A Markdown Interpreter for TEX

Vít Novotný
witiko@mail.muni.cz

Version 2.15.4-0-g4cbe4e3
2022/07/29

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 unicode
(function()
  local ran_ok
  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.

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

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

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 packages:

expl3 A package that enables the expl3 language from the LaTeX3 kernel in TeX Live ⩽ 2019. It is used to implement reflection capabilities that allow us to enumerate and inspect high-level concepts such as options, renderers, and renderer prototypes.

```
⟨@@=markdown⟩
\ifx\ExplSyntaxOn\undefined
  \input expl3-generic\relax
\fi
```

lt3luabridge A package that allows us to execute Lua code with LuaTeX as well as with other TeX engines that provide the *shell escape* capability, which allows them to execute code with the system's shell.

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$_\varepsilon$ format is loaded,

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

a TeX engine that extends ε-TeX, and all the plain TeX prerequisites (see Section 1.1.2):

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.

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

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.

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$_\varepsilon$, 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.

```
28 local M = {metadata = metadata}
```

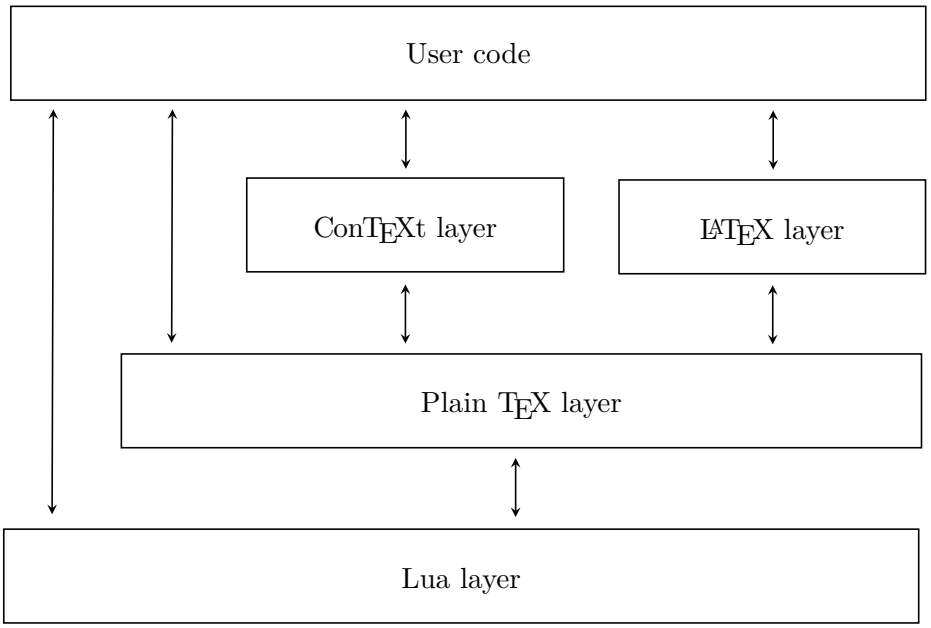

Figure 1: A block diagram of the Markdown package

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.

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

```
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.

```
29 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
\seq_new:N \g_@@_option_layers_seq
\tl_const:Nn \c_@@_option_layer_lua_tl { lua }
\seq_put_right:NV \g_@@_option_layers_seq \c_@@_option_layer_lua_tl
\cs_new:Nn
  \@@_add_lua_option:nnn
  {
    \@@_add_option:Vnnn
      \c_@@_option_layer_lua_tl
      { #1 }
      { #2 }
      { #3 }
  }
\cs_new:Nn
  \@@_add_option:nnnn
  {
    \seq_put_right:cn
      { g_@@_ #1 _options_seq }
      { #2 }
    \prop_put:cnn
      { g_@@_ #1 _option_types_prop }
      { #2 }
      { #3 }
    \prop_put:cnn
      { g_@@_default_ #1 _options_prop }
      { #2 }
      { #4 }
    \@@_typecheck_option:n
      { #2 }
  }
\cs_generate_variant:Nn
  \@@_add_option:nnnn
  { Vnnn }
\tl_const:Nn \c_@@_option_value_true_tl  { true  }
\tl_const:Nn \c_@@_option_value_false_tl { false }
\cs_new:Nn \@@_typecheck_option:n
  {
    \@@_get_option_type:nN
      { #1 }
      \l_tmpa_tl
```

```
    \str_case_e:Vn
      \l_tmpa_tl
      {
        { \c_@@_option_type_boolean_tl }
          {
            \@@_get_option_value:nN
              { #1 }
              \l_tmpa_tl
            \bool_if:nF
              {
                \str_if_eq_p:VV
                  \l_tmpa_tl
                  \c_@@_option_value_true_tl ||
                \str_if_eq_p:VV
                  \l_tmpa_tl
                  \c_@@_option_value_false_tl
              }
              {
                \msg_error:nnnV
                  { @@ }
                  { failed-typecheck-for-boolean-option }
                  { #1 }
                  \l_tmpa_tl
              }
          }
      }
  }
\msg_new:nnn
  { @@ }
  { failed-typecheck-for-boolean-option }
  {
    Option~#1~has~value~#2,~
    but~a~boolean~(true~or~false)~was~expected.
  }
\cs_generate_variant:Nn
  \str_case_e:nn
  { Vn }
\cs_generate_variant:Nn
  \msg_error:nnnn
  { nnnV }
\seq_new:N \g_@@_option_types_seq
\tl_const:Nn \c_@@_option_type_counter_tl { counter }
\seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_counter_tl
\tl_const:Nn \c_@@_option_type_boolean_tl { boolean }
\seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_boolean_tl
\tl_const:Nn \c_@@_option_type_number_tl  { number  }
\seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_number_tl
```

```
\tl_const:Nn \c_@@_option_type_path_tl    { path    }
\seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_path_tl
\tl_const:Nn \c_@@_option_type_slice_tl   { slice   }
\seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_slice_tl
\tl_const:Nn \c_@@_option_type_string_tl  { string  }
\seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_string_tl
\cs_new:Nn
  \@@_get_option_type:nN
  {
    \bool_set_false:N
      \l_tmpa_bool
    \seq_map_inline:Nn
      \g_@@_option_layers_seq
      {
        \prop_get:cnNT
          { g_@@_ ##1 _option_types_prop }
          { #1 }
          \l_tmpa_tl
          {
            \bool_set_true:N
              \l_tmpa_bool
            \seq_map_break:
          }
      }
    \bool_if:nF
      \l_tmpa_bool
      {
        \msg_error:nnn
          { @@ }
          { undefined-option }
          { #1 }
      }
    \seq_if_in:NVF
      \g_@@_option_types_seq
      \l_tmpa_tl
      {
        \msg_error:nnnV
          { @@ }
          { unknown-option-type }
          { #1 }
          \l_tmpa_tl
      }
    \tl_set_eq:NN
      #2
      \l_tmpa_tl
  }
\msg_new:nnn
```

```
  { @@ }
  { unknown-option-type }
  {
    Option~#1~has~unknown~type~#2.
  }
\msg_new:nnn
  { @@ }
  { undefined-option }
  {
    Option~#1~is~undefined.
  }
\cs_new:Nn
  \@@_get_default_option_value:nN
  {
    \bool_set_false:N
      \l_tmpa_bool
    \seq_map_inline:Nn
      \g_@@_option_layers_seq
      {
        \prop_get:cnNT
          { g_@@_default_ ##1 _options_prop }
          { #1 }
          #2
          {
            \bool_set_true:N
              \l_tmpa_bool
            \seq_map_break:
          }
      }
    \bool_if:nF
      \l_tmpa_bool
      {
        \msg_error:nnn
          { @@ }
          { undefined-option }
          { #1 }
      }
  }
\cs_new:Nn
  \@@_get_option_value:nN
  {
    \@@_option_tl_to_csname:nN
      { #1 }
      \l_tmpa_tl
    \cs_if_free:cTF
      { \l_tmpa_tl }
      {
```

```
        \@@_get_default_option_value:nN
          { #1 }
          #2
      }
      {
        \@@_get_option_type:nN
          { #1 }
          \l_tmpa_tl
        \str_if_eq:NNTF
          \c_@@_option_type_counter_tl
          \l_tmpa_tl
          {
            \@@_option_tl_to_csname:nN
              { #1 }
              \l_tmpa_tl
            \tl_set:Nx
              #2
              { \the \cs:w \l_tmpa_tl \cs_end: }
          }
          {
            \@@_option_tl_to_csname:nN
              { #1 }
              \l_tmpa_tl
            \tl_set:Nv
              #2
              { \l_tmpa_tl }
          }
      }
  }
\cs_new:Nn \@@_option_tl_to_csname:nN
  {
    \tl_set:Nn
      \l_tmpa_tl
      { \str_uppercase:n { #1 } }
    \tl_set:Nx
      #2
      {
        markdownOption
        \tl_head:f { \l_tmpa_tl }
        \tl_tail:n { #1 }
      }
  }
```

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.

```
\@@_add_lua_option:nnn
  { cacheDir }
  { path }
  { \markdownOptionOutputDir / _markdown_\jobname }
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.

```
\@@_add_lua_option:nnn
  { frozenCacheFileName }
  { path }
  { \markdownOptionCacheDir / 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.

```
\@@_add_lua_option:nnn
  { blankBeforeBlockquote }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { blankBeforeCodeFence }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { blankBeforeHeading }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { breakableBlockquotes }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { citationNbsps }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { citations }
  { boolean }
  { 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.''
```

```
\@@_add_lua_option:nnn
  { codeSpans }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { contentBlocks }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { contentBlocksLanguageMap }
  { path }
  { 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.

```
\@@_add_lua_option:nnn
  { definitionLists }
  { boolean }
  { 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. Recursive nesting of markdown document fragments is undefined behavior when `eagerCache` is disabled.

```
\@@_add_lua_option:nnn
  { eagerCache }
  { boolean }
  { 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}
```

```
\@@_add_lua_option:nnn
  { expectJekyllData }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { fencedCode }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { finalizeCache }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { footnotes }
  { boolean }
  { 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*⟩.

```
\@@_add_lua_option:nnn
  { frozenCacheCounter }
  { counter }
  { 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.

```
\@@_add_lua_option:nnn
  { hardLineBreaks }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { hashEnumerators }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { headerAttributes }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { html }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { hybrid }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { inlineFootnotes }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { jekyllData }
  { boolean }
  { false }
defaultOptions.jekyllData = false
```

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

`true` Enable the PHP Markdown pipe table syntax extension:

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

`false` Disable the PHP Markdown pipe table syntax extension.

```
\@@_add_lua_option:nnn
  { pipeTables }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { preserveTabs }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { relativeReferences }
  { boolean }
  { false }
defaultOptions.relativeReferences = false
```

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

`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.

```
\@@_add_lua_option:nnn
  { shiftHeadings }
  { number }
  { 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.

```
\@@_add_lua_option:nnn
  { slice }
  { 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.

```
\@@_add_lua_option:nnn
  { smartEllipses }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { startNumber }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { stripIndent }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { tableCaptions }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { taskLists }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { texComments }
  { boolean }
  { 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.

```
\@@_add_lua_option:nnn
  { tightLists }
  { boolean }
  { true }
```

```
440 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.

```
441 \@@_add_lua_option:nnn
442   { underscores }
443   { boolean }
444   { true }
445 \ExplSyntaxOff

446 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.

```
447
448 local HELP_STRING = [[
449 Usage: texlua ]] .. arg[0] .. [[ [OPTIONS] -- [INPUT_FILE] [OUTPUT_FILE]
450 where OPTIONS are documented in the Lua interface section of the
451 technical Markdown package documentation.
```

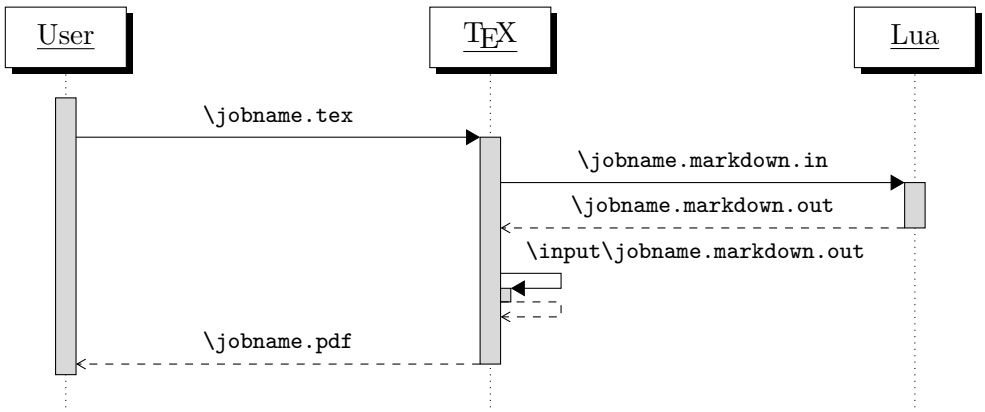

Figure 2: A sequence diagram of the Markdown package typesetting a markdown document using the TEX interface

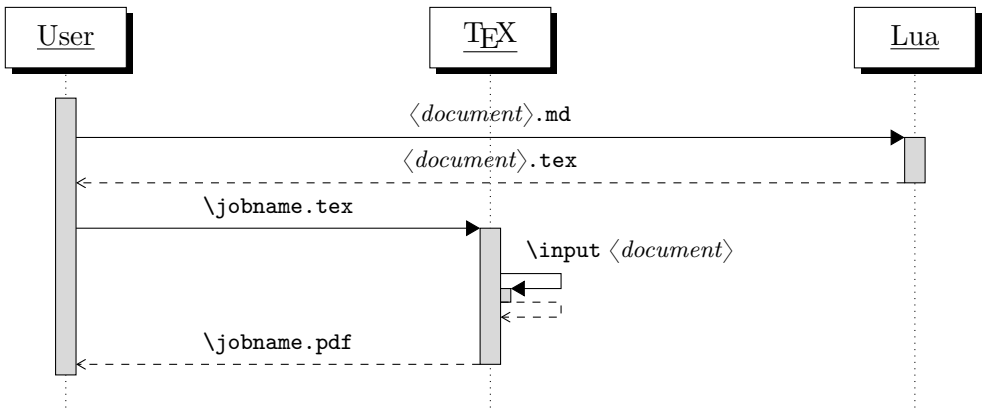

Figure 3: A sequence diagram of the Markdown package typesetting a markdown document using the Lua command-line interface

```
452
453 When OUTPUT_FILE is unspecified, the result of the conversion will be
454 written to the standard output. When INPUT_FILE is also unspecified, the
455 result of the conversion will be read from the standard input.
456
457 Report bugs to: witiko@mail.muni.cz
458 Markdown package home page: <https://github.com/witiko/markdown>]]
459
460 local VERSION_STRING = [[
461 markdown-cli.lua (Markdown) ]] .. metadata.version .. [[
462
463 Copyright (C) ]] .. table.concat(metadata.copyright,
464                                  "\nCopyright (C) ") .. [[
465
466 License: ]] .. metadata.license
467
468 local function warn(s)
469   io.stderr:write("Warning: " .. s .. "\n") end
470
471 local function error(s)
472   io.stderr:write("Error: " .. s .. "\n")
473   os.exit(1) end
474
475 local process_options = true
476 local options = {}
477 local input_filename
478 local output_filename
479 for i = 1, #arg do
480   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.

```
481     if arg[i] == "--" then
482       process_options = false
483       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.

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

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

```
486       local default_type = type(defaultOptions[key])
487       if default_type == "boolean" then
```

```
488         options[key] = (value == "true")
489       elseif default_type == "number" then
490         options[key] = tonumber(value)
491       else
492         if default_type ~= "string" then
493           if default_type == "nil" then
494             warn('Option "' .. key .. '" not recognized.')
495           else
496             warn('Option "' .. key .. '" type not recognized, please file ' ..
497                  'a report to the package maintainer.')
498           end
499           warn('Parsing the ' .. 'value "' .. value ..'" of option "' ..
500                key .. '" as a string.')
501         end
502         options[key] = value
503       end
504       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.

```
505     elseif arg[i] == "--help" or arg[i] == "-h" then
506       print(HELP_STRING)
507       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.

```
508     elseif arg[i] == "--version" or arg[i] == "-v" then
509       print(VERSION_STRING)
510       os.exit()
511     end
512   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.

```
513   if input_filename == nil then
514     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.

```
515   elseif output_filename == nil then
516     output_filename = arg[i]
517   else
518     error('Unexpected argument: "' .. arg[i] .. '".')
519   end
```

```
  ::continue::
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.

```
\def\markdownLastModified{(((LASTMODIFIED)))}%
\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.

```
\let\markdownBegin\relax
\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.

```
526 \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.

To enable the enumeration of plain TEX options, we will maintain the `\g_@@_plain_tex_options_seq` sequence.

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

To enable the reflection of default plain TEX options and their types, we will maintain the `\g_@@_default_plain_tex_options_prop` and `\g_@@_plain_tex_option_types_prop` property lists, respectively.

```
\prop_new:N \g_@@_plain_tex_option_types_prop
\prop_new:N \g_@@_default_plain_tex_options_prop
\tl_const:Nn \c_@@_option_layer_plain_tex_tl { plain_tex }
\seq_put_right:NV \g_@@_option_layers_seq \c_@@_option_layer_plain_tex_tl
\cs_new:Nn
  \@@_add_plain_tex_option:nnn
  {
    \@@_add_option:Vnnn
      \c_@@_option_layer_plain_tex_tl
      { #1 }
      { #2 }
      { #3 }
  }
```

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.

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`.

```
\@@_add_plain_tex_option:nnn
  { frozenCache }
  { boolean }
  { 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.

```
\@@_add_plain_tex_option:nnn
  { helperScriptFileName }
  { path }
  { \jobname.markdown.lua }
```

The `\markdownOptionHelperScriptFileName` macro has been deprecated and will be removed in Markdown 3.0.0. To control the filename of the helper Lua script file, use the `\g_luabridge_helper_script_filename_str` macro from the lt3luabridge package.

```
\str_new:N
  \g_luabridge_helper_script_filename_str
\tl_gset:Nn
  \g_luabridge_helper_script_filename_str
  { \markdownOptionHelperScriptFileName }
```

The `\markdownOptionInputTempFileName` macro sets the filename of the temporary input file that is created during the buffering of markdown text from a TEX source. It defaults to `\jobname.markdown.in`. The same limitations as in the case of the `\markdownOptionHelperScriptFileName` macro apply here.

```
\@@_add_plain_tex_option:nnn
  { inputTempFileName }
  { path }
  { \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.

```
\@@_add_plain_tex_option:nnn
  { outputTempFileName }
  { path }
  { \jobname.markdown.out }
```

The \markdownOptionOutputTempFileName macro has been deprecated and will be removed in Markdown 3.0.0. To control the filename of the temporary file for Lua output, use the \g_luabridge_error_output_filename_str macro from the lt3luabridge package.

```
\str_new:N
  \g_luabridge_standard_output_filename_str
\tl_gset:Nn
  \g_luabridge_standard_output_filename_str
  { \markdownOptionOutputTempFileName }
```

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.

```
\@@_add_plain_tex_option:nnn
  { errorTempFileName }
  { path }
  { \jobname.markdown.err }
```

The \markdownOptionErrorTempFileName macro has been deprecated and will be removed in Markdown 3.0.0. To control the filename of the temporary file for Lua errors, use the \g_luabridge_error_output_filename_str macro from the lt3luabridge package.

```
\str_new:N
  \g_luabridge_error_output_filename_str
\tl_gset:Nn
  \g_luabridge_error_output_filename_str
  { \markdownOptionErrorTempFileName }
```

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.

```
\@@_add_plain_tex_option:nnn
  { outputDir }
  { path }
  { . }
```

The \markdownOptionOutputDir macro is also used to set the \g_luabridge_output_dirname_str macro from the lt3luabridge package.

```
\str_new:N
  \g_luabridge_output_dirname_str
```

```
\tl_gset:Nn
  \g_luabridge_output_dirname_str
  { \markdownOptionOutputDir }
```

Here, we automatically define plain TEX macros for the above plain TEX options.

Furthemore, we also define macros that map directly to the options recognized by the Lua interface, such as `\markdownOptionHybrid` for the `hybrid` Lua option (see Section 2.1.2), which are not processed by the plain TEX implementation, only passed along to Lua.

