8981,
["target"] = 8982,
["ulcorn"] = 8988,
```

```
["ulcorner"] = 8988,
["urcorn"] = 8989,
["urcorner"] = 8989,
["dlcorn"] = 8990,
["llcorner"] = 8990,
["drcorn"] = 8991,
["lrcorner"] = 8991,
["frown"] = 8994,
["sfrown"] = 8994,
["smile"] = 8995,
["ssmile"] = 8995,
["cylcty"] = 9005,
["profalar"] = 9006,
["topbot"] = 9014,
["ovbar"] = 9021,
["solbar"] = 9023,
["angzarr"] = 9084,
["lmoust"] = 9136,
["lmoustache"] = 9136,
["rmoust"] = 9137,
["rmoustache"] = 9137,
["tbrk"] = 9140,
["OverBracket"] = 9140,
["bbrk"] = 9141,
["UnderBracket"] = 9141,
["bbrktbrk"] = 9142,
["OverParenthesis"] = 9180,
["UnderParenthesis"] = 9181,
["OverBrace"] = 9182,
["UnderBrace"] = 9183,
["trpezium"] = 9186,
["elinters"] = 9191,
["blank"] = 9251,
["oS"] = 9416,
["circledS"] = 9416,
["boxh"] = 9472,
["HorizontalLine"] = 9472,
["boxv"] = 9474,
["boxdr"] = 9484,
["boxdl"] = 9488,
["boxur"] = 9492,
["boxul"] = 9496,
["boxvr"] = 9500,
["boxvl"] = 9508,
["boxhd"] = 9516,
["boxhu"] = 9524,
["boxvh"] = 9532,
```

```
  ["boxH"] = 9552,
  ["boxV"] = 9553,
  ["boxdR"] = 9554,
  ["boxDr"] = 9555,
  ["boxDR"] = 9556,
  ["boxdL"] = 9557,
  ["boxDl"] = 9558,
  ["boxDL"] = 9559,
  ["boxuR"] = 9560,
  ["boxUr"] = 9561,
  ["boxUR"] = 9562,
  ["boxuL"] = 9563,
  ["boxUl"] = 9564,
  ["boxUL"] = 9565,
  ["boxvR"] = 9566,
  ["boxVr"] = 9567,
  ["boxVR"] = 9568,
  ["boxvL"] = 9569,
  ["boxVl"] = 9570,
  ["boxVL"] = 9571,
  ["boxHd"] = 9572,
  ["boxhD"] = 9573,
  ["boxHD"] = 9574,
  ["boxHu"] = 9575,
  ["boxhU"] = 9576,
  ["boxHU"] = 9577,
  ["boxvH"] = 9578,
  ["boxVh"] = 9579,
  ["boxVH"] = 9580,
  ["uhblk"] = 9600,
  ["lhblk"] = 9604,
  ["block"] = 9608,
  ["blk14"] = 9617,
  ["blk12"] = 9618,
  ["blk34"] = 9619,
  ["squ"] = 9633,
  ["square"] = 9633,
  ["Square"] = 9633,
  ["squf"] = 9642,
  ["squarf"] = 9642,
  ["blacksquare"] = 9642,
  ["FilledVerySmallSquare"] = 9642,
  ["EmptyVerySmallSquare"] = 9643,
  ["rect"] = 9645,
  ["marker"] = 9646,
  ["fltns"] = 9649,
  ["xutri"] = 9651,
```

```
3439  ["bigtriangleup"] = 9651,
3440  ["utrif"] = 9652,
3441  ["blacktriangle"] = 9652,
3442  ["utri"] = 9653,
3443  ["triangle"] = 9653,
3444  ["rtrif"] = 9656,
3445  ["blacktriangleright"] = 9656,
3446  ["rtri"] = 9657,
3447  ["triangleright"] = 9657,
3448  ["xdtri"] = 9661,
3449  ["bigtriangledown"] = 9661,
3450  ["dtrif"] = 9662,
3451  ["blacktriangledown"] = 9662,
3452  ["dtri"] = 9663,
3453  ["triangledown"] = 9663,
3454  ["ltrif"] = 9666,
3455  ["blacktriangleleft"] = 9666,
3456  ["ltri"] = 9667,
3457  ["triangleleft"] = 9667,
3458  ["loz"] = 9674,
3459  ["lozenge"] = 9674,
3460  ["cir"] = 9675,
3461  ["tridot"] = 9708,
3462  ["xcirc"] = 9711,
3463  ["bigcirc"] = 9711,
3464  ["ultri"] = 9720,
3465  ["urtri"] = 9721,
3466  ["lltri"] = 9722,
3467  ["EmptySmallSquare"] = 9723,
3468  ["FilledSmallSquare"] = 9724,
3469  ["starf"] = 9733,
3470  ["bigstar"] = 9733,
3471  ["star"] = 9734,
3472  ["phone"] = 9742,
3473  ["female"] = 9792,
3474  ["male"] = 9794,
3475  ["spades"] = 9824,
3476  ["spadesuit"] = 9824,
3477  ["clubs"] = 9827,
3478  ["clubsuit"] = 9827,
3479  ["hearts"] = 9829,
3480  ["heartsuit"] = 9829,
3481  ["diams"] = 9830,
3482  ["diamondsuit"] = 9830,
3483  ["sung"] = 9834,
3484  ["flat"] = 9837,
3485  ["natur"] = 9838,
```

```
["natural"] = 9838,
["sharp"] = 9839,
["check"] = 10003,
["checkmark"] = 10003,
["cross"] = 10007,
["malt"] = 10016,
["maltese"] = 10016,
["sext"] = 10038,
["VerticalSeparator"] = 10072,
["lbbrk"] = 10098,
["rbbrk"] = 10099,
["lobrk"] = 10214,
["LeftDoubleBracket"] = 10214,
["robrk"] = 10215,
["RightDoubleBracket"] = 10215,
["lang"] = 10216,
["LeftAngleBracket"] = 10216,
["langle"] = 10216,
["rang"] = 10217,
["RightAngleBracket"] = 10217,
["rangle"] = 10217,
["Lang"] = 10218,
["Rang"] = 10219,
["loang"] = 10220,
["roang"] = 10221,
["xlarr"] = 10229,
["longleftarrow"] = 10229,
["LongLeftArrow"] = 10229,
["xrarr"] = 10230,
["longrightarrow"] = 10230,
["LongRightArrow"] = 10230,
["xharr"] = 10231,
["longleftrightarrow"] = 10231,
["LongLeftRightArrow"] = 10231,
["xlArr"] = 10232,
["Longleftarrow"] = 10232,
["DoubleLongLeftArrow"] = 10232,
["xrArr"] = 10233,
["Longrightarrow"] = 10233,
["DoubleLongRightArrow"] = 10233,
["xhArr"] = 10234,
["Longleftrightarrow"] = 10234,
["DoubleLongLeftRightArrow"] = 10234,
["xmap"] = 10236,
["longmapsto"] = 10236,
["dzigrarr"] = 10239,
["nvlArr"] = 10498,
```

```
  ["nvrArr"] = 10499,
  ["nvHarr"] = 10500,
  ["Map"] = 10501,
  ["lbarr"] = 10508,
  ["rbarr"] = 10509,
  ["bkarow"] = 10509,
  ["lBarr"] = 10510,
  ["rBarr"] = 10511,
  ["dbkarow"] = 10511,
  ["RBarr"] = 10512,
  ["drbkarow"] = 10512,
  ["DDotrahd"] = 10513,
  ["UpArrowBar"] = 10514,
  ["DownArrowBar"] = 10515,
  ["Rarrtl"] = 10518,
  ["latail"] = 10521,
  ["ratail"] = 10522,
  ["lAtail"] = 10523,
  ["rAtail"] = 10524,
  ["larrfs"] = 10525,
  ["rarrfs"] = 10526,
  ["larrbfs"] = 10527,
  ["rarrbfs"] = 10528,
  ["nwarhk"] = 10531,
  ["nearhk"] = 10532,
  ["searhk"] = 10533,
  ["hksearow"] = 10533,
  ["swarhk"] = 10534,
  ["hkswarow"] = 10534,
  ["nwnear"] = 10535,
  ["nesear"] = 10536,
  ["toea"] = 10536,
  ["seswar"] = 10537,
  ["tosa"] = 10537,
  ["swnwar"] = 10538,
  ["rarrc"] = 10547,
  ["cudarrr"] = 10549,
  ["ldca"] = 10550,
  ["rdca"] = 10551,
  ["cudarrl"] = 10552,
  ["larrpl"] = 10553,
  ["curarrm"] = 10556,
  ["cularrp"] = 10557,
  ["rarrpl"] = 10565,
  ["harrcir"] = 10568,
  ["Uarrocir"] = 10569,
  ["lurdshar"] = 10570,
```

```
["ldrushar"] = 10571,
["LeftRightVector"] = 10574,
["RightUpDownVector"] = 10575,
["DownLeftRightVector"] = 10576,
["LeftUpDownVector"] = 10577,
["LeftVectorBar"] = 10578,
["RightVectorBar"] = 10579,
["RightUpVectorBar"] = 10580,
["RightDownVectorBar"] = 10581,
["DownLeftVectorBar"] = 10582,
["DownRightVectorBar"] = 10583,
["LeftUpVectorBar"] = 10584,
["LeftDownVectorBar"] = 10585,
["LeftTeeVector"] = 10586,
["RightTeeVector"] = 10587,
["RightUpTeeVector"] = 10588,
["RightDownTeeVector"] = 10589,
["DownLeftTeeVector"] = 10590,
["DownRightTeeVector"] = 10591,
["LeftUpTeeVector"] = 10592,
["LeftDownTeeVector"] = 10593,
["lHar"] = 10594,
["uHar"] = 10595,
["rHar"] = 10596,
["dHar"] = 10597,
["luruhar"] = 10598,
["ldrdhar"] = 10599,
["ruluhar"] = 10600,
["rdldhar"] = 10601,
["lharul"] = 10602,
["llhard"] = 10603,
["rharul"] = 10604,
["lrhard"] = 10605,
["udhar"] = 10606,
["UpEquilibrium"] = 10606,
["duhar"] = 10607,
["ReverseUpEquilibrium"] = 10607,
["RoundImplies"] = 10608,
["erarr"] = 10609,
["simrarr"] = 10610,
["larrsim"] = 10611,
["rarrsim"] = 10612,
["rarrap"] = 10613,
["ltlarr"] = 10614,
["gtrarr"] = 10616,
["subrarr"] = 10617,
["suplarr"] = 10619,
```

```
  ["lfisht"] = 10620,
  ["rfisht"] = 10621,
  ["ufisht"] = 10622,
  ["dfisht"] = 10623,
  ["lopar"] = 10629,
  ["ropar"] = 10630,
  ["lbrke"] = 10635,
  ["rbrke"] = 10636,
  ["lbrkslu"] = 10637,
  ["rbrksld"] = 10638,
  ["lbrksld"] = 10639,
  ["rbrkslu"] = 10640,
  ["langd"] = 10641,
  ["rangd"] = 10642,
  ["lparlt"] = 10643,
  ["rpargt"] = 10644,
  ["gtlPar"] = 10645,
  ["ltrPar"] = 10646,
  ["vzigzag"] = 10650,
  ["vangrt"] = 10652,
  ["angrtvbd"] = 10653,
  ["ange"] = 10660,
  ["range"] = 10661,
  ["dwangle"] = 10662,
  ["uwangle"] = 10663,
  ["angmsdaa"] = 10664,
  ["angmsdab"] = 10665,
  ["angmsdac"] = 10666,
  ["angmsdad"] = 10667,
  ["angmsdae"] = 10668,
  ["angmsdaf"] = 10669,
  ["angmsdag"] = 10670,
  ["angmsdah"] = 10671,
  ["bemptyv"] = 10672,
  ["demptyv"] = 10673,
  ["cemptyv"] = 10674,
  ["raemptyv"] = 10675,
  ["laemptyv"] = 10676,
  ["ohbar"] = 10677,
  ["omid"] = 10678,
  ["opar"] = 10679,
  ["operp"] = 10681,
  ["olcross"] = 10683,
  ["odsold"] = 10684,
  ["olcir"] = 10686,
  ["ofcir"] = 10687,
  ["olt"] = 10688,
```

```
["ogt"] = 10689,
["cirscir"] = 10690,
["cirE"] = 10691,
["solb"] = 10692,
["bsolb"] = 10693,
["boxbox"] = 10697,
["trisb"] = 10701,
["rtriltri"] = 10702,
["LeftTriangleBar"] = 10703,
["RightTriangleBar"] = 10704,
["race"] = 10714,
["iinfin"] = 10716,
["infintie"] = 10717,
["nvinfin"] = 10718,
["eparsl"] = 10723,
["smeparsl"] = 10724,
["eqvparsl"] = 10725,
["lozf"] = 10731,
["blacklozenge"] = 10731,
["RuleDelayed"] = 10740,
["dsol"] = 10742,
["xodot"] = 10752,
["bigodot"] = 10752,
["xoplus"] = 10753,
["bigoplus"] = 10753,
["xotime"] = 10754,
["bigotimes"] = 10754,
["xuplus"] = 10756,
["biguplus"] = 10756,
["xsqcup"] = 10758,
["bigsqcup"] = 10758,
["qint"] = 10764,
["iiiint"] = 10764,
["fpartint"] = 10765,
["cirfnint"] = 10768,
["awint"] = 10769,
["rppolint"] = 10770,
["scpolint"] = 10771,
["npolint"] = 10772,
["pointint"] = 10773,
["quatint"] = 10774,
["intlarhk"] = 10775,
["pluscir"] = 10786,
["plusacir"] = 10787,
["simplus"] = 10788,
["plusdu"] = 10789,
["plussim"] = 10790,
```

```
3721  ["plustwo"] = 10791,
3722  ["mcomma"] = 10793,
3723  ["minusdu"] = 10794,
3724  ["loplus"] = 10797,
3725  ["roplus"] = 10798,
3726  ["Cross"] = 10799,
3727  ["timesd"] = 10800,
3728  ["timesbar"] = 10801,
3729  ["smashp"] = 10803,
3730  ["lotimes"] = 10804,
3731  ["rotimes"] = 10805,
3732  ["otimesas"] = 10806,
3733  ["Otimes"] = 10807,
3734  ["odiv"] = 10808,
3735  ["triplus"] = 10809,
3736  ["triminus"] = 10810,
3737  ["tritime"] = 10811,
3738  ["iprod"] = 10812,
3739  ["intprod"] = 10812,
3740  ["amalg"] = 10815,
3741  ["capdot"] = 10816,
3742  ["ncup"] = 10818,
3743  ["ncap"] = 10819,
3744  ["capand"] = 10820,
3745  ["cupor"] = 10821,
3746  ["cupcap"] = 10822,
3747  ["capcup"] = 10823,
3748  ["cupbrcap"] = 10824,
3749  ["capbrcup"] = 10825,
3750  ["cupcup"] = 10826,
3751  ["capcap"] = 10827,
3752  ["ccups"] = 10828,
3753  ["ccaps"] = 10829,
3754  ["ccupssm"] = 10832,
3755  ["And"] = 10835,
3756  ["Or"] = 10836,
3757  ["andand"] = 10837,
3758  ["oror"] = 10838,
3759  ["orslope"] = 10839,
3760  ["andslope"] = 10840,
3761  ["andv"] = 10842,
3762  ["orv"] = 10843,
3763  ["andd"] = 10844,
3764  ["ord"] = 10845,
3765  ["wedbar"] = 10847,
3766  ["sdote"] = 10854,
3767  ["simdot"] = 10858,
```

```
["congdot"] = 10861,
["easter"] = 10862,
["apacir"] = 10863,
["apE"] = 10864,
["eplus"] = 10865,
["pluse"] = 10866,
["Esim"] = 10867,
["Colone"] = 10868,
["Equal"] = 10869,
["eDDot"] = 10871,
["ddotseq"] = 10871,
["equivDD"] = 10872,
["ltcir"] = 10873,
["gtcir"] = 10874,
["ltquest"] = 10875,
["gtquest"] = 10876,
["les"] = 10877,
["LessSlantEqual"] = 10877,
["leqslant"] = 10877,
["ges"] = 10878,
["GreaterSlantEqual"] = 10878,
["geqslant"] = 10878,
["lesdot"] = 10879,
["gesdot"] = 10880,
["lesdoto"] = 10881,
["gesdoto"] = 10882,
["lesdotor"] = 10883,
["gesdotol"] = 10884,
["lap"] = 10885,
["lessapprox"] = 10885,
["gap"] = 10886,
["gtrapprox"] = 10886,
["lne"] = 10887,
["lneq"] = 10887,
["gne"] = 10888,
["gneq"] = 10888,
["lnap"] = 10889,
["lnapprox"] = 10889,
["gnap"] = 10890,
["gnapprox"] = 10890,
["lEg"] = 10891,
["lesseqqgtr"] = 10891,
["gEl"] = 10892,
["gtreqqless"] = 10892,
["lsime"] = 10893,
["gsime"] = 10894,
["lsimg"] = 10895,
```

```
["gsiml"] = 10896,
["lgE"] = 10897,
["glE"] = 10898,
["lesges"] = 10899,
["gesles"] = 10900,
["els"] = 10901,
["eqslantless"] = 10901,
["egs"] = 10902,
["eqslantgtr"] = 10902,
["elsdot"] = 10903,
["egsdot"] = 10904,
["el"] = 10905,
["eg"] = 10906,
["siml"] = 10909,
["simg"] = 10910,
["simlE"] = 10911,
["simgE"] = 10912,
["LessLess"] = 10913,
["GreaterGreater"] = 10914,
["glj"] = 10916,
["gla"] = 10917,
["ltcc"] = 10918,
["gtcc"] = 10919,
["lescc"] = 10920,
["gescc"] = 10921,
["smt"] = 10922,
["lat"] = 10923,
["smte"] = 10924,
["late"] = 10925,
["bumpE"] = 10926,
["pre"] = 10927,
["preceq"] = 10927,
["PrecedesEqual"] = 10927,
["sce"] = 10928,
["succeq"] = 10928,
["SucceedsEqual"] = 10928,
["prE"] = 10931,
["scE"] = 10932,
["prnE"] = 10933,
["precneqq"] = 10933,
["scnE"] = 10934,
["succneqq"] = 10934,
["prap"] = 10935,
["precapprox"] = 10935,
["scap"] = 10936,
["succapprox"] = 10936,
["prnap"] = 10937,
```

```
["precnapprox"] = 10937,
["scnap"] = 10938,
["succnapprox"] = 10938,
["Pr"] = 10939,
["Sc"] = 10940,
["subdot"] = 10941,
["supdot"] = 10942,
["subplus"] = 10943,
["supplus"] = 10944,
["submult"] = 10945,
["supmult"] = 10946,
["subedot"] = 10947,
["supedot"] = 10948,
["subE"] = 10949,
["subseteqq"] = 10949,
["supE"] = 10950,
["supseteqq"] = 10950,
["subsim"] = 10951,
["supsim"] = 10952,
["subnE"] = 10955,
["subsetneqq"] = 10955,
["supnE"] = 10956,
["supsetneqq"] = 10956,
["csub"] = 10959,
["csup"] = 10960,
["csube"] = 10961,
["csupe"] = 10962,
["subsup"] = 10963,
["supsub"] = 10964,
["subsub"] = 10965,
["supsup"] = 10966,
["suphsub"] = 10967,
["supdsub"] = 10968,
["forkv"] = 10969,
["topfork"] = 10970,
["mlcp"] = 10971,
["Dashv"] = 10980,
["DoubleLeftTee"] = 10980,
["Vdashl"] = 10982,
["Barv"] = 10983,
["vBar"] = 10984,
["vBarv"] = 10985,
["Vbar"] = 10987,
["Not"] = 10988,
["bNot"] = 10989,
["rnmid"] = 10990,
["cirmid"] = 10991,
```

```
3909  ["midcir"] = 10992,
3910  ["topcir"] = 10993,
3911  ["nhpar"] = 10994,
3912  ["parsim"] = 10995,
3913  ["parsl"] = 11005,
3914  ["fflig"] = 64256,
3915  ["filig"] = 64257,
3916  ["fllig"] = 64258,
3917  ["ffilig"] = 64259,
3918  ["ffllig"] = 64260,
3919  ["Ascr"] = 119964,
3920  ["Cscr"] = 119966,
3921  ["Dscr"] = 119967,
3922  ["Gscr"] = 119970,
3923  ["Jscr"] = 119973,
3924  ["Kscr"] = 119974,
3925  ["Nscr"] = 119977,
3926  ["Oscr"] = 119978,
3927  ["Pscr"] = 119979,
3928  ["Qscr"] = 119980,
3929  ["Sscr"] = 119982,
3930  ["Tscr"] = 119983,
3931  ["Uscr"] = 119984,
3932  ["Vscr"] = 119985,
3933  ["Wscr"] = 119986,
3934  ["Xscr"] = 119987,
3935  ["Yscr"] = 119988,
3936  ["Zscr"] = 119989,
3937  ["ascr"] = 119990,
3938  ["bscr"] = 119991,
3939  ["cscr"] = 119992,
3940  ["dscr"] = 119993,
3941  ["fscr"] = 119995,
3942  ["hscr"] = 119997,
3943  ["iscr"] = 119998,
3944  ["jscr"] = 119999,
3945  ["kscr"] = 120000,
3946  ["lscr"] = 120001,
3947  ["mscr"] = 120002,
3948  ["nscr"] = 120003,
3949  ["pscr"] = 120005,
3950  ["qscr"] = 120006,
3951  ["rscr"] = 120007,
3952  ["sscr"] = 120008,
3953  ["tscr"] = 120009,
3954  ["uscr"] = 120010,
3955  ["vscr"] = 120011,
```

```
3956  ["wscr"] = 120012,
3957  ["xscr"] = 120013,
3958  ["yscr"] = 120014,
3959  ["zscr"] = 120015,
3960  ["Afr"] = 120068,
3961  ["Bfr"] = 120069,
3962  ["Dfr"] = 120071,
3963  ["Efr"] = 120072,
3964  ["Ffr"] = 120073,
3965  ["Gfr"] = 120074,
3966  ["Jfr"] = 120077,
3967  ["Kfr"] = 120078,
3968  ["Lfr"] = 120079,
3969  ["Mfr"] = 120080,
3970  ["Nfr"] = 120081,
3971  ["Ofr"] = 120082,
3972  ["Pfr"] = 120083,
3973  ["Qfr"] = 120084,
3974  ["Sfr"] = 120086,
3975  ["Tfr"] = 120087,
3976  ["Ufr"] = 120088,
3977  ["Vfr"] = 120089,
3978  ["Wfr"] = 120090,
3979  ["Xfr"] = 120091,
3980  ["Yfr"] = 120092,
3981  ["afr"] = 120094,
3982  ["bfr"] = 120095,
3983  ["cfr"] = 120096,
3984  ["dfr"] = 120097,
3985  ["efr"] = 120098,
3986  ["ffr"] = 120099,
3987  ["gfr"] = 120100,
3988  ["hfr"] = 120101,
3989  ["ifr"] = 120102,
3990  ["jfr"] = 120103,
3991  ["kfr"] = 120104,
3992  ["lfr"] = 120105,
3993  ["mfr"] = 120106,
3994  ["nfr"] = 120107,
3995  ["ofr"] = 120108,
3996  ["pfr"] = 120109,
3997  ["qfr"] = 120110,
3998  ["rfr"] = 120111,
3999  ["sfr"] = 120112,
4000  ["tfr"] = 120113,
4001  ["ufr"] = 120114,
4002  ["vfr"] = 120115,
```

```
4003    ["wfr"] = 120116,
4004    ["xfr"] = 120117,
4005    ["yfr"] = 120118,
4006    ["zfr"] = 120119,
4007    ["Aopf"] = 120120,
4008    ["Bopf"] = 120121,
4009    ["Dopf"] = 120123,
4010    ["Eopf"] = 120124,
4011    ["Fopf"] = 120125,
4012    ["Gopf"] = 120126,
4013    ["Iopf"] = 120128,
4014    ["Jopf"] = 120129,
4015    ["Kopf"] = 120130,
4016    ["Lopf"] = 120131,
4017    ["Mopf"] = 120132,
4018    ["Oopf"] = 120134,
4019    ["Sopf"] = 120138,
4020    ["Topf"] = 120139,
4021    ["Uopf"] = 120140,
4022    ["Vopf"] = 120141,
4023    ["Wopf"] = 120142,
4024    ["Xopf"] = 120143,
4025    ["Yopf"] = 120144,
4026    ["aopf"] = 120146,
4027    ["bopf"] = 120147,
4028    ["copf"] = 120148,
4029    ["dopf"] = 120149,
4030    ["eopf"] = 120150,
4031    ["fopf"] = 120151,
4032    ["gopf"] = 120152,
4033    ["hopf"] = 120153,
4034    ["iopf"] = 120154,
4035    ["jopf"] = 120155,
4036    ["kopf"] = 120156,
4037    ["lopf"] = 120157,
4038    ["mopf"] = 120158,
4039    ["nopf"] = 120159,
4040    ["oopf"] = 120160,
4041    ["popf"] = 120161,
4042    ["qopf"] = 120162,
4043    ["ropf"] = 120163,
4044    ["sopf"] = 120164,
4045    ["topf"] = 120165,
4046    ["uopf"] = 120166,
4047    ["vopf"] = 120167,
4048    ["wopf"] = 120168,
4049    ["xopf"] = 120169,
```

```
4050   ["yopf"] = 120170,
4051   ["zopf"] = 120171,
4052 }
```

Given a string `s` of decimal digits, the `entities.dec_entity` returns the corresponding UTF8-encoded Unicode codepoint.

