

A Markdown Interpreter for T_EX

Vít Novotný
witiko@mail.muni.cz

Version 2.15.3-0-g7c8e03d
2022/06/27

Contents

1	Introduction	1	3	Implementation	87
1.1	Requirements	2	3.1	Lua Implementation . . .	87
1.2	Feedback	5	3.2	Plain T _E X Implementation	197
1.3	Acknowledgements	6	3.3	L ^A T _E X Implementation . .	212
2	Interfaces	6	3.4	ConT _E Xt Implementation	232
2.1	Lua Interface	6			
2.2	Plain T _E X Interface	34			
2.3	L ^A T _E X Interface	72			
2.4	ConT _E Xt Interface	86			
				References	237
				Index	238

List of Figures

1	A block diagram of the Markdown package	7
2	A sequence diagram of typesetting a document using the T _E X interface . .	31
3	A sequence diagram of typesetting a document using the Lua CLI	31
4	Various formats of mathematical formulae	79
5	The banner of the Markdown package	80
6	A pushdown automaton that recognizes T _E X comments	149

1 Introduction

The Markdown package¹ converts markdown² markup to T_EX commands. The functionality is provided both as a Lua module and as plain T_EX, L^AT_EX, and ConT_EXt macro packages that can be used to directly typeset T_EX documents containing markdown markup. Unlike other converters, 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

¹See <https://ctan.org/pkg/markdown>.

²See <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

```
1 local metadata = {
2   version   = "(((VERSION)))",
3   comment   = "A module for the conversion from markdown to plain TeX",
4   author    = "John MacFarlane, Hans Hagen, Vít Novotný",
5   copyright = {"2009-2016 John MacFarlane, Hans Hagen",
6               "2016-2022 Vít Novotný"},
7   license   = "LPPL 1.3c"
8 }
9
10 if not modules then modules = { } end
11 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 ≥ 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 ≥ 0.10 is included in LuaTeX $\geq 0.72.0$ (TeXLive ≥ 2013).

```
12 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 ≥ 2008).

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

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

```
14 if not ran_ok then
15   unicode = {"utf8"}={char=utf8.char}}
16 end
```

³See <http://mirrors.ctan.org/macros/generic/markdown/markdown.html>.

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

```
17 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 `jeekyllData` 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 L^AT_EX3 kernel in TeX Live \leq 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.

```
18 <@@=markdown>
19 \ifx\ExplSyntaxOn\undefined
20   \input expl3-generic\relax
21 \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 \geq 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 Lua \TeX `\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 pdf \TeX and Xe \TeX) 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 $\text{\LaTeX} 2_\epsilon$ format is loaded,

22 `\NeedsTeXFormat{LaTeX2e}%`

a \TeX engine that extends ϵ - \TeX , all the plain \TeX prerequisites (see Section 1.1.2), and the following $\text{\LaTeX} 2_\epsilon$ packages:

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

23 `\RequirePackage{keyval}`

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

24 `\RequirePackage{xstring}`

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

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

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

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

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

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

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

gobble A package that provides the `\@gobblethree` TeX command that is used in the default renderer prototype for citations. The package is included in TeXLive \geq 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` L^AT_EX 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 $2006 \leq \text{TeX Live} \leq 2019$. Since TeX Live ≥ 2020 , the functionality of the package has been integrated in the L^AT_EX 2_ε kernel. It is used in the `witiko/dot` and `witiko/graphicx/http` L^AT_EX 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 L^AT_EX3 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 L^AT_EX themes (see Section 3.3.2.1).

25 `\RequirePackage{expl3}`

1.1.4 ConT_EXt Prerequisites

The ConT_EXt 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 ConT_EXt 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 GitHub⁴ to report bugs and submit feature requests. If you do not want to report a bug or request a feature but are simply

⁴See <https://github.com/witiko/markdown/issues>.

in need of assistance, you might want to consider posting your question to the T_EX-L^AT_EX Stack Exchange.⁵ community question answering web site under the `markdown` tag.

1.3 Acknowledgements

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

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 T_EX implementation of the package draws inspiration from several sources including the source code of L^AT_EX 2_ε, the minted package by Geoffrey M. Poore, which likewise tackles the issue of interfacing with an external interpreter from T_EX, 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 T_EX 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 T_EX *token renderers* is exposed by the Lua layer. The plain T_EX layer exposes the conversion capabilities of Lua as T_EX macros. The L^AT_EX and ConT_EXt layers provide syntactic sugar on top of plain T_EX 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 T_EX. This interface is used by the plain T_EX implementation (see Section 3.2) and will be of interest to the developers of other packages and Lua modules.

⁵See <https://tex.stackexchange.com>.