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.

```
\cs_new:Nn \@@_plain_tex_define_option_commands:
  {
    \seq_map_inline:Nn
      \g_@@_option_layers_seq
      {
        \seq_map_inline:cn
          { g_@@_ ##1 _options_seq }
          {
              \@@_plain_tex_define_option_command:n
                { ####1 }
          }
      }
  }
\cs_new:Nn \@@_plain_tex_define_option_command:n
  {
    \@@_get_default_option_value:nN
      { #1 }
      \l_tmpa_tl
    \@@_set_option_value:nV
      { #1 }
      \l_tmpa_tl
  }
\cs_new:Nn
  \@@_set_option_value:nn
  {
    \@@_define_option:n
      { #1 }
    \@@_get_option_type:nN
      { #1 }
      \l_tmpa_tl
    \str_if_eq:NNTF
      \c_@@_option_type_counter_tl
      \l_tmpa_tl
      {
        \@@_option_tl_to_csname:nN
```

```
          { #1 }
          \l_tmpa_tl
        \int_gset:cn
          { \l_tmpa_tl }
          { #2 }
      }
      {
        \@@_option_tl_to_csname:nN
          { #1 }
          \l_tmpa_tl
        \cs_set:cpn
          { \l_tmpa_tl }
          { #2 }
      }
  }
\cs_generate_variant:Nn
  \@@_set_option_value:nn
  { nV }
\cs_new:Nn
  \@@_define_option:n
  {
    \@@_option_tl_to_csname:nN
      { #1 }
      \l_tmpa_tl
    \cs_if_free:cT
      { \l_tmpa_tl }
      {
        \@@_get_option_type:nN
          { #1 }
          \l_tmpb_tl
        \str_if_eq:NNT
          \c_@@_option_type_counter_tl
          \l_tmpb_tl
          {
            \@@_option_tl_to_csname:nN
              { #1 }
              \l_tmpa_tl
            \int_new:c
              { \l_tmpa_tl }
          }
      }
  }
\@@_plain_tex_define_option_commands:
```

2.2.2.3 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 3.2.4) 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`.

```
\seq_put_right:Nn
  \g_@@_plain_tex_options_seq
  { stripPercentSigns }
\prop_put:Nnn
  \g_@@_plain_tex_option_types_prop
  { stripPercentSigns }
  { boolean }
\prop_put:Nnx
  \g_@@_default_plain_tex_options_prop
  { stripPercentSigns }
  { false }
\ExplSyntaxOff
```

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 Renderers

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 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 }
```

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

```
\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 `\markdownRendererHeaderAttri` 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).

```
\ExplSyntaxOn
\cs_new:Nn \@@_plaintex_define_renderer_prototypes:
  {
    \seq_map_function:NN
      \g_@@_renderers_seq
      \@@_plaintex_define_renderer_prototype:n
    \let\markdownRendererBlockHtmlCommentBeginPrototype=\iffalse
    \let\markdownRendererBlockHtmlCommentBegin=\iffalse
    \let\markdownRendererBlockHtmlCommentEndPrototype=\fi
    \let\markdownRendererBlockHtmlCommentEnd=\fi
  }
\cs_new:Nn \@@_plaintex_define_renderer_prototype:n
  {
    \@@_renderer_prototype_tl_to_csname:nN
      { #1 }
      \l_tmpa_tl
    \prop_get:NnN
      \g_@@_renderer_arities_prop
```

```
      { #1 }
      \l_tmpb_tl
    \@@_plaintex_define_renderer_prototype:cV
      { \l_tmpa_tl }
      \l_tmpb_tl
  }
\cs_new:Nn \@@_renderer_prototype_tl_to_csname:nN
  {
    \tl_set:Nn
      \l_tmpa_tl
      { \str_uppercase:n { #1 } }
    \tl_set:Nx
      #2
      {
        markdownRenderer
        \tl_head:f { \l_tmpa_tl }
        \tl_tail:n { #1 }
        Prototype
      }
  }
\cs_new:Nn \@@_plaintex_define_renderer_prototype:Nn
  {
    \cs_generate_from_arg_count:NNnn
      #1
      \cs_set:Npn
      { #2 }
      { }
  }
\cs_generate_variant:Nn
  \@@_plaintex_define_renderer_prototype:Nn
  { cV }
\@@_plaintex_define_renderer_prototypes:
\ExplSyntaxOff
```

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 3.2.6), 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
- `3` – The lt3luabridge Lua package

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.

The `\markdownMode` macro has been deprecated and will be removed in Markdown 3.0.0. The code that corresponds to `\markdownMode` value of `3` will be the only implementation.

```
\ExplSyntaxOn
\cs_if_exist:NF
  \markdownMode
  {
    \file_if_exist:nTF
```

```
      { lt3luabridge.tex }
      {
        \cs_new:Npn
          \markdownMode
          { 3 }
      }
      {
        \cs_if_exist:NTF
          \directlua
          {
            \cs_new:Npn
              \markdownMode
              { 2 }
          }
          {
            \cs_new:Npn
              \markdownMode
              { 0 }
          }
      }
  }
\int_compare:nF
  { \markdownMode = 3 }
  {
    \int_new:N
      \g_luabridge_method_int
    \int_gset:Nn
      \g_luabridge_method_int
      { \markdownMode }
  }
\ExplSyntaxOff
```

The `\markdownLuaRegisterIBCallback` and `\markdownLuaUnregisterIBCallback` macros have been deprecated and will be removed in Markdown 3.0.0:

```
\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 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
```

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 2_ε 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}
```

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

```
% ...
\end{document}
```

```
% ...
\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:

```
\ExplSyntaxOn
\cs_new:Nn
  \@@_setup:n
  {
    \keys_set:nn
      { markdown/latex-options }
      { #1 }
  }
\let\markdownSetup=\@@_setup:n
\ExplSyntaxOff
```

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.

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

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

To enable the reflection of default LaTeX options and their types, we will maintain the `\g_@@_default_latex_options_prop` and `\g_@@_latex_option_types_prop` property lists, respectively.

```
\prop_new:N \g_@@_latex_option_types_prop
\prop_new:N \g_@@_default_latex_options_prop
\tl_const:Nn \c_@@_option_layer_latex_tl { latex }
\seq_put_right:NV \g_@@_option_layers_seq \c_@@_option_layer_latex_tl
\cs_new:Nn
  \@@_add_latex_option:nnn
  {
    \@@_add_option:Vnnn
      \c_@@_option_layer_latex_tl
      { #1 }
      { #2 }
      { #3 }
  }
```

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 3.2.2) and prevent the soft LaTeX prerequisites in Section 1.1.3 from being loaded: The plain option must be set before or when loading the package. Setting the option after loading the package will have no effect.

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

```
\@@_add_latex_option:nnn
  { plain }
  { boolean }
  { false }
\ExplSyntaxOff
```

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}
\ExplSyntaxOn
\tl_new:N \markdownLaTeXThemePackageName
\cs_new:Nn
  \@@_set_latex_theme:n
  {
    \str_if_in:NnF
      { #1 }
      { / }
      {
        \markdownError
        { Won't load theme with unqualified name #1 }
        { Theme names must contain at least one forward slash }
      }
    \tl_set:Nn \markdownLaTeXThemePackageName { #1 }
    \str_replace_all:Nnn
      \markdownLaTeXThemePackageName
      { / }
      { _ }
    \edef\markdownLaTeXThemePackageName{
      markdowntheme\markdownLaTeXThemePackageName}
    \expandafter\markdownLaTeXThemeLoad\expandafter{
      \markdownLaTeXThemePackageName}{#1/}
  }
\keys_define:nn
  { markdown/latex-options }
  {
    theme .code:n = { \@@_set_latex_theme:n { #1 } },
  }
\ExplSyntaxOff
```

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.

```
\ExplSyntaxOn
\@onlypreamble
  \@@_set_latex_theme:n
\ExplSyntaxOff
```

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.

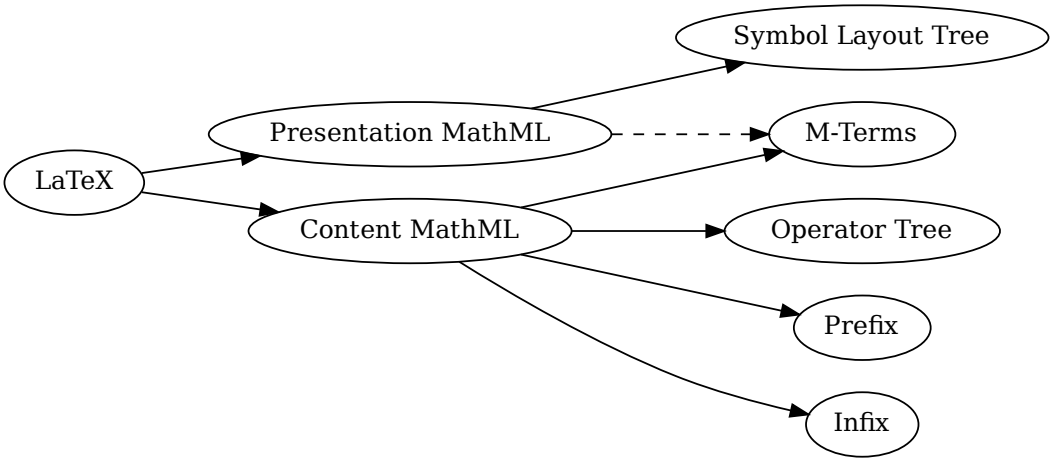


**Figure 4: Various formats of mathemathical formulae**

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}
\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

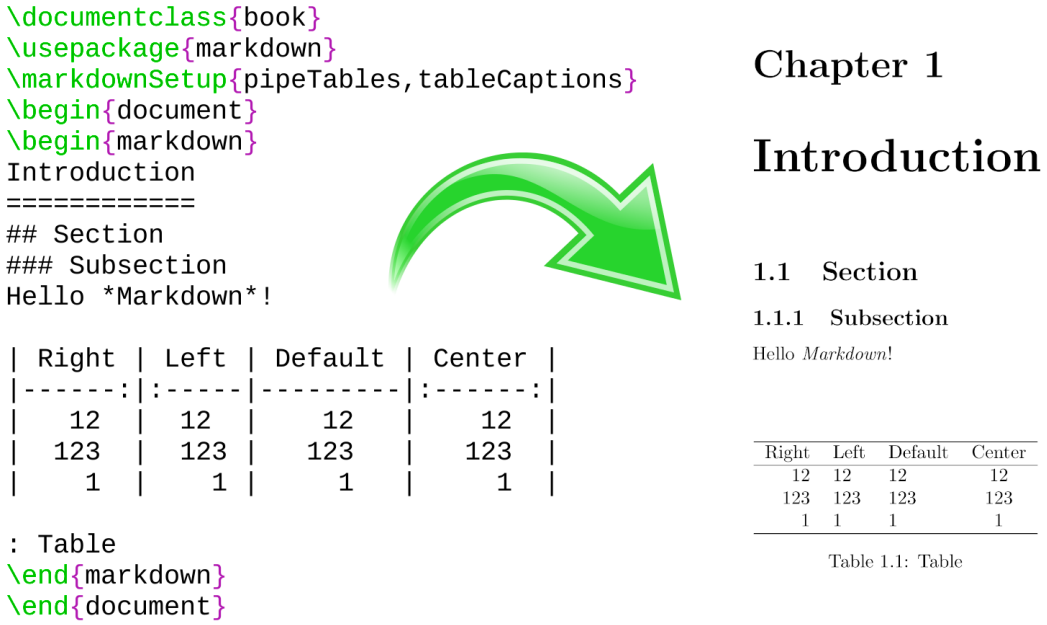

Figure 5: The banner of the Markdown package

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.

```
1796 \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}
```

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*⟩:

```
\ExplSyntaxOn
\keys_define:nn
  { markdown/latex-options }
  {
    snippet .code:n = {
      \markdownIfSnippetExists{#1}
        {
          \expandafter\markdownSetup\expandafter{
            \the\csname markdownLaTeXSetupSnippet
            \markdownLaTeXThemeName#1\endcsname}
        }{
          \markdownError
            {Can't~invoke~setup~snippet~#1}
            {The~setup~snippet~is~undefined}
        }
    }
  }
\ExplSyntaxOff
```

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

Here, we automatically define plain TEX macros and the ⟨*key*⟩=⟨*value*⟩ interface for the above LATEX options.

```
\ExplSyntaxOn
\cs_new:Nn \@@_latex_define_option_commands_and_keyvals:
  {
    \seq_map_inline:Nn
      \g_@@_latex_options_seq
      {
          \@@_plain_tex_define_option_command:n
            { ##1 }
      }
```

Furthermore, we also define the ⟨*key*⟩=⟨*value*⟩ interface for all option macros recognized by the Lua plain TEX interfaces.

```
    \seq_map_inline:Nn
      \g_@@_option_layers_seq
      {
        \seq_map_inline:cn
          { g_@@_ ##1 _options_seq }
          {
              \@@_latex_define_option_keyval:nn
                { ##1 }
                { ####1 }
          }
      }
  }
\cs_new:Nn \@@_latex_define_option_keyval:nn
  {
    \prop_get:cnN
      { g_@@_ #1 _option_types_prop }
      { #2 }
      \l_tmpa_tl
    \keys_define:nn
      { markdown/latex-options }
      {
```

```
        #2 .code:n = {
          \@@_set_option_value:nn
            { #2 }
            { ##1 }
        },
      }
    \str_if_eq:VVT
      \l_tmpa_tl
      \c_@@_option_type_boolean_tl
      {
        \keys_define:nn
          { markdown/latex-options }
          {
            #2 .default:n = { true },
          }
      }
  }
\@@_latex_define_option_commands_and_keyvals:
\ExplSyntaxOff
```

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.

```
\DeclareOption{finalizecache}{\markdownSetup{finalizeCache}}
\DeclareOption{frozencache}{\markdownSetup{frozenCache}}
```

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
  {
    \@@_renderer_tl_to_csname:nN
      { #1 }
      \l_tmpa_tl
    \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 \@@_renderer_tl_to_csname:nN
  {
    \tl_set:Nn
      \l_tmpa_tl
      { \str_uppercase:n { #1 } }
    \tl_set:Nx
      #2
      {
        markdownRenderer
        \tl_head:f { \l_tmpa_tl }
        \tl_tail:n { #1 }
      }
  }
\cs_new:Nn \@@_latex_define_renderer:nNn
  {
    \keys_define:nn
```

```
      { markdown/latex-options/renderers }
      {
        #1 .code:n = {
          \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
  {
    \@@_renderer_prototype_tl_to_csname:nN
      { #1 }
      \l_tmpa_tl
    \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
  {
    \keys_define:nn
      { markdown/latex-options/renderer-prototypes }
      {
        #1 .code:n = {
          \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, which provide the conversion from markdown to plain TEX, and `extension` objects, which provide syntax extensions for the `writer` and `reader` objects.

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).

```
1963 local upper, gsub, format, length =
1964   string.upper, string.gsub, string.format, string.len
1965 local P, R, S, V, C, Cg, Cb, Cmt, Cc, Ct, B, Cs, any =
1966   lpeg.P, lpeg.R, lpeg.S, lpeg.V, lpeg.C, lpeg.Cg, lpeg.Cb,
1967   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.

```
1968 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.

```
1969 function util.err(msg, exit_code)
1970   io.stderr:write("markdown.lua: " .. msg .. "\n")
1971   os.exit(exit_code or 1)
1972 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.

```
1973 function util.cache(dir, string, salt, transform, suffix)
1974   local digest = md5.sumhexa(string .. (salt or ""))
1975   local name = util.pathname(dir, digest .. suffix)
1976   local file = io.open(name, "r")
1977   if file == nil then -- If no cache entry exists, then create a new one.
1978     file = assert(io.open(name, "w"),
1979       [[could not open file "]] .. name .. [[" for writing]])
1980     local result = string
1981     if transform ~= nil then
1982       result = transform(result)
1983     end
1984     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 $(\texttt{k}, \texttt{v}) \in \texttt{string\_escapes}$. 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.

```
  if string_escapes then
    for k,v in pairs(string_escapes) do
      escapable = P(k) / v + escapable
    end
  end
```

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

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

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

```
  return function(s)
    return lpeg.match(escape_string, s)
  end
end
```

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