```
4053 function entities.dec_entity(s)
4054   return unicode.utf8.char(tonumber(s))
4055 end
```

Given a string `s` of hexadecimal digits, the `entities.hex_entity` returns the corresponding UTF8-encoded Unicode codepoint.

```
4056 function entities.hex_entity(s)
4057   return unicode.utf8.char(tonumber("0x"..s))
4058 end
```

Given a character entity name `s` (like `ouml`), the `entities.char_entity` returns the corresponding UTF8-encoded Unicode codepoint.

```
4059 function entities.char_entity(s)
4060   local n = character_entities[s]
4061   if n == nil then
4062     return "&" .. s .. ";"
4063   end
4064   return unicode.utf8.char(n)
4065 end
```

3.1.3 Plain TEX Writer

This section documents the `writer` object, which implements the routines for producing the TEX output. The object is an amalgamate of the generic, TEX, LATEX writer objects that were located in the `lunamark/writer/generic.lua`, `lunamark/writer/tex.lua`, and `lunamark/writer/latex.lua` files in the Lunamark Lua module.

Although not specified in the Lua interface (see Section 2.1), the `writer` object is exported, so that the curious user could easily tinker with the methods of the objects produced by the `writer.new` method described below. The user should be aware, however, that the implementation may change in a future revision.

```
4066 M.writer = {}
```

The `writer.new` method creates and returns a new TEX writer object associated with the Lua interface options (see Section 2.1.2) `options`. When `options` are unspecified, it is assumed that an empty table was passed to the method.

The objects produced by the `writer.new` method expose instance methods and variables of their own. As a convention, I will refer to these ⟨*member*⟩s as `writer->`⟨*member*⟩. All member variables are immutable unless explicitly stated otherwise.

```
function M.writer.new(options)
  local self = {}
  options = options or {}
```

Make the `options` table inherit from the `defaultOptions` table.

```
  setmetatable(options, { __index = function (_, key)
    return defaultOptions[key] end })
```

Parse the `slice` option and define `writer->slice_begin` `writer->slice_end`, and `writer->is_writing`. The `writer->is_writing` member variable is mutable.

```
  local slice_specifiers = {}
  for specifier in options.slice:gmatch("[^%s]+") do
    table.insert(slice_specifiers, specifier)
  end

  if #slice_specifiers == 2 then
    self.slice_begin, self.slice_end = table.unpack(slice_specifiers)
    local slice_begin_type = self.slice_begin:sub(1, 1)
    if slice_begin_type ~= "^" and slice_begin_type ~= "$" then
      self.slice_begin = "^" .. self.slice_begin
    end
    local slice_end_type = self.slice_end:sub(1, 1)
    if slice_end_type ~= "^" and slice_end_type ~= "$" then
      self.slice_end = "$" .. self.slice_end
    end
  elseif #slice_specifiers == 1 then
    self.slice_begin = "^" .. slice_specifiers[1]
    self.slice_end = "$" .. slice_specifiers[1]
  end

  if self.slice_begin == "^" and self.slice_end ~= "^" then
    self.is_writing = true
  else
    self.is_writing = false
  end
```

Define `writer->suffix` as the suffix of the produced cache files.

```
  self.suffix = ".tex"
```

Define `writer->space` as the output format of a space character.

```
  self.space = " "
```

Define `writer->nbsp` as the output format of a non-breaking space character.

```
  self.nbsp = "\\markdownRendererNbsp{}"
```

Define `writer->plain` as a function that will transform an input plain text block `s` to the output format.

```
  function self.plain(s)
    return s
  end
```

Define `writer->paragraph` as a function that will transform an input paragraph `s` to the output format.

```
4103  function self.paragraph(s)
4104    if not self.is_writing then return "" end
4105    return s
4106  end
```

Define `writer->pack` as a function that will take the filename `name` of the output file prepared by the reader and transform it to the output format.

```
4107  function self.pack(name)
4108    return [[\input ]] .. name .. [[\relax]]
4109  end
```

Define `writer->interblocksep` as the output format of a block element separator.

```
4110  function self.interblocksep()
4111    if not self.is_writing then return "" end
4112    return "\\markdownRendererInterblockSeparator\n{}"
4113  end
```

Define `writer->linebreak` as the output format of a forced line break.

```
4114  self.linebreak = "\\markdownRendererLineBreak\n{}"
```

Define `writer->ellipsis` as the output format of an ellipsis.

```
4115  self.ellipsis = "\\markdownRendererEllipsis{}"
```

Define `writer->hrule` as the output format of a horizontal rule.

```
4116  function self.hrule()
4117    if not self.is_writing then return "" end
4118    return "\\markdownRendererHorizontalRule{}"
4119  end
```

Define tables `escaped_uri_chars`, `escaped_citation_chars`, and `escaped_minimal_strings` containing the mapping from special plain characters and character strings that always need to be escaped.

```
4120  local escaped_uri_chars = {
4121    ["{"] = "\\markdownRendererLeftBrace{}",
4122    ["}"] = "\\markdownRendererRightBrace{}",
4123    ["%"] = "\\markdownRendererPercentSign{}",
4124    ["\\"] = "\\markdownRendererBackslash{}",
4125  }
4126  local escaped_citation_chars = {
4127    ["{"] = "\\markdownRendererLeftBrace{}",
4128    ["}"] = "\\markdownRendererRightBrace{}",
4129    ["%"] = "\\markdownRendererPercentSign{}",
4130    ["\\"] = "\\markdownRendererBackslash{}",
4131    ["#"] = "\\markdownRendererHash{}",
4132  }
4133  local escaped_minimal_strings = {
4134    ["^^"] = "\\markdownRendererCircumflex\\markdownRendererCircumflex ",
```

```
4135      ["☒"] = "\\markdownRendererTickedBox{}",
4136      ["⊡"] = "\\markdownRendererHalfTickedBox{}",
4137      ["▫"] = "\\markdownRendererUntickedBox{}",
4138    }
```

Define a table `escaped_chars` containing the mapping from special plain TEX characters (including the active pipe character (`|`) of ConTEXt) that need to be escaped for typeset content.

```
4139  local escaped_chars = {
4140    ["{"] = "\\markdownRendererLeftBrace{}",
4141    ["}"] = "\\markdownRendererRightBrace{}",
4142    ["%"] = "\\markdownRendererPercentSign{}",
4143    ["\\"] = "\\markdownRendererBackslash{}",
4144    ["#"] = "\\markdownRendererHash{}",
4145    ["$"] = "\\markdownRendererDollarSign{}",
4146    ["&"] = "\\markdownRendererAmpersand{}",
4147    ["_"] = "\\markdownRendererUnderscore{}",
4148    ["^"] = "\\markdownRendererCircumflex{}",
4149    ["~"] = "\\markdownRendererTilde{}",
4150    ["|"] = "\\markdownRendererPipe{}",
4151  }
```

Use the `escaped_chars`, `escaped_uri_chars`, `escaped_citation_chars`, and `escaped_minimal_strings` tables to create the `escape`, `escape_citation`, `escape_uri`, and `escape_minimal` escaper functions.

```
4152  local escape = util.escaper(escaped_chars, escaped_minimal_strings)
4153  local escape_citation = util.escaper(escaped_citation_chars,
4154    escaped_minimal_strings)
4155  local escape_uri = util.escaper(escaped_uri_chars, escaped_minimal_strings)
4156  local escape_minimal = util.escaper({}, escaped_minimal_strings)
```

Define `writer->string` as a function that will transform an input plain text span `s` to the output format, `writer->citation` as a function that will transform an input citation name `c` to the output format, and `writer->uri` as a function that will transform an input URI `u` to the output format. If the `hybrid` option is enabled, use the `escape_minimal`. Otherwise, use the `escape`, `escape_citation`, and `escape_uri` functions.

```
4157  if options.hybrid then
4158    self.string = escape_minimal
4159    self.citation = escape_minimal
4160    self.uri = escape_minimal
4161  else
4162    self.string = escape
4163    self.citation = escape_citation
4164    self.uri = escape_uri
4165  end
```

Define `writer->escape` as a function that will transform an input plain text span to the output format. Unlike the `writer->string` function, `writer->escape` always uses the `escape` function, even when the `hybrid` option is enabled.

```
4166  self.escape = escape
```

Define `writer->code` as a function that will transform an input inlined code span `s` to the output format.

```
4167  function self.code(s)
4168    return {"\\markdownRendererCodeSpan{",self.escape(s),"}"}
4169  end
```

Define `writer->link` as a function that will transform an input hyperlink to the output format, where `lab` corresponds to the label, `src` to URI, and `tit` to the title of the link.

```
4170  function self.link(lab,src,tit)
4171    return {"\\markdownRendererLink{",lab,"}",
4172                           "{",self.escape(src),"}",
4173                           "{",self.uri(src),"}",
4174                           "{",self.string(tit or ""),"}"}
4175  end
```

Define `writer->table` as a function that will transform an input table to the output format, where `rows` is a sequence of columns and a column is a sequence of cell texts.

```
4176  function self.table(rows, caption)
4177    if not self.is_writing then return "" end
4178    local buffer = {"\\markdownRendererTable{",
4179      caption or "", "}{", #rows - 1, "}{", #rows[1], "}"}
4180    local temp = rows[2] -- put alignments on the first row
4181    rows[2] = rows[1]
4182    rows[1] = temp
4183    for i, row in ipairs(rows) do
4184      table.insert(buffer, "{")
4185      for _, column in ipairs(row) do
4186        if i > 1 then -- do not use braces for alignments
4187          table.insert(buffer, "{")
4188        end
4189        table.insert(buffer, column)
4190        if i > 1 then
4191          table.insert(buffer, "}")
4192        end
4193      end
4194      table.insert(buffer, "}")
4195    end
4196    return buffer
4197  end
```

Define `writer->image` as a function that will transform an input image to the output format, where `lab` corresponds to the label, `src` to the URL, and `tit` to the title of the image.

```
function self.image(lab,src,tit)
  return {"\\markdownRendererImage{",lab,"}",
                           "{",self.string(src),"}",
                           "{",self.uri(src),"}",
                           "{",self.string(tit or ""),"}"}
end
```

The `languages_json` table maps programming language filename extensions to fence infostrings. All `options.contentBlocksLanguageMap` files located by the KPathSea library are loaded into a chain of tables. `languages_json` corresponds to the first table and is chained with the rest via Lua metatables.

```
local languages_json = (function()
  local ran_ok, kpse = pcall(require, "kpse")
  if ran_ok then
    kpse.set_program_name("luatex")
```

If the KPathSea library is unavailable, perhaps because we are using LuaMetaTEX, we will only locate the `options.contentBlocksLanguageMap` in the current working directory:

```
  else
    kpse = {lookup=function(filename, options) return filename end}
  end
  local base, prev, curr
  for _, filename in ipairs{kpse.lookup(options.contentBlocksLanguageMap,
                                        { all=true })} do
    local file = io.open(filename, "r")
    if not file then goto continue end
    json = file:read("*all"):gsub('("[^\n]-"):','[%1]=')
    curr = (function()
      local _ENV={ json=json, load=load } -- run in sandbox
      return load("return "..json)()
    end)()
    if type(curr) == "table" then
      if base == nil then
        base = curr
      else
        setmetatable(prev, { __index = curr })
      end
      prev = curr
    end
    ::continue::
  end
  return base or {}
end)()
```

Define `writer->contentblock` as a function that will transform an input iA Writer content block to the output format, where `src` corresponds to the URI prefix, `suf` to the URI extension, `type` to the type of the content block (`localfile` or `onlineimage`), and `tit` to the title of the content block.

```
4233  function self.contentblock(src,suf,type,tit)
4234    if not self.is_writing then return "" end
4235    src = src.."."..suf
4236    suf = suf:lower()
4237    if type == "onlineimage" then
4238      return {"\\markdownRendererContentBlockOnlineImage{",suf,"}",
4239                               "{",self.string(src),"}",
4240                               "{",self.uri(src),"}",
4241                               "{",self.string(tit or ""),"}"}
4242    elseif languages_json[suf] then
4243      return {"\\markdownRendererContentBlockCode{",suf,"}",
4244                               "{",self.string(languages_json[suf]),"}",
4245                               "{",self.string(src),"}",
4246                               "{",self.uri(src),"}",
4247                               "{",self.string(tit or ""),"}"}
4248    else
4249      return {"\\markdownRendererContentBlock{",suf,"}",
4250                               "{",self.string(src),"}",
4251                               "{",self.uri(src),"}",
4252                               "{",self.string(tit or ""),"}"}
4253    end
4254  end
```

Define `writer->bulletlist` as a function that will transform an input bulleted list to the output format, where `items` is an array of the list items and `tight` specifies, whether the list is tight or not.

```
4255  local function ulitem(s)
4256    return {"\\markdownRendererUlItem ",s,
4257            "\\markdownRendererUlItemEnd "}
4258  end
4259
4260  function self.bulletlist(items,tight)
4261    if not self.is_writing then return "" end
4262    local buffer = {}
4263    for _,item in ipairs(items) do
4264      buffer[#buffer + 1] = ulitem(item)
4265    end
4266    local contents = util.intersperse(buffer,"\n")
4267    if tight and options.tightLists then
4268      return {"\\markdownRendererUlBeginTight\n",contents,
4269        "\n\\markdownRendererUlEndTight "}
4270    else
4271      return {"\\markdownRendererUlBegin\n",contents,
```

```
4272          "\n\\markdownRendererUlEnd "}
4273      end
4274    end
```

Define `writer->ollist` as a function that will transform an input ordered list to the output format, where `items` is an array of the list items and `tight` specifies, whether the list is tight or not. If the optional parameter `startnum` is present, it should be used as the number of the first list item.

```
4275    local function olitem(s,num)
4276      if num ~= nil then
4277        return {"\\markdownRendererOlItemWithNumber{",num,"}",s,
4278                "\\markdownRendererOlItemEnd "}
4279      else
4280        return {"\\markdownRendererOlItem ",s,
4281                "\\markdownRendererOlItemEnd "}
4282      end
4283    end
4284
4285    function self.orderedlist(items,tight,startnum)
4286      if not self.is_writing then return "" end
4287      local buffer = {}
4288      local num = startnum
4289      for _,item in ipairs(items) do
4290        buffer[#buffer + 1] = olitem(item,num)
4291        if num ~= nil then
4292          num = num + 1
4293        end
4294      end
4295      local contents = util.intersperse(buffer,"\n")
4296      if tight and options.tightLists then
4297        return {"\\markdownRendererOlBeginTight\n",contents,
4298                "\n\\markdownRendererOlEndTight "}
4299      else
4300        return {"\\markdownRendererOlBegin\n",contents,
4301                "\n\\markdownRendererOlEnd "}
4302      end
4303    end
```

Define `writer->inline_html_comment` as a function that will transform the contents of an inline HTML comment, to the output format, where `contents` are the contents of the HTML comment.

```
4304    function self.inline_html_comment(contents)
4305      return {"\\markdownRendererInlineHtmlComment{",contents,"}"}
4306    end
```

Define `writer->block_html_comment` as a function that will transform the contents of a block HTML comment, to the output format, where `contents` are the contents of the HTML comment.

```
4307  function self.block_html_comment(contents)
4308    if not self.is_writing then return "" end
4309    return {"\\markdownRendererBlockHtmlCommentBegin\n",contents,
4310            "\n\\markdownRendererBlockHtmlCommentEnd "}
4311  end
```

Define `writer->inline_html_tag` as a function that will transform the contents of an opening, closing, or empty inline HTML tag to the output format, where `contents` are the contents of the HTML tag.

```
4312  function self.inline_html_tag(contents)
4313    return {"\\markdownRendererInlineHtmlTag{",self.string(contents),"}"}
4314  end
```

Define `writer->block_html_element` as a function that will transform the contents of a block HTML element to the output format, where `s` are the contents of the HTML element.

```
4315  function self.block_html_element(s)
4316    if not self.is_writing then return "" end
4317    local name = util.cache(options.cacheDir, s, nil, nil, ".verbatim")
4318    return {"\\markdownRendererInputBlockHtmlElement{",name,"}"}
4319  end
```

Define `writer->definitionlist` as a function that will transform an input definition list to the output format, where `items` is an array of tables, each of the form `{ term = t, definitions = defs }`, where `t` is a term and `defs` is an array of definitions. `tight` specifies, whether the list is tight or not.

```
4320  local function dlitem(term, defs)
4321    local retVal = {"\\markdownRendererDlItem{",term,"}"}
4322    for _, def in ipairs(defs) do
4323      retVal[#retVal+1] = {"\\markdownRendererDlDefinitionBegin ",def,
4324                           "\\markdownRendererDlDefinitionEnd "}
4325    end
4326    retVal[#retVal+1] = "\\markdownRendererDlItemEnd "
4327    return retVal
4328  end
4329
4330  function self.definitionlist(items,tight)
4331    if not self.is_writing then return "" end
4332    local buffer = {}
4333    for _,item in ipairs(items) do
4334      buffer[#buffer + 1] = dlitem(item.term, item.definitions)
4335    end
4336    if tight and options.tightLists then
4337      return {"\\markdownRendererDlBeginTight\n", buffer,
4338        "\n\\markdownRendererDlEndTight"}
4339    else
4340      return {"\\markdownRendererDlBegin\n", buffer,
```

```
4341          "\n\\markdownRendererDlEnd"}
4342      end
4343    end
```

Define `writer->emphasis` as a function that will transform an emphasized span `s` of input text to the output format.

```
4344    function self.emphasis(s)
4345      return {"\\markdownRendererEmphasis{",s,"}"}
4346    end
```

Define `writer->tickbox` as a function that will transform a number `f` to the output format.

```
4347    function self.tickbox(f)
4348      if f == 1.0 then
4349        return "☒ "
4350      elseif f == 0.0 then
4351        return "☐ "
4352      else
4353        return "☐ "
4354      end
4355    end
```

Define `writer->strong` as a function that will transform a strongly emphasized span `s` of input text to the output format.

```
4356    function self.strong(s)
4357      return {"\\markdownRendererStrongEmphasis{",s,"}"}
4358    end
```

Define `writer->blockquote` as a function that will transform an input block quote `s` to the output format.

```
4359    function self.blockquote(s)
4360      if #util.rope_to_string(s) == 0 then return "" end
4361      return {"\\markdownRendererBlockQuoteBegin\n",s,
4362        "\n\\markdownRendererBlockQuoteEnd "}
4363    end
```

Define `writer->verbatim` as a function that will transform an input code block `s` to the output format.

```
4364    function self.verbatim(s)
4365      if not self.is_writing then return "" end
4366      s = string.gsub(s, '[\r\n%s]*$', '')
4367      local name = util.cache(options.cacheDir, s, nil, nil, ".verbatim")
4368      return {"\\markdownRendererInputVerbatim{",name,"}"}
4369    end
```

Define `writer->codeFence` as a function that will transform an input fenced code block `s` with the infostring `i` to the output format.

```
4370    function self.fencedCode(i, s)
4371      if not self.is_writing then return "" end
```

```
4372    s = string.gsub(s, '[\r\n%s]*$', '')
4373    local name = util.cache(options.cacheDir, s, nil, nil, ".verbatim")
4374    return {"\\markdownRendererInputFencedCode{",name,"}{",i,"}"}
4375  end
```

Define `writer->document` as a function that will transform a document `d` to the output format.

```
4376  function self.document(d)
4377    local active_attributes = self.active_attributes
4378    local buf = {"\\markdownRendererDocumentBegin\n", d}
4379
4380    -- pop attributes for sections that have ended
4381    if options.headerAttributes and self.is_writing then
4382      while #active_attributes > 0 do
4383        local attributes = active_attributes[#active_attributes]
4384        if #attributes > 0 then
4385          table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
4386        end
4387        table.remove(active_attributes, #active_attributes)
4388      end
4389    end
4390
4391    table.insert(buf, "\\markdownRendererDocumentEnd")
4392
4393    return buf
4394  end
```

Define `writer->jekyllData` as a function that will transform an input YAML table `d` to the output format. The table is the value for the key `p` in the parent table; if `p` is nil, then the table has no parent. All scalar keys and values encountered in the table will be cast to a string following YAML serialization rules. String values will also be transformed using the function `t`.

```
4395  function self.jekyllData(d, t, p)
4396    if not self.is_writing then return "" end
4397
4398    local buf = {}
4399
4400    local keys = {}
4401    for k, _ in pairs(d) do
4402      table.insert(keys, k)
4403    end
4404    table.sort(keys)
4405
4406    if not p then
4407      table.insert(buf, "\\markdownRendererJekyllDataBegin")
4408    end
4409
```

```
    if #d > 0 then
        table.insert(buf, "\\markdownRendererJekyllDataSequenceBegin{")
        table.insert(buf, self.uri(p or "null"))
        table.insert(buf, "}{")
        table.insert(buf, #keys)
        table.insert(buf, "}")
    else
        table.insert(buf, "\\markdownRendererJekyllDataMappingBegin{")
        table.insert(buf, self.uri(p or "null"))
        table.insert(buf, "}{")
        table.insert(buf, #keys)
        table.insert(buf, "}")
    end

    for _, k in ipairs(keys) do
      local v = d[k]
      local typ = type(v)
      k = tostring(k or "null")
      if typ == "table" and next(v) ~= nil then
        table.insert(
          buf,
          self.jekyllData(v, t, k)
        )
      else
        k = self.uri(k)
        v = tostring(v)
        if typ == "boolean" then
          table.insert(buf, "\\markdownRendererJekyllDataBoolean{")
          table.insert(buf, k)
          table.insert(buf, "}{")
          table.insert(buf, v)
          table.insert(buf, "}")
        elseif typ == "number" then
          table.insert(buf, "\\markdownRendererJekyllDataNumber{")
          table.insert(buf, k)
          table.insert(buf, "}{")
          table.insert(buf, v)
          table.insert(buf, "}")
        elseif typ == "string" then
          table.insert(buf, "\\markdownRendererJekyllDataString{")
          table.insert(buf, k)
          table.insert(buf, "}{")
          table.insert(buf, t(v))
          table.insert(buf, "}")
        elseif typ == "table" then
          table.insert(buf, "\\markdownRendererJekyllDataEmpty{")
          table.insert(buf, k)
```

```
          table.insert(buf, "}")
        else
          error(format("Unexpected type %s for value of " ..
                       "YAML key %s", typ, k))
        end
      end
    end

    if #d > 0 then
      table.insert(buf, "\\markdownRendererJekyllDataSequenceEnd")
    else
      table.insert(buf, "\\markdownRendererJekyllDataMappingEnd")
    end

    if not p then
      table.insert(buf, "\\markdownRendererJekyllDataEnd")
    end

    return buf
  end
```

Define `writer->active_attributes` as a stack of attributes of the headings that are currently active. The `writer->active_headings` member variable is mutable.

```
  self.active_attributes = {}
```

Define `writer->heading` as a function that will transform an input heading `s` at level `level` with identifiers `identifiers` to the output format.