```
function util.pathname(dir, file)
  if #dir == 0 then
    return file
  else
    return dir .. "/" .. file
  end
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.

```
local entities = {}

local character_entities = {
  ["Tab"] = 9,
  ["NewLine"] = 10,
  ["excl"] = 33,
  ["quot"] = 34,
  ["QUOT"] = 34,
  ["num"] = 35,
  ["dollar"] = 36,
  ["percnt"] = 37,
  ["amp"] = 38,
  ["AMP"] = 38,
  ["apos"] = 39,
  ["lpar"] = 40,
```

```
2112  ["rpar"] = 41,
2113  ["ast"] = 42,
2114  ["midast"] = 42,
2115  ["plus"] = 43,
2116  ["comma"] = 44,
2117  ["period"] = 46,
2118  ["sol"] = 47,
2119  ["colon"] = 58,
2120  ["semi"] = 59,
2121  ["lt"] = 60,
2122  ["LT"] = 60,
2123  ["equals"] = 61,
2124  ["gt"] = 62,
2125  ["GT"] = 62,
2126  ["quest"] = 63,
2127  ["commat"] = 64,
2128  ["lsqb"] = 91,
2129  ["lbrack"] = 91,
2130  ["bsol"] = 92,
2131  ["rsqb"] = 93,
2132  ["rbrack"] = 93,
2133  ["Hat"] = 94,
2134  ["lowbar"] = 95,
2135  ["grave"] = 96,
2136  ["DiacriticalGrave"] = 96,
2137  ["lcub"] = 123,
2138  ["lbrace"] = 123,
2139  ["verbar"] = 124,
2140  ["vert"] = 124,
2141  ["VerticalLine"] = 124,
2142  ["rcub"] = 125,
2143  ["rbrace"] = 125,
2144  ["nbsp"] = 160,
2145  ["NonBreakingSpace"] = 160,
2146  ["iexcl"] = 161,
2147  ["cent"] = 162,
2148  ["pound"] = 163,
2149  ["curren"] = 164,
2150  ["yen"] = 165,
2151  ["brvbar"] = 166,
2152  ["sect"] = 167,
2153  ["Dot"] = 168,
2154  ["die"] = 168,
2155  ["DoubleDot"] = 168,
2156  ["uml"] = 168,
2157  ["copy"] = 169,
2158  ["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,
```

```
2206   ["Icirc"] = 206,
2207   ["Iuml"] = 207,
2208   ["ETH"] = 208,
2209   ["Ntilde"] = 209,
2210   ["Ograve"] = 210,
2211   ["Oacute"] = 211,
2212   ["Ocirc"] = 212,
2213   ["Otilde"] = 213,
2214   ["Ouml"] = 214,
2215   ["times"] = 215,
2216   ["Oslash"] = 216,
2217   ["Ugrave"] = 217,
2218   ["Uacute"] = 218,
2219   ["Ucirc"] = 219,
2220   ["Uuml"] = 220,
2221   ["Yacute"] = 221,
2222   ["THORN"] = 222,
2223   ["szlig"] = 223,
2224   ["agrave"] = 224,
2225   ["aacute"] = 225,
2226   ["acirc"] = 226,
2227   ["atilde"] = 227,
2228   ["auml"] = 228,
2229   ["aring"] = 229,
2230   ["aelig"] = 230,
2231   ["ccedil"] = 231,
2232   ["egrave"] = 232,
2233   ["eacute"] = 233,
2234   ["ecirc"] = 234,
2235   ["euml"] = 235,
2236   ["igrave"] = 236,
2237   ["iacute"] = 237,
2238   ["icirc"] = 238,
2239   ["iuml"] = 239,
2240   ["eth"] = 240,
2241   ["ntilde"] = 241,
2242   ["ograve"] = 242,
2243   ["oacute"] = 243,
2244   ["ocirc"] = 244,
2245   ["otilde"] = 245,
2246   ["ouml"] = 246,
2247   ["divide"] = 247,
2248   ["div"] = 247,
2249   ["oslash"] = 248,
2250   ["ugrave"] = 249,
2251   ["uacute"] = 250,
2252   ["ucirc"] = 251,
```

```
2253   ["uuml"] = 252,
2254   ["yacute"] = 253,
2255   ["thorn"] = 254,
2256   ["yuml"] = 255,
2257   ["Amacr"] = 256,
2258   ["amacr"] = 257,
2259   ["Abreve"] = 258,
2260   ["abreve"] = 259,
2261   ["Aogon"] = 260,
2262   ["aogon"] = 261,
2263   ["Cacute"] = 262,
2264   ["cacute"] = 263,
2265   ["Ccirc"] = 264,
2266   ["ccirc"] = 265,
2267   ["Cdot"] = 266,
2268   ["cdot"] = 267,
2269   ["Ccaron"] = 268,
2270   ["ccaron"] = 269,
2271   ["Dcaron"] = 270,
2272   ["dcaron"] = 271,
2273   ["Dstrok"] = 272,
2274   ["dstrok"] = 273,
2275   ["Emacr"] = 274,
2276   ["emacr"] = 275,
2277   ["Edot"] = 278,
2278   ["edot"] = 279,
2279   ["Eogon"] = 280,
2280   ["eogon"] = 281,
2281   ["Ecaron"] = 282,
2282   ["ecaron"] = 283,
2283   ["Gcirc"] = 284,
2284   ["gcirc"] = 285,
2285   ["Gbreve"] = 286,
2286   ["gbreve"] = 287,
2287   ["Gdot"] = 288,
2288   ["gdot"] = 289,
2289   ["Gcedil"] = 290,
2290   ["Hcirc"] = 292,
2291   ["hcirc"] = 293,
2292   ["Hstrok"] = 294,
2293   ["hstrok"] = 295,
2294   ["Itilde"] = 296,
2295   ["itilde"] = 297,
2296   ["Imacr"] = 298,
2297   ["imacr"] = 299,
2298   ["Iogon"] = 302,
2299   ["iogon"] = 303,
```

```
2300    ["Idot"] = 304,
2301    ["imath"] = 305,
2302    ["inodot"] = 305,
2303    ["IJlig"] = 306,
2304    ["ijlig"] = 307,
2305    ["Jcirc"] = 308,
2306    ["jcirc"] = 309,
2307    ["Kcedil"] = 310,
2308    ["kcedil"] = 311,
2309    ["kgreen"] = 312,
2310    ["Lacute"] = 313,
2311    ["lacute"] = 314,
2312    ["Lcedil"] = 315,
2313    ["lcedil"] = 316,
2314    ["Lcaron"] = 317,
2315    ["lcaron"] = 318,
2316    ["Lmidot"] = 319,
2317    ["lmidot"] = 320,
2318    ["Lstrok"] = 321,
2319    ["lstrok"] = 322,
2320    ["Nacute"] = 323,
2321    ["nacute"] = 324,
2322    ["Ncedil"] = 325,
2323    ["ncedil"] = 326,
2324    ["Ncaron"] = 327,
2325    ["ncaron"] = 328,
2326    ["napos"] = 329,
2327    ["ENG"] = 330,
2328    ["eng"] = 331,
2329    ["Omacr"] = 332,
2330    ["omacr"] = 333,
2331    ["Odblac"] = 336,
2332    ["odblac"] = 337,
2333    ["OElig"] = 338,
2334    ["oelig"] = 339,
2335    ["Racute"] = 340,
2336    ["racute"] = 341,
2337    ["Rcedil"] = 342,
2338    ["rcedil"] = 343,
2339    ["Rcaron"] = 344,
2340    ["rcaron"] = 345,
2341    ["Sacute"] = 346,
2342    ["sacute"] = 347,
2343    ["Scirc"] = 348,
2344    ["scirc"] = 349,
2345    ["Scedil"] = 350,
2346    ["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,
  ["ogon"] = 731,
  ["tilde"] = 732,
  ["DiacriticalTilde"] = 732,
  ["dblac"] = 733,
```

```
2394  ["DiacriticalDoubleAcute"] = 733,
2395  ["DownBreve"] = 785,
2396  ["UnderBar"] = 818,
2397  ["Alpha"] = 913,
2398  ["Beta"] = 914,
2399  ["Gamma"] = 915,
2400  ["Delta"] = 916,
2401  ["Epsilon"] = 917,
2402  ["Zeta"] = 918,
2403  ["Eta"] = 919,
2404  ["Theta"] = 920,
2405  ["Iota"] = 921,
2406  ["Kappa"] = 922,
2407  ["Lambda"] = 923,
2408  ["Mu"] = 924,
2409  ["Nu"] = 925,
2410  ["Xi"] = 926,
2411  ["Omicron"] = 927,
2412  ["Pi"] = 928,
2413  ["Rho"] = 929,
2414  ["Sigma"] = 931,
2415  ["Tau"] = 932,
2416  ["Upsilon"] = 933,
2417  ["Phi"] = 934,
2418  ["Chi"] = 935,
2419  ["Psi"] = 936,
2420  ["Omega"] = 937,
2421  ["alpha"] = 945,
2422  ["beta"] = 946,
2423  ["gamma"] = 947,
2424  ["delta"] = 948,
2425  ["epsiv"] = 949,
2426  ["varepsilon"] = 949,
2427  ["epsilon"] = 949,
2428  ["zeta"] = 950,
2429  ["eta"] = 951,
2430  ["theta"] = 952,
2431  ["iota"] = 953,
2432  ["kappa"] = 954,
2433  ["lambda"] = 955,
2434  ["mu"] = 956,
2435  ["nu"] = 957,
2436  ["xi"] = 958,
2437  ["omicron"] = 959,
2438  ["pi"] = 960,
2439  ["rho"] = 961,
2440  ["sigmav"] = 962,
```

```
2441   ["varsigma"] = 962,
2442   ["sigmaf"] = 962,
2443   ["sigma"] = 963,
2444   ["tau"] = 964,
2445   ["upsi"] = 965,
2446   ["upsilon"] = 965,
2447   ["phi"] = 966,
2448   ["phiv"] = 966,
2449   ["varphi"] = 966,
2450   ["chi"] = 967,
2451   ["psi"] = 968,
2452   ["omega"] = 969,
2453   ["thetav"] = 977,
2454   ["vartheta"] = 977,
2455   ["thetasym"] = 977,
2456   ["Upsi"] = 978,
2457   ["upsih"] = 978,
2458   ["straightphi"] = 981,
2459   ["piv"] = 982,
2460   ["varpi"] = 982,
2461   ["Gammad"] = 988,
2462   ["gammad"] = 989,
2463   ["digamma"] = 989,
2464   ["kappav"] = 1008,
2465   ["varkappa"] = 1008,
2466   ["rhov"] = 1009,
2467   ["varrho"] = 1009,
2468   ["epsi"] = 1013,
2469   ["straightepsilon"] = 1013,
2470   ["bepsi"] = 1014,
2471   ["backepsilon"] = 1014,
2472   ["IOcy"] = 1025,
2473   ["DJcy"] = 1026,
2474   ["GJcy"] = 1027,
2475   ["Jukcy"] = 1028,
2476   ["DScy"] = 1029,
2477   ["Iukcy"] = 1030,
2478   ["YIcy"] = 1031,
2479   ["Jsercy"] = 1032,
2480   ["LJcy"] = 1033,
2481   ["NJcy"] = 1034,
2482   ["TSHcy"] = 1035,
2483   ["KJcy"] = 1036,
2484   ["Ubrcy"] = 1038,
2485   ["DZcy"] = 1039,
2486   ["Acy"] = 1040,
2487   ["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,
```

```
2535    ["scy"] = 1089,
2536    ["tcy"] = 1090,
2537    ["ucy"] = 1091,
2538    ["fcy"] = 1092,
2539    ["khcy"] = 1093,
2540    ["tscy"] = 1094,
2541    ["chcy"] = 1095,
2542    ["shcy"] = 1096,
2543    ["shchcy"] = 1097,
2544    ["hardcy"] = 1098,
2545    ["ycy"] = 1099,
2546    ["softcy"] = 1100,
2547    ["ecy"] = 1101,
2548    ["yucy"] = 1102,
2549    ["yacy"] = 1103,
2550    ["iocy"] = 1105,
2551    ["djcy"] = 1106,
2552    ["gjcy"] = 1107,
2553    ["jukcy"] = 1108,
2554    ["dscy"] = 1109,
2555    ["iukcy"] = 1110,
2556    ["yicy"] = 1111,
2557    ["jsercy"] = 1112,
2558    ["ljcy"] = 1113,
2559    ["njcy"] = 1114,
2560    ["tshcy"] = 1115,
2561    ["kjcy"] = 1116,
2562    ["ubrcy"] = 1118,
2563    ["dzcy"] = 1119,
2564    ["ensp"] = 8194,
2565    ["emsp"] = 8195,
2566    ["emsp13"] = 8196,
2567    ["emsp14"] = 8197,
2568    ["numsp"] = 8199,
2569    ["puncsp"] = 8200,
2570    ["thinsp"] = 8201,
2571    ["ThinSpace"] = 8201,
2572    ["hairsp"] = 8202,
2573    ["VeryThinSpace"] = 8202,
2574    ["ZeroWidthSpace"] = 8203,
2575    ["NegativeVeryThinSpace"] = 8203,
2576    ["NegativeThinSpace"] = 8203,
2577    ["NegativeMediumSpace"] = 8203,
2578    ["NegativeThickSpace"] = 8203,
2579    ["zwnj"] = 8204,
2580    ["zwj"] = 8205,
2581    ["lrm"] = 8206,
```

```
["rlm"] = 8207,
["hyphen"] = 8208,
["dash"] = 8208,
["ndash"] = 8211,
["mdash"] = 8212,
["horbar"] = 8213,
["Verbar"] = 8214,
["Vert"] = 8214,
["lsquo"] = 8216,
["OpenCurlyQuote"] = 8216,
["rsquo"] = 8217,
["rsquor"] = 8217,
["CloseCurlyQuote"] = 8217,
["lsquor"] = 8218,
["sbquo"] = 8218,
["ldquo"] = 8220,
["OpenCurlyDoubleQuote"] = 8220,
["rdquo"] = 8221,
["rdquor"] = 8221,
["CloseCurlyDoubleQuote"] = 8221,
["ldquor"] = 8222,
["bdquo"] = 8222,
["dagger"] = 8224,
["Dagger"] = 8225,
["ddagger"] = 8225,
["bull"] = 8226,
["bullet"] = 8226,
["nldr"] = 8229,
["hellip"] = 8230,
["mldr"] = 8230,
["permil"] = 8240,
["pertenk"] = 8241,
["prime"] = 8242,
["Prime"] = 8243,
["tprime"] = 8244,
["bprime"] = 8245,
["backprime"] = 8245,
["lsaquo"] = 8249,
["rsaquo"] = 8250,
["oline"] = 8254,
["caret"] = 8257,
["hybull"] = 8259,
["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,
```

```
2676  ["realine"] = 8475,
2677  ["real"] = 8476,
2678  ["Re"] = 8476,
2679  ["realpart"] = 8476,
2680  ["Rfr"] = 8476,
2681  ["reals"] = 8477,
2682  ["Ropf"] = 8477,
2683  ["rx"] = 8478,
2684  ["trade"] = 8482,
2685  ["TRADE"] = 8482,
2686  ["integers"] = 8484,
2687  ["Zopf"] = 8484,
2688  ["ohm"] = 8486,
2689  ["mho"] = 8487,
2690  ["Zfr"] = 8488,
2691  ["zeetrf"] = 8488,
2692  ["iiota"] = 8489,
2693  ["angst"] = 8491,
2694  ["bernou"] = 8492,
2695  ["Bernoullis"] = 8492,
2696  ["Bscr"] = 8492,
2697  ["Cfr"] = 8493,
2698  ["Cayleys"] = 8493,
2699  ["escr"] = 8495,
2700  ["Escr"] = 8496,
2701  ["expectation"] = 8496,
2702  ["Fscr"] = 8497,
2703  ["Fouriertrf"] = 8497,
2704  ["phmmat"] = 8499,
2705  ["Mellintrf"] = 8499,
2706  ["Mscr"] = 8499,
2707  ["order"] = 8500,
2708  ["orderof"] = 8500,
2709  ["oscr"] = 8500,
2710  ["alefsym"] = 8501,
2711  ["aleph"] = 8501,
2712  ["beth"] = 8502,
2713  ["gimel"] = 8503,
2714  ["daleth"] = 8504,
2715  ["CapitalDifferentialD"] = 8517,
2716  ["DD"] = 8517,
2717  ["DifferentialD"] = 8518,
2718  ["dd"] = 8518,
2719  ["ExponentialE"] = 8519,
2720  ["exponentiale"] = 8519,
2721  ["ee"] = 8519,
2722  ["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,
```

```
2770  ["swarrow"] = 8601,
2771  ["LowerLeftArrow"] = 8601,
2772  ["nlarr"] = 8602,
2773  ["nleftarrow"] = 8602,
2774  ["nrarr"] = 8603,
2775  ["nrightarrow"] = 8603,
2776  ["rarrw"] = 8605,
2777  ["rightsquigarrow"] = 8605,
2778  ["Larr"] = 8606,
2779  ["twoheadleftarrow"] = 8606,
2780  ["Uarr"] = 8607,
2781  ["Rarr"] = 8608,
2782  ["twoheadrightarrow"] = 8608,
2783  ["Darr"] = 8609,
2784  ["larrtl"] = 8610,
2785  ["leftarrowtail"] = 8610,
2786  ["rarrtl"] = 8611,
2787  ["rightarrowtail"] = 8611,
2788  ["LeftTeeArrow"] = 8612,
2789  ["mapstoleft"] = 8612,
2790  ["UpTeeArrow"] = 8613,
2791  ["mapstoup"] = 8613,
2792  ["map"] = 8614,
2793  ["RightTeeArrow"] = 8614,
2794  ["mapsto"] = 8614,
2795  ["DownTeeArrow"] = 8615,
2796  ["mapstodown"] = 8615,
2797  ["larrhk"] = 8617,
2798  ["hookleftarrow"] = 8617,
2799  ["rarrhk"] = 8618,
2800  ["hookrightarrow"] = 8618,
2801  ["larrlp"] = 8619,
2802  ["looparrowleft"] = 8619,
2803  ["rarrlp"] = 8620,
2804  ["looparrowright"] = 8620,
2805  ["harrw"] = 8621,
2806  ["leftrightsquigarrow"] = 8621,
2807  ["nharr"] = 8622,
2808  ["nleftrightarrow"] = 8622,
2809  ["lsh"] = 8624,
2810  ["Lsh"] = 8624,
2811  ["rsh"] = 8625,
2812  ["Rsh"] = 8625,
2813  ["ldsh"] = 8626,
2814  ["rdsh"] = 8627,
2815  ["crarr"] = 8629,
2816  ["cularr"] = 8630,
```

```
["curvearrowleft"] = 8630,
["curarr"] = 8631,
["curvearrowright"] = 8631,
["olarr"] = 8634,
["circlearrowleft"] = 8634,
["orarr"] = 8635,
["circlearrowright"] = 8635,
["lharu"] = 8636,
["LeftVector"] = 8636,
["leftharpoonup"] = 8636,
["lhard"] = 8637,
["leftharpoondown"] = 8637,
["DownLeftVector"] = 8637,
["uharr"] = 8638,
["upharpoonright"] = 8638,
["RightUpVector"] = 8638,
["uharl"] = 8639,
["upharpoonleft"] = 8639,
["LeftUpVector"] = 8639,
["rharu"] = 8640,
["RightVector"] = 8640,
["rightharpoonup"] = 8640,
["rhard"] = 8641,
["rightharpoondown"] = 8641,
["DownRightVector"] = 8641,
["dharr"] = 8642,
["RightDownVector"] = 8642,
["downharpoonright"] = 8642,
["dharl"] = 8643,
["LeftDownVector"] = 8643,
["downharpoonleft"] = 8643,
["rlarr"] = 8644,
["rightleftarrows"] = 8644,
["RightArrowLeftArrow"] = 8644,
["udarr"] = 8645,
["UpArrowDownArrow"] = 8645,
["lrarr"] = 8646,
["leftrightarrows"] = 8646,
["LeftArrowRightArrow"] = 8646,
["llarr"] = 8647,
["leftleftarrows"] = 8647,
["uuarr"] = 8648,
["upuparrows"] = 8648,
["rrarr"] = 8649,
["rightrightarrows"] = 8649,
["ddarr"] = 8650,
["downdownarrows"] = 8650,
```

```
["lrhar"] = 8651,
["ReverseEquilibrium"] = 8651,
["leftrightharpoons"] = 8651,
["rlhar"] = 8652,
["rightleftharpoons"] = 8652,
["Equilibrium"] = 8652,
["nlArr"] = 8653,
["nLeftarrow"] = 8653,
["nhArr"] = 8654,
["nLeftrightarrow"] = 8654,
["nrArr"] = 8655,
["nRightarrow"] = 8655,
["lArr"] = 8656,
["Leftarrow"] = 8656,
["DoubleLeftArrow"] = 8656,
["uArr"] = 8657,
["Uparrow"] = 8657,
["DoubleUpArrow"] = 8657,
["rArr"] = 8658,
["Rightarrow"] = 8658,
["Implies"] = 8658,
["DoubleRightArrow"] = 8658,
["dArr"] = 8659,
["Downarrow"] = 8659,
["DoubleDownArrow"] = 8659,
["hArr"] = 8660,
["Leftrightarrow"] = 8660,
["DoubleLeftRightArrow"] = 8660,
["iff"] = 8660,
["vArr"] = 8661,
["Updownarrow"] = 8661,
["DoubleUpDownArrow"] = 8661,
["nwArr"] = 8662,
["neArr"] = 8663,
["seArr"] = 8664,
["swArr"] = 8665,
["lAarr"] = 8666,
["Lleftarrow"] = 8666,
["rAarr"] = 8667,
["Rrightarrow"] = 8667,
["zigrarr"] = 8669,
["larrb"] = 8676,
["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,
["MinusPlus"] = 8723,
["plusdo"] = 8724,
["dotplus"] = 8724,
["setmn"] = 8726,
```

```
["setminus"] = 8726,
["Backslash"] = 8726,
["ssetmn"] = 8726,
["smallsetminus"] = 8726,
["lowast"] = 8727,
["compfn"] = 8728,
["SmallCircle"] = 8728,
["radic"] = 8730,
["Sqrt"] = 8730,
["prop"] = 8733,
["propto"] = 8733,
["Proportional"] = 8733,
["vprop"] = 8733,
["varpropto"] = 8733,
["infin"] = 8734,
["angrt"] = 8735,
["ang"] = 8736,
["angle"] = 8736,
["angmsd"] = 8737,
["measuredangle"] = 8737,
["angsph"] = 8738,
["mid"] = 8739,
["VerticalBar"] = 8739,
["smid"] = 8739,
["shortmid"] = 8739,
["nmid"] = 8740,
["NotVerticalBar"] = 8740,
["nsmid"] = 8740,
["nshortmid"] = 8740,
["par"] = 8741,
["parallel"] = 8741,
["DoubleVerticalBar"] = 8741,
["spar"] = 8741,
["shortparallel"] = 8741,
["npar"] = 8742,
["nparallel"] = 8742,
["NotDoubleVerticalBar"] = 8742,
["nspar"] = 8742,
["nshortparallel"] = 8742,
["and"] = 8743,
["wedge"] = 8743,
["or"] = 8744,
["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,
["ominus"] = 8854,
["CircleMinus"] = 8854,
["otimes"] = 8855,
["CircleTimes"] = 8855,
```

```
["osol"] = 8856,
["odot"] = 8857,
["CircleDot"] = 8857,
["ocir"] = 8858,
["circledcirc"] = 8858,
["oast"] = 8859,
["circledast"] = 8859,
["odash"] = 8861,
["circleddash"] = 8861,
["plusb"] = 8862,
["boxplus"] = 8862,
["minusb"] = 8863,
["boxminus"] = 8863,
["timesb"] = 8864,
["boxtimes"] = 8864,
["sdotb"] = 8865,
["dotsquare"] = 8865,
["vdash"] = 8866,
["RightTee"] = 8866,
["dashv"] = 8867,
["LeftTee"] = 8867,
["top"] = 8868,
["DownTee"] = 8868,
["bottom"] = 8869,
["bot"] = 8869,
["perp"] = 8869,
["UpTee"] = 8869,
["models"] = 8871,
["vDash"] = 8872,
["DoubleRightTee"] = 8872,
["Vdash"] = 8873,
["Vvdash"] = 8874,
["VDash"] = 8875,
["nvdash"] = 8876,
["nvDash"] = 8877,
["nVdash"] = 8878,
["nVDash"] = 8879,
["prurel"] = 8880,
["vltri"] = 8882,
["vartriangleleft"] = 8882,
["LeftTriangle"] = 8882,
["vrtri"] = 8883,
["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,
  ["bigtriangleup"] = 9651,
  ["utrif"] = 9652,
  ["blacktriangle"] = 9652,
  ["utri"] = 9653,
```

```
  ["triangle"] = 9653,
  ["rtrif"] = 9656,
  ["blacktriangleright"] = 9656,
  ["rtri"] = 9657,
  ["triangleright"] = 9657,
  ["xdtri"] = 9661,
  ["bigtriangledown"] = 9661,
  ["dtrif"] = 9662,
  ["blacktriangledown"] = 9662,
  ["dtri"] = 9663,
  ["triangledown"] = 9663,
  ["ltrif"] = 9666,
  ["blacktriangleleft"] = 9666,
  ["ltri"] = 9667,
  ["triangleleft"] = 9667,
  ["loz"] = 9674,
  ["lozenge"] = 9674,
  ["cir"] = 9675,
  ["tridot"] = 9708,
  ["xcirc"] = 9711,
  ["bigcirc"] = 9711,
  ["ultri"] = 9720,
  ["urtri"] = 9721,
  ["lltri"] = 9722,
  ["EmptySmallSquare"] = 9723,
  ["FilledSmallSquare"] = 9724,
  ["starf"] = 9733,
  ["bigstar"] = 9733,
  ["star"] = 9734,
  ["phone"] = 9742,
  ["female"] = 9792,
  ["male"] = 9794,
  ["spades"] = 9824,
  ["spadesuit"] = 9824,
  ["clubs"] = 9827,
  ["clubsuit"] = 9827,
  ["hearts"] = 9829,
  ["heartsuit"] = 9829,
  ["diams"] = 9830,
  ["diamondsuit"] = 9830,
  ["sung"] = 9834,
  ["flat"] = 9837,
  ["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,
```

```
3710    ["lopar"] = 10629,
3711    ["ropar"] = 10630,
3712    ["lbrke"] = 10635,
3713    ["rbrke"] = 10636,
3714    ["lbrkslu"] = 10637,
3715    ["rbrksld"] = 10638,
3716    ["lbrksld"] = 10639,
3717    ["rbrkslu"] = 10640,
3718    ["langd"] = 10641,
3719    ["rangd"] = 10642,
3720    ["lparlt"] = 10643,
3721    ["rpargt"] = 10644,
3722    ["gtlPar"] = 10645,
3723    ["ltrPar"] = 10646,
3724    ["vzigzag"] = 10650,
3725    ["vangrt"] = 10652,
3726    ["angrtvbd"] = 10653,
3727    ["ange"] = 10660,
3728    ["range"] = 10661,
3729    ["dwangle"] = 10662,
3730    ["uwangle"] = 10663,
3731    ["angmsdaa"] = 10664,
3732    ["angmsdab"] = 10665,
3733    ["angmsdac"] = 10666,
3734    ["angmsdad"] = 10667,
3735    ["angmsdae"] = 10668,
3736    ["angmsdaf"] = 10669,
3737    ["angmsdag"] = 10670,
3738    ["angmsdah"] = 10671,
3739    ["bemptyv"] = 10672,
3740    ["demptyv"] = 10673,
3741    ["cemptyv"] = 10674,
3742    ["raemptyv"] = 10675,
3743    ["laemptyv"] = 10676,
3744    ["ohbar"] = 10677,
3745    ["omid"] = 10678,
3746    ["opar"] = 10679,
3747    ["operp"] = 10681,
3748    ["olcross"] = 10683,
3749    ["odsold"] = 10684,
3750    ["olcir"] = 10686,
3751    ["ofcir"] = 10687,
3752    ["olt"] = 10688,
3753    ["ogt"] = 10689,
3754    ["cirscir"] = 10690,
3755    ["cirE"] = 10691,
3756    ["solb"] = 10692,
```

```
3757  ["bsolb"] = 10693,
3758  ["boxbox"] = 10697,
3759  ["trisb"] = 10701,
3760  ["rtriltri"] = 10702,
3761  ["LeftTriangleBar"] = 10703,
3762  ["RightTriangleBar"] = 10704,
3763  ["race"] = 10714,
3764  ["iinfin"] = 10716,
3765  ["infintie"] = 10717,
3766  ["nvinfin"] = 10718,
3767  ["eparsl"] = 10723,
3768  ["smeparsl"] = 10724,
3769  ["eqvparsl"] = 10725,
3770  ["lozf"] = 10731,
3771  ["blacklozenge"] = 10731,
3772  ["RuleDelayed"] = 10740,
3773  ["dsol"] = 10742,
3774  ["xodot"] = 10752,
3775  ["bigodot"] = 10752,
3776  ["xoplus"] = 10753,
3777  ["bigoplus"] = 10753,
3778  ["xotime"] = 10754,
3779  ["bigotimes"] = 10754,
3780  ["xuplus"] = 10756,
3781  ["biguplus"] = 10756,
3782  ["xsqcup"] = 10758,
3783  ["bigsqcup"] = 10758,
3784  ["qint"] = 10764,
3785  ["iiiint"] = 10764,
3786  ["fpartint"] = 10765,
3787  ["cirfnint"] = 10768,
3788  ["awint"] = 10769,
3789  ["rppolint"] = 10770,
3790  ["scpolint"] = 10771,
3791  ["npolint"] = 10772,
3792  ["pointint"] = 10773,
3793  ["quatint"] = 10774,
3794  ["intlarhk"] = 10775,
3795  ["pluscir"] = 10786,
3796  ["plusacir"] = 10787,
3797  ["simplus"] = 10788,
3798  ["plusdu"] = 10789,
3799  ["plussim"] = 10790,
3800  ["plustwo"] = 10791,
3801  ["mcomma"] = 10793,
3802  ["minusdu"] = 10794,
3803  ["loplus"] = 10797,
```

```
3804    ["roplus"] = 10798,
3805    ["Cross"] = 10799,
3806    ["timesd"] = 10800,
3807    ["timesbar"] = 10801,
3808    ["smashp"] = 10803,
3809    ["lotimes"] = 10804,
3810    ["rotimes"] = 10805,
3811    ["otimesas"] = 10806,
3812    ["Otimes"] = 10807,
3813    ["odiv"] = 10808,
3814    ["triplus"] = 10809,
3815    ["triminus"] = 10810,
3816    ["tritime"] = 10811,
3817    ["iprod"] = 10812,
3818    ["intprod"] = 10812,
3819    ["amalg"] = 10815,
3820    ["capdot"] = 10816,
3821    ["ncup"] = 10818,
3822    ["ncap"] = 10819,
3823    ["capand"] = 10820,
3824    ["cupor"] = 10821,
3825    ["cupcap"] = 10822,
3826    ["capcup"] = 10823,
3827    ["cupbrcap"] = 10824,
3828    ["capbrcup"] = 10825,
3829    ["cupcup"] = 10826,
3830    ["capcap"] = 10827,
3831    ["ccups"] = 10828,
3832    ["ccaps"] = 10829,
3833    ["ccupssm"] = 10832,
3834    ["And"] = 10835,
3835    ["Or"] = 10836,
3836    ["andand"] = 10837,
3837    ["oror"] = 10838,
3838    ["orslope"] = 10839,
3839    ["andslope"] = 10840,
3840    ["andv"] = 10842,
3841    ["orv"] = 10843,
3842    ["andd"] = 10844,
3843    ["ord"] = 10845,
3844    ["wedbar"] = 10847,
3845    ["sdote"] = 10854,
3846    ["simdot"] = 10858,
3847    ["congdot"] = 10861,
3848    ["easter"] = 10862,
3849    ["apacir"] = 10863,
3850    ["apE"] = 10864,
```

```
3851  ["eplus"] = 10865,
3852  ["pluse"] = 10866,
3853  ["Esim"] = 10867,
3854  ["Colone"] = 10868,
3855  ["Equal"] = 10869,
3856  ["eDDot"] = 10871,
3857  ["ddotseq"] = 10871,
3858  ["equivDD"] = 10872,
3859  ["ltcir"] = 10873,
3860  ["gtcir"] = 10874,
3861  ["ltquest"] = 10875,
3862  ["gtquest"] = 10876,
3863  ["les"] = 10877,
3864  ["LessSlantEqual"] = 10877,
3865  ["leqslant"] = 10877,
3866  ["ges"] = 10878,
3867  ["GreaterSlantEqual"] = 10878,
3868  ["geqslant"] = 10878,
3869  ["lesdot"] = 10879,
3870  ["gesdot"] = 10880,
3871  ["lesdoto"] = 10881,
3872  ["gesdoto"] = 10882,
3873  ["lesdotor"] = 10883,
3874  ["gesdotol"] = 10884,
3875  ["lap"] = 10885,
3876  ["lessapprox"] = 10885,
3877  ["gap"] = 10886,
3878  ["gtrapprox"] = 10886,
3879  ["lne"] = 10887,
3880  ["lneq"] = 10887,
3881  ["gne"] = 10888,
3882  ["gneq"] = 10888,
3883  ["lnap"] = 10889,
3884  ["lnapprox"] = 10889,
3885  ["gnap"] = 10890,
3886  ["gnapprox"] = 10890,
3887  ["lEg"] = 10891,
3888  ["lesseqqgtr"] = 10891,
3889  ["gEl"] = 10892,
3890  ["gtreqqless"] = 10892,
3891  ["lsime"] = 10893,
3892  ["gsime"] = 10894,
3893  ["lsimg"] = 10895,
3894  ["gsiml"] = 10896,
3895  ["lgE"] = 10897,
3896  ["glE"] = 10898,
3897  ["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,
["midcir"] = 10992,
["topcir"] = 10993,
["nhpar"] = 10994,
["parsim"] = 10995,
```

```
["parsl"] = 11005,
["fflig"] = 64256,
["filig"] = 64257,
["fllig"] = 64258,
["ffilig"] = 64259,
["ffllig"] = 64260,
["Ascr"] = 119964,
["Cscr"] = 119966,
["Dscr"] = 119967,
["Gscr"] = 119970,
["Jscr"] = 119973,
["Kscr"] = 119974,
["Nscr"] = 119977,
["Oscr"] = 119978,
["Pscr"] = 119979,
["Qscr"] = 119980,
["Sscr"] = 119982,
["Tscr"] = 119983,
["Uscr"] = 119984,
["Vscr"] = 119985,
["Wscr"] = 119986,
["Xscr"] = 119987,
["Yscr"] = 119988,
["Zscr"] = 119989,
["ascr"] = 119990,
["bscr"] = 119991,
["cscr"] = 119992,
["dscr"] = 119993,
["fscr"] = 119995,
["hscr"] = 119997,
["iscr"] = 119998,
["jscr"] = 119999,
["kscr"] = 120000,
["lscr"] = 120001,
["mscr"] = 120002,
["nscr"] = 120003,
["pscr"] = 120005,
["qscr"] = 120006,
["rscr"] = 120007,
["sscr"] = 120008,
["tscr"] = 120009,
["uscr"] = 120010,
["vscr"] = 120011,
["wscr"] = 120012,
["xscr"] = 120013,
["yscr"] = 120014,
["zscr"] = 120015,
```

```
4039  ["Afr"] = 120068,
4040  ["Bfr"] = 120069,
4041  ["Dfr"] = 120071,
4042  ["Efr"] = 120072,
4043  ["Ffr"] = 120073,
4044  ["Gfr"] = 120074,
4045  ["Jfr"] = 120077,
4046  ["Kfr"] = 120078,
4047  ["Lfr"] = 120079,
4048  ["Mfr"] = 120080,
4049  ["Nfr"] = 120081,
4050  ["Ofr"] = 120082,
4051  ["Pfr"] = 120083,
4052  ["Qfr"] = 120084,
4053  ["Sfr"] = 120086,
4054  ["Tfr"] = 120087,
4055  ["Ufr"] = 120088,
4056  ["Vfr"] = 120089,
4057  ["Wfr"] = 120090,
4058  ["Xfr"] = 120091,
4059  ["Yfr"] = 120092,
4060  ["afr"] = 120094,
4061  ["bfr"] = 120095,
4062  ["cfr"] = 120096,
4063  ["dfr"] = 120097,
4064  ["efr"] = 120098,
4065  ["ffr"] = 120099,
4066  ["gfr"] = 120100,
4067  ["hfr"] = 120101,
4068  ["ifr"] = 120102,
4069  ["jfr"] = 120103,
4070  ["kfr"] = 120104,
4071  ["lfr"] = 120105,
4072  ["mfr"] = 120106,
4073  ["nfr"] = 120107,
4074  ["ofr"] = 120108,
4075  ["pfr"] = 120109,
4076  ["qfr"] = 120110,
4077  ["rfr"] = 120111,
4078  ["sfr"] = 120112,
4079  ["tfr"] = 120113,
4080  ["ufr"] = 120114,
4081  ["vfr"] = 120115,
4082  ["wfr"] = 120116,
4083  ["xfr"] = 120117,
4084  ["yfr"] = 120118,
4085  ["zfr"] = 120119,
```

```
  ["Aopf"] = 120120,
  ["Bopf"] = 120121,
  ["Dopf"] = 120123,
  ["Eopf"] = 120124,
  ["Fopf"] = 120125,
  ["Gopf"] = 120126,
  ["Iopf"] = 120128,
  ["Jopf"] = 120129,
  ["Kopf"] = 120130,
  ["Lopf"] = 120131,
  ["Mopf"] = 120132,
  ["Oopf"] = 120134,
  ["Sopf"] = 120138,
  ["Topf"] = 120139,
  ["Uopf"] = 120140,
  ["Vopf"] = 120141,
  ["Wopf"] = 120142,
  ["Xopf"] = 120143,
  ["Yopf"] = 120144,
  ["aopf"] = 120146,
  ["bopf"] = 120147,
  ["copf"] = 120148,
  ["dopf"] = 120149,
  ["eopf"] = 120150,
  ["fopf"] = 120151,
  ["gopf"] = 120152,
  ["hopf"] = 120153,
  ["iopf"] = 120154,
  ["jopf"] = 120155,
  ["kopf"] = 120156,
  ["lopf"] = 120157,
  ["mopf"] = 120158,
  ["nopf"] = 120159,
  ["oopf"] = 120160,
  ["popf"] = 120161,
  ["qopf"] = 120162,
  ["ropf"] = 120163,
  ["sopf"] = 120164,
  ["topf"] = 120165,
  ["uopf"] = 120166,
  ["vopf"] = 120167,
  ["wopf"] = 120168,
  ["xopf"] = 120169,
  ["yopf"] = 120170,
  ["zopf"] = 120171,
}
```

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

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

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

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

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

```
function entities.char_entity(s)
  local n = character_entities[s]
  if n == nil then
    return "&" .. s .. ";"
  end
  return unicode.utf8.char(n)
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.

```
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 = {}
```

Make `options.cacheDir` available as `writer->cacheDir`, so that it is accessible from extensions.

```
4148  self.cacheDir = options.cacheDir
```

Make `options.hybrid` available as `writer->hybrid`, so that it is accessible from extensions.

```
4149  self.hybrid = options.hybrid
```

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

```
4150  local slice_specifiers = {}
4151  for specifier in options.slice:gmatch("[^%s]+") do
4152    table.insert(slice_specifiers, specifier)
4153  end
4154
4155  if #slice_specifiers == 2 then
4156    self.slice_begin, self.slice_end = table.unpack(slice_specifiers)
4157    local slice_begin_type = self.slice_begin:sub(1, 1)
4158    if slice_begin_type ~= "^" and slice_begin_type ~= "$" then
4159      self.slice_begin = "^" .. self.slice_begin
4160    end
4161    local slice_end_type = self.slice_end:sub(1, 1)
4162    if slice_end_type ~= "^" and slice_end_type ~= "$" then
4163      self.slice_end = "$" .. self.slice_end
4164    end
4165  elseif #slice_specifiers == 1 then
4166    self.slice_begin = "^" .. slice_specifiers[1]
4167    self.slice_end = "$" .. slice_specifiers[1]
4168  end
4169
4170  if self.slice_begin == "^" and self.slice_end ~= "^" then
4171    self.is_writing = true
4172  else
4173    self.is_writing = false
4174  end
```

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

```
4175  self.suffix = ".tex"
```

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

```
4176  self.space = " "
```

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

```
4177  self.nbsp = "\\markdownRendererNbsp{}"
```

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

```
4178  function self.plain(s)
4179    return s
```

```
4180    end
```

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

```
4181    function self.paragraph(s)
4182      if not self.is_writing then return "" end
4183      return s
4184    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.

```
4185    function self.pack(name)
4186      return [[\input ]] .. name .. [[\relax]]
4187    end
```

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

```
4188    function self.interblocksep()
4189      if not self.is_writing then return "" end
4190      return "\\markdownRendererInterblockSeparator\n{}"
4191    end
```

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

```
4192    self.linebreak = "\\markdownRendererLineBreak\n{}"
```

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

```
4193    self.ellipsis = "\\markdownRendererEllipsis{}"
```

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

```
4194    function self.hrule()
4195      if not self.is_writing then return "" end
4196      return "\\markdownRendererHorizontalRule{}"
4197    end
```

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

```
4198    self.escaped_uri_chars = {
4199      ["{"] = "\\markdownRendererLeftBrace{}",
4200      ["}"] = "\\markdownRendererRightBrace{}",
4201      ["%"] = "\\markdownRendererPercentSign{}",
4202      ["\\"] = "\\markdownRendererBackslash{}",
4203    }
4204    self.escaped_minimal_strings = {
4205      ["^^"] = "\\markdownRendererCircumflex\\markdownRendererCircumflex ",
4206      ["☒"] = "\\markdownRendererTickedBox{}",
4207      ["⊡"] = "\\markdownRendererHalfTickedBox{}",
4208      ["▫"] = "\\markdownRendererUntickedBox{}",
4209    }
```

Define a table `writer->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.

```
self.escaped_chars = {
   ["{"] = "\\markdownRendererLeftBrace{}",
   ["}"] = "\\markdownRendererRightBrace{}",
   ["%"] = "\\markdownRendererPercentSign{}",
   ["\\"] = "\\markdownRendererBackslash{}",
   ["#"] = "\\markdownRendererHash{}",
   ["$"] = "\\markdownRendererDollarSign{}",
   ["&"] = "\\markdownRendererAmpersand{}",
   ["_"] = "\\markdownRendererUnderscore{}",
   ["^"] = "\\markdownRendererCircumflex{}",
   ["~"] = "\\markdownRendererTilde{}",
   ["|"] = "\\markdownRendererPipe{}",
 }
```

Use the `writer->escaped_chars`, `writer->escaped_uri_chars`, and `writer->escaped_minima` tables to create the `writer->escape`, `writer->escape_uri`, and `writer->escape_minimal` escaper functions.

```
self.escape = util.escaper(self.escaped_chars, self.escaped_minimal_strings)
self.escape_uri = util.escaper(self.escaped_uri_chars, self.escaped_minimal_strings)
self.escape_minimal = util.escaper({}, self.escaped_minimal_strings)
```

Define `writer->string` as a function that will transform an input plain text span `s` 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 `writer->escape_minimal`. Otherwise, use the `writer->escape`, and `writer->escape_uri` functions.

```
if options.hybrid then
  self.string = self.escape_minimal
  self.uri = self.escape_minimal
else
  self.string = self.escape
  self.uri = self.escape_uri
end
```

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

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

```
function self.link(lab,src,tit)
```

```
4237    return {"\\markdownRendererLink{",lab,"}",
4238                            "{",self.escape(src),"}",
4239                            "{",self.uri(src),"}",
4240                            "{",self.string(tit or ""),"}"}
4241  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.

```
4242  function self.image(lab,src,tit)
4243    return {"\\markdownRendererImage{",lab,"}",
4244                             "{",self.string(src),"}",
4245                             "{",self.uri(src),"}",
4246                             "{",self.string(tit or ""),"}"}
4247  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.

```
4248  local function ulitem(s)
4249    return {"\\markdownRendererUlItem ",s,
4250            "\\markdownRendererUlItemEnd "}
4251  end
4252
4253  function self.bulletlist(items,tight)
4254    if not self.is_writing then return "" end
4255    local buffer = {}
4256    for _,item in ipairs(items) do
4257      buffer[#buffer + 1] = ulitem(item)
4258    end
4259    local contents = util.intersperse(buffer,"\n")
4260    if tight and options.tightLists then
4261      return {"\\markdownRendererUlBeginTight\n",contents,
4262        "\n\\markdownRendererUlEndTight "}
4263    else
4264      return {"\\markdownRendererUlBegin\n",contents,
4265        "\n\\markdownRendererUlEnd "}
4266    end
4267  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.