```
  function self.heading(s, level, attributes)
    attributes = attributes or {}
    for i = 1, #attributes do
      attributes[attributes[i]] = true
    end

    local active_attributes = self.active_attributes
    local slice_begin_type = self.slice_begin:sub(1, 1)
    local slice_begin_identifier = self.slice_begin:sub(2) or ""
    local slice_end_type = self.slice_end:sub(1, 1)
    local slice_end_identifier = self.slice_end:sub(2) or ""

    local buf = {}

    -- push empty attributes for implied sections
    while #active_attributes < level-1 do
      table.insert(active_attributes, {})
    end

    -- pop attributes for sections that have ended
    while #active_attributes >= level do
```

```
      local active_identifiers = active_attributes[#active_attributes]
      -- tear down all active attributes at slice end
      if active_identifiers["#" .. slice_end_identifier] ~= nil
          and slice_end_type == "$" then
        for header_level = #active_attributes, 1, -1 do
          if options.headerAttributes and #active_attributes[header_level] > 0 then
            table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
          end
        end
        self.is_writing = false
      end
      table.remove(active_attributes, #active_attributes)
      if self.is_writing and options.headerAttributes and #active_identifiers > 0 then
        table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
      end
      -- apply all active attributes at slice beginning
      if active_identifiers["#" .. slice_begin_identifier] ~= nil
          and slice_begin_type == "$" then
        for header_level = 1, #active_attributes do
          if options.headerAttributes and #active_attributes[header_level] > 0 then
            table.insert(buf, "\\markdownRendererHeaderAttributeContextBegin")
          end
        end
        self.is_writing = true
      end
    end

    -- tear down all active attributes at slice end
    if attributes["#" .. slice_end_identifier] ~= nil
        and slice_end_type == "^" then
      for header_level = #active_attributes, 1, -1 do
        if options.headerAttributes and #active_attributes[header_level] > 0 then
          table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
        end
      end
      self.is_writing = false
    end

    -- push attributes for the new section
    table.insert(active_attributes, attributes)
    if self.is_writing and options.headerAttributes and #attributes > 0 then
      table.insert(buf, "\\markdownRendererHeaderAttributeContextBegin")
    end

    -- apply all active attributes at slice beginning
    if attributes["#" .. slice_begin_identifier] ~= nil
        and slice_begin_type == "^" then
```

```
      for header_level = 1, #active_attributes do
        if options.headerAttributes and #active_attributes[header_level] > 0 then
          table.insert(buf, "\\markdownRendererHeaderAttributeContextBegin")
        end
      end
      self.is_writing = true
    end

    if self.is_writing then
      table.sort(attributes)
      local key, value
      for i = 1, #attributes do
        if attributes[i]:sub(1, 1) == "#" then
          table.insert(buf, {"\\markdownRendererAttributeIdentifier{",
                             attributes[i]:sub(2), "}"})
        elseif attributes[i]:sub(1, 1) == "." then
          table.insert(buf, {"\\markdownRendererAttributeClassName{",
                             attributes[i]:sub(2), "}"})
        else
          key, value = attributes[i]:match("(%w+)=(%w+)")
          table.insert(buf, {"\\markdownRendererAttributeKeyValue{",
                             key, "}{", value, "}"})
        end
      end
    end

    local cmd
    level = level + options.shiftHeadings
    if level <= 1 then
      cmd = "\\markdownRendererHeadingOne"
    elseif level == 2 then
      cmd = "\\markdownRendererHeadingTwo"
    elseif level == 3 then
      cmd = "\\markdownRendererHeadingThree"
    elseif level == 4 then
      cmd = "\\markdownRendererHeadingFour"
    elseif level == 5 then
      cmd = "\\markdownRendererHeadingFive"
    elseif level >= 6 then
      cmd = "\\markdownRendererHeadingSix"
    else
      cmd = ""
    end
    if self.is_writing then
      table.insert(buf, {cmd, "{", s, "}"})
    end

```

```
4593      return buf
4594    end
```

Define `writer->note` as a function that will transform an input footnote `s` to the output format.

```
4595    function self.note(s)
4596      return {"\\markdownRendererFootnote{",s,"}"}
4597    end
```

Define `writer->citations` as a function that will transform an input array of citations `cites` to the output format. If `text_cites` is enabled, the citations should be rendered in-text, when applicable. The `cites` array contains tables with the following keys and values:

- `suppress_author` – If the value of the key is true, then the author of the work should be omitted in the citation, when applicable.
- `prenote` – The value of the key is either `nil` or a rope that should be inserted before the citation.
- `postnote` – The value of the key is either `nil` or a rope that should be inserted after the citation.
- `name` – The value of this key is the citation name.

```
4598    function self.citations(text_cites, cites)
4599      local buffer = {"\\markdownRenderer", text_cites and "TextCite" or "Cite",
4600        "{", #cites, "}"}
4601      for _,cite in ipairs(cites) do
4602        buffer[#buffer+1] = {cite.suppress_author and "-" or "+", "{",
4603          cite.prenote or "", "}{", cite.postnote or "", "}{", cite.name, "}"}
4604      end
4605      return buffer
4606    end
```

Define `writer->get_state` as a function that returns the current state of the writer, where the state of a writer are its mutable member variables.

```
4607    function self.get_state()
4608      return {
4609        is_writing=self.is_writing,
4610        active_attributes={table.unpack(self.active_attributes)},
4611      }
4612    end
```

Define `writer->set_state` as a function that restores the input state `s` and returns the previous state of the writer.

```
4613    function self.set_state(s)
4614      local previous_state = self.get_state()
```

```
    for key, value in pairs(s) do
      self[key] = value
    end
    return previous_state
  end
```

Define `writer->defer_call` as a function that will encapsulate the input function `f`, so that `f` is called with the state of the writer at the time of calling `writer->defer_call`.

```
  function self.defer_call(f)
    local previous_state = self.get_state()
    return function(...)
      local state = self.set_state(previous_state)
      local return_value = f(...)
      self.set_state(state)
      return return_value
    end
  end

  return self
end
```

3.1.4 Parsers

The `parsers` hash table stores PEG patterns that are static and can be reused between different `reader` objects.

```
local parsers                 = {}
```

3.1.4.1 Basic Parsers

```
parsers.percent               = P("%")
parsers.at                    = P("@")
parsers.comma                 = P(",")
parsers.asterisk              = P("*")
parsers.dash                  = P("-")
parsers.plus                  = P("+")
parsers.underscore            = P("_")
parsers.period                = P(".")
parsers.hash                  = P("#")
parsers.ampersand             = P("&")
parsers.backtick              = P("`")
parsers.less                  = P("<")
parsers.more                  = P(">")
parsers.space                 = P(" ")
parsers.squote                = P("'")
parsers.dquote                = P('"')
parsers.lparent               = P("(")
```

```
parsers.rparent               = P(")")
parsers.lbracket              = P("[")
parsers.rbracket              = P("]")
parsers.lbrace                = P("{")
parsers.rbrace                = P("}")
parsers.circumflex            = P("^")
parsers.slash                 = P("/")
parsers.equal                 = P("=")
parsers.colon                 = P(":")
parsers.semicolon             = P(";")
parsers.exclamation           = P("!")
parsers.pipe                  = P("|")
parsers.tilde                 = P("~")
parsers.backslash             = P("\\")
parsers.tab                   = P("\t")
parsers.newline               = P("\n")
parsers.tightblocksep         = P("\001")

parsers.digit                 = R("09")
parsers.hexdigit              = R("09","af","AF")
parsers.letter                = R("AZ","az")
parsers.alphanumeric          = R("AZ","az","09")
parsers.keyword               = parsers.letter
                              * parsers.alphanumeric^0
parsers.citation_chars        = parsers.alphanumeric
                              + S("#$%&-+<>~/_")
parsers.internal_punctuation  = S(":;,.?")

parsers.doubleasterisks       = P("**")
parsers.doubleunderscores     = P("__")
parsers.fourspaces            = P("    ")

parsers.any                   = P(1)
parsers.fail                  = parsers.any - 1

parsers.escapable             = S("\\`*_{}[]()+_.!<>#-~:^@;")
parsers.anyescaped            = parsers.backslash / "" * parsers.escapable
                              + parsers.any

parsers.spacechar             = S("\t ")
parsers.spacing               = S(" \n\r\t")
parsers.nonspacechar          = parsers.any - parsers.spacing
parsers.optionalspace         = parsers.spacechar^0

parsers.specialchar           = S("*_`&[]<!\\.@-^")

parsers.normalchar            = parsers.any - (parsers.specialchar
```

```
                                                + parsers.spacing
                                                + parsers.tightblocksep)
parsers.eof                    = -parsers.any
parsers.nonindentspace         = parsers.space^-3 * - parsers.spacechar
parsers.indent                 = parsers.space^-3 * parsers.tab
                               + parsers.fourspaces / ""
parsers.linechar               = P(1 - parsers.newline)

parsers.blankline              = parsers.optionalspace
                               * parsers.newline / "\n"
parsers.blanklines             = parsers.blankline^0
parsers.skipblanklines         = (parsers.optionalspace * parsers.newline)^0
parsers.indentedline           = parsers.indent    /""
                               * C(parsers.linechar^1 * parsers.newline^-
1)
parsers.optionallyindentedline = parsers.indent^-1 /""
                               * C(parsers.linechar^1 * parsers.newline^-
1)
parsers.sp                     = parsers.spacing^0
parsers.spnl                   = parsers.optionalspace
                               * (parsers.newline * parsers.optionalspace)^-
1
parsers.line                   = parsers.linechar^0 * parsers.newline
parsers.nonemptyline           = parsers.line - parsers.blankline
```

The `parsers.commented_line^1` parser recognizes the regular language of TEX comments, see an equivalent finite automaton in Figure 6.

```
parsers.commented_line_letter  = parsers.linechar
                               + parsers.newline
                               - parsers.backslash
                               - parsers.percent
parsers.commented_line         = Cg(Cc(""), "backslashes")
                               * ((#(parsers.commented_line_letter
                                    - parsers.newline)
                                  * Cb("backslashes")
                                  * Cs(parsers.commented_line_letter
                                    - parsers.newline)^1  -- initial
                                  * Cg(Cc(""), "backslashes"))
                                 + #(parsers.backslash * parsers.backslash)
                                 * Cg((parsers.backslash  -- even backslash
                                      * parsers.backslash)^1, "backslashes")
                                 + (parsers.backslash
                                   * (#parsers.percent
                                     * Cb("backslashes")
                                     / function(backslashes)
                                       return string.rep("\\", #backslashes / 2)
                                     end
```

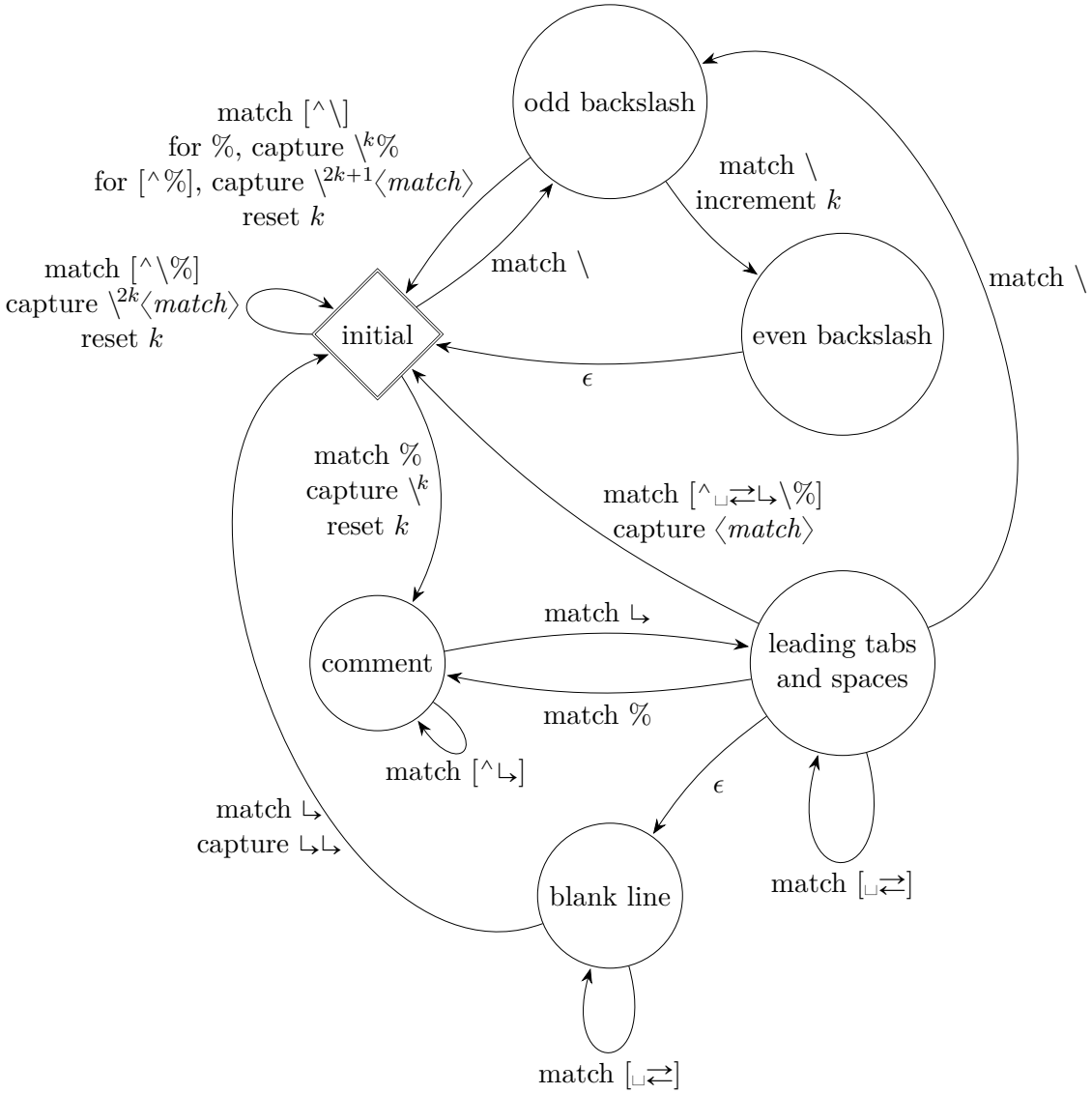

Figure 6: A pushdown automaton that recognizes TEX comments

```
                                      * C(parsers.percent)
                                      + #parsers.commented_line_letter
                                      * Cb("backslashes")
                                      * Cc("\\")
                                      * C(parsers.commented_line_letter))
                                    * Cg(Cc(""), "backslashes")))^0
                                  * (#parsers.percent
                                    * Cb("backslashes")
                                    / function(backslashes)
                                      return string.rep("\\", #backslashes / 2)
                                    end
                                    * ((parsers.percent  -- comment
                                       * parsers.line
                                       * #parsers.blankline) -- blank line
                                      / "\n"
                                      + parsers.percent  -- comment
                                      * parsers.line
                                      * parsers.optionalspace)  -- leading tabs and spaces
                                    + #(parsers.newline)
                                    * Cb("backslashes")
                                    * C(parsers.newline))

parsers.chunk                     = parsers.line * (parsers.optionallyindentedline
                                                    - parsers.blankline)^0

parsers.css_identifier_char       = R("AZ", "az", "09") + S("-_")
parsers.css_identifier            = (parsers.hash + parsers.period)
                                  * (((parsers.css_identifier_char
                                      - parsers.dash - parsers.digit)
                                     * parsers.css_identifier_char^1)
                                    + (parsers.dash
                                      * (parsers.css_identifier_char
                                        - parsers.digit)
                                      * parsers.css_identifier_char^0))
parsers.attribute_key_char        = parsers.any - parsers.space
                                  - parsers.squote - parsers.dquote
                                  - parsers.more - parsers.slash
                                  - parsers.equal
parsers.attribute_value_char      = parsers.any - parsers.space
                                  - parsers.dquote - parsers.more

-- block followed by 0 or more optionally
-- indented blocks with first line indented.
parsers.indented_blocks = function(bl)
  return Cs( bl
         * (parsers.blankline^1 * parsers.indent * -parsers.blankline * bl)^0
         * (parsers.blankline^1 + parsers.eof) )
```

```
4785 end
```

3.1.4.2 Parsers Used for Markdown Lists

```
4786 parsers.bulletchar = C(parsers.plus + parsers.asterisk + parsers.dash)
4787
4788 parsers.bullet = ( parsers.bulletchar * #parsers.spacing
4789                                       * (parsers.tab + parsers.space^-3)
4790                  + parsers.space * parsers.bulletchar * #parsers.spacing
4791                                  * (parsers.tab + parsers.space^-2)
4792                  + parsers.space * parsers.space * parsers.bulletchar
4793                                  * #parsers.spacing
4794                                  * (parsers.tab + parsers.space^-1)
4795                  + parsers.space * parsers.space * parsers.space
4796                                  * parsers.bulletchar * #parsers.spacing
4797                  )
4798
4799 local function tickbox(interior)
4800   return parsers.optionalspace * parsers.lbracket
4801         * interior * parsers.rbracket * parsers.spacechar^1
4802 end
4803
4804 parsers.ticked_box = tickbox(S("xX")) * Cc(1.0)
4805 parsers.halfticked_box = tickbox(S("./")) * Cc(0.5)
4806 parsers.unticked_box = tickbox(parsers.spacechar^1) * Cc(0.0)
4807
```

3.1.4.3 Parsers Used for Markdown Code Spans

```
4808 parsers.openticks   = Cg(parsers.backtick^1, "ticks")
4809
4810 local function captures_equal_length(s,i,a,b)
4811   return #a == #b and i
4812 end
4813
4814 parsers.closeticks  = parsers.space^-1
4815                     * Cmt(C(parsers.backtick^1)
4816                         * Cb("ticks"), captures_equal_length)
4817
4818 parsers.intickschar = (parsers.any - S(" \n\r`"))
4819                     + (parsers.newline * -parsers.blankline)
4820                     + (parsers.space - parsers.closeticks)
4821                     + (parsers.backtick^1 - parsers.closeticks)
4822
4823 parsers.inticks     = parsers.openticks * parsers.space^-1
4824                     * C(parsers.intickschar^0) * parsers.closeticks
```

3.1.4.4 Parsers Used for Fenced Code Blocks

```
local function captures_geq_length(s,i,a,b)
  return #a >= #b and i
end

parsers.infostring     = (parsers.linechar - (parsers.backtick
                       + parsers.space^1 * (parsers.newline + parsers.eof)))^0

local fenceindent
parsers.fencehead    = function(char)
  return               C(parsers.nonindentspace) / function(s) fenceindent = #s end
                     * Cg(char^3, "fencelength")
                     * parsers.optionalspace * C(parsers.infostring)
                     * parsers.optionalspace * (parsers.newline + parsers.eof)
end

parsers.fencetail    = function(char)
  return               parsers.nonindentspace
                     * Cmt(C(char^3) * Cb("fencelength"), captures_geq_length)
                     * parsers.optionalspace * (parsers.newline + parsers.eof)
                     + parsers.eof
end

parsers.fencedline   = function(char)
  return               C(parsers.line - parsers.fencetail(char))
                     / function(s)
                         i = 1
                         remaining = fenceindent
                         while true do
                           c = s:sub(i, i)
                           if c == " " and remaining > 0 then
                             remaining = remaining - 1
                             i = i + 1
                           elseif c == "\t" and remaining > 3 then
                             remaining = remaining - 4
                             i = i + 1
                           else
                             break
                           end
                         end
                         return s:sub(i)
                       end
end
```

3.1.4.5 Parsers Used for Markdown Tags and Links

```
parsers.leader      = parsers.space^-3
```

```
4868
4869 -- content in balanced brackets, parentheses, or quotes:
4870 parsers.bracketed   = P{ parsers.lbracket
4871                        * ((parsers.anyescaped - (parsers.lbracket
4872                                                + parsers.rbracket
4873                                                + parsers.blankline^2)
4874                           ) + V(1))^0
4875                        * parsers.rbracket }
4876
4877 parsers.inparens    = P{ parsers.lparent
4878                        * ((parsers.anyescaped - (parsers.lparent
4879                                                + parsers.rparent
4880                                                + parsers.blankline^2)
4881                           ) + V(1))^0
4882                        * parsers.rparent }
4883
4884 parsers.squoted     = P{ parsers.squote * parsers.alphanumeric
4885                        * ((parsers.anyescaped - (parsers.squote
4886                                                + parsers.blankline^2)
4887                           ) + V(1))^0
4888                        * parsers.squote }
4889
4890 parsers.dquoted     = P{ parsers.dquote * parsers.alphanumeric
4891                        * ((parsers.anyescaped - (parsers.dquote
4892                                                + parsers.blankline^2)
4893                           ) + V(1))^0
4894                        * parsers.dquote }
4895
4896 -- bracketed tag for markdown links, allowing nested brackets:
4897 parsers.tag         = parsers.lbracket
4898                     * Cs((parsers.alphanumeric^1
4899                          + parsers.bracketed
4900                          + parsers.inticks
4901                          + (parsers.anyescaped
4902                            - (parsers.rbracket + parsers.blankline^2)))^0)
4903                     * parsers.rbracket
4904
4905 -- url for markdown links, allowing nested brackets:
4906 parsers.url         = parsers.less * Cs((parsers.anyescaped
4907                                         - parsers.more)^0)
4908                                    * parsers.more
4909                     + Cs((parsers.inparens + (parsers.anyescaped
4910                                             - parsers.spacing
4911                                             - parsers.rparent))^1)
4912
4913 -- quoted text, possibly with nested quotes:
4914 parsers.title_s     = parsers.squote * Cs(((parsers.anyescaped-parsers.squote)
```

```
                                        + parsers.squoted)^0)
                                    * parsers.squote

parsers.title_d     = parsers.dquote * Cs(((parsers.anyescaped-parsers.dquote)
                                        + parsers.dquoted)^0)
                                    * parsers.dquote

parsers.title_p     = parsers.lparent
                    * Cs((parsers.inparens + (parsers.anyescaped-parsers.rparent))^0)
                    * parsers.rparent

parsers.title       = parsers.title_d + parsers.title_s + parsers.title_p

parsers.optionaltitle
                    = parsers.spnl * parsers.title * parsers.spacechar^0
                    + Cc("")
```

3.1.4.6 Parsers Used for iA Writer Content Blocks

```
parsers.contentblock_tail
                    = parsers.optionaltitle
                    * (parsers.newline + parsers.eof)

-- case insensitive online image suffix:
parsers.onlineimagesuffix
                    = (function(...)
                        local parser = nil
                        for _,suffix in ipairs({...}) do
                          local pattern=nil
                          for i=1,#suffix do
                            local char=suffix:sub(i,i)
                            char = S(char:lower()..char:upper())
                            if pattern == nil then
                              pattern = char
                            else
                              pattern = pattern * char
                            end
                          end
                          if parser == nil then
                            parser = pattern
                          else
                            parser = parser + pattern
                          end
                        end
                        return parser
                      end)("png", "jpg", "jpeg", "gif", "tif", "tiff")
```

```
-- online image url for iA Writer content blocks with mandatory suffix,
-- allowing nested brackets:
parsers.onlineimageurl
                     = (parsers.less
                       * Cs((parsers.anyescaped
                            - parsers.more
                            - #(parsers.period
                               * parsers.onlineimagesuffix
                               * parsers.more
                               * parsers.contentblock_tail))^0)
                       * parsers.period
                       * Cs(parsers.onlineimagesuffix)
                       * parsers.more
                       + (Cs((parsers.inparens
                             + (parsers.anyescaped
                               - parsers.spacing
                               - parsers.rparent
                               - #(parsers.period
                                  * parsers.onlineimagesuffix
                                  * parsers.contentblock_tail)))^0)
                         * parsers.period
                         * Cs(parsers.onlineimagesuffix))
                       ) * Cc("onlineimage")

-- filename for iA Writer content blocks with mandatory suffix:
parsers.localfilepath
                     = parsers.slash
                     * Cs((parsers.anyescaped
                          - parsers.tab
                          - parsers.newline
                          - #(parsers.period
                             * parsers.alphanumeric^1
                             * parsers.contentblock_tail))^1)
                     * parsers.period
                     * Cs(parsers.alphanumeric^1)
                     * Cc("localfile")
```

3.1.4.7 Parsers Used for Citations

```
parsers.citation_name = Cs(parsers.dash^-1) * parsers.at
                       * Cs(parsers.citation_chars
                           * (((parsers.citation_chars + parsers.internal_punctuation
                               - parsers.comma - parsers.semicolon)
                              * -#((parsers.internal_punctuation - parsers.comma
                                   - parsers.semicolon)^0
                                  * -(parsers.citation_chars + parsers.internal_punctuati
                                     - parsers.comma - parsers.semicolon)))^0
```

```
                            * parsers.citation_chars)^-1)

parsers.citation_body_prenote
                      = Cs((parsers.alphanumeric^1
                           + parsers.bracketed
                           + parsers.inticks
                           + (parsers.anyescaped
                             - (parsers.rbracket + parsers.blankline^2))
                           - (parsers.spnl * parsers.dash^-1 * parsers.at))^0)

parsers.citation_body_postnote
                      = Cs((parsers.alphanumeric^1
                           + parsers.bracketed
                           + parsers.inticks
                           + (parsers.anyescaped
                             - (parsers.rbracket + parsers.semicolon
                               + parsers.blankline^2))
                           - (parsers.spnl * parsers.rbracket))^0)

parsers.citation_body_chunk
                      = parsers.citation_body_prenote
                      * parsers.spnl * parsers.citation_name
                      * (parsers.internal_punctuation - parsers.semicolon)^-
1
                      * parsers.spnl * parsers.citation_body_postnote

parsers.citation_body
                      = parsers.citation_body_chunk
                      * (parsers.semicolon * parsers.spnl
                        * parsers.citation_body_chunk)^0

parsers.citation_headless_body_postnote
                      = Cs((parsers.alphanumeric^1
                           + parsers.bracketed
                           + parsers.inticks
                           + (parsers.anyescaped
                             - (parsers.rbracket + parsers.at
                               + parsers.semicolon + parsers.blankline^2))
                           - (parsers.spnl * parsers.rbracket))^0)

parsers.citation_headless_body
                      = parsers.citation_headless_body_postnote
                      * (parsers.sp * parsers.semicolon * parsers.spnl
                        * parsers.citation_body_chunk)^0
```

3.1.4.8 Parsers Used for Footnotes

```
local function strip_first_char(s)
  return s:sub(2)
end

parsers.RawNoteRef = #(parsers.lbracket * parsers.circumflex)
                   * parsers.tag / strip_first_char
```

3.1.4.9 Parsers Used for Tables

```
local function make_pipe_table_rectangular(rows)
  local num_columns = #rows[2]
  local rectangular_rows = {}
  for i = 1, #rows do
    local row = rows[i]
    local rectangular_row = {}
    for j = 1, num_columns do
      rectangular_row[j] = row[j] or ""
    end
    table.insert(rectangular_rows, rectangular_row)
  end
  return rectangular_rows
end

local function pipe_table_row(allow_empty_first_column
                              , nonempty_column
                              , column_separator
                              , column)
  local row_beginning
  if allow_empty_first_column then
    row_beginning = -- empty first column
                    #(parsers.spacechar^4
                     * column_separator)
                  * parsers.optionalspace
                  * column
                  * parsers.optionalspace
                  -- non-empty first column
                  + parsers.nonindentspace
                  * nonempty_column^-1
                  * parsers.optionalspace
  else
    row_beginning = parsers.nonindentspace
                  * nonempty_column^-1
                  * parsers.optionalspace
  end

  return Ct(row_beginning
            * (-- single column with no leading pipes
```

```
               #(column_separator
                 * parsers.optionalspace
                 * parsers.newline)
               * column_separator
               * parsers.optionalspace
               -- single column with leading pipes or
               -- more than a single column
               + (column_separator
                 * parsers.optionalspace
                 * column
                 * parsers.optionalspace)^1
               * (column_separator
                 * parsers.optionalspace)^-1))
end

parsers.table_hline_separator = parsers.pipe + parsers.plus
parsers.table_hline_column = (parsers.dash
                              - #(parsers.dash
                                 * (parsers.spacechar
                                   + parsers.table_hline_separator
                                   + parsers.newline)))^1
                            * (parsers.colon * Cc("r")
                              + parsers.dash * Cc("d"))
                            + parsers.colon
                            * (parsers.dash
                              - #(parsers.dash
                                 * (parsers.spacechar
                                   + parsers.table_hline_separator
                                   + parsers.newline)))^1
                            * (parsers.colon * Cc("c")
                              + parsers.dash * Cc("l"))
parsers.table_hline = pipe_table_row(false
                                     , parsers.table_hline_column
                                     , parsers.table_hline_separator
                                     , parsers.table_hline_column)
parsers.table_caption_beginning = parsers.skipblanklines
                                  * parsers.nonindentspace
                                  * (P("Table")^-1 * parsers.colon)
                                  * parsers.optionalspace
```

3.1.4.10 Parsers Used for HTML

```
-- case-insensitive match (we assume s is lowercase). must be single byte encoding
parsers.keyword_exact = function(s)
  local parser = P(0)
  for i=1,#s do
    local c = s:sub(i,i)
```

```
    local m = c .. upper(c)
    parser = parser * S(m)
  end
  return parser
end

parsers.block_keyword =
    parsers.keyword_exact("address") + parsers.keyword_exact("blockquote") +
    parsers.keyword_exact("center") + parsers.keyword_exact("del") +
    parsers.keyword_exact("dir") + parsers.keyword_exact("div") +
    parsers.keyword_exact("p") + parsers.keyword_exact("pre") +
    parsers.keyword_exact("li") + parsers.keyword_exact("ol") +
    parsers.keyword_exact("ul") + parsers.keyword_exact("dl") +
    parsers.keyword_exact("dd") + parsers.keyword_exact("form") +
    parsers.keyword_exact("fieldset") + parsers.keyword_exact("isindex") +
    parsers.keyword_exact("ins") + parsers.keyword_exact("menu") +
    parsers.keyword_exact("noframes") + parsers.keyword_exact("frameset") +
    parsers.keyword_exact("h1") + parsers.keyword_exact("h2") +
    parsers.keyword_exact("h3") + parsers.keyword_exact("h4") +
    parsers.keyword_exact("h5") + parsers.keyword_exact("h6") +
    parsers.keyword_exact("hr") + parsers.keyword_exact("script") +
    parsers.keyword_exact("noscript") + parsers.keyword_exact("table") +
    parsers.keyword_exact("tbody") + parsers.keyword_exact("tfoot") +
    parsers.keyword_exact("thead") + parsers.keyword_exact("th") +
    parsers.keyword_exact("td") + parsers.keyword_exact("tr")

-- There is no reason to support bad html, so we expect quoted attributes
parsers.htmlattributevalue
                          = parsers.squote * (parsers.any - (parsers.blankline
                                                           + parsers.squote))^0
                                           * parsers.squote
                          + parsers.dquote * (parsers.any - (parsers.blankline
                                                           + parsers.dquote))^0
                                           * parsers.dquote

parsers.htmlattribute     = parsers.spacing^1
                          * (parsers.alphanumeric + S("_-"))^1
                          * parsers.sp * parsers.equal * parsers.sp
                          * parsers.htmlattributevalue

parsers.htmlcomment       = P("<!--")
                          * parsers.optionalspace
                          * Cs((parsers.any - parsers.optionalspace * P("-->"))^0)
                          * parsers.optionalspace
                          * P("-->")

parsers.htmlinstruction   = P("<?") * (parsers.any - P("?>"))^0 * P("?>")
```

```
5181
5182 parsers.openelt_any = parsers.less * parsers.keyword * parsers.htmlattribute^0
5183                     * parsers.sp * parsers.more
5184
5185 parsers.openelt_exact = function(s)
5186   return parsers.less * parsers.sp * parsers.keyword_exact(s)
5187        * parsers.htmlattribute^0 * parsers.sp * parsers.more
5188 end
5189
5190 parsers.openelt_block = parsers.sp * parsers.block_keyword
5191                       * parsers.htmlattribute^0 * parsers.sp * parsers.more
5192
5193 parsers.closeelt_any = parsers.less * parsers.sp * parsers.slash
5194                      * parsers.keyword * parsers.sp * parsers.more
5195
5196 parsers.closeelt_exact = function(s)
5197   return parsers.less * parsers.sp * parsers.slash * parsers.keyword_exact(s)
5198        * parsers.sp * parsers.more
5199 end
5200
5201 parsers.emptyelt_any = parsers.less * parsers.sp * parsers.keyword
5202                      * parsers.htmlattribute^0 * parsers.sp * parsers.slash
5203                      * parsers.more
5204
5205 parsers.emptyelt_block = parsers.less * parsers.sp * parsers.block_keyword
5206                        * parsers.htmlattribute^0 * parsers.sp * parsers.slash
5207                        * parsers.more
5208
5209 parsers.displaytext = (parsers.any - parsers.less)^1
5210
5211 -- return content between two matched HTML tags
5212 parsers.in_matched = function(s)
5213   return { parsers.openelt_exact(s)
5214          * (V(1) + parsers.displaytext
5215            + (parsers.less - parsers.closeelt_exact(s)))^0
5216          * parsers.closeelt_exact(s) }
5217 end
5218
5219 local function parse_matched_tags(s,pos)
5220   local t = string.lower(lpeg.match(C(parsers.keyword),s,pos))
5221   return lpeg.match(parsers.in_matched(t),s,pos-1)
5222 end
5223
5224 parsers.in_matched_block_tags = parsers.less
5225                               * Cmt(#parsers.openelt_block, parse_matched_tags)
5226
```

3.1.4.11 Parsers Used for HTML Entities

```
parsers.hexentity = parsers.ampersand * parsers.hash * S("Xx")
                  * C(parsers.hexdigit^1) * parsers.semicolon
parsers.decentity = parsers.ampersand * parsers.hash
                  * C(parsers.digit^1) * parsers.semicolon
parsers.tagentity = parsers.ampersand * C(parsers.alphanumeric^1)
                  * parsers.semicolon
```

3.1.4.12 Helpers for References

```
-- parse a reference definition:  [foo]: /bar "title"
parsers.define_reference_parser = parsers.leader * parsers.tag * parsers.colon
                                * parsers.spacechar^0 * parsers.url
                                * parsers.optionaltitle * parsers.blankline^1
```

3.1.4.13 Inline Elements

```
parsers.Inline         = V("Inline")
parsers.IndentedInline = V("IndentedInline")

-- parse many p between starter and ender
parsers.between = function(p, starter, ender)
  local ender2 = B(parsers.nonspacechar) * ender
  return (starter * #parsers.nonspacechar * Ct(p * (p - ender2)^0) * ender2)
end

parsers.urlchar      = parsers.anyescaped - parsers.newline - parsers.more
```

3.1.4.14 Block Elements

```
parsers.OnlineImageURL
                     = parsers.leader
                     * parsers.onlineimageurl
                     * parsers.optionaltitle

parsers.LocalFilePath
                     = parsers.leader
                     * parsers.localfilepath
                     * parsers.optionaltitle

parsers.TildeFencedCode
                     = parsers.fencehead(parsers.tilde)
                     * Cs(parsers.fencedline(parsers.tilde)^0)
                     * parsers.fencetail(parsers.tilde)

parsers.BacktickFencedCode
                     = parsers.fencehead(parsers.backtick)
                     * Cs(parsers.fencedline(parsers.backtick)^0)
```

```
                      * parsers.fencetail(parsers.backtick)

parsers.JekyllFencedCode
                      = parsers.fencehead(parsers.dash)
                      * Cs(parsers.fencedline(parsers.dash)^0)
                      * parsers.fencetail(parsers.dash)

parsers.lineof = function(c)
    return (parsers.leader * (P(c) * parsers.optionalspace)^3
            * (parsers.newline * parsers.blankline^1
              + parsers.newline^-1 * parsers.eof))
end
```

3.1.4.15 Lists

```
parsers.defstartchar = S("~:")
parsers.defstart     = ( parsers.defstartchar * #parsers.spacing
                                              * (parsers.tab + parsers.space^-
3)
                      + parsers.space * parsers.defstartchar * #parsers.spacing
                                      * (parsers.tab + parsers.space^-2)
                      + parsers.space * parsers.space * parsers.defstartchar
                                      * #parsers.spacing
                                      * (parsers.tab + parsers.space^-1)
                      + parsers.space * parsers.space * parsers.space
                                      * parsers.defstartchar * #parsers.spacing
                      )

parsers.dlchunk = Cs(parsers.line * (parsers.indentedline - parsers.blankline)^0)
```

3.1.4.16 Headings

```
parsers.heading_attribute = C(parsers.css_identifier)
                          + C((parsers.attribute_key_char
                             - parsers.rbrace)^1
                             * parsers.equal
                             * (parsers.attribute_value_char
                               - parsers.rbrace)^1)
parsers.HeadingAttributes = parsers.lbrace
                          * parsers.heading_attribute
                          * (parsers.spacechar^1
                            * parsers.heading_attribute)^0
                          * parsers.rbrace

-- parse Atx heading start and return level
parsers.HeadingStart = #parsers.hash * C(parsers.hash^-6)
                      * -parsers.hash / length

```

```
-- parse setext header ending and return level
parsers.HeadingLevel = parsers.equal^1 * Cc(1) + parsers.dash^1 * Cc(2)

local function strip_atx_end(s)
  return s:gsub("[#%s]*\n$","")
end
```

3.1.5 Markdown Reader

This section documents the `reader` object, which implements the routines for parsing the markdown input. The object corresponds to the markdown reader object that was located in the `lunamark/reader/markdown.lua` file in the Lunamark Lua module.

Although not specified in the Lua interface (see Section 2.1), the `reader` object is exported, so that the curious user could easily tinker with the methods of the objects produced by the `reader.new` method described below. The user should be aware, however, that the implementation may change in a future revision.

The `reader.new` method creates and returns a new TEX reader object associated with the Lua interface options (see Section 2.1.2) `options` and with a writer object `writer`. When `options` are unspecified, it is assumed that an empty table was passed to the method.

The objects produced by the `reader.new` method expose instance methods and variables of their own. As a convention, I will refer to these ⟨*member*⟩s as `reader->`⟨*member*⟩.

```
M.reader = {}
function M.reader.new(writer, options)
  local self = {}
  options = options or {}
```

Make the `options` table inherit from the `defaultOptions` table.

```
  setmetatable(options, { __index = function (_, key)
    return defaultOptions[key] end })
```

3.1.5.1 Top-Level Helper Functions

Define `normalize_tag` as a function that normalizes a markdown reference tag by lowercasing it, and by collapsing any adjacent whitespace characters.

```
  local function normalize_tag(tag)
    return string.lower(
      gsub(util.rope_to_string(tag), "[ \n\r\t]+", " "))
  end
```

Define `iterlines` as a function that iterates over the lines of the input string `s`, transforms them using an input function `f`, and reassembles them into a new string, which it returns.

```
  local function iterlines(s, f)
    rope = lpeg.match(Ct((parsers.line / f)^1), s)
```

```
5324    return util.rope_to_string(rope)
5325  end
```

Define `expandtabs` either as an identity function, when the `preserveTabs` Lua interface option is enabled, or to a function that expands tabs into spaces otherwise.

```
5326  local expandtabs
5327  if options.preserveTabs then
5328    expandtabs = function(s) return s end
5329  else
5330    expandtabs = function(s)
5331                   if s:find("\t") then
5332                     return iterlines(s, util.expand_tabs_in_line)
5333                   else
5334                     return s
5335                   end
5336                 end
5337  end
```

The `larsers` (as in ‘`local \luamref{parsers}'') hash table stores \acro{peg} patterns` tions‘, which impedes their reuse between different `reader` objects.

```
5338  local larsers    = {}
```

3.1.5.2 Top-Level Parser Functions

```
5339  local function create_parser(name, grammar, toplevel)
5340    return function(str)
```

If the parser is top-level and the `stripIndent` Lua option is enabled, we will first expand tabs in the input string `str` into spaces and then we will count the minimum indent across all lines, skipping blank lines. Next, we will remove the minimum indent from all lines.

```
5341      if toplevel and options.stripIndent then
5342          local min_prefix_length, min_prefix = nil, ''
5343          str = iterlines(str, function(line)
5344              if lpeg.match(parsers.nonemptyline, line) == nil then
5345                  return line
5346              end
5347              line = util.expand_tabs_in_line(line)
5348              prefix = lpeg.match(C(parsers.optionalspace), line)
5349              local prefix_length = #prefix
5350              local is_shorter = min_prefix_length == nil
5351              is_shorter = is_shorter or prefix_length < min_prefix_length
5352              if is_shorter then
5353                  min_prefix_length, min_prefix = prefix_length, prefix
5354              end
5355              return line
5356          end)
5357          str = str:gsub('^' .. min_prefix, '')
```

```
5358      end
```

If the parser is top-level and the `texComments` or `hybrid` Lua options are enabled, we will strip all plain TEX comments from the input string `str` together with the trailing newline characters.

```
5359      if toplevel and (options.texComments or options.hybrid) then
5360          str = lpeg.match(Ct(parsers.commented_line^1), str)
5361          str = util.rope_to_string(str)
5362      end
5363      local res = lpeg.match(grammar(), str)
5364      if res == nil then
5365        error(format("%s failed on:\n%s", name, str:sub(1,20)))
5366      else
5367        return res
5368      end
5369    end
5370  end
5371
5372  local parse_blocks
5373    = create_parser("parse_blocks",
5374                    function()
5375                      return larsers.blocks
5376                    end, true)
5377
5378  local parse_blocks_nested
5379    = create_parser("parse_blocks_nested",
5380                    function()
5381                      return larsers.blocks_nested
5382                    end, false)
5383
5384  local parse_inlines
5385    = create_parser("parse_inlines",
5386                    function()
5387                      return larsers.inlines
5388                    end, false)
5389
5390  local parse_inlines_no_link
5391    = create_parser("parse_inlines_no_link",
5392                    function()
5393                      return larsers.inlines_no_link
5394                    end, false)
5395
5396  local parse_inlines_no_inline_note
5397    = create_parser("parse_inlines_no_inline_note",
5398                    function()
5399                      return larsers.inlines_no_inline_note
5400                    end, false)
```

```
5401
5402  local parse_inlines_no_html
5403    = create_parser("parse_inlines_no_html",
5404                    function()
5405                      return larsers.inlines_no_html
5406                    end, false)
5407
5408  local parse_inlines_nbsp
5409    = create_parser("parse_inlines_nbsp",
5410                    function()
5411                      return larsers.inlines_nbsp
5412                    end, false)
```

3.1.5.3 Parsers Used for Markdown Lists (local)

```
5413  if options.hashEnumerators then
5414    larsers.dig = parsers.digit + parsers.hash
5415  else
5416    larsers.dig = parsers.digit
5417  end
5418
5419  larsers.enumerator = C(larsers.dig^3 * parsers.period) * #parsers.spacing
5420                     + C(larsers.dig^2 * parsers.period) * #parsers.spacing
5421                                         * (parsers.tab + parsers.space^1)
5422                     + C(larsers.dig * parsers.period) * #parsers.spacing
5423                                       * (parsers.tab + parsers.space^-2)
5424                     + parsers.space * C(larsers.dig^2 * parsers.period)
5425                                       * #parsers.spacing
5426                     + parsers.space * C(larsers.dig * parsers.period)
5427                                       * #parsers.spacing
5428                                       * (parsers.tab + parsers.space^-1)
5429                     + parsers.space * parsers.space * C(larsers.dig^1
5430                                       * parsers.period) * #parsers.spacing
```

3.1.5.4 Parsers Used for Blockquotes (local)

```
5431  -- strip off leading > and indents, and run through blocks
5432  larsers.blockquote_body = ((parsers.leader * parsers.more * parsers.space^-
    1)/""
5433                              * parsers.linechar^0 * parsers.newline)^1
5434                            * (-(parsers.leader * parsers.more
5435                                + parsers.blankline) * parsers.linechar^1
5436                              * parsers.newline)^0
5437
5438  if not options.breakableBlockquotes then
5439    larsers.blockquote_body = larsers.blockquote_body
5440                            * (parsers.blankline^0 / "")
5441  end
```

3.1.5.5 Parsers Used for Citations (local)

```
larsers.citations = function(text_cites, raw_cites)
    local function normalize(str)
        if str == "" then
            str = nil
        else
            str = (options.citationNbsps and parse_inlines_nbsp or
              parse_inlines)(str)
        end
        return str
    end

    local cites = {}
    for i = 1,#raw_cites,4 do
        cites[#cites+1] = {
            prenote = normalize(raw_cites[i]),
            suppress_author = raw_cites[i+1] == "-",
            name = writer.citation(raw_cites[i+2]),
            postnote = normalize(raw_cites[i+3]),
        }
    end
    return writer.citations(text_cites, cites)
end
```

3.1.5.6 Parsers Used for Footnotes (local)

```
local rawnotes = {}

-- like indirect_link
local function lookup_note(ref)
  return writer.defer_call(function()
    local found = rawnotes[normalize_tag(ref)]
    if found then
      return writer.note(parse_blocks_nested(found))
    else
      return {"[", parse_inlines("^" .. ref), "]"}
    end
  end)
end

local function register_note(ref,rawnote)
  rawnotes[normalize_tag(ref)] = rawnote
  return ""
end

larsers.NoteRef    = parsers.RawNoteRef / lookup_note

```

```
5485
5486   larsers.NoteBlock  = parsers.leader * parsers.RawNoteRef * parsers.colon
5487                      * parsers.spnl * parsers.indented_blocks(parsers.chunk)
5488                      / register_note
5489
5490   larsers.InlineNote = parsers.circumflex
5491                      * (parsers.tag / parse_inlines_no_inline_note) -- no notes inside n
5492                      / writer.note
```

3.1.5.7 Parsers Used for Tables (local)

```
5493 larsers.table_row = pipe_table_row(true
5494                                    , (C((parsers.linechar - parsers.pipe)^1)
5495                                      / parse_inlines)
5496                                    , parsers.pipe
5497                                    , (C((parsers.linechar - parsers.pipe)^0)
5498                                      / parse_inlines))
5499
5500 if options.tableCaptions then
5501   larsers.table_caption = #parsers.table_caption_beginning
5502                         * parsers.table_caption_beginning
5503                         * Ct(parsers.IndentedInline^1)
5504                         * parsers.newline
5505 else
5506   larsers.table_caption = parsers.fail
5507 end
5508
5509 larsers.PipeTable = Ct(larsers.table_row * parsers.newline
5510                       * parsers.table_hline
5511                       * (parsers.newline * larsers.table_row)^0)
5512                     / make_pipe_table_rectangular
5513                     * larsers.table_caption^-1
5514                     / writer.table
```

3.1.5.8 Helpers for Links and References (local)

```
5515   -- List of references defined in the document
5516   local references
5517
5518   -- add a reference to the list
5519   local function register_link(tag,url,title)
5520       references[normalize_tag(tag)] = { url = url, title = title }
5521       return ""
5522   end
5523
5524   -- lookup link reference and return either
5525   -- the link or nil and fallback text.
5526   local function lookup_reference(label,sps,tag)
```

```
5527      local tagpart
5528      if not tag then
5529          tag = label
5530          tagpart = ""
5531      elseif tag == "" then
5532          tag = label
5533          tagpart = "[]"
5534      else
5535          tagpart = {"[", parse_inlines(tag), "]"}
5536      end
5537      if sps then
5538        tagpart = {sps, tagpart}
5539      end
5540      local r = references[normalize_tag(tag)]
5541      if r then
5542        return r
5543      else
5544        return nil, {"[", parse_inlines(label), "]", tagpart}
5545      end
5546  end
5547
5548  -- lookup link reference and return a link, if the reference is found,
5549  -- or a bracketed label otherwise.
5550  local function indirect_link(label,sps,tag)
5551    return writer.defer_call(function()
5552      local r,fallback = lookup_reference(label,sps,tag)
5553      if r then
5554        return writer.link(parse_inlines_no_link(label), r.url, r.title)
5555      else
5556        return fallback
5557      end
5558    end)
5559  end
5560
5561  -- lookup image reference and return an image, if the reference is found,
5562  -- or a bracketed label otherwise.
5563  local function indirect_image(label,sps,tag)
5564    return writer.defer_call(function()
5565      local r,fallback = lookup_reference(label,sps,tag)
5566      if r then
5567        return writer.image(writer.string(label), r.url, r.title)
5568      else
5569        return {"!", fallback}
5570      end
5571    end)
5572  end
```

3.1.5.9 Inline Elements (local)

```
larsers.Str      = (parsers.normalchar * (parsers.normalchar + parsers.at)^0)
                 / writer.string

larsers.Symbol   = (parsers.specialchar - parsers.tightblocksep)
                 / writer.string

larsers.Ellipsis = P("...") / writer.ellipsis

larsers.Smart    = larsers.Ellipsis

larsers.Code     = parsers.inticks / writer.code

if options.blankBeforeBlockquote then
  larsers.bqstart = parsers.fail
else
  larsers.bqstart = parsers.more
end

if options.blankBeforeHeading then
  larsers.headerstart = parsers.fail
else
  larsers.headerstart = parsers.hash
                      + (parsers.line * (parsers.equal^1 + parsers.dash^1)
                      * parsers.optionalspace * parsers.newline)
end

if not options.fencedCode or options.blankBeforeCodeFence then
  larsers.fencestart = parsers.fail
else
  larsers.fencestart = parsers.fencehead(parsers.backtick)
                     + parsers.fencehead(parsers.tilde)
end

larsers.Endline   = parsers.newline * -( -- newline, but not before...
                      parsers.blankline -- paragraph break
                    + parsers.tightblocksep  -- nested list
                    + parsers.eof       -- end of document
                    + larsers.bqstart
                    + larsers.headerstart
                    + larsers.fencestart
                   ) * parsers.spacechar^0
                   / (options.hardLineBreaks and writer.linebreak
                                             or writer.space)

larsers.OptionalIndent
                    = parsers.spacechar^1 / writer.space
```

```
5619
5620  larsers.Space      = parsers.spacechar^2 * larsers.Endline / writer.linebreak
5621                     + parsers.spacechar^1 * larsers.Endline^-1 * parsers.eof / ""
5622                     + parsers.spacechar^1 * larsers.Endline
5623                                           * parsers.optionalspace
5624                                           / (options.hardLineBreaks
5625                                              and writer.linebreak
5626                                               or writer.space)
5627                     + parsers.spacechar^1 * parsers.optionalspace
5628                                           / writer.space
5629
5630  larsers.NonbreakingEndline
5631                    = parsers.newline * -( -- newline, but not before...
5632                        parsers.blankline -- paragraph break
5633                      + parsers.tightblocksep  -- nested list
5634                      + parsers.eof       -- end of document
5635                      + larsers.bqstart
5636                      + larsers.headerstart
5637                      + larsers.fencestart
5638                    ) * parsers.spacechar^0
5639                    / (options.hardLineBreaks and writer.linebreak
5640                                              or writer.nbsp)
5641
5642  larsers.NonbreakingSpace
5643                  = parsers.spacechar^2 * larsers.Endline / writer.linebreak
5644                  + parsers.spacechar^1 * larsers.Endline^-1 * parsers.eof / ""
5645                  + parsers.spacechar^1 * larsers.Endline
5646                                        * parsers.optionalspace
5647                                        / (options.hardLineBreaks
5648                                           and writer.linebreak
5649                                            or writer.nbsp)
5650                  + parsers.spacechar^1 * parsers.optionalspace
5651                                        / writer.nbsp
5652
5653  if options.underscores then
5654    larsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
5655                                       parsers.doubleasterisks)
5656                     + parsers.between(parsers.Inline, parsers.doubleunderscores,
5657                                       parsers.doubleunderscores)
5658                     ) / writer.strong
5659
5660    larsers.Emph   = ( parsers.between(parsers.Inline, parsers.asterisk,
5661                                       parsers.asterisk)
5662                     + parsers.between(parsers.Inline, parsers.underscore,
5663                                       parsers.underscore)
5664                     ) / writer.emphasis
5665  else
```