```
4268  local function olitem(s,num)
4269    if num ~= nil then
4270      return {"\\markdownRendererOlItemWithNumber{",num,"}",s,
```

```
4271                "\\markdownRendererOlItemEnd "}
4272     else
4273       return {"\\markdownRendererOlItem ",s,
4274               "\\markdownRendererOlItemEnd "}
4275     end
4276   end
4277
4278   function self.orderedlist(items,tight,startnum)
4279     if not self.is_writing then return "" end
4280     local buffer = {}
4281     local num = startnum
4282     for _,item in ipairs(items) do
4283       buffer[#buffer + 1] = olitem(item,num)
4284       if num ~= nil then
4285         num = num + 1
4286       end
4287     end
4288     local contents = util.intersperse(buffer,"\n")
4289     if tight and options.tightLists then
4290       return {"\\markdownRendererOlBeginTight\n",contents,
4291               "\n\\markdownRendererOlEndTight "}
4292     else
4293       return {"\\markdownRendererOlBegin\n",contents,
4294               "\n\\markdownRendererOlEnd "}
4295     end
4296   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.

```
4297   function self.inline_html_comment(contents)
4298     return {"\\markdownRendererInlineHtmlComment{",contents,"}"}
4299   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.

```
4300   function self.block_html_comment(contents)
4301     if not self.is_writing then return "" end
4302     return {"\\markdownRendererBlockHtmlCommentBegin\n",contents,
4303             "\n\\markdownRendererBlockHtmlCommentEnd "}
4304   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.

```
4305   function self.inline_html_tag(contents)
4306     return {"\\markdownRendererInlineHtmlTag{",self.string(contents),"}"}
```

```
4307    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.

```
4308    function self.block_html_element(s)
4309      if not self.is_writing then return "" end
4310      local name = util.cache(self.cacheDir, s, nil, nil, ".verbatim")
4311      return {"\\markdownRendererInputBlockHtmlElement{",name,"}"}
4312    end
```

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

```
4313    function self.emphasis(s)
4314      return {"\\markdownRendererEmphasis{",s,"}"}
4315    end
```

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

```
4316    function self.tickbox(f)
4317      if f == 1.0 then
4318        return "☒ "
4319      elseif f == 0.0 then
4320        return "☐ "
4321      else
4322        return "☒ "
4323      end
4324    end
```

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

```
4325    function self.strong(s)
4326      return {"\\markdownRendererStrongEmphasis{",s,"}"}
4327    end
```

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

```
4328    function self.blockquote(s)
4329      if #util.rope_to_string(s) == 0 then return "" end
4330      return {"\\markdownRendererBlockQuoteBegin\n",s,
4331        "\n\\markdownRendererBlockQuoteEnd "}
4332    end
```

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

```
4333    function self.verbatim(s)
4334      if not self.is_writing then return "" end
4335      s = string.gsub(s, '[\r\n%s]*$', '')
4336      local name = util.cache(self.cacheDir, s, nil, nil, ".verbatim")
```

```
4337    return {"\\markdownRendererInputVerbatim{",name,"}"}
4338  end
```

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

```
4339  function self.document(d)
4340    local active_attributes = self.active_attributes
4341    local buf = {"\\markdownRendererDocumentBegin\n", d}
4342
4343    -- pop attributes for sections that have ended
4344    if options.headerAttributes and self.is_writing then
4345      while #active_attributes > 0 do
4346        local attributes = active_attributes[#active_attributes]
4347        if #attributes > 0 then
4348          table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
4349        end
4350        table.remove(active_attributes, #active_attributes)
4351      end
4352    end
4353
4354    table.insert(buf, "\\markdownRendererDocumentEnd")
4355
4356    return buf
4357  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.

```
4358  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.

```
4359  function self.heading(s, level, attributes)
4360    attributes = attributes or {}
4361    for i = 1, #attributes do
4362      attributes[attributes[i]] = true
4363    end
4364
4365    local active_attributes = self.active_attributes
4366    local slice_begin_type = self.slice_begin:sub(1, 1)
4367    local slice_begin_identifier = self.slice_begin:sub(2) or ""
4368    local slice_end_type = self.slice_end:sub(1, 1)
4369    local slice_end_identifier = self.slice_end:sub(2) or ""
4370
4371    local buf = {}
4372
4373    -- push empty attributes for implied sections
4374    while #active_attributes < level-1 do
4375      table.insert(active_attributes, {})
```

```
4376    end
4377
4378    -- pop attributes for sections that have ended
4379    while #active_attributes >= level do
4380      local active_identifiers = active_attributes[#active_attributes]
4381      -- tear down all active attributes at slice end
4382      if active_identifiers["#" .. slice_end_identifier] ~= nil
4383          and slice_end_type == "$" then
4384        for header_level = #active_attributes, 1, -1 do
4385          if options.headerAttributes and #active_attributes[header_level] > 0 then
4386            table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
4387          end
4388        end
4389        self.is_writing = false
4390      end
4391      table.remove(active_attributes, #active_attributes)
4392      if self.is_writing and options.headerAttributes and #active_identifiers > 0 then
4393        table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
4394      end
4395      -- apply all active attributes at slice beginning
4396      if active_identifiers["#" .. slice_begin_identifier] ~= nil
4397          and slice_begin_type == "$" then
4398        for header_level = 1, #active_attributes do
4399          if options.headerAttributes and #active_attributes[header_level] > 0 then
4400            table.insert(buf, "\\markdownRendererHeaderAttributeContextBegin")
4401          end
4402        end
4403        self.is_writing = true
4404      end
4405    end
4406
4407    -- tear down all active attributes at slice end
4408    if attributes["#" .. slice_end_identifier] ~= nil
4409        and slice_end_type == "^" then
4410      for header_level = #active_attributes, 1, -1 do
4411        if options.headerAttributes and #active_attributes[header_level] > 0 then
4412          table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
4413        end
4414      end
4415      self.is_writing = false
4416    end
4417
4418    -- push attributes for the new section
4419    table.insert(active_attributes, attributes)
4420    if self.is_writing and options.headerAttributes and #attributes > 0 then
4421      table.insert(buf, "\\markdownRendererHeaderAttributeContextBegin")
4422    end
```

```
4423
4424      -- apply all active attributes at slice beginning
4425      if attributes["#" .. slice_begin_identifier] ~= nil
4426          and slice_begin_type == "^" then
4427        for header_level = 1, #active_attributes do
4428          if options.headerAttributes and #active_attributes[header_level] > 0 then
4429            table.insert(buf, "\\markdownRendererHeaderAttributeContextBegin")
4430          end
4431        end
4432        self.is_writing = true
4433      end
4434
4435      if self.is_writing then
4436        table.sort(attributes)
4437        local key, value
4438        for i = 1, #attributes do
4439          if attributes[i]:sub(1, 1) == "#" then
4440            table.insert(buf, {"\\markdownRendererAttributeIdentifier{",
4441                               attributes[i]:sub(2), "}"})
4442          elseif attributes[i]:sub(1, 1) == "." then
4443            table.insert(buf, {"\\markdownRendererAttributeClassName{",
4444                               attributes[i]:sub(2), "}"})
4445          else
4446            key, value = attributes[i]:match("(%w+)=(%w+)")
4447            table.insert(buf, {"\\markdownRendererAttributeKeyValue{",
4448                               key, "}{", value, "}"})
4449          end
4450        end
4451      end
4452
4453      local cmd
4454      level = level + options.shiftHeadings
4455      if level <= 1 then
4456        cmd = "\\markdownRendererHeadingOne"
4457      elseif level == 2 then
4458        cmd = "\\markdownRendererHeadingTwo"
4459      elseif level == 3 then
4460        cmd = "\\markdownRendererHeadingThree"
4461      elseif level == 4 then
4462        cmd = "\\markdownRendererHeadingFour"
4463      elseif level == 5 then
4464        cmd = "\\markdownRendererHeadingFive"
4465      elseif level >= 6 then
4466        cmd = "\\markdownRendererHeadingSix"
4467      else
4468        cmd = ""
4469      end
```

```
    if self.is_writing then
      table.insert(buf, {cmd, "{", s, "}"})
    end

    return buf
  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.

```
  function self.get_state()
    return {
      is_writing=self.is_writing,
      active_attributes={table.unpack(self.active_attributes)},
    }
  end
```

Define `writer->set_state` as a function that restores the input state `s` and returns the previous state of the writer.

```
  function self.set_state(s)
    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.internal_punctuation = S(":;,.?")

parsers.doubleasterisks     = P("**")
parsers.doubleunderscores   = P("__")
parsers.fourspaces          = P("    ")
```

```
4548
4549 parsers.any                     = P(1)
4550 parsers.fail                    = parsers.any - 1
4551
4552 parsers.escapable               = S("\\`*_{}[]()+_.!<>#-~:^@;")
4553 parsers.anyescaped              = parsers.backslash / "" * parsers.escapable
4554                                 + parsers.any
4555
4556 parsers.spacechar               = S("\t ")
4557 parsers.spacing                 = S(" \n\r\t")
4558 parsers.nonspacechar            = parsers.any - parsers.spacing
4559 parsers.optionalspace           = parsers.spacechar^0
4560
4561 parsers.normalchar              = parsers.any - (V("SpecialChar")
4562                                                  + parsers.spacing
4563                                                  + parsers.tightblocksep)
4564 parsers.eof                     = -parsers.any
4565 parsers.nonindentspace          = parsers.space^-3 * - parsers.spacechar
4566 parsers.indent                  = parsers.space^-3 * parsers.tab
4567                                 + parsers.fourspaces / ""
4568 parsers.linechar                = P(1 - parsers.newline)
4569
4570 parsers.blankline               = parsers.optionalspace
4571                                 * parsers.newline / "\n"
4572 parsers.blanklines              = parsers.blankline^0
4573 parsers.skipblanklines          = (parsers.optionalspace * parsers.newline)^0
4574 parsers.indentedline            = parsers.indent    /""
4575                                 * C(parsers.linechar^1 * parsers.newline^-
    1)
4576 parsers.optionallyindentedline = parsers.indent^-1 /""
4577                                 * C(parsers.linechar^1 * parsers.newline^-
    1)
4578 parsers.sp                      = parsers.spacing^0
4579 parsers.spnl                    = parsers.optionalspace
4580                                 * (parsers.newline * parsers.optionalspace)^-
    1
4581 parsers.line                    = parsers.linechar^0 * parsers.newline
4582 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.

```
4583 parsers.commented_line_letter  = parsers.linechar
4584                                 + parsers.newline
4585                                 - parsers.backslash
4586                                 - parsers.percent
4587 parsers.commented_line          = Cg(Cc(""), "backslashes")
4588                                 * ((#(parsers.commented_line_letter
```

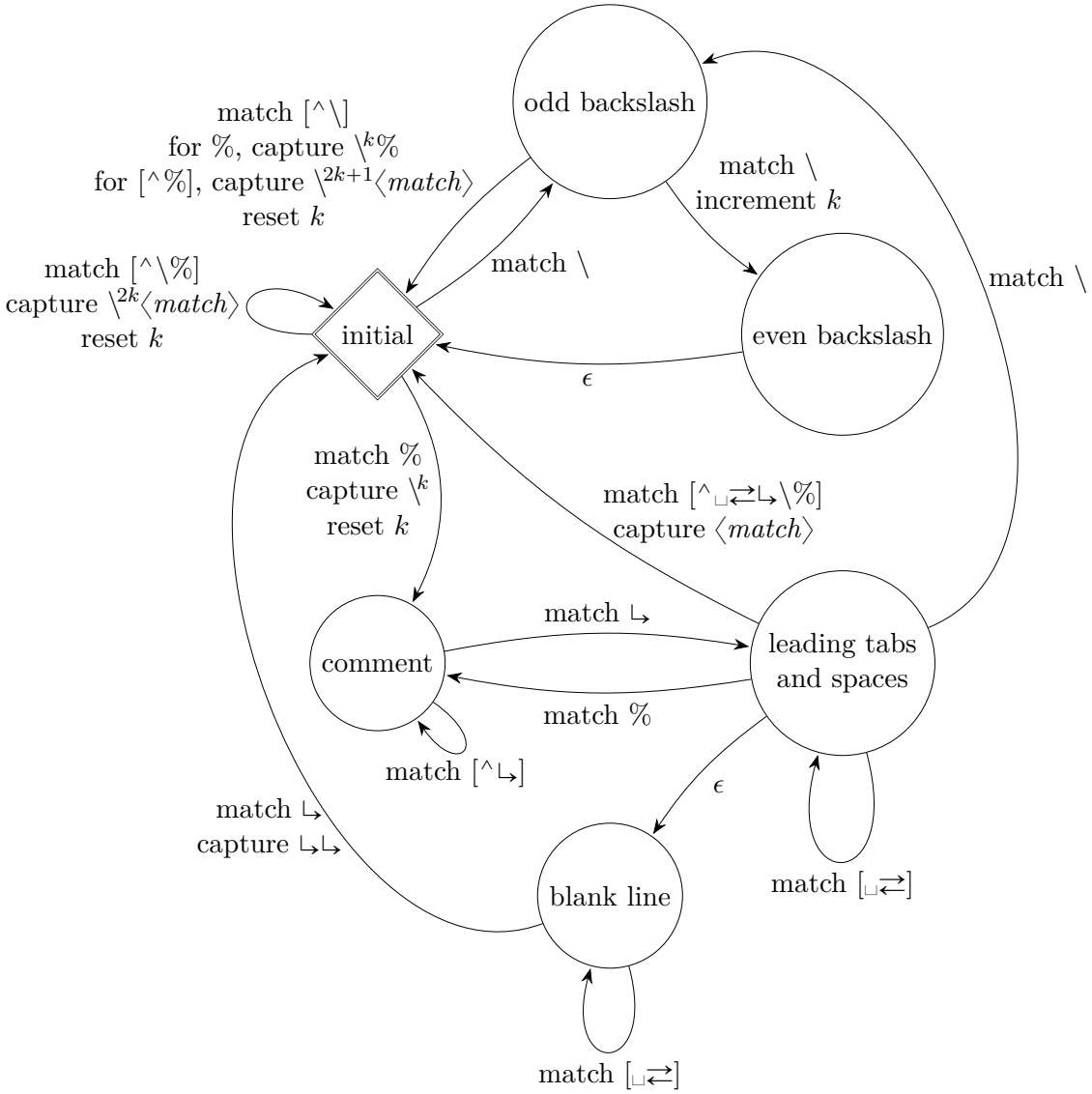

Figure 6: A pushdown automaton that recognizes TEX comments

```
                                  - 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
                                   * 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) )
end
```

3.1.4.2 Parsers Used for Markdown Lists

```
parsers.bulletchar = C(parsers.plus + parsers.asterisk + parsers.dash)

parsers.bullet = ( parsers.bulletchar * #parsers.spacing
                                      * (parsers.tab + parsers.space^-3)
                 + parsers.space * parsers.bulletchar * #parsers.spacing
                                 * (parsers.tab + parsers.space^-2)
                 + parsers.space * parsers.space * parsers.bulletchar
                                 * #parsers.spacing
                                 * (parsers.tab + parsers.space^-1)
                 + parsers.space * parsers.space * parsers.space
                                 * parsers.bulletchar * #parsers.spacing
                 )

local function tickbox(interior)
  return parsers.optionalspace * parsers.lbracket
       * interior * parsers.rbracket * parsers.spacechar^1
end

parsers.ticked_box = tickbox(S("xX")) * Cc(1.0)
parsers.halfticked_box = tickbox(S("./")) * Cc(0.5)
parsers.unticked_box = tickbox(parsers.spacechar^1) * Cc(0.0)
```

3.1.4.3 Parsers Used for Markdown Code Spans

```
parsers.openticks   = Cg(parsers.backtick^1, "ticks")

local function captures_equal_length(_,i,a,b)
  return #a == #b and i
end
```

```
4678
4679 parsers.closeticks  = parsers.space^-1
4680                   * Cmt(C(parsers.backtick^1)
4681                       * Cb("ticks"), captures_equal_length)
4682
4683 parsers.intickschar = (parsers.any - S(" \n\r`"))
4684                   + (parsers.newline * -parsers.blankline)
4685                   + (parsers.space - parsers.closeticks)
4686                   + (parsers.backtick^1 - parsers.closeticks)
4687
4688 parsers.inticks     = parsers.openticks * parsers.space^-1
4689                   * C(parsers.intickschar^0) * parsers.closeticks
```

3.1.4.4 Parsers Used for Fenced Code Blocks

```
4690 local function captures_geq_length(_,i,a,b)
4691   return #a >= #b and i
4692 end
4693
4694 parsers.infostring     = (parsers.linechar - (parsers.backtick
4695                        + parsers.space^1 * (parsers.newline + parsers.eof)))^0
4696
4697 local fenceindent
4698 parsers.fencehead    = function(char)
4699   return               C(parsers.nonindentspace) / function(s) fenceindent = #s end
4700                      * Cg(char^3, "fencelength")
4701                      * parsers.optionalspace * C(parsers.infostring)
4702                      * parsers.optionalspace * (parsers.newline + parsers.eof)
4703 end
4704
4705 parsers.fencetail    = function(char)
4706   return               parsers.nonindentspace
4707                      * Cmt(C(char^3) * Cb("fencelength"), captures_geq_length)
4708                      * parsers.optionalspace * (parsers.newline + parsers.eof)
4709                      + parsers.eof
4710 end
4711
4712 parsers.fencedline   = function(char)
4713   return               C(parsers.line - parsers.fencetail(char))
4714                      / function(s)
4715                          local i = 1
4716                          local remaining = fenceindent
4717                          while true do
4718                            local c = s:sub(i, i)
4719                            if c == " " and remaining > 0 then
4720                              remaining = remaining - 1
4721                              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

-- content in balanced brackets, parentheses, or quotes:
parsers.bracketed   = P{ parsers.lbracket
                         * ((parsers.anyescaped - (parsers.lbracket
                                                 + parsers.rbracket
                                                 + parsers.blankline^2)
                            ) + V(1))^0
                         * parsers.rbracket }

parsers.inparens    = P{ parsers.lparent
                         * ((parsers.anyescaped - (parsers.lparent
                                                 + parsers.rparent
                                                 + parsers.blankline^2)
                            ) + V(1))^0
                         * parsers.rparent }

parsers.squoted     = P{ parsers.squote * parsers.alphanumeric
                         * ((parsers.anyescaped - (parsers.squote
                                                 + parsers.blankline^2)
                            ) + V(1))^0
                         * parsers.squote }

parsers.dquoted     = P{ parsers.dquote * parsers.alphanumeric
                         * ((parsers.anyescaped - (parsers.dquote
                                                 + parsers.blankline^2)
                            ) + V(1))^0
                         * parsers.dquote }

-- bracketed tag for markdown links, allowing nested brackets:
parsers.tag         = parsers.lbracket
                    * Cs((parsers.alphanumeric^1
                         + parsers.bracketed
                         + parsers.inticks
```

```
                        + (parsers.anyescaped
                          - (parsers.rbracket + parsers.blankline^2)))^0)
                    * parsers.rbracket

-- url for markdown links, allowing nested brackets:
parsers.url         = parsers.less * Cs((parsers.anyescaped
                                       - parsers.more)^0)
                                   * parsers.more
                    + Cs((parsers.inparens + (parsers.anyescaped
                                              - parsers.spacing
                                              - parsers.rparent))^1)

-- quoted text, possibly with nested quotes:
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 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("?>")

parsers.openelt_any = parsers.less * parsers.keyword * parsers.htmlattribute^0
                    * parsers.sp * parsers.more

parsers.openelt_exact = function(s)
  return parsers.less * parsers.sp * parsers.keyword_exact(s)
       * parsers.htmlattribute^0 * parsers.sp * parsers.more
end

```

```
parsers.openelt_block = parsers.sp * parsers.block_keyword
                      * parsers.htmlattribute^0 * parsers.sp * parsers.more

parsers.closeelt_any = parsers.less * parsers.sp * parsers.slash
                     * parsers.keyword * parsers.sp * parsers.more

parsers.closeelt_exact = function(s)
  return parsers.less * parsers.sp * parsers.slash * parsers.keyword_exact(s)
       * parsers.sp * parsers.more
end

parsers.emptyelt_any = parsers.less * parsers.sp * parsers.keyword
                     * parsers.htmlattribute^0 * parsers.sp * parsers.slash
                     * parsers.more

parsers.emptyelt_block = parsers.less * parsers.sp * parsers.block_keyword
                       * parsers.htmlattribute^0 * parsers.sp * parsers.slash
                       * parsers.more

parsers.displaytext = (parsers.any - parsers.less)^1

-- return content between two matched HTML tags
parsers.in_matched = function(s)
  return { parsers.openelt_exact(s)
         * (V(1) + parsers.displaytext
           + (parsers.less - parsers.closeelt_exact(s)))^0
         * parsers.closeelt_exact(s) }
end

local function parse_matched_tags(s,pos)
  local t = string.lower(lpeg.match(C(parsers.keyword),s,pos))
  return lpeg.match(parsers.in_matched(t),s,pos-1)
end

parsers.in_matched_block_tags = parsers.less
                              * Cmt(#parsers.openelt_block, parse_matched_tags)

```

3.1.4.7 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
```

3.1.4.8 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
```

3.1.4.9 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
```

3.1.4.10 Block Elements

```
parsers.lineof = function(c)
    return (parsers.leader * (P(c) * parsers.optionalspace)^3
            * (parsers.newline * parsers.blankline^1
              + parsers.newline^-1 * parsers.eof))
end
```

3.1.4.11 Lists

3.1.4.12 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, extensions)
  local self = {}
```

Make the `writer` parameter available as `reader->writer`, so that it is accessible from extensions.