```
5666    larsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
5667                                       parsers.doubleasterisks)
5668                     ) / writer.strong
5669
5670    larsers.Emph   = ( parsers.between(parsers.Inline, parsers.asterisk,
5671                                       parsers.asterisk)
5672                     ) / writer.emphasis
5673  end
5674
5675  larsers.AutoLinkUrl    = parsers.less
5676                           * C(parsers.alphanumeric^1 * P("://") * parsers.urlchar^1)
5677                           * parsers.more
5678                           / function(url)
5679                               return writer.link(writer.escape(url), url)
5680                             end
5681
5682  larsers.AutoLinkEmail = parsers.less
5683                          * C((parsers.alphanumeric + S("-._+"))^1
5684                          * P("@") * parsers.urlchar^1)
5685                          * parsers.more
5686                          / function(email)
5687                              return writer.link(writer.escape(email),
5688                                                 "mailto:"..email)
5689                            end
5690
5691  larsers.AutoLinkRelativeReference
5692                           = parsers.less
5693                           * C(parsers.urlchar^1)
5694                           * parsers.more
5695                           / function(url)
5696                               return writer.link(writer.escape(url), url)
5697                             end
5698
5699  larsers.DirectLink    = (parsers.tag / parse_inlines_no_link)  -- no links inside linl
5700                          * parsers.spnl
5701                          * parsers.lparent
5702                          * (parsers.url + Cc(""))  -- link can be empty [foo]()
5703                          * parsers.optionaltitle
5704                          * parsers.rparent
5705                          / writer.link
5706
5707  larsers.IndirectLink  = parsers.tag * (C(parsers.spnl) * parsers.tag)^-
    1
5708                          / indirect_link
5709
5710  -- parse a link or image (direct or indirect)
5711  larsers.Link          = larsers.DirectLink + larsers.IndirectLink
```

```
5712
5713  larsers.DirectImage   = parsers.exclamation
5714                        * (parsers.tag / parse_inlines)
5715                        * parsers.spnl
5716                        * parsers.lparent
5717                        * (parsers.url + Cc(""))  -- link can be empty [foo]()
5718                        * parsers.optionaltitle
5719                        * parsers.rparent
5720                        / writer.image
5721
5722  larsers.IndirectImage = parsers.exclamation * parsers.tag
5723                        * (C(parsers.spnl) * parsers.tag)^-1 / indirect_image
5724
5725  larsers.Image         = larsers.DirectImage + larsers.IndirectImage
5726
5727  larsers.TextCitations = Ct((parsers.spnl
5728                        * Cc("")
5729                        * parsers.citation_name
5730                        * ((parsers.spnl
5731                            * parsers.lbracket
5732                            * parsers.citation_headless_body
5733                            * parsers.rbracket) + Cc("")))^1)
5734                        / function(raw_cites)
5735                            return larsers.citations(true, raw_cites)
5736                          end
5737
5738  larsers.ParenthesizedCitations
5739                        = Ct((parsers.spnl
5740                        * parsers.lbracket
5741                        * parsers.citation_body
5742                        * parsers.rbracket)^1)
5743                        / function(raw_cites)
5744                            return larsers.citations(false, raw_cites)
5745                          end
5746
5747  larsers.Citations     = larsers.TextCitations + larsers.ParenthesizedCitations
5748
5749  -- avoid parsing long strings of * or _ as emph/strong
5750  larsers.UlOrStarLine  = parsers.asterisk^4 + parsers.underscore^4
5751                        / writer.string
5752
5753  larsers.EscapedChar   = parsers.backslash * C(parsers.escapable) / writer.string
5754
5755  larsers.InlineHtml    = parsers.emptyelt_any / writer.inline_html_tag
5756                        + (parsers.htmlcomment / parse_inlines_no_html)
5757                        / writer.inline_html_comment
5758                        + parsers.htmlinstruction
```

```
                        + parsers.openelt_any / writer.inline_html_tag
                        + parsers.closeelt_any / writer.inline_html_tag

larsers.HtmlEntity    = parsers.hexentity / entities.hex_entity  / writer.string
                        + parsers.decentity / entities.dec_entity  / writer.string
                        + parsers.tagentity / entities.char_entity / writer.string
```

3.1.5.10 Block Elements (local)

```
larsers.ContentBlock = parsers.leader
                       * (parsers.localfilepath + parsers.onlineimageurl)
                       * parsers.contentblock_tail
                       / writer.contentblock

larsers.DisplayHtml  = (parsers.htmlcomment / parse_blocks_nested)
                       / writer.block_html_comment
                       + parsers.emptyelt_block / writer.block_html_element
                       + parsers.openelt_exact("hr") / writer.block_html_element
                       + parsers.in_matched_block_tags / writer.block_html_element
                       + parsers.htmlinstruction

larsers.Verbatim     = Cs( (parsers.blanklines
                          * ((parsers.indentedline - parsers.blankline))^1)^1
                          ) / expandtabs / writer.verbatim

larsers.FencedCode   = (parsers.TildeFencedCode
                        + parsers.BacktickFencedCode)
                       / function(infostring, code)
                           return writer.fencedCode(writer.string(infostring),
                                                    expandtabs(code))
                         end

larsers.JekyllData   = Cmt( C((parsers.line - P("---") - P("..."))^0)
                            , function(s, i, text)
                                local data
                                local ran_ok, error = pcall(function()
                                  local tinyyaml = require("markdown-tinyyaml")
                                  data = tinyyaml.parse(text, {timestamps=false})
                                end)
                                if ran_ok and data ~= nil then
                                  return true, writer.jekyllData(data, function(s)
                                    return parse_blocks_nested(s)
                                  end, nil)
                                else
                                  return false
                                end
                              end
```

```
                            )

  larsers.UnexpectedJekyllData
                        = P("---")
                        * parsers.blankline / 0
                        * #(-parsers.blankline)  -- if followed by blank, it's an hrule
                        * larsers.JekyllData
                        * (P("---") + P("..."))

  larsers.ExpectedJekyllData
                        = ( P("---")
                          * parsers.blankline / 0
                          * #(-parsers.blankline)  -- if followed by blank, it's an hrule
                          )^-1
                        * larsers.JekyllData
                        * (P("---") + P("..."))^-1

  larsers.Blockquote    = Cs(larsers.blockquote_body^1)
                        / parse_blocks_nested / writer.blockquote

  larsers.HorizontalRule = ( parsers.lineof(parsers.asterisk)
                           + parsers.lineof(parsers.dash)
                           + parsers.lineof(parsers.underscore)
                           ) / writer.hrule

  larsers.Reference     = parsers.define_reference_parser / register_link

  larsers.Paragraph     = parsers.nonindentspace * Ct(parsers.Inline^1)
                        * ( parsers.newline
                        * ( parsers.blankline^1
                          + #parsers.hash
                          + #(parsers.leader * parsers.more * parsers.space^-
1)
                          + parsers.eof
                          )
                        + parsers.eof )
                        / writer.paragraph

  larsers.Plain         = parsers.nonindentspace * Ct(parsers.Inline^1)
                        / writer.plain
```

3.1.5.11 Lists (local)

```
  larsers.starter = parsers.bullet + larsers.enumerator

  if options.taskLists then
    larsers.tickbox = ( parsers.ticked_box
```

```
                      + parsers.halfticked_box
                      + parsers.unticked_box
                      ) / writer.tickbox
  else
     larsers.tickbox = parsers.fail
  end

  -- we use \001 as a separator between a tight list item and a
  -- nested list under it.
  larsers.NestedList            = Cs((parsers.optionallyindentedline
                                     - larsers.starter)^1)
                                / function(a) return "\001"..a end

  larsers.ListBlockLine         = parsers.optionallyindentedline
                                - parsers.blankline - (parsers.indent^-1
                                                     * larsers.starter)

  larsers.ListBlock             = parsers.line * larsers.ListBlockLine^0

  larsers.ListContinuationBlock = parsers.blanklines * (parsers.indent / "")
                                * larsers.ListBlock

  larsers.TightListItem = function(starter)
      return -larsers.HorizontalRule
             * (Cs(starter / "" * larsers.tickbox^-1 * larsers.ListBlock * larsers.Neste
1)
               / parse_blocks_nested)
             * -(parsers.blanklines * parsers.indent)
  end

  larsers.LooseListItem = function(starter)
      return -larsers.HorizontalRule
             * Cs( starter / "" * larsers.tickbox^-1 * larsers.ListBlock * Cc("\n")
               * (larsers.NestedList + larsers.ListContinuationBlock^0)
               * (parsers.blanklines / "\n\n")
               ) / parse_blocks_nested
  end

  larsers.BulletList = ( Ct(larsers.TightListItem(parsers.bullet)^1) * Cc(true)
                       * parsers.skipblanklines * -parsers.bullet
                       + Ct(larsers.LooseListItem(parsers.bullet)^1) * Cc(false)
                       * parsers.skipblanklines )
                     / writer.bulletlist

  local function ordered_list(items,tight,startNumber)
    if options.startNumber then
      startNumber = tonumber(startNumber) or 1  -- fallback for '#'
```

```
5892      if startNumber ~= nil then
5893        startNumber = math.floor(startNumber)
5894      end
5895    else
5896      startNumber = nil
5897    end
5898    return writer.orderedlist(items,tight,startNumber)
5899  end
5900
5901  larsers.OrderedList = Cg(larsers.enumerator, "listtype") *
5902                        ( Ct(larsers.TightListItem(Cb("listtype"))
5903                            * larsers.TightListItem(larsers.enumerator)^0)
5904                        * Cc(true) * parsers.skipblanklines * -larsers.enumerator
5905                        + Ct(larsers.LooseListItem(Cb("listtype"))
5906                            * larsers.LooseListItem(larsers.enumerator)^0)
5907                        * Cc(false) * parsers.skipblanklines
5908                        ) * Cb("listtype") / ordered_list
5909
5910  local function definition_list_item(term, defs, tight)
5911    return { term = parse_inlines(term), definitions = defs }
5912  end
5913
5914  larsers.DefinitionListItemLoose = C(parsers.line) * parsers.skipblanklines
5915                                  * Ct((parsers.defstart
5916                                       * parsers.indented_blocks(parsers.dlchunk)
5917                                       / parse_blocks_nested)^1)
5918                                  * Cc(false) / definition_list_item
5919
5920  larsers.DefinitionListItemTight = C(parsers.line)
5921                                  * Ct((parsers.defstart * parsers.dlchunk
5922                                       / parse_blocks_nested)^1)
5923                                  * Cc(true) / definition_list_item
5924
5925  larsers.DefinitionList = ( Ct(larsers.DefinitionListItemLoose^1) * Cc(false)
5926                           + Ct(larsers.DefinitionListItemTight^1)
5927                           * (parsers.skipblanklines
5928                             * -larsers.DefinitionListItemLoose * Cc(true))
5929                           ) / writer.definitionlist
```

3.1.5.12 Blank (local)

```
5930  larsers.Blank        = parsers.blankline / ""
5931                       + larsers.NoteBlock
5932                       + larsers.Reference
5933                       + (parsers.tightblocksep / "\n")
```

3.1.5.13 Headings (local)

```
5934  -- parse atx header
5935  if options.headerAttributes then
5936    larsers.AtxHeading = Cg(parsers.HeadingStart,"level")
5937                         * parsers.optionalspace
5938                         * (C(((parsers.linechar
5939                               - ((parsers.hash^1
5940                                   * parsers.optionalspace
5941                                   * parsers.HeadingAttributes^-1
5942                                   + parsers.HeadingAttributes)
5943                                  * parsers.optionalspace
5944                                  * parsers.newline))
5945                              * (parsers.linechar
5946                                - parsers.hash
5947                                - parsers.lbrace)^0)^1)
5948                             / parse_inlines)
5949                         * Cg(Ct(parsers.newline
5950                                 + (parsers.hash^1
5951                                   * parsers.optionalspace
5952                                   * parsers.HeadingAttributes^-1
5953                                   + parsers.HeadingAttributes)
5954                                 * parsers.optionalspace
5955                                 * parsers.newline), "attributes")
5956                         * Cb("level")
5957                         * Cb("attributes")
5958                         / writer.heading
5959
5960    larsers.SetextHeading = #(parsers.line * S("=-"))
5961                            * (C(((parsers.linechar
5962                                   - (parsers.HeadingAttributes
5963                                     * parsers.optionalspace
5964                                     * parsers.newline))
5965                                  * (parsers.linechar
5966                                    - parsers.lbrace)^0)^1)
5967                                / parse_inlines)
5968                            * Cg(Ct(parsers.newline
5969                                    + (parsers.HeadingAttributes
5970                                      * parsers.optionalspace
5971                                      * parsers.newline)), "attributes")
5972                            * parsers.HeadingLevel
5973                            * Cb("attributes")
5974                            * parsers.optionalspace
5975                            * parsers.newline
5976                            / writer.heading
5977  else
5978    larsers.AtxHeading = Cg(parsers.HeadingStart,"level")
5979                         * parsers.optionalspace
5980                         * (C(parsers.line) / strip_atx_end / parse_inlines)
```

```
                        * Cb("level")
                        / writer.heading

  larsers.SetextHeading = #(parsers.line * S("=-"))
                          * Ct(parsers.linechar^1 / parse_inlines)
                          * parsers.newline
                          * parsers.HeadingLevel
                          * parsers.optionalspace
                          * parsers.newline
                          / writer.heading
end

larsers.Heading = larsers.AtxHeading + larsers.SetextHeading
```

3.1.5.14 Syntax Specification

```
local syntax =
  { "Blocks",

    Blocks                = ( V("ExpectedJekyllData")
                            * (V("Blank")^0 / writer.interblocksep)
                            )^-1
                          * V("Blank")^0
                          * V("Block")^-1
                          * (V("Blank")^0 / writer.interblocksep
                            * V("Block"))^0
                          * V("Blank")^0 * parsers.eof,

    Blank                 = larsers.Blank,

    UnexpectedJekyllData  = larsers.UnexpectedJekyllData,
    ExpectedJekyllData    = larsers.ExpectedJekyllData,

    Block                 = V("ContentBlock")
                          + V("UnexpectedJekyllData")
                          + V("Blockquote")
                          + V("PipeTable")
                          + V("Verbatim")
                          + V("FencedCode")
                          + V("HorizontalRule")
                          + V("BulletList")
                          + V("OrderedList")
                          + V("Heading")
                          + V("DefinitionList")
                          + V("DisplayHtml")
                          + V("Paragraph")
                          + V("Plain"),
```

```
6025
6026        ContentBlock          = larsers.ContentBlock,
6027        Blockquote            = larsers.Blockquote,
6028        Verbatim              = larsers.Verbatim,
6029        FencedCode            = larsers.FencedCode,
6030        HorizontalRule        = larsers.HorizontalRule,
6031        BulletList            = larsers.BulletList,
6032        OrderedList           = larsers.OrderedList,
6033        Heading               = larsers.Heading,
6034        DefinitionList        = larsers.DefinitionList,
6035        DisplayHtml           = larsers.DisplayHtml,
6036        Paragraph             = larsers.Paragraph,
6037        PipeTable             = larsers.PipeTable,
6038        Plain                 = larsers.Plain,
6039
6040        Inline                = V("Str")
6041                              + V("Space")
6042                              + V("Endline")
6043                              + V("UlOrStarLine")
6044                              + V("Strong")
6045                              + V("Emph")
6046                              + V("InlineNote")
6047                              + V("NoteRef")
6048                              + V("Citations")
6049                              + V("Link")
6050                              + V("Image")
6051                              + V("Code")
6052                              + V("AutoLinkUrl")
6053                              + V("AutoLinkEmail")
6054                              + V("AutoLinkRelativeReference")
6055                              + V("InlineHtml")
6056                              + V("HtmlEntity")
6057                              + V("EscapedChar")
6058                              + V("Smart")
6059                              + V("Symbol"),
6060
6061        IndentedInline        = V("Str")
6062                              + V("OptionalIndent")
6063                              + V("Endline")
6064                              + V("UlOrStarLine")
6065                              + V("Strong")
6066                              + V("Emph")
6067                              + V("InlineNote")
6068                              + V("NoteRef")
6069                              + V("Citations")
6070                              + V("Link")
6071                              + V("Image")
```

```
                            + V("Code")
                            + V("AutoLinkUrl")
                            + V("AutoLinkEmail")
                            + V("AutoLinkRelativeReference")
                            + V("InlineHtml")
                            + V("HtmlEntity")
                            + V("EscapedChar")
                            + V("Smart")
                            + V("Symbol"),

    Str                     = larsers.Str,
    Space                   = larsers.Space,
    OptionalIndent          = larsers.OptionalIndent,
    Endline                 = larsers.Endline,
    UlOrStarLine            = larsers.UlOrStarLine,
    Strong                  = larsers.Strong,
    Emph                    = larsers.Emph,
    InlineNote              = larsers.InlineNote,
    NoteRef                 = larsers.NoteRef,
    Citations               = larsers.Citations,
    Link                    = larsers.Link,
    Image                   = larsers.Image,
    Code                    = larsers.Code,
    AutoLinkUrl             = larsers.AutoLinkUrl,
    AutoLinkEmail           = larsers.AutoLinkEmail,
    AutoLinkRelativeReference
                            = larsers.AutoLinkRelativeReference,
    InlineHtml              = larsers.InlineHtml,
    HtmlEntity              = larsers.HtmlEntity,
    EscapedChar             = larsers.EscapedChar,
    Smart                   = larsers.Smart,
    Symbol                  = larsers.Symbol,
  }

if not options.citations then
  syntax.Citations = parsers.fail
end

if not options.contentBlocks then
  syntax.ContentBlock = parsers.fail
end

if not options.codeSpans then
  syntax.Code = parsers.fail
end

if not options.definitionLists then
```

```
6119    syntax.DefinitionList = parsers.fail
6120  end
6121
6122  if not options.fencedCode then
6123    syntax.FencedCode = parsers.fail
6124  end
6125
6126  if not options.footnotes then
6127    syntax.NoteRef = parsers.fail
6128  end
6129
6130  if not options.html then
6131    syntax.DisplayHtml = parsers.fail
6132    syntax.InlineHtml = parsers.fail
6133    syntax.HtmlEntity  = parsers.fail
6134  end
6135
6136  if not options.inlineFootnotes then
6137    syntax.InlineNote = parsers.fail
6138  end
6139
6140  if not options.jekyllData then
6141    syntax.UnexpectedJekyllData = parsers.fail
6142  end
6143
6144  if not options.jekyllData or not options.expectJekyllData then
6145    syntax.ExpectedJekyllData = parsers.fail
6146  end
6147
6148  if options.preserveTabs then
6149    options.stripIndent = false
6150  end
6151
6152  if not options.pipeTables then
6153    syntax.PipeTable = parsers.fail
6154  end
6155
6156  if not options.smartEllipses then
6157    syntax.Smart = parsers.fail
6158  end
6159
6160  if not options.relativeReferences then
6161    syntax.AutoLinkRelativeReference = parsers.fail
6162  end
6163
6164  local blocks_nested_t = util.table_copy(syntax)
6165  blocks_nested_t.ExpectedJekyllData = parsers.fail
```

```
6166  larsers.blocks_nested = Ct(blocks_nested_t)
6167
6168  larsers.blocks = Ct(syntax)
6169
6170  local inlines_t = util.table_copy(syntax)
6171  inlines_t[1] = "Inlines"
6172  inlines_t.Inlines = parsers.Inline^0 * (parsers.spacing^0 * parsers.eof / "")
6173  larsers.inlines = Ct(inlines_t)
6174
6175  local inlines_no_link_t = util.table_copy(inlines_t)
6176  inlines_no_link_t.Link = parsers.fail
6177  larsers.inlines_no_link = Ct(inlines_no_link_t)
6178
6179  local inlines_no_inline_note_t = util.table_copy(inlines_t)
6180  inlines_no_inline_note_t.InlineNote = parsers.fail
6181  larsers.inlines_no_inline_note = Ct(inlines_no_inline_note_t)
6182
6183  local inlines_no_html_t = util.table_copy(inlines_t)
6184  inlines_no_html_t.DisplayHtml = parsers.fail
6185  inlines_no_html_t.InlineHtml = parsers.fail
6186  inlines_no_html_t.HtmlEntity = parsers.fail
6187  larsers.inlines_no_html = Ct(inlines_no_html_t)
6188
6189  local inlines_nbsp_t = util.table_copy(inlines_t)
6190  inlines_nbsp_t.Endline = larsers.NonbreakingEndline
6191  inlines_nbsp_t.Space = larsers.NonbreakingSpace
6192  larsers.inlines_nbsp = Ct(inlines_nbsp_t)
```

3.1.5.15 Exported Conversion Function

Define `reader->convert` as a function that converts markdown string `input` into a plain TEX output and returns it. Note that the converter assumes that the input has UNIX line endings.

```
6193  function self.convert(input)
6194    references = {}
```

When determining the name of the cache file, create salt for the hashing function out of the package version and the passed options recognized by the Lua interface (see Section 2.1.2). The `cacheDir` option is disregarded.

```
6195    local opt_string = {}
6196    for k,_ in pairs(defaultOptions) do
6197      local v = options[k]
6198      if k ~= "cacheDir" then
6199        opt_string[#opt_string+1] = k .. "=" .. tostring(v)
6200      end
6201    end
6202    table.sort(opt_string)
6203    local salt = table.concat(opt_string, ",") .. "," .. metadata.version
```

```
    local output
```

If we cache markdown documents, produce the cache file and transform its filename to plain TEX output via the `writer->pack` method.

```
    local function convert(input)
      local document = parse_blocks(input)
      return util.rope_to_string(writer.document(document))
    end
    if options.eagerCache or options.finalizeCache then
      local name = util.cache(options.cacheDir, input, salt, convert, ".md" .. writer.su
      output = writer.pack(name)
```

Otherwise, return the result of the conversion directly.

```
    else
      output = convert(input)
    end
```

If the `finalizeCache` option is enabled, populate the frozen cache in the file `frozenCacheFileName` with an entry for markdown document number `frozenCacheCounter`.

```
    if options.finalizeCache then
      local file, mode
      if options.frozenCacheCounter > 0 then
        mode = "a"
      else
        mode = "w"
      end
      file = assert(io.open(options.frozenCacheFileName, mode),
        [[could not open file "]] .. options.frozenCacheFileName
        .. [[" for writing]])
      assert(file:write([[\expandafter\global\expandafter\def\csname ]]
        .. [[markdownFrozenCache]] .. options.frozenCacheCounter
        .. [[\endcsname{]] .. output .. [[}]] .. "\n"))
      assert(file:close())
    end
    return output
  end
  return self
end
```

3.1.6 Conversion from Markdown to Plain TEX

The `new` method returns the `reader->convert` function of a reader object associated with the Lua interface options (see Section 2.1.2) `options` and with a writer object associated with `options`.

```
function M.new(options)
  local writer = M.writer.new(options)
```

```
  local reader = M.reader.new(writer, options)
  return reader.convert
end

return M
```

3.1.7 Command-Line Implementation

The command-line implementation provides the actual conversion routine for the command-line interface described in Section 2.1.5.

```

local input
if input_filename then
  local input_file = assert(io.open(input_filename, "r"),
    [[could not open file "]] .. input_filename .. [[" for reading]])
  input = assert(input_file:read("*a"))
  assert(input_file:close())
else
  input = assert(io.read("*a"))
end

```

First, ensure that the `options.cacheDir` directory exists.

```
local lfs = require("lfs")
if options.cacheDir and not lfs.isdir(options.cacheDir) then
  assert(lfs.mkdir(options["cacheDir"]))
end

local ran_ok, kpse = pcall(require, "kpse")
if ran_ok then kpse.set_program_name("luatex") end
local md = require("markdown")
```

Since we are loading the rest of the Lua implementation dynamically, check that both the `markdown` module and the command line implementation are the same version.

```
if metadata.version ~= md.metadata.version then
  warn("markdown-cli.lua " .. metadata.version .. " used with " ..
       "markdown.lua " .. md.metadata.version .. ".")
end
local convert = md.new(options)
```

Since the Lua converter expects UNIX line endings, normalize the input. Also add a line ending at the end of the file in case the input file has none.

```
local output = convert(input:gsub("\r\n?", "\n") .. "\n")

if output_filename then
  local output_file = assert(io.open(output_filename, "w"),
    [[could not open file "]] .. output_filename .. [[" for writing]])
  assert(output_file:write(output))
```

```
  assert(output_file:close())
else
  assert(io.write(output))
end
```

3.2 Plain TEX Implementation

The plain TEX implementation provides macros for the interfacing between TEX and Lua and for the buffering of input text. These macros are then used to implement the macros for the conversion from markdown to plain TEX exposed by the plain TEX interface (see Section 2.2).

3.2.1 Logging Facilities

```
\ifx\markdownInfo\undefined
  \def\markdownInfo#1{%
    \immediate\write-1{(l.\the\inputlineno) markdown.tex info: #1.}}%
\fi
\ifx\markdownWarning\undefined
  \def\markdownWarning#1{%
    \immediate\write16{(l.\the\inputlineno) markdown.tex warning: #1}}%
\fi
\ifx\markdownError\undefined
  \def\markdownError#1#2{%
    \errhelp{#2.}%
    \errmessage{(l.\the\inputlineno) markdown.tex error: #1}}%
\fi
```

3.2.2 Finalizing and Freezing the Cache

When the `\markdownOptionFinalizeCache` option is enabled, then the `\markdownFrozenCacheCounter` counter is used to enumerate the markdown documents using the Lua interface `frozenCacheCounter` option.

When the `\markdownOptionFrozenCache` option is enabled, then the `\markdownFrozenCacheCounter` counter is used to render markdown documents from the frozen cache without invoking Lua.