```
  self.writer = writer
```

Create a `reader->parsers` hash table that stores PEG patterns that depend on the received `options`. Make `reader->parsers` inherit from the global `parsers` table.

```
  self.parsers = {}
  (function(parsers)
    setmetatable(self.parsers, {
      __index = function (_, key)
        return parsers[key]
      end
    })
  end)(parsers)
```

Make `reader->parsers` available as a local `parsers` variable that will shadow the global `parsers` table and will make `reader->parsers` easier to type in the rest of the reader code.

```
  local parsers = self.parsers
```

3.1.5.1 Top-Level Helper Functions

Define `reader->normalize_tag` as a function that normalizes a markdown reference tag by lowercasing it, and by collapsing any adjacent whitespace characters.

```
4954  function self.normalize_tag(tag)
4955    return string.lower(
4956      gsub(util.rope_to_string(tag), "[ \n\r\t]+", " "))
4957  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.

```
4958  local function iterlines(s, f)
4959    local rope = lpeg.match(Ct((parsers.line / f)^1), s)
4960    return util.rope_to_string(rope)
4961  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.

```
4962  if options.preserveTabs then
4963    self.expandtabs = function(s) return s end
4964  else
4965    self.expandtabs = function(s)
4966                        if s:find("\t") then
4967                          return iterlines(s, util.expand_tabs_in_line)
4968                        else
4969                          return s
4970                        end
4971                      end
4972  end
```

3.1.5.2 High-Level Parser Functions

Create a `reader->parser_functions` hash table that stores high-level parser functions. Define `reader->create_parser` as a function that will create a high-level parser function `reader->parser_functions.name`, that matches input using grammar `grammar`. If `toplevel` is true, the input is expected to come straight from the user, not from a recursive call, and will be preprocessed.

```
4973  self.parser_functions = {}
4974  self.create_parser = function(name, grammar, toplevel)
4975    self.parser_functions[name] = function(str)
```

If the parser function 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.

```
4976      if toplevel and options.stripIndent then
4977          local min_prefix_length, min_prefix = nil, ''
```

```
4978            str = iterlines(str, function(line)
4979                if lpeg.match(parsers.nonemptyline, line) == nil then
4980                    return line
4981                end
4982                line = util.expand_tabs_in_line(line)
4983                local prefix = lpeg.match(C(parsers.optionalspace), line)
4984                local prefix_length = #prefix
4985                local is_shorter = min_prefix_length == nil
4986                is_shorter = is_shorter or prefix_length < min_prefix_length
4987                if is_shorter then
4988                    min_prefix_length, min_prefix = prefix_length, prefix
4989                end
4990                return line
4991            end)
4992            str = str:gsub('^' .. min_prefix, '')
4993        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.

```
4994        if toplevel and (options.texComments or options.hybrid) then
4995            str = lpeg.match(Ct(parsers.commented_line^1), str)
4996            str = util.rope_to_string(str)
4997        end
4998        local res = lpeg.match(grammar(), str)
4999        if res == nil then
5000          error(format("%s failed on:\n%s", name, str:sub(1,20)))
5001        else
5002          return res
5003        end
5004      end
5005    end
5006
5007    self.create_parser("parse_blocks",
5008                       function()
5009                         return parsers.blocks
5010                       end, true)
5011
5012    self.create_parser("parse_blocks_nested",
5013                       function()
5014                         return parsers.blocks_nested
5015                       end, false)
5016
5017    self.create_parser("parse_inlines",
5018                       function()
5019                         return parsers.inlines
5020                       end, false)
```

```
5021
5022  self.create_parser("parse_inlines_no_link",
5023                     function()
5024                       return parsers.inlines_no_link
5025                     end, false)
5026
5027  self.create_parser("parse_inlines_no_inline_note",
5028                     function()
5029                       return parsers.inlines_no_inline_note
5030                     end, false)
5031
5032  self.create_parser("parse_inlines_no_html",
5033                     function()
5034                       return parsers.inlines_no_html
5035                     end, false)
5036
5037  self.create_parser("parse_inlines_nbsp",
5038                     function()
5039                       return parsers.inlines_nbsp
5040                     end, false)
```

3.1.5.3 Parsers Used for Markdown Lists (local)

```
5041  if options.hashEnumerators then
5042    parsers.dig = parsers.digit + parsers.hash
5043  else
5044    parsers.dig = parsers.digit
5045  end
5046
5047  parsers.enumerator = C(parsers.dig^3 * parsers.period) * #parsers.spacing
5048                     + C(parsers.dig^2 * parsers.period) * #parsers.spacing
5049                                         * (parsers.tab + parsers.space^1)
5050                     + C(parsers.dig * parsers.period) * #parsers.spacing
5051                                        * (parsers.tab + parsers.space^-2)
5052                     + parsers.space * C(parsers.dig^2 * parsers.period)
5053                                        * #parsers.spacing
5054                     + parsers.space * C(parsers.dig * parsers.period)
5055                                        * #parsers.spacing
5056                                        * (parsers.tab + parsers.space^-1)
5057                     + parsers.space * parsers.space * C(parsers.dig^1
5058                                        * parsers.period) * #parsers.spacing
```

3.1.5.4 Parsers Used for Blockquotes (local)

```
5059  -- strip off leading > and indents, and run through blocks
5060  parsers.blockquote_body = ((parsers.leader * parsers.more * parsers.space^-
    1)/""
5061                             * parsers.linechar^0 * parsers.newline)^1
```

```
                              * (-(parsers.leader * parsers.more
                                  + parsers.blankline) * parsers.linechar^1
                                * parsers.newline)^0

if not options.breakableBlockquotes then
  parsers.blockquote_body = parsers.blockquote_body
                              * (parsers.blankline^0 / "")
end
```

3.1.5.5 Parsers Used for Footnotes (local)

3.1.5.6 Helpers for Links and References (local)

```
-- List of references defined in the document
local references

-- add a reference to the list
local function register_link(tag,url,title)
    references[self.normalize_tag(tag)] = { url = url, title = title }
    return ""
end

-- lookup link reference and return either
-- the link or nil and fallback text.
local function lookup_reference(label,sps,tag)
    local tagpart
    if not tag then
        tag = label
        tagpart = ""
    elseif tag == "" then
        tag = label
        tagpart = "[]"
    else
        tagpart = {"[",
          self.parser_functions.parse_inlines(tag),
          "]"}
    end
    if sps then
      tagpart = {sps, tagpart}
    end
    local r = references[self.normalize_tag(tag)]
    if r then
      return r
    else
      return nil, {"[",
        self.parser_functions.parse_inlines(label),
        "]", tagpart}
```

```
    end
end

-- lookup link reference and return a link, if the reference is found,
-- or a bracketed label otherwise.
local function indirect_link(label,sps,tag)
  return writer.defer_call(function()
    local r,fallback = lookup_reference(label,sps,tag)
    if r then
      return writer.link(
        self.parser_functions.parse_inlines_no_link(label),
        r.url, r.title)
    else
      return fallback
    end
  end)
end

-- lookup image reference and return an image, if the reference is found,
-- or a bracketed label otherwise.
local function indirect_image(label,sps,tag)
  return writer.defer_call(function()
    local r,fallback = lookup_reference(label,sps,tag)
    if r then
      return writer.image(writer.string(label), r.url, r.title)
    else
      return {"!", fallback}
    end
  end)
end
```

3.1.5.7 Inline Elements (local)

```
parsers.Str      = (parsers.normalchar * (parsers.normalchar + parsers.at)^0)
                 / writer.string

parsers.Symbol   = (V("SpecialChar") - parsers.tightblocksep)
                 / writer.string

parsers.Ellipsis = P("...") / writer.ellipsis

parsers.Smart    = parsers.Ellipsis

parsers.Code     = parsers.inticks / writer.code

if options.blankBeforeBlockquote then
  parsers.bqstart = parsers.fail
```

```
5148  else
5149    parsers.bqstart = parsers.more
5150  end
5151
5152  if options.blankBeforeHeading then
5153    parsers.headerstart = parsers.fail
5154  else
5155    parsers.headerstart = parsers.hash
5156                        + (parsers.line * (parsers.equal^1 + parsers.dash^1)
5157                        * parsers.optionalspace * parsers.newline)
5158  end
5159
5160  parsers.EndlineExceptions
5161                     = parsers.blankline -- paragraph break
5162                     + parsers.tightblocksep  -- nested list
5163                     + parsers.eof       -- end of document
5164                     + parsers.bqstart
5165                     + parsers.headerstart
5166
5167  parsers.Endline   = parsers.newline
5168                    * -V("EndlineExceptions")
5169                    * parsers.spacechar^0
5170                    / (options.hardLineBreaks and writer.linebreak
5171                                              or writer.space)
5172
5173  parsers.OptionalIndent
5174                     = parsers.spacechar^1 / writer.space
5175
5176  parsers.Space      = parsers.spacechar^2 * parsers.Endline / writer.linebreak
5177                     + parsers.spacechar^1 * parsers.Endline^-1 * parsers.eof / ""
5178                     + parsers.spacechar^1 * parsers.Endline
5179                                           * parsers.optionalspace
5180                                           / (options.hardLineBreaks
5181                                              and writer.linebreak
5182                                               or writer.space)
5183                     + parsers.spacechar^1 * parsers.optionalspace
5184                                           / writer.space
5185
5186  parsers.NonbreakingEndline
5187                    = parsers.newline
5188                    * -V("EndlineExceptions")
5189                    * parsers.spacechar^0
5190                    / (options.hardLineBreaks and writer.linebreak
5191                                              or writer.nbsp)
5192
5193  parsers.NonbreakingSpace
5194                  = parsers.spacechar^2 * parsers.Endline / writer.linebreak
```

```
                  + parsers.spacechar^1 * parsers.Endline^-1 * parsers.eof / ""
                  + parsers.spacechar^1 * parsers.Endline
                                        * parsers.optionalspace
                                        / (options.hardLineBreaks
                                           and writer.linebreak
                                            or writer.nbsp)
                  + parsers.spacechar^1 * parsers.optionalspace
                                        / writer.nbsp

if options.underscores then
  parsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
                                     parsers.doubleasterisks)
                   + parsers.between(parsers.Inline, parsers.doubleunderscores,
                                     parsers.doubleunderscores)
                   ) / writer.strong

  parsers.Emph   = ( parsers.between(parsers.Inline, parsers.asterisk,
                                     parsers.asterisk)
                   + parsers.between(parsers.Inline, parsers.underscore,
                                     parsers.underscore)
                   ) / writer.emphasis
else
  parsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
                                     parsers.doubleasterisks)
                   ) / writer.strong

  parsers.Emph   = ( parsers.between(parsers.Inline, parsers.asterisk,
                                     parsers.asterisk)
                   ) / writer.emphasis
end

parsers.AutoLinkUrl    = parsers.less
                         * C(parsers.alphanumeric^1 * P("://") * parsers.urlchar^1)
                         * parsers.more
                         / function(url)
                             return writer.link(writer.escape(url), url)
                           end

parsers.AutoLinkEmail = parsers.less
                        * C((parsers.alphanumeric + S("-._+"))^1
                        * P("@") * parsers.urlchar^1)
                        * parsers.more
                        / function(email)
                            return writer.link(writer.escape(email),
                                               "mailto:"..email)
                          end

```

```
  parsers.AutoLinkRelativeReference
                          = parsers.less
                          * C(parsers.urlchar^1)
                          * parsers.more
                          / function(url)
                              return writer.link(writer.escape(url), url)
                            end

  parsers.DirectLink    = (parsers.tag / self.parser_functions.parse_inlines_no_link)
                          * parsers.spnl
                          * parsers.lparent
                          * (parsers.url + Cc(""))  -- link can be empty [foo]()
                          * parsers.optionaltitle
                          * parsers.rparent
                          / writer.link

  parsers.IndirectLink  = parsers.tag * (C(parsers.spnl) * parsers.tag)^-
1
                          / indirect_link

  -- parse a link or image (direct or indirect)
  parsers.Link          = parsers.DirectLink + parsers.IndirectLink

  parsers.DirectImage   = parsers.exclamation
                          * (parsers.tag / self.parser_functions.parse_inlines)
                          * parsers.spnl
                          * parsers.lparent
                          * (parsers.url + Cc(""))  -- link can be empty [foo]()
                          * parsers.optionaltitle
                          * parsers.rparent
                          / writer.image

  parsers.IndirectImage = parsers.exclamation * parsers.tag
                          * (C(parsers.spnl) * parsers.tag)^-1 / indirect_image

  parsers.Image         = parsers.DirectImage + parsers.IndirectImage

  -- avoid parsing long strings of * or _ as emph/strong
  parsers.UlOrStarLine  = parsers.asterisk^4 + parsers.underscore^4
                          / writer.string

  parsers.EscapedChar   = parsers.backslash * C(parsers.escapable) / writer.string

  parsers.InlineHtml    = parsers.emptyelt_any / writer.inline_html_tag
                          + (parsers.htmlcomment / self.parser_functions.parse_inlines_no_
                          / writer.inline_html_comment
                          + parsers.htmlinstruction
```

```
                        + parsers.openelt_any / writer.inline_html_tag
                        + parsers.closeelt_any / writer.inline_html_tag

parsers.HtmlEntity    = parsers.hexentity / entities.hex_entity  / writer.string
                        + parsers.decentity / entities.dec_entity  / writer.string
                        + parsers.tagentity / entities.char_entity / writer.string
```

3.1.5.8 Block Elements (local)

```
parsers.DisplayHtml  = (parsers.htmlcomment / self.parser_functions.parse_blocks_neste
                       / 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

parsers.Verbatim     = Cs( (parsers.blanklines
                           * ((parsers.indentedline - parsers.blankline))^1)^1
                           ) / self.expandtabs / writer.verbatim

parsers.Blockquote   = Cs(parsers.blockquote_body^1)
                       / self.parser_functions.parse_blocks_nested
                       / writer.blockquote

parsers.HorizontalRule = ( parsers.lineof(parsers.asterisk)
                           + parsers.lineof(parsers.dash)
                           + parsers.lineof(parsers.underscore)
                           ) / writer.hrule

parsers.Reference    = parsers.define_reference_parser / register_link

parsers.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

parsers.Plain        = parsers.nonindentspace * Ct(parsers.Inline^1)
                       / writer.plain
```

3.1.5.9 Lists (local)

```
parsers.starter = parsers.bullet + parsers.enumerator
```

```
5329
5330  if options.taskLists then
5331    parsers.tickbox = ( parsers.ticked_box
5332                      + parsers.halfticked_box
5333                      + parsers.unticked_box
5334                      ) / writer.tickbox
5335  else
5336     parsers.tickbox = parsers.fail
5337  end
5338
5339  -- we use \001 as a separator between a tight list item and a
5340  -- nested list under it.
5341  parsers.NestedList            = Cs((parsers.optionallyindentedline
5342                                      - parsers.starter)^1)
5343                                / function(a) return "\001"..a end
5344
5345  parsers.ListBlockLine         = parsers.optionallyindentedline
5346                                - parsers.blankline - (parsers.indent^-1
5347                                                     * parsers.starter)
5348
5349  parsers.ListBlock             = parsers.line * parsers.ListBlockLine^0
5350
5351  parsers.ListContinuationBlock = parsers.blanklines * (parsers.indent / "")
5352                                * parsers.ListBlock
5353
5354  parsers.TightListItem = function(starter)
5355      return -parsers.HorizontalRule
5356             * (Cs(starter / "" * parsers.tickbox^-1 * parsers.ListBlock * parsers.Neste
    1)
5357               / self.parser_functions.parse_blocks_nested)
5358             * -(parsers.blanklines * parsers.indent)
5359  end
5360
5361  parsers.LooseListItem = function(starter)
5362      return -parsers.HorizontalRule
5363             * Cs( starter / "" * parsers.tickbox^-1 * parsers.ListBlock * Cc("\n")
5364               * (parsers.NestedList + parsers.ListContinuationBlock^0)
5365               * (parsers.blanklines / "\n\n")
5366               ) / self.parser_functions.parse_blocks_nested
5367  end
5368
5369  parsers.BulletList = ( Ct(parsers.TightListItem(parsers.bullet)^1) * Cc(true)
5370                       * parsers.skipblanklines * -parsers.bullet
5371                       + Ct(parsers.LooseListItem(parsers.bullet)^1) * Cc(false)
5372                       * parsers.skipblanklines )
5373                     / writer.bulletlist
5374
```

```
5375  local function ordered_list(items,tight,startNumber)
5376    if options.startNumber then
5377      startNumber = tonumber(startNumber) or 1  -- fallback for '#'
5378      if startNumber ~= nil then
5379        startNumber = math.floor(startNumber)
5380      end
5381    else
5382      startNumber = nil
5383    end
5384    return writer.orderedlist(items,tight,startNumber)
5385  end
5386
5387  parsers.OrderedList = Cg(parsers.enumerator, "listtype") *
5388                        ( Ct(parsers.TightListItem(Cb("listtype"))
5389                            * parsers.TightListItem(parsers.enumerator)^0)
5390                        * Cc(true) * parsers.skipblanklines * -parsers.enumerator
5391                        + Ct(parsers.LooseListItem(Cb("listtype"))
5392                            * parsers.LooseListItem(parsers.enumerator)^0)
5393                        * Cc(false) * parsers.skipblanklines
5394                        ) * Cb("listtype") / ordered_list
```

3.1.5.10 Blank (local)

```
5395  parsers.Blank        = parsers.blankline / ""
5396                       + parsers.Reference
5397                       + (parsers.tightblocksep / "\n")
```

3.1.5.11 Headings (local)

```
5398  -- parse atx header
5399  parsers.AtxHeading = Cg(parsers.HeadingStart,"level")
5400                     * parsers.optionalspace
5401                     * (C(parsers.line)
5402                       / strip_atx_end
5403                       / self.parser_functions.parse_inlines)
5404                     * Cb("level")
5405                     / writer.heading
5406
5407  parsers.SetextHeading = #(parsers.line * S("=-"))
5408                          * Ct(parsers.linechar^1
5409                              / self.parser_functions.parse_inlines)
5410                          * parsers.newline
5411                          * parsers.HeadingLevel
5412                          * parsers.optionalspace
5413                          * parsers.newline
5414                          / writer.heading
5415
5416  parsers.Heading = parsers.AtxHeading + parsers.SetextHeading
```