```
\newcount\markdownFrozenCacheCounter
```

3.2.3 Token Renderer Prototypes

The following definitions should be considered placeholder.

```
\def\markdownRendererInterblockSeparatorPrototype{\par}%
\def\markdownRendererLineBreakPrototype{\hfil\break}%
\let\markdownRendererEllipsisPrototype\dots
\def\markdownRendererNbspPrototype{~}%
\def\markdownRendererLeftBracePrototype{\char`\{}%
```

```
\def\markdownRendererRightBracePrototype{\char`\}}%
\def\markdownRendererDollarSignPrototype{\char`$}%
\def\markdownRendererPercentSignPrototype{\char`\%}%
\def\markdownRendererAmpersandPrototype{\&}%
\def\markdownRendererUnderscorePrototype{\char`_}%
\def\markdownRendererHashPrototype{\char`\#}%
\def\markdownRendererCircumflexPrototype{\char`^}%
\def\markdownRendererBackslashPrototype{\char`\\}%
\def\markdownRendererTildePrototype{\char`~}%
\def\markdownRendererPipePrototype{|}%
\def\markdownRendererCodeSpanPrototype#1{{\tt#1}}%
\def\markdownRendererLinkPrototype#1#2#3#4{#2}%
\def\markdownRendererContentBlockPrototype#1#2#3#4{%
  \markdownInput{#3}}%
\def\markdownRendererContentBlockOnlineImagePrototype{%
  \markdownRendererImage}%
\def\markdownRendererContentBlockCodePrototype#1#2#3#4#5{%
  \markdownRendererInputFencedCode{#3}{#2}}%
\def\markdownRendererImagePrototype#1#2#3#4{#2}%
\def\markdownRendererUlBeginPrototype{}%
\def\markdownRendererUlBeginTightPrototype{}%
\def\markdownRendererUlItemPrototype{}%
\def\markdownRendererUlItemEndPrototype{}%
\def\markdownRendererUlEndPrototype{}%
\def\markdownRendererUlEndTightPrototype{}%
\def\markdownRendererOlBeginPrototype{}%
\def\markdownRendererOlBeginTightPrototype{}%
\def\markdownRendererOlItemPrototype{}%
\def\markdownRendererOlItemWithNumberPrototype#1{}%
\def\markdownRendererOlItemEndPrototype{}%
\def\markdownRendererOlEndPrototype{}%
\def\markdownRendererOlEndTightPrototype{}%
\def\markdownRendererDlBeginPrototype{}%
\def\markdownRendererDlBeginTightPrototype{}%
\def\markdownRendererDlItemPrototype#1{#1}%
\def\markdownRendererDlItemEndPrototype{}%
\def\markdownRendererDlDefinitionBeginPrototype{}%
\def\markdownRendererDlDefinitionEndPrototype{\par}%
\def\markdownRendererDlEndPrototype{}%
\def\markdownRendererDlEndTightPrototype{}%
\def\markdownRendererEmphasisPrototype#1{{\it#1}}%
\def\markdownRendererStrongEmphasisPrototype#1{{\bf#1}}%
\def\markdownRendererBlockQuoteBeginPrototype{\par\begingroup\it}%
\def\markdownRendererBlockQuoteEndPrototype{\endgroup\par}%
\def\markdownRendererInputVerbatimPrototype#1{%
  \par{\tt\input#1\relax{}}\par}%
\def\markdownRendererInputFencedCodePrototype#1#2{%
```

```
  \markdownRendererInputVerbatimPrototype{#1}}%
\def\markdownRendererHeadingOnePrototype#1{#1}%
\def\markdownRendererHeadingTwoPrototype#1{#1}%
\def\markdownRendererHeadingThreePrototype#1{#1}%
\def\markdownRendererHeadingFourPrototype#1{#1}%
\def\markdownRendererHeadingFivePrototype#1{#1}%
\def\markdownRendererHeadingSixPrototype#1{#1}%
\def\markdownRendererHorizontalRulePrototype{}%
\def\markdownRendererFootnotePrototype#1{#1}%
\def\markdownRendererCitePrototype#1{}%
\def\markdownRendererTextCitePrototype#1{}%
\def\markdownRendererTickedBoxPrototype{[X]}%
\def\markdownRendererHalfTickedBoxPrototype{[/]}%
\def\markdownRendererUntickedBoxPrototype{[ ]}%
```

3.2.3.1 YAML Metadata Renderer Prototypes

To keep track of the current type of structure we inhabit when we are traversing a YAML document, we will maintain the `\g_@@_jekyll_data_datatypes_seq` stack. At every step of the traversal, the stack will contain one of the following constants at any position p:

`\c_@@_jekyll_data_sequence_tl` The currently traversed branch of the YAML document contains a sequence at depth p.

`\c_@@_jekyll_data_mapping_tl` The currently traversed branch of the YAML document contains a mapping at depth p.

`\c_@@_jekyll_data_scalar_tl` The currently traversed branch of the YAML document contains a scalar value at depth p.

```
\ExplSyntaxOn
\seq_new:N   \g_@@_jekyll_data_datatypes_seq
\tl_const:Nn \c_@@_jekyll_data_sequence_tl   { sequence }
\tl_const:Nn \c_@@_jekyll_data_mapping_tl    { mapping  }
\tl_const:Nn \c_@@_jekyll_data_scalar_tl     { scalar   }
```

To keep track of our current place when we are traversing a YAML document, we will maintain the `\g_@@_jekyll_data_wildcard_absolute_address_seq` stack of keys using the `\markdown_jekyll_data_push_address_segment:n` macro.

```
\seq_new:N \g_@@_jekyll_data_wildcard_absolute_address_seq
\cs_new:Nn \markdown_jekyll_data_push_address_segment:n
  {
    \seq_if_empty:NF
      \g_@@_jekyll_data_datatypes_seq
      {
        \seq_get_right:NN
          \g_@@_jekyll_data_datatypes_seq
          \l_tmpa_tl
```

If we are currently in a sequence, we will put an asterisk (`*`) instead of a key into `\g_@@_jekyll_data_wildcard_absolute_address_seq` to make it represent a *wildcard*. Keeping a wildcard instead of a precise address makes it easy for the users to react to *any* item of a sequence regardless of how many there are, which can often be useful.

```
      \tl_if_eq:NNTF
        \l_tmpa_tl
        \c_@@_jekyll_data_sequence_tl
        {
          \seq_put_right:Nn
            \g_@@_jekyll_data_wildcard_absolute_address_seq
            { * }
        }
        {
          \seq_put_right:Nn
            \g_@@_jekyll_data_wildcard_absolute_address_seq
            { #1 }
        }
      }
  }
```

Out of `\g_@@_jekyll_data_wildcard_absolute_address_seq`, we will construct the following two token lists:

`\g_@@_jekyll_data_wildcard_absolute_address_tl` An *absolute wildcard*: The wildcard from the root of the document prefixed with a slash (`/`) with individual keys and asterisks also delimited by slashes. Allows the users to react to complex context-sensitive structures with ease.

For example, the `name` key in the following YAML document would correspond to the `/*/person/name` absolute wildcard:

```
[{person: {name: Elon, surname: Musk}}]
```

`\g_@@_jekyll_data_wildcard_relative_address_tl` A *relative wildcard*: The rightmost segment of the wildcard. Allows the users to react to simple context-free structures.

For example, the `name` key in the following YAML document would correspond to the `name` relative wildcard:

```
[{person: {name: Elon, surname: Musk}}]
```

We will construct `\g_@@_jekyll_data_wildcard_absolute_address_tl` using the `\markdown_jekyll_data_concatenate_address:NN` macro and we will construct both token lists using the `\markdown_jekyll_data_update_address_tls:` macro.

```
\tl_new:N  \g_@@_jekyll_data_wildcard_absolute_address_tl
\tl_new:N  \g_@@_jekyll_data_wildcard_relative_address_tl
\cs_new:Nn \markdown_jekyll_data_concatenate_address:NN
  {
    \seq_pop_left:NN #1 \l_tmpa_tl
    \tl_set:Nx #2 { / \seq_use:Nn #1 { / } }
    \seq_put_left:NV #1 \l_tmpa_tl
  }
\cs_new:Nn \markdown_jekyll_data_update_address_tls:
  {
    \markdown_jekyll_data_concatenate_address:NN
      \g_@@_jekyll_data_wildcard_absolute_address_seq
      \g_@@_jekyll_data_wildcard_absolute_address_tl
    \seq_get_right:NN
      \g_@@_jekyll_data_wildcard_absolute_address_seq
      \g_@@_jekyll_data_wildcard_relative_address_tl
  }
```

To make sure that the stacks and token lists stay in sync, we will use the `\markdown_jekyll_data_push:nN` and `\markdown_jekyll_data_pop:` macros.

```
\cs_new:Nn \markdown_jekyll_data_push:nN
  {
    \markdown_jekyll_data_push_address_segment:n
      { #1 }
    \seq_put_right:NV
     \g_@@_jekyll_data_datatypes_seq
     #2
    \markdown_jekyll_data_update_address_tls:
  }
\cs_new:Nn \markdown_jekyll_data_pop:
  {
    \seq_pop_right:NN
      \g_@@_jekyll_data_wildcard_absolute_address_seq
      \l_tmpa_tl
    \seq_pop_right:NN
      \g_@@_jekyll_data_datatypes_seq
      \l_tmpa_tl
    \markdown_jekyll_data_update_address_tls:
  }
```

To set a single key–value, we will use the `\markdown_jekyll_data_set_keyval:Nn` macro, ignoring unknown keys. To set key–values for both absolute and relative wildcards, we will use the `\markdown_jekyll_data_set_keyvals:nn` macro.

```
\cs_new:Nn \markdown_jekyll_data_set_keyval:nn
  {
    \keys_set_known:nn
      { markdown/jekyllData }
      { { #1 } = { #2 } }
```

```
  }
\cs_generate_variant:Nn
  \markdown_jekyll_data_set_keyval:nn
  { Vn }
\cs_new:Nn \markdown_jekyll_data_set_keyvals:nn
  {
    \markdown_jekyll_data_push:nN
      { #1 }
      \c_@@_jekyll_data_scalar_tl
    \markdown_jekyll_data_set_keyval:Vn
      \g_@@_jekyll_data_wildcard_absolute_address_tl
      { #2 }
    \markdown_jekyll_data_set_keyval:Vn
      \g_@@_jekyll_data_wildcard_relative_address_tl
      { #2 }
    \markdown_jekyll_data_pop:
  }
```

Finally, we will register our macros as token renderer prototypes to be able to react to the traversal of a YAML document.

```
\def\markdownRendererJekyllDataSequenceBeginPrototype#1#2{
  \markdown_jekyll_data_push:nN
    { #1 }
    \c_@@_jekyll_data_sequence_tl
}
\def\markdownRendererJekyllDataMappingBeginPrototype#1#2{
  \markdown_jekyll_data_push:nN
    { #1 }
    \c_@@_jekyll_data_mapping_tl
}
\def\markdownRendererJekyllDataSequenceEndPrototype{
  \markdown_jekyll_data_pop:
}
\def\markdownRendererJekyllDataMappingEndPrototype{
  \markdown_jekyll_data_pop:
}
\def\markdownRendererJekyllDataBooleanPrototype#1#2{
  \markdown_jekyll_data_set_keyvals:nn
    { #1 }
    { #2 }
}
\def\markdownRendererJekyllDataEmptyPrototype#1{}
\def\markdownRendererJekyllDataNumberPrototype#1#2{
  \markdown_jekyll_data_set_keyvals:nn
    { #1 }
    { #2 }
}
```

```
\def\markdownRendererJekyllDataStringPrototype#1#2{
  \markdown_jekyll_data_set_keyvals:nn
    { #1 }
    { #2 }
}
\ExplSyntaxOff
```

3.2.4 Lua Snippets

After the \markdownPrepareLuaOptions macro has been fully expanded, the \markdownLuaOptions macro will expands to a Lua table that contains the plain TEX options (see Section 2.2.2) in a format recognized by Lua (see Section 2.1.2).

```
\ExplSyntaxOn
\tl_new:N \g_@@_formatted_lua_options_tl
\tl_const:Nn \c_@@_lua_option_type_boolean { boolean }
\tl_const:Nn \c_@@_lua_option_type_counter { counter }
\tl_const:Nn \c_@@_lua_option_type_number  { number  }
\tl_const:Nn \c_@@_lua_option_type_string  { string  }
\cs_new:Nn \@@_format_lua_options:
  {
    \tl_gclear:N
      \g_@@_formatted_lua_options_tl
    \seq_map_function:NN
      \g_@@_lua_options_seq
      \@@_format_lua_option:n
  }
\cs_new:Nn \@@_format_lua_option:n
  {
    \@@_typecheck_lua_option:n { #1 }
    \tl_set:Nn
      \l_tmpb_tl
%     TODO: Replace with \str_uppercase:n in TeX Live 2020.
      { \str_upper_case:n { #1 } }
    \tl_set:Nx
      \l_tmpa_tl
      {
        markdownOption
        \tl_head:f { \l_tmpb_tl }
        \tl_tail:n { #1 }
      }
    \prop_get:NnN
      \g_@@_lua_option_types_prop
      { #1 }
      \l_tmpb_tl
    \cs_if_free:cTF
      { \l_tmpa_tl }
```

```
      { }
      {
        \tl_case:NnF
          \l_tmpb_tl
          {
            \c_@@_lua_option_type_string
              {
                \tl_gput_right:Nx
                  \g_@@_formatted_lua_options_tl
                  { #1~=~"     \cs:w \l_tmpa_tl \cs_end: ",~ }
              }
            \c_@@_lua_option_type_counter
              {
                \tl_gput_right:Nx
                  \g_@@_formatted_lua_options_tl
                  { #1~=~ \the \cs:w \l_tmpa_tl \cs_end: ,~ }
              }
          }
          {
            \tl_gput_right:Nx
              \g_@@_formatted_lua_options_tl
              {     #1~=~      \cs:w \l_tmpa_tl \cs_end: ,~ }
          }
      }
  }
\msg_new:nnn
  { markdown }
  { undefined-lua-option }
  {
    Lua~option~#1~is~undefined.
  }
\msg_new:nnn
  { markdown }
  { failed-typecheck-for-boolean-lua-option }
  {
    Lua~option~#1~has~value~#2,~
    but~a~boolean~(true~or~false)~was~expected.
  }
\cs_new:Nn \@@_typecheck_lua_option:n
  {
    \tl_set:Nn
      \l_tmpb_tl
%     TODO: Replace with \str_uppercase:n in TeX Live 2020.
      { \str_upper_case:n { #1 } }
    \tl_set:Nx
      \l_tmpa_tl
      {
```

```
        markdownOption
        \tl_head:f { \l_tmpb_tl }
        \tl_tail:n { #1 }
      }
    \prop_get:NnNTF
      \g_@@_lua_option_types_prop
      { #1 }
      \l_tmpb_tl
      {
        \cs_if_free:cTF
          { \l_tmpa_tl }
          { }
          {
            \tl_case:Nn
              \l_tmpb_tl
              {
                \c_@@_lua_option_type_boolean
                  {
                    \tl_case:cnF
                      { \l_tmpa_tl }
                      {
                        \c_@@_lua_option_value_true  { }
                        \c_@@_lua_option_value_false { }
                      }
                      {
                        \msg_error:nnxx
                          { markdown }
                          { failed-typecheck-for-boolean-lua-option }
                          { #1 }
                          { \l_tmpa_tl }
                      }
                  }
              }
          }
      }
      {
        \msg_error:nnn
          { markdown }
          { undefined-lua-option }
          { #1 }
      }
  }
\let\markdownPrepareLuaOptions=\@@_format_lua_options:
\def\markdownLuaOptions{{ \g_@@_formatted_lua_options_tl }}
\ExplSyntaxOff
```

The \markdownPrepare macro contains the Lua code that is executed prior to

any conversion from markdown to plain TEX. It exposes the `convert` function for the use by any further Lua code.

```
\def\markdownPrepare{%
```

First, ensure that the `\markdownOptionCacheDir` directory exists.

```
  local lfs = require("lfs")
  local cacheDir = "\markdownOptionCacheDir"
  if not lfs.isdir(cacheDir) then
    assert(lfs.mkdir(cacheDir))
  end
```

Next, load the `markdown` module and create a converter function using the plain TEX options, which were serialized to a Lua table via the `\markdownLuaOptions` macro.

```
  local md = require("markdown")
  local convert = md.new(\markdownLuaOptions)
}%
```

3.2.5 Buffering Markdown Input

The `\markdownIfOption{`⟨*name*⟩`}{`⟨*iftrue*⟩`}{`⟨*iffalse*⟩`}` macro is provided for testing, whether the value of `\markdownOption`⟨*name*⟩ is `true`. If the value is `true`, then ⟨*iftrue*⟩ is expanded, otherwise ⟨*iffalse*⟩ is expanded.

```
\ExplSyntaxOn
\tl_const:Nn \c_@@_lua_option_value_true  { true  }
\tl_const:Nn \c_@@_lua_option_value_false { false }
\cs_new:Nn \@@_if_option:nTF
  {
    \tl_set:Nn
      \l_tmpb_tl
%     TODO: Replace with \str_uppercase:n in TeX Live 2020.
      { \str_upper_case:n { #1 } }
    \tl_set:Nx
      \l_tmpa_tl
      {
        markdownOption
        \tl_head:f { \l_tmpb_tl }
        \tl_tail:n { #1 }
      }
    \cs_if_free:cTF
      { \l_tmpa_tl }
      {
        \prop_get:NnN
          \g_@@_default_lua_options_prop
          { #1 }
          \l_tmpb_tl
      }
      {
```

```
        \tl_set:Nf
          \l_tmpb_tl
          { \cs:w \l_tmpa_tl \cs_end: }
      }
    \tl_if_eq:NNTF
      \l_tmpb_tl
      \c_@@_lua_option_value_true
      { #2 }
      { #3 }
  }
\let\markdownIfOption=\@@_if_option:nTF
\ExplSyntaxOff
```

The macros `\markdownInputFileStream` and `\markdownOutputFileStream` contain the number of the input and output file streams that will be used for the IO operations of the package.

```
\csname newread\endcsname\markdownInputFileStream
\csname newwrite\endcsname\markdownOutputFileStream
```

The `\markdownReadAndConvertTab` macro contains the tab character literal.

```
\begingroup
  \catcode`\^^I=12%
  \gdef\markdownReadAndConvertTab{^^I}%
\endgroup
```

The `\markdownReadAndConvert` macro is largely a rewrite of the LaTeX 2_ε `\filecontents` macro to plain TeX.

```
\begingroup
```

Make the newline and tab characters active and swap the character codes of the backslash symbol (`\`) and the pipe symbol (`|`), so that we can use the backslash as an ordinary character inside the macro definition. Likewise, swap the character codes of the percent sign (`%`) and the ampersand (`@`), so that we can remove percent signs from the beginning of lines when `\markdownOptionStripPercentSigns` is enabled.

```
  \catcode`\^^M=13%
  \catcode`\^^I=13%
  \catcode`|=0%
  \catcode`\\=12%
  |catcode`@=14%
  |catcode`|%=12@
  |gdef|markdownReadAndConvert#1#2{@
    |begingroup@
```

If we are not reading markdown documents from the frozen cache, open the `\markdownOptionInputTempFileName` file for writing.

```
    |markdownIfOption{frozenCache}{}{@
      |immediate|openout|markdownOutputFileStream@
        |markdownOptionInputTempFileName|relax@
```

```
      |markdownInfo{Buffering markdown input into the temporary @
        input file "|markdownOptionInputTempFileName" and scanning @
        for the closing token sequence "#1"}@
    }@
```

Locally change the category of the special plain TEX characters to *other* in order to prevent unwanted interpretation of the input. Change also the category of the space character, so that we can retrieve it unaltered.

```
    |def|do##1{|catcode`##1=12}|dospecials@
    |catcode`| =12@
    |markdownMakeOther@
```

The `\markdownReadAndConvertStripPercentSigns` macro will process the individual lines of output, stipping away leading percent signs (`%`) when `\markdownOptionStripPercentSigns` is enabled. Notice the use of the comments (`@`) to ensure that the entire macro is at a single line and therefore no (active) newline symbols (`^^M`) are produced.

```
    |def|markdownReadAndConvertStripPercentSign##1{@
      |markdownIfOption{stripPercentSigns}{@
        |if##1%@
          |expandafter|expandafter|expandafter@
            |markdownReadAndConvertProcessLine@
        |else@
          |expandafter|expandafter|expandafter@
            |markdownReadAndConvertProcessLine@
            |expandafter|expandafter|expandafter##1@
        |fi@
      }{@
        |expandafter@
          |markdownReadAndConvertProcessLine@
          |expandafter##1@
      }@
    }@
```

The `\markdownReadAndConvertProcessLine` macro will process the individual lines of output. Notice the use of the comments (`@`) to ensure that the entire macro is at a single line and therefore no (active) newline symbols (`^^M`) are produced.

```
    |def|markdownReadAndConvertProcessLine##1#1##2#1##3|relax{@
```

If we are not reading markdown documents from the frozen cache and the ending token sequence does not appear in the line, store the line in the `\markdownOptionInputTempFileName` file. If we are reading markdown documents from the frozen cache and the ending token sequence does not appear in the line, gobble the line.

```
      |ifx|relax##3|relax@
        |markdownIfOption{frozenCache}{}{@
          |immediate|write|markdownOutputFileStream{##1}@
```

```
      }@
    |else@
```

When the ending token sequence appears in the line, make the next newline character close the `\markdownOptionInputTempFileName` file, return the character categories back to the former state, convert the `\markdownOptionInputTempFileName` file from markdown to plain TEX, `\input` the result of the conversion, and expand the ending control sequence.

```
      |def^^M{@
        |markdownInfo{The ending token sequence was found}@
        |markdownIfOption{frozenCache}{}{@
          |immediate|closeout|markdownOutputFileStream@
        }@
        |endgroup@
        |markdownInput{@
          |markdownOptionOutputDir@
          /|markdownOptionInputTempFileName@
        }@
        #2}@
    |fi@
```

Repeat with the next line.

```
    ^^M}@
```

Make the tab character active at expansion time and make it expand to a literal tab character.

```
  |catcode`|^^I=13@
  |def^^I{|markdownReadAndConvertTab}@
```

Make the newline character active at expansion time and make it consume the rest of the line on expansion. Throw away the rest of the first line and pass the second line to the `\markdownReadAndConvertProcessLine` macro.

```
  |catcode`|^^M=13@
  |def^^M##1^^M{@
    |def^^M####1^^M{@
      |markdownReadAndConvertStripPercentSign####1#1#1|relax}@
    ^^M}@
  ^^M}@
```

Reset the character categories back to the former state.

```
|endgroup
```

3.2.6 Lua Shell Escape Bridge

The following TEX code is intended for TEX engines that do not provide direct access to Lua, but expose the shell of the operating system. This corresponds to the `\markdownMode` values of `0` and `1`.

The `\markdownLuaExecute` macro defined here and in Section 3.2.7 are meant to be indistinguishable to the remaining code.

The package assumes that although the user is not using the LuaTEX engine, their TEX distribution contains it, and uses shell access to produce and execute Lua scripts using the TEXLua interpreter [1, Section 3.1.1].

```
\ifnum\markdownMode<2\relax
\ifnum\markdownMode=0\relax
  \markdownInfo{Using mode 0: Shell escape via write18}%
\else
  \markdownInfo{Using mode 1: Shell escape via os.execute}%
\fi
```

The `\markdownExecuteShellEscape` macro contains the numeric value indicating whether the shell access is enabled (`1`), disabled (`0`), or restricted (`2`).

Inherit the value of the the `\pdfshellescape` (LuaTEX, PdfTEX) or the `\shellescape` (XETEX) commands. If neither of these commands is defined and Lua is available, attempt to access the `status.shell_escape` configuration item.

If you cannot detect, whether the shell access is enabled, act as if it were.

```
\ifx\pdfshellescape\undefined
  \ifx\shellescape\undefined
    \ifnum\markdownMode=0\relax
      \def\markdownExecuteShellEscape{1}%
    \else
      \def\markdownExecuteShellEscape{%
        \directlua{tex.sprint(status.shell_escape or "1")}}%
    \fi
  \else
    \let\markdownExecuteShellEscape\shellescape
  \fi
\else
  \let\markdownExecuteShellEscape\pdfshellescape
\fi
```

The `\markdownExecuteDirect` macro executes the code it has received as its first argument by writing it to the output file stream 18, if Lua is unavailable, or by using the Lua `os.execute` method otherwise.

```
\ifnum\markdownMode=0\relax
  \def\markdownExecuteDirect#1{\immediate\write18{#1}}%
\else
  \def\markdownExecuteDirect#1{%
    \directlua{os.execute("\luaescapestring{#1}")}}%
\fi
```

The `\markdownExecute` macro is a wrapper on top of `\markdownExecuteDirect` that checks the value of `\markdownExecuteShellEscape` and prints an error message if the shell is inaccessible.

```
\def\markdownExecute#1{%
  \ifnum\markdownExecuteShellEscape=1\relax
    \markdownExecuteDirect{#1}%
  \else
    \markdownError{I can not access the shell}{Either run the TeX
      compiler with the --shell-escape or the --enable-write18 flag,
      or set shell_escape=t in the texmf.cnf file}%
  \fi}%
```

The `\markdownLuaExecute` macro executes the Lua code it has received as its first argument. The Lua code may not directly interact with the TEX engine, but it can use the `print` function in the same manner it would use the `tex.print` method.

```
\begingroup
```

Swap the category code of the backslash symbol and the pipe symbol, so that we may use the backslash symbol freely inside the Lua code.

```
  \catcode`|=0%
  \catcode`\\=12%
  |gdef|markdownLuaExecute#1{%
```

Create the file `\markdownOptionHelperScriptFileName` and fill it with the input Lua code prepended with kpathsea initialization, so that Lua modules from the TEX distribution are available.

```
    |immediate|openout|markdownOutputFileStream=%
      |markdownOptionHelperScriptFileName
    |markdownInfo{Writing a helper Lua script to the file
      "|markdownOptionHelperScriptFileName"}%
    |immediate|write|markdownOutputFileStream{%
      local ran_ok, error = pcall(function()
        local ran_ok, kpse = pcall(require, "kpse")
        if ran_ok then kpse.set_program_name("luatex") end
        #1
      end)
```

If there was an error, use the file `\markdownOptionErrorTempFileName` to store the error message.

```
      if not ran_ok then
        local file = io.open("%
          |markdownOptionOutputDir
          /|markdownOptionErrorTempFileName", "w")
        if file then
          file:write(error .. "\n")
          file:close()
        end
        print('\\markdownError{An error was encountered while executing
               Lua code}{For further clues, examine the file
               "|markdownOptionOutputDir
               /|markdownOptionErrorTempFileName"}')
```

```
      end}%
    |immediate|closeout|markdownOutputFileStream
```

Execute the generated `\markdownOptionHelperScriptFileName` Lua script using the TEXLua binary and store the output in the `\markdownOptionOutputTempFileName` file.

```
    |markdownInfo{Executing a helper Lua script from the file
      "|markdownOptionHelperScriptFileName" and storing the result in the
      file "|markdownOptionOutputTempFileName"}%
    |markdownExecute{texlua "|markdownOptionOutputDir
      /|markdownOptionHelperScriptFileName" > %
      "|markdownOptionOutputDir
      /|markdownOptionOutputTempFileName"}%
```

`\input` the generated `\markdownOptionOutputTempFileName` file.

```
    |input|markdownOptionOutputTempFileName|relax}%
|endgroup
```

3.2.7 Direct Lua Access

The following TEX code is intended for TEX engines that provide direct access to Lua (LuaTEX). The macro `\markdownLuaExecute` defined here and in Section 3.2.6 are meant to be indistinguishable to the remaining code. This corresponds to the `\markdownMode` value of 2.

```
\else
\markdownInfo{Using mode 2: Direct Lua access}%
```

The direct Lua access version of the `\markdownLuaExecute` macro is defined in terms of the `\directlua` primitive. The `print` function is set as an alias to the `\tex.print` method in order to mimic the behaviour of the `\markdownLuaExecute` definition from Section 3.2.6,

```
\begingroup
```

Swap the category code of the backslash symbol and the pipe symbol, so that we may use the backslash symbol freely inside the Lua code.

```
  \catcode`|=0%
  \catcode`\\=12%
  |gdef|markdownLuaExecute#1{%
    |directlua{%
      local function print(input)
        local output = {}
        for line in input:gmatch("[^\r\n]+") do
          table.insert(output, line)
        end
        tex.print(output)
      end
      #1
```

```
    }%
  }%
|endgroup
\fi
```

3.2.8 Typesetting Markdown

The `\markdownInput` macro uses an implementation of the `\markdownLuaExecute` macro to convert the contents of the file whose filename it has received as its single argument from markdown to plain TEX.

```
\begingroup
```

Swap the category code of the backslash symbol and the pipe symbol, so that we may use the backslash symbol freely inside the Lua code.

```
  \catcode`|=0%
  \catcode`\\=12%
  |gdef|markdownInput#1{%
```

Change the category code of the percent sign (`%`) to other, so that a user of the `hybrid` Lua option or a malevolent actor can't produce TeX comments in the plain TeX output of the Markdown package.

```
    |begingroup
    |catcode`|%=12
```

If we are reading from the frozen cache, input it, expand the corresponding `\markdownFrozenCache`⟨*number*⟩ macro, and increment `\markdownFrozenCacheCounter`.

```
    |markdownIfOption{frozenCache}{%
      |ifnum|markdownFrozenCacheCounter=0|relax
        |markdownInfo{Reading frozen cache from
          "|markdownOptionFrozenCacheFileName"}%
        |input|markdownOptionFrozenCacheFileName|relax
      |fi
      |markdownInfo{Including markdown document number
        "|the|markdownFrozenCacheCounter" from frozen cache}%
      |csname markdownFrozenCache|the|markdownFrozenCacheCounter|endcsname
      |global|advance|markdownFrozenCacheCounter by 1|relax
    }{%
      |markdownInfo{Including markdown document "#1"}%
```

Attempt to open the markdown document to record it in the `.log` and `.fls` files. This allows external programs such as LATEXMk to track changes to the markdown document.

```
      |openin|markdownInputFileStream#1
      |closein|markdownInputFileStream
      |markdownPrepareLuaOptions
      |markdownLuaExecute{%
        |markdownPrepare
```

```
        local file = assert(io.open("#1", "r"),
          [[could not open file "#1" for reading]])
        local input = assert(file:read("*a"))
        assert(file:close())
```

Since the Lua converter expects UNIX line endings, normalize the input. Also add a line ending at the end of the file in case the input file has none.

```
        print(convert(input:gsub("\r\n?", "\n") .. "\n"))}%
```

If we are finalizing the frozen cache, increment `\markdownFrozenCacheCounter`.

```
      |markdownIfOption{finalizeCache}{%
        |global|advance|markdownFrozenCacheCounter by 1|relax
      }%
    }%
    |endgroup
  }%
|endgroup
```

3.3 LaTeX Implementation

The LaTeX implemenation makes use of the fact that, apart from some subtle differences, LaTeX implements the majority of the plain TeX format [10, Section 9]. As a consequence, we can directly reuse the existing plain TeX implementation.

The LaTeX implementation redefines the plain TeX logging macros (see Section 3.2.1) to use the LaTeX `\PackageInfo`, `\PackageWarning`, and `\PackageError` macros.

```
\newcommand\markdownInfo[1]{\PackageInfo{markdown}{#1}}%
\newcommand\markdownWarning[1]{\PackageWarning{markdown}{#1}}%
\newcommand\markdownError[2]{\PackageError{markdown}{#1}{#2.}}%
\input markdown/markdown
\def\markdownVersionSpace{ }%
\ProvidesPackage{markdown}[\markdownLastModified\markdownVersionSpace v%
  \markdownVersion\markdownVersionSpace markdown renderer]%
```

Use reflection to define the `renderers` and `rendererPrototypes` keys of `\markdownSetup` as well as the keys that correspond to Lua options.

```
\ExplSyntaxOn
\@@_latex_define_renderers:
\@@_latex_define_renderer_prototypes:
\ExplSyntaxOff
```

3.3.1 Logging Facilities

The LaTeX implementation redefines the plain TeX logging macros (see Section 3.2.1) to use the LaTeX `\PackageInfo`, `\PackageWarning`, and `\PackageError` macros.

3.3.2 Typesetting Markdown

The `\markdownInputPlainTeX` macro is used to store the original plain TeX implementation of the `\markdownInput` macro. The `\markdownInput` is then redefined to accept an optional argument with options recognized by the LaTeX interface (see Section 2.3.2).

```
\let\markdownInputPlainTeX\markdownInput
\renewcommand\markdownInput[2][]{%
  \begingroup
    \markdownSetup{#1}%
    \markdownInputPlainTeX{#2}%
  \endgroup}%
```

The `markdown`, and `markdown*` LaTeX environments are implemented using the `\markdownReadAndConvert` macro.

```
\renewenvironment{markdown}{%
  \markdownReadAndConvert@markdown{}}{%
  \markdownEnd}%
\renewenvironment{markdown*}[1]{%
  \markdownSetup{#1}%
  \markdownReadAndConvert@markdown*}{%
  \markdownEnd}%
\begingroup
```

Locally swap the category code of the backslash symbol with the pipe symbol, and of the left (`{`) and right brace (`}`) with the less-than (`<`) and greater-than (`>`) signs. This is required in order that all the special symbols that appear in the first argument of the `markdownReadAndConvert` macro have the category code *other*.

```
  \catcode`\|=0\catcode`\<=1\catcode`\>=2%
  \catcode`\\=12|catcode`|{=12|catcode`|}=12%
  |gdef|markdownReadAndConvert@markdown#1<%
    |markdownReadAndConvert<\end{markdown#1}>%
                           <|end<markdown#1>>>%
|endgroup
```

3.3.2.1 LaTeX Themes

This section implements the theme-loading mechanism and the example themes provided with the Markdown package.

```
\ExplSyntaxOn
```

To keep track of our current place when packages themes have been nested, we will maintain the `\g_@@_latex_themes_seq` stack of theme names.

```
\newcommand\markdownLaTeXThemeName{}
\seq_new:N \g_@@_latex_themes_seq
\seq_put_right:NV
  \g_@@_latex_themes_seq
  \markdownLaTeXThemeName
```

```
\newcommand\markdownLaTeXThemeLoad[2]{
  \def\@tempa{%
    \def\markdownLaTeXThemeName{#2}
    \seq_put_right:NV
      \g_@@_latex_themes_seq
      \markdownLaTeXThemeName
    \RequirePackage{#1}
    \seq_pop_right:NN
      \g_@@_latex_themes_seq
      \l_tmpa_tl
    \seq_get_right:NN
      \g_@@_latex_themes_seq
      \l_tmpa_tl
    \exp_args:NNV
      \def
      \markdownLaTeXThemeName
      \l_tmpa_tl}
  \ifmarkdownLaTeXLoaded
    \@tempa
  \else
    \exp_args:No
      \AtEndOfPackage
      { \@tempa }
  \fi}
\ExplSyntaxOff
```

The `witiko/dot` theme enables the `fencedCode` Lua option:

```
\markdownSetup{fencedCode}%
```

We load the ifthen and grffile packages, see also Section 1.1.3:

```
\RequirePackage{ifthen,grffile}
```

We store the previous definition of the fenced code token renderer prototype:

```
\let\markdown@witiko@dot@oldRendererInputFencedCodePrototype
  \markdownRendererInputFencedCodePrototype
```

If the infostring starts with `dot …`, we redefine the fenced code block token renderer prototype, so that it typesets the code block via Graphviz tools if and only if the `\markdownOptionFrozenCache` plain TEX option is disabled and the code block has not been previously typeset:

```
\renewcommand\markdownRendererInputFencedCode[2]{%
  \def\next##1 ##2\relax{%
    \ifthenelse{\equal{##1}{dot}}{%
      \markdownIfOption{frozenCache}{}{%
        \immediate\write18{%
          if ! test -e #1.pdf.source || ! diff #1 #1.pdf.source;
          then
            dot -Tpdf -o #1.pdf #1;
```

```
            cp #1 #1.pdf.source;
          fi}}%
```

We include the typeset image using the image token renderer:

```
      \markdownRendererImage{Graphviz image}{#1.pdf}{#1.pdf}{##2}%
```

If the infostring does not start with `dot …`, we use the previous definition of the fenced code token renderer prototype:

```
    }{%
      \markdown@witiko@dot@oldRendererInputFencedCodePrototype{#1}{#2}%
    }%
  }%
  \next#2 \relax}%
```

The `witiko/graphicx/http` theme stores the previous definition of the image token renderer prototype:

```
\let\markdown@witiko@graphicx@http@oldRendererImagePrototype
  \markdownRendererImagePrototype
```

We load the catchfile and grffile packages, see also Section 1.1.3:

```
\RequirePackage{catchfile,grffile}
```

We define the `\markdown@witiko@graphicx@http@counter` counter to enumerate the images for caching and the `\markdown@witiko@graphicx@http@filename` command, which will store the pathname of the file containing the pathname of the downloaded image file.

```
\newcount\markdown@witiko@graphicx@http@counter
\markdown@witiko@graphicx@http@counter=0
\newcommand\markdown@witiko@graphicx@http@filename{%
  \markdownOptionCacheDir/witiko_graphicx_http%
  .\the\markdown@witiko@graphicx@http@counter}%
```

We define the `\markdown@witiko@graphicx@http@download` command, which will receive two arguments that correspond to the URL of the online image and to the pathname, where the online image should be downloaded. The command will produce a shell command that tries to downloads the online image to the pathname.

```
\newcommand\markdown@witiko@graphicx@http@download[2]{%
  wget -O #2 #1 || curl --location -o #2 #1 || rm -f #2}
```

We locally swap the category code of the percentage sign with the line feed control character, so that we can use percentage signs in the shell code:

```
\begingroup
\catcode`\%=12
\catcode`\^^A=14
```

We redefine the image token renderer prototype, so that it tries to download an online image.

```
\global\def\markdownRendererImagePrototype#1#2#3#4{^^A
  \begingroup
    \edef\filename{\markdown@witiko@graphicx@http@filename}^^A
```

The image will be downloaded only if the image URL has the http or https protocols and the \markdownOptionFrozenCache plain TEX option is disabled:

```
    \markdownIfOption{frozenCache}{}{^^A
      \immediate\write18{^^A
        if printf '%s' "#3" | grep -q -E '^https?:';
        then
```

The image will be downloaded to the pathname \markdownOptionCacheDir/⟨*the MD5 digest of the image URL*⟩.⟨*the suffix of the image URL*⟩:

```
          OUTPUT_PREFIX="\markdownOptionCacheDir";
          OUTPUT_BODY="$(printf '%s' '#3' | md5sum | cut -d' ' -f1)";
          OUTPUT_SUFFIX="$(printf '%s' '#3' | sed 's/.*[.]//')";
          OUTPUT="$OUTPUT_PREFIX/$OUTPUT_BODY.$OUTPUT_SUFFIX";
```

The image will be downloaded only if it has not already been downloaded:

```
          if ! [ -e "$OUTPUT" ];
          then
            \markdown@witiko@graphicx@http@download{'#3'}{"$OUTPUT"};
            printf '%s' "$OUTPUT" > "\filename";
          fi;
```

If the image does not have the http or https protocols or the image has already been downloaded, the URL will be stored as-is:

```
        else
          printf '%s' '#3' > "\filename";
        fi}}^^A
```

We load the pathname of the downloaded image and we typeset the image using the previous definition of the image renderer prototype:

```
    \CatchFileDef{\filename}{\filename}{\endlinechar=-1}^^A
    \markdown@witiko@graphicx@http@oldRendererImagePrototype^^A
      {#1}{#2}{\filename}{#4}^^A
  \endgroup
  \global\advance\markdown@witiko@graphicx@http@counter by 1\relax}^^A
\endgroup
```

The witiko/tilde theme redefines the tilde token renderer prototype, so that it expands to a non-breaking space:

```
\renewcommand\markdownRendererTildePrototype{~}%
```

3.3.3 Options

The supplied package options are processed using the \markdownSetup macro.

```
\DeclareOption*{%
  \expandafter\markdownSetup\expandafter{\CurrentOption}}%
\ProcessOptions\relax
```

After processing the options, activate the `renderers`, `rendererPrototypes`, and `code` keys. The `code` key is used to immediately expand and execute code, which can be especially useful in LaTeX setup snippets.

```
\define@key{markdownOptions}{renderers}{%
  \setkeys{markdownRenderers}{#1}%
  \def\KV@prefix{KV@markdownOptions@}}%
\define@key{markdownOptions}{rendererPrototypes}{%
  \setkeys{markdownRendererPrototypes}{#1}%
  \def\KV@prefix{KV@markdownOptions@}}%
\define@key{markdownOptions}{code}{#1}%
```

3.3.4 Token Renderer Prototypes

The following configuration should be considered placeholder. If the `plain` package option has been enabled (see Section 2.3.2.1), none of it will take effect.

```
\ifmarkdownLaTeXPlain\else
```

If the `tightLists` Lua option is disabled or the current document class is beamer, do not load the paralist package.

```
\markdownIfOption{tightLists}{
  \@ifclassloaded{beamer}{}{\RequirePackage{paralist}}%
}{}
```

If we loaded the paralist package, define the respective renderer prototypes to make use of the capabilities of the package. Otherwise, define the renderer prototypes to fall back on the corresponding renderers for the non-tight lists.

```
\@ifpackageloaded{paralist}{
  \markdownSetup{rendererPrototypes={
    ulBeginTight = {\begin{compactitem}},
    ulEndTight = {\end{compactitem}},
    olBeginTight = {\begin{compactenum}},
    olEndTight = {\end{compactenum}},
    dlBeginTight = {\begin{compactdesc}},
    dlEndTight = {\end{compactdesc}}}}
}{
  \markdownSetup{rendererPrototypes={
    ulBeginTight = {\markdownRendererUlBegin},
    ulEndTight = {\markdownRendererUlEnd},
    olBeginTight = {\markdownRendererOlBegin},
    olEndTight = {\markdownRendererOlEnd},
    dlBeginTight = {\markdownRendererDlBegin},
    dlEndTight = {\markdownRendererDlEnd}}}}
\RequirePackage{amsmath,ifthen}
```

Unless the unicode-math package has been loaded, load the amssymb package with symbols to be used for tickboxes.

```
\@ifpackageloaded{unicode-math}{
```

```
  \markdownSetup{rendererPrototypes={
    untickedBox = {$\mdlgwhtsquare$},
  }}
}{
  \RequirePackage{amssymb}
  \markdownSetup{rendererPrototypes={
    untickedBox = {$\square$},
  }}
}
\RequirePackage{csvsimple}
\RequirePackage{fancyvrb}
\RequirePackage{graphicx}
\markdownSetup{rendererPrototypes={
  lineBreak = {\\},
  leftBrace = {\textbraceleft},
  rightBrace = {\textbraceright},
  dollarSign = {\textdollar},
  underscore = {\textunderscore},
  circumflex = {\textasciicircum},
  backslash = {\textbackslash},
  tilde = {\textasciitilde},
  pipe = {\textbar},
```

We can capitalize on the fact that the expansion of renderers is performed by TEX during the typesetting. Therefore, even if we don't know whether a span of text is part of math formula or not when we are parsing markdown,[8] we can reliably detect math mode inside the renderer.

Here, we will redefine the code span renderer prototype to typeset upright text in math formulae and typewriter text outside math formulae.

```
  codeSpan = {%
    \ifmmode
      \text{#1}%
    \else
      \texttt{#1}%
    \fi
  },
  contentBlock = {%
    \ifthenelse{\equal{#1}{csv}}{%
      \begin{table}%
        \begin{center}%
          \csvautotabular{#3}%
        \end{center}
        \ifx\empty#4\empty\else
```

[8] This property may actually be undecidable. Suppose a span of text is a part of a macro definition. Then, whether the span of text is part of a math formula or not depends on where the macro is later used, which may easily be *both* inside and outside a math formula.

```
          \caption{#4}%
        \fi
      \end{table}%
    }{%
      \ifthenelse{\equal{#1}{tex}}{%
        \catcode`\%=14\relax
        \input #3\relax
        \catcode`\%=12\relax
      }{%
        \markdownInput{#3}%
      }%
    }%
  },
  image = {%
    \begin{figure}%
      \begin{center}%
        \includegraphics{#3}%
      \end{center}%
      \ifx\empty#4\empty\else
        \caption{#4}%
      \fi
    \end{figure}},
  ulBegin = {\begin{itemize}},
  ulEnd = {\end{itemize}},
  olBegin = {\begin{enumerate}},
  olItem = {\item{}},
  olItemWithNumber = {\item[#1.]},
  olEnd = {\end{enumerate}},
  dlBegin = {\begin{description}},
  dlItem = {\item[#1]},
  dlEnd = {\end{description}},
  emphasis = {\emph{#1}},
  tickedBox = {$\boxtimes$},
  halfTickedBox = {$\boxdot$},
```

If identifier attributes appear at the beginning of a section, we make the next heading produce the `\label` macro.

```
  headerAttributeContextBegin = {
    \markdownSetup{
      rendererPrototypes = {
        attributeIdentifier = {%
          \begingroup
          \def\next####1{%
            \def####1########1{%
              \endgroup
              ####1{########1}%
              \label{##1}%
```

```
            }%
          }%
          \next\markdownRendererHeadingOne
          \next\markdownRendererHeadingTwo
          \next\markdownRendererHeadingThree
          \next\markdownRendererHeadingFour
          \next\markdownRendererHeadingFive
          \next\markdownRendererHeadingSix
        },
      },
    }%
  },
  blockQuoteBegin = {\begin{quotation}},
  blockQuoteEnd = {\end{quotation}},
  inputVerbatim = {\VerbatimInput{#1}},
  inputFencedCode = {%
    \ifx\relax#2\relax
      \VerbatimInput{#1}%
    \else
      \@ifundefined{minted@code}{%
        \@ifundefined{lst@version}{%
          \markdownRendererInputFencedCode{#1}{}%
```

When the listings package is loaded, use it for syntax highlighting.

```
        }{%
          \lstinputlisting[language=#2]{#1}%
        }%
```

When the minted package is loaded, use it for syntax highlighting. The minted package is preferred over listings.

```
      }{%
        \inputminted{#2}{#1}%
      }%
    \fi},
  horizontalRule = {\noindent\rule[0.5ex]{\linewidth}{1pt}},
  footnote = {\footnote{#1}}}}
```

Support the nesting of strong emphasis.

```
\def\markdownLATEXStrongEmphasis#1{%
  \IfSubStr\f@series{b}{\textnormal{#1}}{\textbf{#1}}}
\markdownSetup{rendererPrototypes={strongEmphasis={%
  \protect\markdownLATEXStrongEmphasis{#1}}}}
```

Support LaTeX document classes that do not provide chapters.

```
\@ifundefined{chapter}{%
  \markdownSetup{rendererPrototypes = {
    headingOne = {\section{#1}},
    headingTwo = {\subsection{#1}},
    headingThree = {\subsubsection{#1}},
```

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

3.3.4.1 Tickboxes

If the `taskLists` option is enabled, we will hide bullets in unordered list items with tickboxes.

```
\markdownSetup{
  rendererPrototypes = {
    ulItem = {%
      \futurelet\markdownLaTeXCheckbox\markdownLaTeXUlItem
    },
  },
}
\def\markdownLaTeXUlItem{%
  \ifx\markdownLaTeXCheckbox\markdownRendererTickedBox
    \item[\markdownLaTeXCheckbox]%
    \expandafter\@gobble
  \else
    \ifx\markdownLaTeXCheckbox\markdownRendererHalfTickedBox
      \item[\markdownLaTeXCheckbox]%
      \expandafter\expandafter\expandafter\@gobble
    \else
      \ifx\markdownLaTeXCheckbox\markdownRendererUntickedBox
        \item[\markdownLaTeXCheckbox]%
        \expandafter\expandafter\expandafter\expandafter
          \expandafter\expandafter\expandafter\@gobble
      \else
        \item{}%
      \fi
    \fi
  \fi
}
```

3.3.4.2 HTML elements

If the `html` option is enabled and we are using TEX4ht[9], we will pass HTML elements to the output HTML document unchanged.

```
\@ifundefined{HCode}{}{
```

[9] See https://tug.org/tex4ht/.

```
  \markdownSetup{
    rendererPrototypes = {
      inlineHtmlTag = {%
        \ifvmode
          \IgnorePar
          \EndP
        \fi
        \HCode{#1}%
      },
      inputBlockHtmlElement = {%
        \ifvmode
          \IgnorePar
        \fi
        \EndP
        \special{t4ht*<#1}%
        \par
        \ShowPar
      },
    },
  }
}
```

3.3.4.3 Citations

Here is a basic implementation for citations that uses the LaTeX `\cite` macro. There are also implementations that use the natbib `\citep`, and `\citet` macros, and the BibLaTeX `\autocites` and `\textcites` macros. These implementations will be used, when the respective packages are loaded.