3.1.5.12 Syntax Specification

Create a `reader->syntax` hash table that stores the PEG grammar.

```
5417  self.syntax =
5418    { "Blocks",
5419
5420      Blocks                = ( V("ExpectedJekyllData")
5421                              * (V("Blank")^0 / writer.interblocksep)
5422                              )^-1
5423                            * V("Blank")^0
5424                            * V("Block")^-1
5425                            * (V("Blank")^0 / writer.interblocksep
5426                              * V("Block"))^0
5427                            * V("Blank")^0 * parsers.eof,
5428
5429      Blank                 = parsers.Blank,
5430
5431      UnexpectedJekyllData  = parsers.fail,
5432      ExpectedJekyllData    = parsers.fail,
5433
5434      Block                 = V("ContentBlock")
5435                            + V("UnexpectedJekyllData")
5436                            + V("Blockquote")
5437                            + V("PipeTable")
5438                            + V("Verbatim")
5439                            + V("FencedCode")
5440                            + V("HorizontalRule")
5441                            + V("BulletList")
5442                            + V("OrderedList")
5443                            + V("Heading")
5444                            + V("DefinitionList")
5445                            + V("DisplayHtml")
5446                            + V("Paragraph")
5447                            + V("Plain"),
5448
5449      ContentBlock          = parsers.fail,
5450      Blockquote            = parsers.Blockquote,
5451      Verbatim              = parsers.Verbatim,
5452      FencedCode            = parsers.fail,
5453      HorizontalRule        = parsers.HorizontalRule,
5454      BulletList            = parsers.BulletList,
5455      OrderedList           = parsers.OrderedList,
5456      Heading               = parsers.Heading,
5457      DefinitionList        = parsers.fail,
5458      DisplayHtml           = parsers.DisplayHtml,
5459      Paragraph             = parsers.Paragraph,
5460      PipeTable             = parsers.fail,
5461      Plain                 = parsers.Plain,
```

```
5462        EndlineExceptions     = parsers.EndlineExceptions,
5463
5464        Inline                = V("Str")
5465                              + V("Space")
5466                              + V("Endline")
5467                              + V("UlOrStarLine")
5468                              + V("Strong")
5469                              + V("Emph")
5470                              + V("InlineNote")
5471                              + V("NoteRef")
5472                              + V("Citations")
5473                              + V("Link")
5474                              + V("Image")
5475                              + V("Code")
5476                              + V("AutoLinkUrl")
5477                              + V("AutoLinkEmail")
5478                              + V("AutoLinkRelativeReference")
5479                              + V("InlineHtml")
5480                              + V("HtmlEntity")
5481                              + V("EscapedChar")
5482                              + V("Smart")
5483                              + V("Symbol"),
5484
5485        IndentedInline        = V("Str")
5486                              + V("OptionalIndent")
5487                              + V("Endline")
5488                              + V("UlOrStarLine")
5489                              + V("Strong")
5490                              + V("Emph")
5491                              + V("InlineNote")
5492                              + V("NoteRef")
5493                              + V("Citations")
5494                              + V("Link")
5495                              + V("Image")
5496                              + V("Code")
5497                              + V("AutoLinkUrl")
5498                              + V("AutoLinkEmail")
5499                              + V("AutoLinkRelativeReference")
5500                              + V("InlineHtml")
5501                              + V("HtmlEntity")
5502                              + V("EscapedChar")
5503                              + V("Smart")
5504                              + V("Symbol"),
5505
5506        Str                   = parsers.Str,
5507        Space                 = parsers.Space,
5508        OptionalIndent        = parsers.OptionalIndent,
```

```
      Endline               = parsers.Endline,
      UlOrStarLine          = parsers.UlOrStarLine,
      Strong                = parsers.Strong,
      Emph                  = parsers.Emph,
      InlineNote            = parsers.fail,
      NoteRef               = parsers.fail,
      Citations             = parsers.fail,
      Link                  = parsers.Link,
      Image                 = parsers.Image,
      Code                  = parsers.Code,
      AutoLinkUrl           = parsers.AutoLinkUrl,
      AutoLinkEmail         = parsers.AutoLinkEmail,
      AutoLinkRelativeReference
                            = parsers.AutoLinkRelativeReference,
      InlineHtml            = parsers.InlineHtml,
      HtmlEntity            = parsers.HtmlEntity,
      EscapedChar           = parsers.EscapedChar,
      Smart                 = parsers.Smart,
      Symbol                = parsers.Symbol,
      SpecialChar           = parsers.fail,
    }
```

Define a hash table of all characters with a special meaning that can be escaped and add method `reader->add_special_character` that extends the hash table and updates the PEG grammar.

```
  local special_characters = {"*", "`", "[", "]", "<", "!", "\\"}
  self.add_special_character = function(c)
    table.insert(special_characters, c)
    self.syntax.SpecialChar = S(table.concat(special_characters, ""))
  end
  self.syntax.SpecialChar = S(table.concat(special_characters, ""))
```

Apply syntax extensions.

```
  for _, extension in ipairs(extensions) do
    extension.extend_writer(writer)
    extension.extend_reader(self)
  end

  if options.underscores then
    self.add_special_character("_")
  end

  if not options.codeSpans then
    self.syntax.Code = parsers.fail
  end

  if not options.html then
    self.syntax.DisplayHtml = parsers.fail
```

```
    self.syntax.InlineHtml = parsers.fail
    self.syntax.HtmlEntity  = parsers.fail
  else
    self.add_special_character("&")
  end

  if options.preserveTabs then
    options.stripIndent = false
  end

  if not options.smartEllipses then
    self.syntax.Smart = parsers.fail
  else
    self.add_special_character(".")
  end

  if not options.relativeReferences then
    self.syntax.AutoLinkRelativeReference = parsers.fail
  end

  local blocks_nested_t = util.table_copy(self.syntax)
  blocks_nested_t.ExpectedJekyllData = parsers.fail
  parsers.blocks_nested = Ct(blocks_nested_t)

  parsers.blocks = Ct(self.syntax)

  local inlines_t = util.table_copy(self.syntax)
  inlines_t[1] = "Inlines"
  inlines_t.Inlines = parsers.Inline^0 * (parsers.spacing^0 * parsers.eof / "")
  parsers.inlines = Ct(inlines_t)

  local inlines_no_link_t = util.table_copy(inlines_t)
  inlines_no_link_t.Link = parsers.fail
  parsers.inlines_no_link = Ct(inlines_no_link_t)

  local inlines_no_inline_note_t = util.table_copy(inlines_t)
  inlines_no_inline_note_t.InlineNote = parsers.fail
  parsers.inlines_no_inline_note = Ct(inlines_no_inline_note_t)

  local inlines_no_html_t = util.table_copy(inlines_t)
  inlines_no_html_t.DisplayHtml = parsers.fail
  inlines_no_html_t.InlineHtml = parsers.fail
  inlines_no_html_t.HtmlEntity = parsers.fail
  parsers.inlines_no_html = Ct(inlines_no_html_t)

  local inlines_nbsp_t = util.table_copy(inlines_t)
  inlines_nbsp_t.Endline = parsers.NonbreakingEndline
```

```
  inlines_nbsp_t.Space = parsers.NonbreakingSpace
  parsers.inlines_nbsp = Ct(inlines_nbsp_t)
```

3.1.5.13 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.

```
  function self.convert(input)
    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.

```
    local opt_string = {}
    for k,_ in pairs(defaultOptions) do
      local v = options[k]
      if k ~= "cacheDir" then
        opt_string[#opt_string+1] = k .. "=" .. tostring(v)
      end
    end
    table.sort(opt_string)
    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 = self.parser_functions.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 Syntax Extensions for Markdown

Create `extensions` hash table that contains syntax extensions. Syntax extensions are functions that produce objects with two methods: `extend_writer` and `extend_reader`. The `extend_writer` object takes a `writer` object as the only parameter and mutates it. Similarly, `extend_reader` takes a `reader` object as the only parameter and mutates it.

```
M.extensions = {}
```

3.1.6.1 Citations

The `extensions.citations` function implements the Pandoc citation syntax extension. When the `citation_nbsps` parameter is `true`, the syntax extension will replace regular spaces with non-breaking spaces inside the prenotes and postnotes of citations.

```
M.extensions.citations = function(citation_nbsps)
```

Define table `escaped_citation_chars` containing the characters to escape in citations.

```
  local escaped_citation_chars = {
    ["{"] = "\\markdownRendererLeftBrace{}",
    ["}"] = "\\markdownRendererRightBrace{}",
    ["%"] = "\\markdownRendererPercentSign{}",
    ["\\"] = "\\markdownRendererBackslash{}",
    ["#"] = "\\markdownRendererHash{}",
  }
  return {
    extend_writer = function(self)
```

Use the `escaped_citation_chars` to create the `escape_citation` escaper functions.

```
      local escape_citation = util.escaper(
        escaped_citation_chars,
        self.escaped_minimal_strings)
```

Define `writer->citation` as a function that will transform an input citation name `c` to the output format. If `writer->hybrid` is `true`, use the `writer->escape_minimal` function. Otherwise, use the `escape_citation` function.

```
5655       if self.hybrid then
5656         self.citation = self.escape_minimal
5657       else
5658         self.citation = escape_citation
5659       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.

```
5660       function self.citations(text_cites, cites)
5661         local buffer = {"\\markdownRenderer", text_cites and "TextCite" or "Cite",
5662           "{", #cites, "}"}
5663         for _,cite in ipairs(cites) do
5664           buffer[#buffer+1] = {cite.suppress_author and "-" or "+", "{",
5665             cite.prenote or "", "}{", cite.postnote or "", "}{", cite.name, "}"}
5666         end
5667         return buffer
5668       end
5669     end, extend_reader = function(self)
5670       local parsers = self.parsers
5671       local syntax = self.syntax
5672       local writer = self.writer
5673
5674       local citation_chars
5675                     = parsers.alphanumeric
5676                     + S("#$%&-+<>~/_")
5677
5678       local citation_name
5679                     = Cs(parsers.dash^-1) * parsers.at
5680                     * Cs(citation_chars
5681                         * (((citation_chars + parsers.internal_punctuation
```

```
                            - parsers.comma - parsers.semicolon)
                           * -#((parsers.internal_punctuation - parsers.comma
                                - parsers.semicolon)^0
                              * -(citation_chars + parsers.internal_punctuation
                                 - parsers.comma - parsers.semicolon)))^0
                         * citation_chars)^-1)

      local 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)

      local 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)

      local citation_body_chunk
                    = citation_body_prenote
                    * parsers.spnl * citation_name
                    * (parsers.internal_punctuation - parsers.semicolon)^-
1
                    * parsers.spnl * citation_body_postnote

      local citation_body
                    = citation_body_chunk
                    * (parsers.semicolon * parsers.spnl
                      * citation_body_chunk)^0

      local 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)

      local citation_headless_body
                    = citation_headless_body_postnote
```

```
                * (parsers.sp * parsers.semicolon * parsers.spnl
                  * citation_body_chunk)^0

    local citations
                = function(text_cites, raw_cites)
        local function normalize(str)
            if str == "" then
                str = nil
            else
                str = (citation_nbsps and
                  self.parser_functions.parse_inlines_nbsp or
                  self.parser_functions.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

    local TextCitations
                = Ct((parsers.spnl
                * Cc("")
                * citation_name
                * ((parsers.spnl
                    * parsers.lbracket
                    * citation_headless_body
                    * parsers.rbracket) + Cc("")))^1)
                / function(raw_cites)
                    return citations(true, raw_cites)
                  end

    local ParenthesizedCitations
                = Ct((parsers.spnl
                * parsers.lbracket
                * citation_body
                * parsers.rbracket)^1)
                / function(raw_cites)
                    return citations(false, raw_cites)
```

```
                        end

      local Citations = TextCitations + ParenthesizedCitations

      syntax.Citations = Citations

      self.add_special_character("@")
      self.add_special_character("-")
    end
  }
end
```

3.1.6.2 Content Blocks

The `extensions.content_blocks` function implements the iA Writer content blocks syntax extension. The `language_map` parameter specifies the filename of the JSON file that maps filename extensions to programming language names.

```
M.extensions.content_blocks = function(language_map)
```

The `languages_json` table maps programming language filename extensions to fence infostrings. All `language_map` 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, _) return filename end}
    end
    local base, prev, curr
    for _, filename in ipairs{kpse.lookup(language_map, { all=true })} do
      local file = io.open(filename, "r")
      if not file then goto continue end
      local json = file:read("*all"):gsub('("[^\n]-"):','[%1]=')
      curr = (function()
        local _ENV={ json=json, load=load } -- luacheck: ignore _ENV
        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)()

  return {
    extend_writer = function(self)
```

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.

```
      function self.contentblock(src,suf,type,tit)
        if not self.is_writing then return "" end
        src = src.."."..suf
        suf = suf:lower()
        if type == "onlineimage" then
          return {"\\markdownRendererContentBlockOnlineImage{",suf,"}",
                                  "{",self.string(src),"}",
                                  "{",self.uri(src),"}",
                                  "{",self.string(tit or ""),"}"}
        elseif languages_json[suf] then
          return {"\\markdownRendererContentBlockCode{",suf,"}",
                                  "{",self.string(languages_json[suf]),"}",
                                  "{",self.string(src),"}",
                                  "{",self.uri(src),"}",
                                  "{",self.string(tit or ""),"}"}
        else
          return {"\\markdownRendererContentBlock{",suf,"}",
                                  "{",self.string(src),"}",
                                  "{",self.uri(src),"}",
                                  "{",self.string(tit or ""),"}"}
        end
      end
    end, extend_reader = function(self)
      local parsers = self.parsers
      local syntax = self.syntax
      local writer = self.writer

      local contentblock_tail
                    = parsers.optionaltitle
                    * (parsers.newline + parsers.eof)

```

```
5849        -- case insensitive online image suffix:
5850        local onlineimagesuffix
5851                      = (function(...)
5852                          local parser = nil
5853                          for _, suffix in ipairs({...}) do
5854                            local pattern=nil
5855                            for i=1,#suffix do
5856                              local char=suffix:sub(i,i)
5857                              char = S(char:lower()..char:upper())
5858                              if pattern == nil then
5859                                pattern = char
5860                              else
5861                                pattern = pattern * char
5862                              end
5863                            end
5864                            if parser == nil then
5865                              parser = pattern
5866                            else
5867                              parser = parser + pattern
5868                            end
5869                          end
5870                          return parser
5871                        end)("png", "jpg", "jpeg", "gif", "tif", "tiff")
5872
5873        -- online image url for iA Writer content blocks with mandatory suffix,
5874        -- allowing nested brackets:
5875        local onlineimageurl
5876                      = (parsers.less
5877                        * Cs((parsers.anyescaped
5878                             - parsers.more
5879                             - #(parsers.period
5880                                * onlineimagesuffix
5881                                * parsers.more
5882                                * contentblock_tail))^0)
5883                        * parsers.period
5884                        * Cs(onlineimagesuffix)
5885                        * parsers.more
5886                        + (Cs((parsers.inparens
5887                              + (parsers.anyescaped
5888                                - parsers.spacing
5889                                - parsers.rparent
5890                                - #(parsers.period
5891                                   * onlineimagesuffix
5892                                   * contentblock_tail)))^0)
5893                        * parsers.period
5894                        * Cs(onlineimagesuffix))
5895                        ) * Cc("onlineimage")
```

```
5896
5897        -- filename for iA Writer content blocks with mandatory suffix:
5898        local localfilepath
5899                      = parsers.slash
5900                      * Cs((parsers.anyescaped
5901                           - parsers.tab
5902                           - parsers.newline
5903                           - #(parsers.period
5904                              * parsers.alphanumeric^1
5905                              * contentblock_tail))^1)
5906                      * parsers.period
5907                      * Cs(parsers.alphanumeric^1)
5908                      * Cc("localfile")
5909
5910        local ContentBlock
5911                      = parsers.leader
5912                      * (localfilepath + onlineimageurl)
5913                      * contentblock_tail
5914                      / writer.contentblock
5915
5916        syntax.ContentBlock = ContentBlock
5917      end
5918    }
5919  end
```

3.1.6.3 Definition Lists

The `extensions.definition_lists` function implements the definition list syntax extension. If the `tight_lists` parameter is `true`, tight lists will produce special right item renderers.

```
5920  M.extensions.definition_lists = function(tight_lists)
5921    return {
5922      extend_writer = function(self)
```

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.

```
5923        local function dlitem(term, defs)
5924          local retVal = {"\\markdownRendererDlItem{",term,"}"}
5925          for _, def in ipairs(defs) do
5926            retVal[#retVal+1] = {"\\markdownRendererDlDefinitionBegin ",def,
5927                                 "\\markdownRendererDlDefinitionEnd "}
5928          end
5929          retVal[#retVal+1] = "\\markdownRendererDlItemEnd "
5930          return retVal
5931        end
5932
```

```
5933        function self.definitionlist(items,tight)
5934          if not self.is_writing then return "" end
5935          local buffer = {}
5936          for _,item in ipairs(items) do
5937            buffer[#buffer + 1] = dlitem(item.term, item.definitions)
5938          end
5939          if tight and tight_lists then
5940            return {"\\markdownRendererDlBeginTight\n", buffer,
5941              "\n\\markdownRendererDlEndTight"}
5942          else
5943            return {"\\markdownRendererDlBegin\n", buffer,
5944              "\n\\markdownRendererDlEnd"}
5945          end
5946        end
5947      end, extend_reader = function(self)
5948        local parsers = self.parsers
5949        local syntax = self.syntax
5950        local writer = self.writer
5951
5952        local defstartchar = S("~:")
5953
5954        local defstart = ( defstartchar * #parsers.spacing
5955                                        * (parsers.tab + parsers.space^-3)
5956                         + parsers.space * defstartchar * #parsers.spacing
5957                                        * (parsers.tab + parsers.space^-2)
5958                         + parsers.space * parsers.space * defstartchar
5959                                        * #parsers.spacing
5960                                        * (parsers.tab + parsers.space^-1)
5961                         + parsers.space * parsers.space * parsers.space
5962                                        * defstartchar * #parsers.spacing
5963                         )
5964
5965        local dlchunk = Cs(parsers.line * (parsers.indentedline - parsers.blankline)^0)
5966
5967        local function definition_list_item(term, defs, _)
5968          return { term = self.parser_functions.parse_inlines(term),
5969                   definitions = defs }
5970        end
5971
5972        local DefinitionListItemLoose
5973                     = C(parsers.line) * parsers.skipblanklines
5974                     * Ct((defstart
5975                          * parsers.indented_blocks(dlchunk)
5976                          / self.parser_functions.parse_blocks_nested)^1)
5977                     * Cc(false) / definition_list_item
5978
5979        local DefinitionListItemTight
```

```
                    = C(parsers.line)
                    * Ct((defstart * dlchunk
                         / self.parser_functions.parse_blocks_nested)^1)
                    * Cc(true) / definition_list_item

    local DefinitionList
                    = ( Ct(DefinitionListItemLoose^1) * Cc(false)
                      + Ct(DefinitionListItemTight^1)
                      * (parsers.skipblanklines
                        * -DefinitionListItemLoose * Cc(true))
                      ) / writer.definitionlist

    syntax.DefinitionList = DefinitionList
  end
 }
end
```

3.1.6.4 Fenced Code

The `extensions.fenced_code` function implements the commonmark fenced code block syntax extension. When the `blank_before_code_fence` parameter is `true`, the syntax extension requires between a paragraph and the following fenced code block.

```
M.extensions.fenced_code = function(blank_before_code_fence)
  return {
    extend_writer = function(self)
```

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

```
      function self.fencedCode(i, s)
        if not self.is_writing then return "" end
        s = string.gsub(s, '[\r\n%s]*$', '')
        local name = util.cache(self.cacheDir, s, nil, nil, ".verbatim")
        return {"\\markdownRendererInputFencedCode{",name,"}{",i,"}"}
      end
    end, extend_reader = function(self)
      local parsers = self.parsers
      local syntax = self.syntax
      local writer = self.writer

      local function captures_geq_length(_,i,a,b)
        return #a >= #b and i
      end

      local infostring   = (parsers.linechar - (parsers.backtick
                          + parsers.space^1 * (parsers.newline + parsers.eof)))^0

      local fenceindent
```

```
6018        local fencehead    = function(char)
6019          return             C(parsers.nonindentspace) / function(s) fenceindent = #s end
6020                           * Cg(char^3, "fencelength")
6021                           * parsers.optionalspace * C(infostring)
6022                           * parsers.optionalspace * (parsers.newline + parsers.eof)
6023        end
6024
6025        local fencetail    = function(char)
6026          return             parsers.nonindentspace
6027                           * Cmt(C(char^3) * Cb("fencelength"), captures_geq_length)
6028                           * parsers.optionalspace * (parsers.newline + parsers.eof)
6029                           + parsers.eof
6030        end
6031
6032        local fencedline   = function(char)
6033          return             C(parsers.line - fencetail(char))
6034                           / function(s)
6035                               local i = 1
6036                               local remaining = fenceindent
6037                               while true do
6038                                 local c = s:sub(i, i)
6039                                 if c == " " and remaining > 0 then
6040                                   remaining = remaining - 1
6041                                   i = i + 1
6042                                 elseif c == "\t" and remaining > 3 then
6043                                   remaining = remaining - 4
6044                                   i = i + 1
6045                                 else
6046                                   break
6047                                 end
6048                               end
6049                               return s:sub(i)
6050                             end
6051        end
6052
6053        local TildeFencedCode
6054               = fencehead(parsers.tilde)
6055               * Cs(fencedline(parsers.tilde)^0)
6056               * fencetail(parsers.tilde)
6057
6058        local BacktickFencedCode
6059               = fencehead(parsers.backtick)
6060               * Cs(fencedline(parsers.backtick)^0)
6061               * fencetail(parsers.backtick)
6062
6063        local FencedCode = (TildeFencedCode
6064                             + BacktickFencedCode)
```

```
                          / function(infostring, code)
                              return writer.fencedCode(writer.string(infostring),
                                                       self.expandtabs(code))
                            end

      syntax.FencedCode = FencedCode

      local fencestart
      if blank_before_code_fence then
        fencestart = parsers.fail
      else
        fencestart = fencehead(parsers.backtick)
                   + fencehead(parsers.tilde)
      end

      parsers.EndlineExceptions = parsers.EndlineExceptions + fencestart
      syntax.EndlineExceptions = parsers.EndlineExceptions
    end
  }
end
```

3.1.6.5 Footnotes

The `extensions.footnotes` function implements the Pandoc footnote and inline footnote syntax extensions. When the `footnote` parameter is `true`, the Pandoc footnote syntax extension will be enabled. When the `inline_footnotes` parameter is `true`, the Pandoc inline footnote syntax extension will be enabled.

```
M.extensions.footnotes = function(footnotes, inline_footnotes)
  assert(footnotes or inline_footnotes)
  return {
    extend_writer = function(self)
```

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

```
      function self.note(s)
        return {"\\markdownRendererFootnote{",s,"}"}
      end
    end, extend_reader = function(self)
      local parsers = self.parsers
      local syntax = self.syntax
      local writer = self.writer

      if footnotes then
        local function strip_first_char(s)
          return s:sub(2)
        end

```

```
6102        local RawNoteRef
6103                      = #(parsers.lbracket * parsers.circumflex)
6104                      * parsers.tag / strip_first_char
6105
6106        local rawnotes = {}
6107
6108        -- like indirect_link
6109        local function lookup_note(ref)
6110          return writer.defer_call(function()
6111            local found = rawnotes[self.normalize_tag(ref)]
6112            if found then
6113              return writer.note(
6114                self.parser_functions.parse_blocks_nested(found))
6115            else
6116              return {"[",
6117                self.parser_functions.parse_inlines("^" .. ref), "]"}
6118            end
6119          end)
6120        end
6121
6122        local function register_note(ref,rawnote)
6123          rawnotes[self.normalize_tag(ref)] = rawnote
6124          return ""
6125        end
6126
6127        local NoteRef = RawNoteRef / lookup_note
6128
6129        local NoteBlock
6130                    = parsers.leader * RawNoteRef * parsers.colon
6131                    * parsers.spnl * parsers.indented_blocks(parsers.chunk)
6132                    / register_note
6133
6134        parsers.Blank = NoteBlock + parsers.Blank
6135        syntax.Blank = parsers.Blank
6136
6137        syntax.NoteRef = NoteRef
6138      end
6139      if inline_footnotes then
6140        local InlineNote
6141                    = parsers.circumflex
6142                    * (parsers.tag / self.parser_functions.parse_inlines_no_inline_note)
6143                    / writer.note
6144        syntax.InlineNote = InlineNote
6145      end
6146
6147      self.add_special_character("^")
6148    end
```

```
  }
end
```

3.1.6.6 Header Attributes

The `extensions.header_attributes` function implements a syntax extension that enables the assignment of HTML attributes to headings.

```
M.extensions.header_attributes = function()
  return {
    extend_writer = function()
    end, extend_reader = function(self)
      local parsers = self.parsers
      local syntax = self.syntax
      local writer = self.writer

      parsers.AtxHeading = Cg(parsers.HeadingStart,"level")
                           * parsers.optionalspace
                           * (C(((parsers.linechar
                                 - ((parsers.hash^1
                                    * parsers.optionalspace
                                    * parsers.HeadingAttributes^-1
                                    + parsers.HeadingAttributes)
                                   * parsers.optionalspace
                                   * parsers.newline))
                                * (parsers.linechar
                                  - parsers.hash
                                  - parsers.lbrace)^0)^1)
                               / self.parser_functions.parse_inlines)
                           * Cg(Ct(parsers.newline
                                 + (parsers.hash^1
                                   * parsers.optionalspace
                                   * parsers.HeadingAttributes^-1
                                   + parsers.HeadingAttributes)
                                 * parsers.optionalspace
                                 * parsers.newline), "attributes")
                           * Cb("level")
                           * Cb("attributes")
                           / writer.heading

      parsers.SetextHeading = #(parsers.line * S("=-"))
                              * (C(((parsers.linechar
                                    - (parsers.HeadingAttributes
                                      * parsers.optionalspace
                                      * parsers.newline))
                                   * (parsers.linechar
                                     - parsers.lbrace)^0)^1)
                                  / self.parser_functions.parse_inlines)
                              * Cg(Ct(parsers.newline
```

```
                                    + (parsers.HeadingAttributes
                                      * parsers.optionalspace
                                      * parsers.newline)), "attributes")
                                * parsers.HeadingLevel
                                * Cb("attributes")
                                * parsers.optionalspace
                                * parsers.newline
                                / writer.heading

      parsers.Heading = parsers.AtxHeading + parsers.SetextHeading
      syntax.Heading = parsers.Heading
    end
  }
end
```

3.1.6.7 YAML Metadata

The `extensions.jekyll_data` function implements the Pandoc `yaml_metadata_block` syntax extension for entering metadata in YAML. When the `expect_jekyll_data` is `true`, then a markdown document may begin directly with YAML metadata and may contain nothing but YAML metadata

```
M.extensions.jekyll_data = function(expect_jekyll_data)
  return {
    extend_writer = function(self)
```

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`.

```
      function self.jekyllData(d, t, p)
        if not self.is_writing then return "" end

        local buf = {}

        local keys = {}
        for k, _ in pairs(d) do
          table.insert(keys, k)
        end
        table.sort(keys)

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

        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
    end, extend_reader = function(self)
      local parsers = self.parsers
      local syntax = self.syntax
      local writer = self.writer

      local JekyllData
                    = Cmt( C((parsers.line - P("---") - P("..."))^0)
                         , function(s, i, text) -- luacheck: ignore s i
                             local data
                             local ran_ok, _ = 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 self.parser_functions.parse_blocks_nested(s)
                               end, nil)
                             else
                               return false
                             end
                           end
                         )

      local UnexpectedJekyllData
                    = P("---")
                    * parsers.blankline / 0
                    * #(-parsers.blankline)  -- if followed by blank, it's an hrule
                    * JekyllData
                    * (P("---") + P("..."))

```

```
      local ExpectedJekyllData
                    = ( P("---")
                      * parsers.blankline / 0
                      * #(-parsers.blankline)  -- if followed by blank, it's an hrule
                      )^-1
                    * JekyllData
                    * (P("---") + P("..."))^-1

      syntax.UnexpectedJekyllData = UnexpectedJekyllData
      if expect_jekyll_data then
        syntax.ExpectedJekyllData = ExpectedJekyllData
      end
    end
  }
end
```

3.1.6.8 Pipe Tables

The `extensions.pipe_table` function implements the PHP Markdown table syntax extension (affectionately known as pipe tables). When the parameter `table_captions` is `true`, the function also implements the Pandoc `table_captions` syntax extension for table captions.

```
M.extensions.pipe_tables = function(table_captions)

  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

  return {
    extend_writer = function(self)
```

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.

```
      function self.table(rows, caption)
        if not self.is_writing then return "" end
        local buffer = {"\\markdownRendererTable{",
          caption or "", "}{", #rows - 1, "}{", #rows[1], "}"}
        local temp = rows[2] -- put alignments on the first row
        rows[2] = rows[1]
        rows[1] = temp
        for i, row in ipairs(rows) do
          table.insert(buffer, "{")
          for _, column in ipairs(row) do
            if i > 1 then -- do not use braces for alignments
              table.insert(buffer, "{")
```

```
            end
            table.insert(buffer, column)
            if i > 1 then
              table.insert(buffer, "}")
            end
          end
          table.insert(buffer, "}")
        end
        return buffer
      end
    end, extend_reader = function(self)
      local parsers = self.parsers
      local syntax = self.syntax
      local writer = self.writer

      local table_hline_separator = parsers.pipe + parsers.plus

      local table_hline_column = (parsers.dash
                                  - #(parsers.dash
                                     * (parsers.spacechar
                                       + table_hline_separator
                                       + parsers.newline)))^1
                               * (parsers.colon * Cc("r")
                                 + parsers.dash * Cc("d"))
                               + parsers.colon
                               * (parsers.dash
                                 - #(parsers.dash
                                    * (parsers.spacechar
                                      + table_hline_separator
                                      + parsers.newline)))^1
                               * (parsers.colon * Cc("c")
                                 + parsers.dash * Cc("l"))

      local table_hline = pipe_table_row(false
                                         , table_hline_column
                                         , table_hline_separator
                                         , table_hline_column)

      local table_caption_beginning = parsers.skipblanklines
                                    * parsers.nonindentspace
                                    * (P("Table")^-1 * parsers.colon)
                                    * parsers.optionalspace

      local table_row = pipe_table_row(true
                                       , (C((parsers.linechar - parsers.pipe)^1)
                                         / self.parser_functions.parse_inlines)
                                       , parsers.pipe
```

```
                                      , (C((parsers.linechar - parsers.pipe)^0)
                                        / self.parser_functions.parse_inlines))

      local table_caption
      if table_captions then
        table_caption = #table_caption_beginning
                      * table_caption_beginning
                      * Ct(parsers.IndentedInline^1)
                      * parsers.newline
      else
        table_caption = parsers.fail
      end

      local PipeTable = Ct(table_row * parsers.newline
                        * table_hline
                        * (parsers.newline * table_row)^0)
                      / make_pipe_table_rectangular
                      * table_caption^-1
                      / writer.table

      syntax.PipeTable = PipeTable
    end
  }
end
```

3.1.7 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)
```

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

```
  options = options or {}
  setmetatable(options, { __index = function (_, key)
    return defaultOptions[key] end })
% \par
% \begin{markdown}
%
% Apply syntax extensions based on `options`.
%
% \end{markdown}
%  \begin{macrocode}
  local extensions = {}

  if options.citations then
    local citations_extension = M.extensions.citations(options.citationNbsps)
```

```
6491      table.insert(extensions, citations_extension)
6492    end
6493
6494    if options.contentBlocks then
6495      local content_blocks_extension = M.extensions.content_blocks(
6496        options.contentBlocksLanguageMap)
6497      table.insert(extensions, content_blocks_extension)
6498    end
6499
6500    if options.definitionLists then
6501      local definition_lists_extension = M.extensions.definition_lists(
6502        options.tightLists)
6503      table.insert(extensions, definition_lists_extension)
6504    end
6505
6506    if options.fencedCode then
6507      local fenced_code_extension = M.extensions.fenced_code(
6508        options.blankBeforeCodeFence)
6509      table.insert(extensions, fenced_code_extension)
6510    end
6511
6512    if options.footnotes or options.inlineFootnotes then
6513      local footnotes_extension = M.extensions.footnotes(
6514        options.footnotes, options.inlineFootnotes)
6515      table.insert(extensions, footnotes_extension)
6516    end
6517
6518    if options.headerAttributes then
6519      local header_attributes_extension = M.extensions.header_attributes()
6520      table.insert(extensions, header_attributes_extension)
6521    end
6522
6523    if options.jekyllData then
6524      local jekyll_data_extension = M.extensions.jekyll_data(
6525        options.expectJekyllData)
6526      table.insert(extensions, jekyll_data_extension)
6527    end
6528
6529    if options.pipeTables then
6530      local pipe_tables_extension = M.extensions.pipe_tables(
6531        options.tableCaptions)
6532      table.insert(extensions, pipe_tables_extension)
6533    end
6534
6535    local writer = M.writer.new(options)
6536    local reader = M.reader.new(writer, options, extensions)
6537
```

```
  return reader.convert
end

return M
```

3.1.8 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 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{[ ]}%
```

3.2.2.1 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.

```
        \str_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
```

3.2.3 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
\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_option:n
      { #1 }
    \@@_get_option_type:nN
      { #1 }
      \l_tmpa_tl
    \bool_if:nTF
      {
        \str_if_eq_p:VV
          \l_tmpa_tl
          \c_@@_option_type_boolean_tl ||
        \str_if_eq_p:VV
          \l_tmpa_tl
          \c_@@_option_type_number_tl ||
        \str_if_eq_p:VV
          \l_tmpa_tl
          \c_@@_option_type_counter_tl
      }
      {
        \@@_get_option_value:nN
          { #1 }
          \l_tmpa_tl
        \tl_gput_right:Nx
          \g_@@_formatted_lua_options_tl
          { #1~=~  \l_tmpa_tl   ,~ }
      }
      {
        \@@_get_option_value:nN
          { #1 }
          \l_tmpa_tl
        \tl_gput_right:Nx
```

```
          \g_@@_formatted_lua_options_tl
          { #1~=~ " \l_tmpa_tl " ,~ }
      }
  }
\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)
}%
```

3.2.4 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
\cs_new:Nn
  \@@_if_option:nTF
  {
    \@@_get_option_type:nN
      { #1 }
      \l_tmpa_tl
    \str_if_eq:NNF
      \l_tmpa_tl
      \c_@@_option_type_boolean_tl
      {
        \msg_error:nnxx
          { @@ }
          { expected-boolean-option }
          { #1 }
          { \l_tmpa_tl }
```

```
      }
    \@@_get_option_value:nN
      { #1 }
      \l_tmpa_tl
    \str_if_eq:NNTF
      \l_tmpa_tl
      \c_@@_option_value_true_tl
      { #2 }
      { #3 }
  }
\msg_new:nnn
  { @@ }
  { expected-boolean-option }
  {
    Option~#1~has~type~#2,~
    but~a~boolean~was~expected.
  }
\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{@
```

```
6882      |begingroup@
```

If we are not reading markdown documents from the frozen cache, open the `\markdownOptionInputTempFileName` file for writing.

```
6883      |markdownIfOption{frozenCache}{}{@
6884        |immediate|openout|markdownOutputFileStream@
6885          |markdownOptionInputTempFileName|relax@
6886        |markdownInfo{Buffering markdown input into the temporary @
6887          input file "|markdownOptionInputTempFileName" and scanning @
6888          for the closing token sequence "#1"}@
6889      }@
```

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.

```
6890      |def|do##1{|catcode`##1=12}|dospecials@
6891      |catcode`| =12@
6892      |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.

```
6893      |def|markdownReadAndConvertStripPercentSign##1{@
6894        |markdownIfOption{stripPercentSigns}{@
6895          |if##1%@
6896            |expandafter|expandafter|expandafter@
6897              |markdownReadAndConvertProcessLine@
6898          |else@
6899            |expandafter|expandafter|expandafter@
6900              |markdownReadAndConvertProcessLine@
6901              |expandafter|expandafter|expandafter##1@
6902          |fi@
6903        }{@
6904          |expandafter@
6905            |markdownReadAndConvertProcessLine@
6906            |expandafter##1@
6907        }@
6908      }@
```

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.

```
6909      |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
```

The following two sections of the implementation have been deprecated and will be removed in Markdown 3.0.0. The code that corresponds to `\markdownMode` value of `3` will be the only implementation.

```
\ExplSyntaxOn
\int_compare:nT
  { \markdownMode = 3 }
  {
    \markdownInfo{Using~mode~3:~The~lt3luabridge~package}
    \file_input:n { lt3luabridge.tex }
    \cs_new:Npn
      \markdownLuaExecute
      { \luabridgeExecute }
  }
\ExplSyntaxOff
```

3.2.5 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 3.2.6 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
  \markdownWarning{Using mode 0: Shell escape via write18
                   (deprecated, to be removed in Markdown 3.0.0)}%
\else
  \markdownWarning{Using mode 1: Shell escape via os.execute
                   (deprecated, to be removed in Markdown 3.0.0)}%
\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
```

3.2.6 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 3.2.5 are meant to be indistinguishable to the remaining code. This corresponds to the `\markdownMode` value of 2.

```
\fi
\ifnum\markdownMode=2\relax
  \markdownWarning{Using mode 2: Direct Lua access
                   (deprecated, to be removed in Markdown 3.0.0)}%
```

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 3.2.5,

```
\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
```

3.2.7 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 frozenCacheCounter.

```
    |markdownIfOption{frozenCache}{%
      |ifnum|markdownOptionFrozenCacheCounter=0|relax
        |markdownInfo{Reading frozen cache from
          "|markdownOptionFrozenCacheFileName"}%
        |input|markdownOptionFrozenCacheFileName|relax
      |fi
      |markdownInfo{Including markdown document number
        "|the|markdownOptionFrozenCacheCounter" from frozen cache}%
      |csname markdownFrozenCache|the|markdownOptionFrozenCacheCounter|endcsname
      |global|advance|markdownOptionFrozenCacheCounter 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"))}%
```

In case we were finalizing the frozen cache, increment frozenCacheCounter.

```
      |global|advance|markdownOptionFrozenCacheCounter 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.

```
\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
        mkdir -p "\markdownOptionCacheDir";
        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.

```
\ExplSyntaxOn
\keys_define:nn
  { markdown/latex-options }
  {
    renderers .code:n = {
      \keys_set:nn
        { markdown/latex-options/renderers }
        { #1 }
    },
    rendererPrototypes .code:n = {
      \keys_set:nn
        { markdown/latex-options/renderer-prototypes }
        { #1 }
    },
    code .code:n = { #1 },
  }
\ExplSyntaxOff
```

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.

```
\markdownIfOption{plain}{\iffalse}{\iftrue}
```

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
          \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}},
```

[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.

```
  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.

```
\ExplSyntaxOn
\def\markdownLATEXStrongEmphasis#1{%
  \str_if_in:NnTF
    \f@series
    { b }
    { \textnormal{#1} }
    { \textbf{#1} }
}
\ExplSyntaxOff
\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}{}{
  \markdownSetup{
    rendererPrototypes = {
      inlineHtmlTag = {%
        \ifvmode
          \IgnorePar
          \EndP
        \fi
        \HCode{#1}%
      },
      inputBlockHtmlElement = {%
        \ifvmode
          \IgnorePar
        \fi
        \EndP
        \special{t4ht*<#1}%
        \par
        \ShowPar
      },
```

[9] See https://tug.org/tex4ht/.

```
    },
  }
}
```

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 label and the fully-escaped URI are equivalent and the title is empty, assume that the link is an autolink. Otherwise, assume that the link is either direct or indirect.

```
    \tl_set:Nn \l_tmpa_tl { ##1 }
    \tl_set:Nn \l_tmpb_tl { ##2 }
    \bool_set:Nn
      \l_tmpa_bool
      {
        \tl_if_eq_p:NN
          \l_tmpa_tl
          \l_tmpb_tl
      }
    \tl_set:Nn \l_tmpa_tl { ##4 }
    \bool_set:Nn
      \l_tmpb_bool
      {
        \tl_if_empty_p:N
          \l_tmpa_tl
      }
    \bool_if:nTF
      {
        \l_tmpa_bool && \l_tmpb_bool
      }
      {
        \markdownLaTeXRendererAutolink { ##2 } { ##3 }
      }{
        \markdownLaTeXRendererDirectOrIndirectLink { ##1 } { ##2 } { ##3 } { ##4 }
      }
  }
  \next
}
\def\markdownLaTeXRendererAutolink#1#2{%
```

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:Nn
    \l_tmpa_tl
    { #2 }
  \tl_trim_spaces:N
    \l_tmpa_tl
  \tl_set:Nx
```

```
    \l_tmpb_tl
    {
      \tl_range:Nnn
        \l_tmpa_tl
        { 1 }
        { 1 }
    }
  \str_if_eq:NNTF
    \l_tmpb_tl
    \c_hash_str
    {
      \tl_set:Nx
        \l_tmpb_tl
        {
          \tl_range:Nnn
            \l_tmpa_tl
            { 2 }
            { -1 }
        }
      \exp_args:NV
        \ref
        \l_tmpb_tl
    }{
      \url { #2 }
    }
}
\ExplSyntaxOff
\def\markdownLaTeXRendererDirectOrIndirectLink#1#2#3#4{%
  #1\footnote{\ifx\empty#4\empty\else#4: \fi\url{#3}}}
```

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{$\boxdot$}
```

```
8028 \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