```
\newcount\markdownLaTeXCitationsCounter

% Basic implementation
\RequirePackage{gobble}
\def\markdownLaTeXBasicCitations#1#2#3#4#5#6{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifx\relax#4\relax
    \ifx\relax#5\relax
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
        \cite{#1#2#6}%  Without prenotes and postnotes, just accumulate cites
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \@gobblethree
      \fi
    \else%  Before a postnote (#5), dump the accumulator
      \ifx\relax#1\relax\else
        \cite{#1}%
      \fi
      \cite[#5]{#6}%
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
```

```
      \else
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \markdownLaTeXBasicCitations
      \fi
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter}%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter}%
      \expandafter\expandafter\expandafter
      \@gobblethree
    \fi
  \else%  Before a prenote (#4), dump the accumulator
    \ifx\relax#1\relax\else
      \cite{#1}%
    \fi
    \ifnum\markdownLaTeXCitationsCounter>1\relax
      \space  % Insert a space before the prenote in later citations
    \fi
    #4~\expandafter\cite\ifx\relax#5\relax{#6}\else[#5]{#6}\fi
    \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
    \else
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter
      \markdownLaTeXBasicCitations
    \fi
    \expandafter\expandafter\expandafter{%
    \expandafter\expandafter\expandafter}%
    \expandafter\expandafter\expandafter{%
    \expandafter\expandafter\expandafter}%
    \expandafter
    \@gobblethree
  \fi\markdownLaTeXBasicCitations{#1#2#6},}
\let\markdownLaTeXBasicTextCitations\markdownLaTeXBasicCitations

% Natbib implementation
\def\markdownLaTeXNatbibCitations#1#2#3#4#5{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifx\relax#3\relax
    \ifx\relax#4\relax
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
```

```
        \citep{#1,#5}%  Without prenotes and postnotes, just accumulate cites
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \@gobbletwo
      \fi
    \else%  Before a postnote (#4), dump the accumulator
      \ifx\relax#1\relax\else
        \citep{#1}%
      \fi
      \citep[][#4]{#5}%
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
      \else
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \markdownLaTeXNatbibCitations
      \fi
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter}%
      \expandafter\expandafter\expandafter
      \@gobbletwo
    \fi
  \else%  Before a prenote (#3), dump the accumulator
    \ifx\relax#1\relax\relax\else
      \citep{#1}%
    \fi
    \citep[#3][#4]{#5}%
    \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
    \else
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter
      \markdownLaTeXNatbibCitations
    \fi
    \expandafter\expandafter\expandafter{%
    \expandafter\expandafter\expandafter}%
    \expandafter
    \@gobbletwo
  \fi\markdownLaTeXNatbibCitations{#1,#5}}
\def\markdownLaTeXNatbibTextCitations#1#2#3#4#5{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifx\relax#3\relax
    \ifx\relax#4\relax
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
        \citet{#1,#5}%  Without prenotes and postnotes, just accumulate cites
```

```
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \@gobbletwo
      \fi
    \else%  After a prenote or a postnote, dump the accumulator
      \ifx\relax#1\relax\else
        \citet{#1}%
      \fi
      , \citet[#3][#4]{#5}%
      \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
        ,
      \else
        \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
          ,
        \fi
      \fi
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter
      \markdownLaTeXNatbibTextCitations
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter}%
      \expandafter\expandafter\expandafter
      \@gobbletwo
    \fi
  \else%  After a prenote or a postnote, dump the accumulator
    \ifx\relax#1\relax\relax\else
      \citet{#1}%
    \fi
    , \citet[#3][#4]{#5}%
    \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
      ,
    \else
      \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
        ,
      \fi
    \fi
    \expandafter\expandafter\expandafter
    \markdownLaTeXNatbibTextCitations
    \expandafter\expandafter\expandafter{%
    \expandafter\expandafter\expandafter}%
    \expandafter
    \@gobbletwo
  \fi\markdownLaTeXNatbibTextCitations{#1,#5}}

% BibLaTeX implementation
```

```
\def\markdownLaTeXBibLaTeXCitations#1#2#3#4#5{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
    \autocites#1[#3][#4]{#5}%
    \expandafter\@gobbletwo
  \fi\markdownLaTeXBibLaTeXCitations{#1[#3][#4]{#5}}}
\def\markdownLaTeXBibLaTeXTextCitations#1#2#3#4#5{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
    \textcites#1[#3][#4]{#5}%
    \expandafter\@gobbletwo
  \fi\markdownLaTeXBibLaTeXTextCitations{#1[#3][#4]{#5}}}

\markdownSetup{rendererPrototypes = {
  cite = {%
    \markdownLaTeXCitationsCounter=1%
    \def\markdownLaTeXCitationsTotal{#1}%
    \@ifundefined{autocites}{%
      \@ifundefined{citep}{%
        \expandafter\expandafter\expandafter
        \markdownLaTeXBasicCitations
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
      }{%
        \expandafter\expandafter\expandafter
        \markdownLaTeXNatbibCitations
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
      }%
    }{%
      \expandafter\expandafter\expandafter
      \markdownLaTeXBibLaTeXCitations
      \expandafter{\expandafter}%
    }},
  textCite = {%
    \markdownLaTeXCitationsCounter=1%
    \def\markdownLaTeXCitationsTotal{#1}%
    \@ifundefined{autocites}{%
      \@ifundefined{citep}{%
        \expandafter\expandafter\expandafter
        \markdownLaTeXBasicTextCitations
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
```

```
      }{%
        \expandafter\expandafter\expandafter
        \markdownLaTeXNatbibTextCitations
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
      }%
    }{%
      \expandafter\expandafter\expandafter
      \markdownLaTeXBibLaTeXTextCitations
      \expandafter{\expandafter}%
    }}}}
```

3.3.4.4 Links

Before consuming the parameters for the hyperlink renderer, we change the category code of the hash sign (`#`) to other, so that it cannot be mistaken for a parameter character.

```
\RequirePackage{url}
\RequirePackage{expl3}
\ExplSyntaxOn
\def\markdownRendererLinkPrototype{
  \begingroup
  \catcode`\#=12
  \def\next##1##2##3##4{
    \endgroup
```

If the URL begins with a hash sign, then we assume that it is a relative reference. Otherwise, we assume that it is an absolute URL.

```
    \tl_set:Nx
      \l_tmpa_tl
      { \str_range:nnn { ##3 } { 1 } { 1 } }
    \str_if_eq:NNTF
      \l_tmpa_tl
      \c_hash_str
      {
        \exp_args:No
          \markdownLaTeXRendererRelativeLink
          { \str_range:nnn { ##3 } { 2 } { -1 } }
      }{
        \markdownLaTeXRendererAbsoluteLink { ##1 } { ##2 } { ##3 } { ##4 }
      }
  }
  \next
}
\ExplSyntaxOff
\def\markdownLaTeXRendererAbsoluteLink#1#2#3#4{%
  #1\footnote{\ifx\empty#4\empty\else#4: \fi\texttt<\url{#3}\texttt>}}
\def\markdownLaTeXRendererRelativeLink#1{%
  \ref{#1}}
```

3.3.4.5 Tables

Here is a basic implementation of tables. If the booktabs package is loaded, then it is used to produce horizontal lines.

```
\newcount\markdownLaTeXRowCounter
\newcount\markdownLaTeXRowTotal
\newcount\markdownLaTeXColumnCounter
\newcount\markdownLaTeXColumnTotal
\newtoks\markdownLaTeXTable
\newtoks\markdownLaTeXTableAlignment
\newtoks\markdownLaTeXTableEnd
\AtBeginDocument{%
  \@ifpackageloaded{booktabs}{%
    \def\markdownLaTeXTopRule{\toprule}%
    \def\markdownLaTeXMidRule{\midrule}%
    \def\markdownLaTeXBottomRule{\bottomrule}%
  }{%
    \def\markdownLaTeXTopRule{\hline}%
    \def\markdownLaTeXMidRule{\hline}%
    \def\markdownLaTeXBottomRule{\hline}%
  }%
}
\markdownSetup{rendererPrototypes={
  table = {%
    \markdownLaTeXTable={}%
    \markdownLaTeXTableAlignment={}%
    \markdownLaTeXTableEnd={%
      \markdownLaTeXBottomRule
      \end{tabular}}%
    \ifx\empty#1\empty\else
      \addto@hook\markdownLaTeXTable{%
        \begin{table}
        \centering}%
      \addto@hook\markdownLaTeXTableEnd{%
        \caption{#1}
        \end{table}}%
    \fi
    \addto@hook\markdownLaTeXTable{\begin{tabular}}%
    \markdownLaTeXRowCounter=0%
    \markdownLaTeXRowTotal=#2%
    \markdownLaTeXColumnTotal=#3%
    \markdownLaTeXRenderTableRow
  }
}}
\def\markdownLaTeXRenderTableRow#1{%
  \markdownLaTeXColumnCounter=0%
  \ifnum\markdownLaTeXRowCounter=0\relax
    \markdownLaTeXReadAlignments#1%
    \markdownLaTeXTable=\expandafter\expandafter\expandafter{%
```

```
      \expandafter\the\expandafter\markdownLaTeXTable\expandafter{%
        \the\markdownLaTeXTableAlignment}}%
    \addto@hook\markdownLaTeXTable{\markdownLaTeXTopRule}%
  \else
    \markdownLaTeXRenderTableCell#1%
  \fi
  \ifnum\markdownLaTeXRowCounter=1\relax
    \addto@hook\markdownLaTeXTable\markdownLaTeXMidRule
  \fi
  \advance\markdownLaTeXRowCounter by 1\relax
  \ifnum\markdownLaTeXRowCounter>\markdownLaTeXRowTotal\relax
    \the\markdownLaTeXTable
    \the\markdownLaTeXTableEnd
    \expandafter\@gobble
  \fi\markdownLaTeXRenderTableRow}
\def\markdownLaTeXReadAlignments#1{%
  \advance\markdownLaTeXColumnCounter by 1\relax
  \if#1d%
    \addto@hook\markdownLaTeXTableAlignment{l}%
  \else
    \addto@hook\markdownLaTeXTableAlignment{#1}%
  \fi
  \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax\else
    \expandafter\@gobble
  \fi\markdownLaTeXReadAlignments}
\def\markdownLaTeXRenderTableCell#1{%
  \advance\markdownLaTeXColumnCounter by 1\relax
  \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax
    \addto@hook\markdownLaTeXTable{#1&}%
  \else
    \addto@hook\markdownLaTeXTable{#1\\}%
    \expandafter\@gobble
  \fi\markdownLaTeXRenderTableCell}
\fi
```

3.3.4.6 YAML Metadata

The default setup of YAML metadata will invoke the `\title`, `\author`, and `\date` macros when scalar values for keys that correspond to the `title`, `author`, and `date` relative wildcards are encountered, respectively.

```
\ExplSyntaxOn
\keys_define:nn
  { markdown/jekyllData }
  {
    author  .code:n = { \author{#1} },
    date    .code:n = { \date{#1}   },
    title   .code:n = { \title{#1}  },
  }
```

To complement the default setup of our key–values, we will use the `\maketitle` macro to typeset the title page of a document at the end of YAML metadata. If we are in the preamble, we will wait macro until after the beginning of the document. Otherwise, we will use the `\maketitle` macro straight away.

```
% TODO: Remove the command definition in TeX Live 2021.
\providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
\markdownSetup{
  rendererPrototypes = {
    jekyllDataEnd = {
%     TODO: Remove the else branch in TeX Live 2021.
      \IfFormatAtLeastTF
        { 2020-10-01 }
        { \AddToHook{begindocument/end}{\maketitle} }
        {
          \ifx\@onlypreamble\@notprerr
            % We are in the document
            \maketitle
          \else
            % We are in the preamble
            \RequirePackage{etoolbox}
            \AfterEndPreamble{\maketitle}
          \fi
        }
    },
  },
}
\ExplSyntaxOff
```

3.3.5 Miscellanea

When buffering user input, we should disable the bytes with the high bit set, since these are made active by the inputenc package. We will do this by redefining the `\markdownMakeOther` macro accordingly. The code is courtesy of Scott Pakin, the creator of the filecontents package.

```
\newcommand\markdownMakeOther{%
  \count0=128\relax
  \loop
    \catcode\count0=11\relax
    \advance\count0 by 1\relax
  \ifnum\count0<256\repeat}%
```

3.4 ConTEXt Implementation

The ConTEXt implementation makes use of the fact that, apart from some subtle differences, the Mark II and Mark IV ConTEXt formats *seem* to implement (the

documentation is scarce) the majority of the plain TEX format required by the plain TEX implementation. As a consequence, we can directly reuse the existing plain TEX implementation after supplying the missing plain TEX macros.

The ConTEXt implementation redefines the plain TEX logging macros (see Section 3.2.1) to use the ConTEXt `\writestatus` macro.

```
\def\markdownInfo#1{\writestatus{markdown}{#1.}}%
\def\markdownWarning#1{\writestatus{markdown\space warn}{#1.}}%
\def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
  \do\#\do\^\do\_\do\%\do\~}%
\input markdown/markdown
```

When buffering user input, we should disable the bytes with the high bit set, since these are made active by the `\enableregime` macro. We will do this by redefining the `\markdownMakeOther` macro accordingly. The code is courtesy of Scott Pakin, the creator of the filecontents LATEX package.

```
\def\markdownMakeOther{%
  \count0=128\relax
  \loop
    \catcode\count0=11\relax
    \advance\count0 by 1\relax
  \ifnum\count0<256\repeat
```

On top of that, make the pipe character (`|`) inactive during the scanning. This is necessary, since the character is active in ConTEXt.

```
  \catcode`|=12}%
```

3.4.1 Typesetting Markdown

The `\startmarkdown` and `\stopmarkdown` macros are implemented using the `\markdownReadAndConvert` macro.

In Knuth's TEX, trailing spaces are removed very early on when a line is being put to the input buffer. [11, sec. 31]. According to Eijkhout [12, sec. 2.2], this is because "these spaces are hard to see in an editor". At the moment, there is no option to suppress this behavior in (Lua)TEX, but ConTEXt MkIV funnels all input through its own input handler. This makes it possible to suppress the removal of trailing spaces in ConTEXt MkIV and therefore to insert hard line breaks into markdown text.

```
\ifx\startluacode\undefined  % MkII
  \begingroup
    \catcode`\|=0%
    \catcode`\\=12%
    |gdef|startmarkdown{%
      |markdownReadAndConvert{\stopmarkdown}%
                             {|stopmarkdown}}%
    |gdef|stopmarkdown{%
      |markdownEnd}%
```

```
  |endgroup
\else  % MkIV
  \startluacode
    document.markdown_buffering = false
    local function preserve_trailing_spaces(line)
      if document.markdown_buffering then
        line = line:gsub("[ \t][ \t]$", "\t\t")
      end
      return line
    end
    resolvers.installinputlinehandler(preserve_trailing_spaces)
  \stopluacode
  \begingroup
    \catcode`\|=0%
    \catcode`\\=12%
    |gdef|startmarkdown{%
      |ctxlua{document.markdown_buffering = true}%
      |markdownReadAndConvert{\stopmarkdown}%
                              {|stopmarkdown}}%
    |gdef|stopmarkdown{%
      |ctxlua{document.markdown_buffering = false}%
      |markdownEnd}%
  |endgroup
\fi
```

3.4.2 Token Renderer Prototypes

The following configuration should be considered placeholder.

```
\def\markdownRendererLineBreakPrototype{\blank}%
\def\markdownRendererLeftBracePrototype{\textbraceleft}%
\def\markdownRendererRightBracePrototype{\textbraceright}%
\def\markdownRendererDollarSignPrototype{\textdollar}%
\def\markdownRendererPercentSignPrototype{\percent}%
\def\markdownRendererUnderscorePrototype{\textunderscore}%
\def\markdownRendererCircumflexPrototype{\textcircumflex}%
\def\markdownRendererBackslashPrototype{\textbackslash}%
\def\markdownRendererTildePrototype{\textasciitilde}%
\def\markdownRendererPipePrototype{\char`|}%
\def\markdownRendererLinkPrototype#1#2#3#4{%
  \useURL[#1][#3][][#4]#1\footnote[#1]{\ifx\empty#4\empty\else#4:
  \fi\tt<\hyphenatedurl{#3}>}}%
\usemodule[database]
\defineseparatedlist
  [MarkdownConTeXtCSV]
  [separator={,},
   before=\bTABLE,after=\eTABLE,
   first=\bTR,last=\eTR,
```

```
    left=\bTD,right=\eTD]
\def\markdownConTeXtCSV{csv}
\def\markdownRendererContentBlockPrototype#1#2#3#4{%
  \def\markdownConTeXtCSV@arg{#1}%
  \ifx\markdownConTeXtCSV@arg\markdownConTeXtCSV
    \placetable[][tab:#1]{#4}{%
      \processseparatedfile[MarkdownConTeXtCSV][#3]}%
  \else
    \markdownInput{#3}%
  \fi}%
\def\markdownRendererImagePrototype#1#2#3#4{%
  \placefigure[][]{#4}{\externalfigure[#3]}}%
\def\markdownRendererUlBeginPrototype{\startitemize}%
\def\markdownRendererUlBeginTightPrototype{\startitemize[packed]}%
\def\markdownRendererUlItemPrototype{\item}%
\def\markdownRendererUlEndPrototype{\stopitemize}%
\def\markdownRendererUlEndTightPrototype{\stopitemize}%
\def\markdownRendererOlBeginPrototype{\startitemize[n]}%
\def\markdownRendererOlBeginTightPrototype{\startitemize[packed,n]}%
\def\markdownRendererOlItemPrototype{\item}%
\def\markdownRendererOlItemWithNumberPrototype#1{\sym{#1.}}%
\def\markdownRendererOlEndPrototype{\stopitemize}%
\def\markdownRendererOlEndTightPrototype{\stopitemize}%
\definedescription
  [MarkdownConTeXtDlItemPrototype]
  [location=hanging,
   margin=standard,
   headstyle=bold]%
\definestartstop
  [MarkdownConTeXtDlPrototype]
  [before=\blank,
   after=\blank]%
\definestartstop
  [MarkdownConTeXtDlTightPrototype]
  [before=\blank\startpacked,
   after=\stoppacked\blank]%
\def\markdownRendererDlBeginPrototype{%
  \startMarkdownConTeXtDlPrototype}%
\def\markdownRendererDlBeginTightPrototype{%
  \startMarkdownConTeXtDlTightPrototype}%
\def\markdownRendererDlItemPrototype#1{%
  \startMarkdownConTeXtDlItemPrototype{#1}}%
\def\markdownRendererDlItemEndPrototype{%
  \stopMarkdownConTeXtDlItemPrototype}%
\def\markdownRendererDlEndPrototype{%
  \stopMarkdownConTeXtDlPrototype}%
\def\markdownRendererDlEndTightPrototype{%
```

```
  \stopMarkdownConTeXtDlTightPrototype}%
\def\markdownRendererEmphasisPrototype#1{{\em#1}}%
\def\markdownRendererStrongEmphasisPrototype#1{{\bf#1}}%
\def\markdownRendererBlockQuoteBeginPrototype{\startquotation}%
\def\markdownRendererBlockQuoteEndPrototype{\stopquotation}%
\def\markdownRendererInputVerbatimPrototype#1{\typefile{#1}}%
\def\markdownRendererInputFencedCodePrototype#1#2{%
  \ifx\relax#2\relax
    \typefile{#1}%
  \else
```

The code fence infostring is used as a name from the ConTEXt `\definetyping` macro. This allows the user to set up code highlighting mapping as follows:

```
\definetyping [latex]
\setuptyping  [latex] [option=TEX]

\starttext
  \startmarkdown
~~~ latex
\documentclass{article}
\begin{document}
  Hello world!
\end{document}
~~~
  \stopmarkdown
\stoptext
```

```
    \typefile[#2][]{#1}%
  \fi}%
\def\markdownRendererHeadingOnePrototype#1{\chapter{#1}}%
\def\markdownRendererHeadingTwoPrototype#1{\section{#1}}%
\def\markdownRendererHeadingThreePrototype#1{\subsection{#1}}%
\def\markdownRendererHeadingFourPrototype#1{\subsubsection{#1}}%
\def\markdownRendererHeadingFivePrototype#1{\subsubsubsection{#1}}%
\def\markdownRendererHeadingSixPrototype#1{\subsubsubsubsection{#1}}%
\def\markdownRendererHorizontalRulePrototype{%
  \blackrule[height=1pt, width=\hsize]}%
\def\markdownRendererFootnotePrototype#1{\footnote{#1}}%
\stopmodule\protect
```

There is a basic implementation of tables.

```
\newcount\markdownConTeXtRowCounter
\newcount\markdownConTeXtRowTotal
\newcount\markdownConTeXtColumnCounter
\newcount\markdownConTeXtColumnTotal
```

```
\newtoks\markdownConTeXtTable
\newtoks\markdownConTeXtTableFloat
\def\markdownRendererTablePrototype#1#2#3{%
  \markdownConTeXtTable={}%
  \ifx\empty#1\empty
    \markdownConTeXtTableFloat={%
      \the\markdownConTeXtTable}%
  \else
    \markdownConTeXtTableFloat={%
      \placetable{#1}{\the\markdownConTeXtTable}}%
  \fi
  \begingroup
  \setupTABLE[r][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]
  \setupTABLE[c][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]
  \setupTABLE[r][1][topframe=on, bottomframe=on]
  \setupTABLE[r][#1][bottomframe=on]
  \markdownConTeXtRowCounter=0%
  \markdownConTeXtRowTotal=#2%
  \markdownConTeXtColumnTotal=#3%
  \markdownConTeXtRenderTableRow}
\def\markdownConTeXtRenderTableRow#1{%
  \markdownConTeXtColumnCounter=0%
  \ifnum\markdownConTeXtRowCounter=0\relax
    \markdownConTeXtReadAlignments#1%
    \markdownConTeXtTable={\bTABLE}%
  \else
    \markdownConTeXtTable=\expandafter{%
      \the\markdownConTeXtTable\bTR}%
    \markdownConTeXtRenderTableCell#1%
    \markdownConTeXtTable=\expandafter{%
      \the\markdownConTeXtTable\eTR}%
  \fi
  \advance\markdownConTeXtRowCounter by 1\relax
  \ifnum\markdownConTeXtRowCounter>\markdownConTeXtRowTotal\relax
    \markdownConTeXtTable=\expandafter{%
      \the\markdownConTeXtTable\eTABLE}%
    \the\markdownConTeXtTableFloat
    \endgroup
    \expandafter\gobbleoneargument
  \fi\markdownConTeXtRenderTableRow}
\def\markdownConTeXtReadAlignments#1{%
  \advance\markdownConTeXtColumnCounter by 1\relax
  \if#1d%
    \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
  \fi\if#1l%
    \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
  \fi\if#1c%
```

```
    \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=middle]
  \fi\if#1r%
    \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=left]
  \fi
  \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
    \expandafter\gobbleoneargument
  \fi\markdownConTeXtReadAlignments}
\def\markdownConTeXtRenderTableCell#1{%
  \advance\markdownConTeXtColumnCounter by 1\relax
  \markdownConTeXtTable=\expandafter{%
    \the\markdownConTeXtTable\bTD#1\eTD}%
  \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
    \expandafter\gobbleoneargument
  \fi\markdownConTeXtRenderTableCell}
\def\markdownRendererTickedBox{$\boxtimes$}
\def\markdownRendererHalfTickedBox{$\boxdot$}
\def\markdownRendererUntickedBox{$\square$}
```

References

[1] LuaTEX development team. *LuaTEX reference manual*. Feb. 2017. URL: http://www.luatex.org/svn/trunk/manual/luatex.pdf (visited on 01/08/2018).

[2] Vít Novotný. *TeXový interpret jazyka Markdown (markdown.sty)*. 2015. URL: https://www.muni.cz/en/research/projects/32984 (visited on 02/19/2018).

[3] Anton Sotkov. *File transclusion syntax for Markdown*. Jan. 19, 2017. URL: https://github.com/iainc/Markdown-Content-Blocks (visited on 01/08/2018).

[4] Donald Ervin Knuth. *The TEXbook*. 3rd ed. Vol. A. Computers & Typesetting. Reading, MA: Addison-Wesley, 1986. ix, 479. ISBN: 0-201-13447-0.

[5] Frank Mittelbach. *The `doc` and `shortvrb` Packages*. Apr. 15, 2017. URL: https://mirrors.ctan.org/macros/latex/base/doc.pdf (visited on 02/19/2018).

[6] Till Tantau, Joseph Wright, and Vedran Miletić. *The Beamer class*. Feb. 10, 2021. URL: https://mirrors.ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf (visited on 02/11/2021).

[7] Vít Novotný. *LATEX 2ε no longer keys packages by pathnames*. Feb. 20, 2021. URL: https://github.com/latex3/latex2e/issues/510 (visited on 02/21/2021).

[8] Geoffrey M. Poore. *The `minted` Package. Highlighted source code in LATEX*. July 19, 2017. URL: https://mirrors.ctan.org/macros/latex/contrib/minted/minted.pdf (visited on 09/01/2020).

[9] Roberto Ierusalimschy. *Programming in Lua*. 3rd ed. Rio de Janeiro: PUC-Rio, 2013. xviii, 347. ISBN: 978-85-903798-5-0.

[10] Johannes Braams et al. *The LaTeX 2ε Sources*. Apr. 15, 2017. URL: https://mirrors.ctan.org/macros/latex/base/source2e.pdf (visited on 01/08/2018).

[11] Donald Ervin Knuth. *TeX: The Program*. Vol. B. Computers & Typesetting. Reading, MA: Addison-Wesley, 1986. xvi, 594. ISBN: 0-201-13437-7.

[12] Victor Eijkhout. *TeX by Topic. A TeXnician's Reference*. Wokingham, England: Addison-Wesley, Feb. 1, 1992. 307 pp. ISBN: 0-201-56882-0.

Index