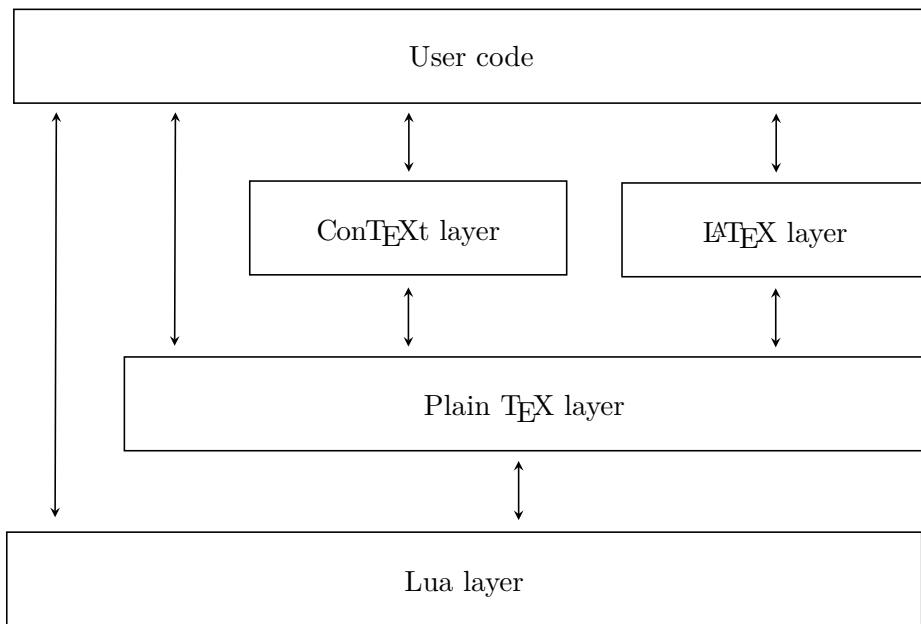


Figure 1: A block diagram of the Markdown package

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

```
26 local M = {metadata = metadata}
```

2.1.1 Conversion from Markdown to Plain T_EX

The Lua interface exposes the `new(options)` method. This method creates converter functions that perform the conversion from markdown to plain T_EX 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 T_EX output using the default options and prints the T_EX 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.

```
27 local defaultOptions = {}
```

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

```
28 \ExplSyntaxOn
```

```
29 \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.

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

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

```
32 \seq_new:N \g_@@_option_layers_seq
```

```
33 \tl_const:Nn \c_@@_option_layer_lua_tl { lua }
```

```
34 \seq_put_right:NV \g_@@_option_layers_seq \c_@@_option_layer_lua_tl
```

```
35 \cs_new:Nn
```

```
36   \@@_add_lua_option:nnn
```

```
37   {
```

```
38     \@@_add_option:Vnnn
```

```
39     \c_@@_option_layer_lua_tl
```

```
40     { #1 }
```

```
41     { #2 }
```

```
42     { #3 }
```

```
43   }
```

```
44 \cs_new:Nn
```

```
45   \@@_add_option:nnnn
```

```
46   {
```

```
47     \seq_put_right:cn
```

```
48     { g_@@_ #1 _options_seq }
```

```
49     { #2 }
```

```
50     \prop_put:cnn
```

```
51     { g_@@_ #1 _option_types_prop }
```

```
52     { #2 }
```

```
53     { #3 }
```

```
54     \prop_put:cnn
```

```
55     { g_@@_default_ #1 _options_prop }
```

```
56     { #2 }
```

```
57     { #4 }
```

```
58     \@@_typecheck_option:n
```

```
59     { #2 }
```

```
60   }
```

```
61 \cs_generate_variant:Nn
```

```
62   \@@_add_option:nnnn
```

```
63   { Vnnn }
```

```
64 \tl_const:Nn \c_@@_option_value_true_tl { true }
```

```
65 \tl_const:Nn \c_@@_option_value_false_tl { false }
```

```
66 \cs_new:Nn \@@_typecheck_option:n
```

```
67   {
```



```

68   \@@_get_option_type:nN
69   { #1 }
70   \l_tmpa_tl
71   \str_case_e:Vn
72   \l_tmpa_tl
73   {
74     { \c_@@_option_type_boolean_tl }
75     {
76       \@@_get_option_value:nN
77       { #1 }
78       \l_tmpa_tl
79       \bool_if:nF
80       {
81         \str_if_eq_p:VV
82         \l_tmpa_tl
83         \c_@@_option_value_true_tl ||
84         \str_if_eq_p:VV
85         \l_tmpa_tl
86         \c_@@_option_value_false_tl
87       }
88       {
89         \msg_error:nnnV
90         { @@ }
91         { failed-typecheck-for-boolean-option }
92         { #1 }
93         \l_tmpa_tl
94       }
95     }
96   }
97 }
98 \msg_new:nnn
99 { @@ }
100 { failed-typecheck-for-boolean-option }
101 {
102   Option~#1~has~value~#2,~
103   but~a~boolean~(true~or~false)~was~expected.
104 }
105 \cs_generate_variant:Nn
106 \str_case_e:nn
107 { Vn }
108 \cs_generate_variant:Nn
109 \msg_error:nnnn
110 { nnnV }
111 \seq_new:N \g_@@_option_types_seq
112 \tl_const:Nn \c_@@_option_type_counter_tl { counter }
113 \seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_counter_tl
114 \tl_const:Nn \c_@@_option_type_boolean_tl { boolean }

```

```

115 \seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_boolean_tl
116 \tl_const:Nn \c_@@_option_type_number_tl { number }
117 \seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_number_tl
118 \tl_const:Nn \c_@@_option_type_path_tl { path }
119 \seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_path_tl
120 \tl_const:Nn \c_@@_option_type_slice_tl { slice }
121 \seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_slice_tl
122 \tl_const:Nn \c_@@_option_type_string_tl { string }
123 \seq_put_right:NV \g_@@_option_types_seq \c_@@_option_type_string_tl
124 \cs_new:Nn
125   \@@_get_option_type:nN
126   {
127     \bool_set_false:N
128       \l_tmpa_bool
129     \seq_map_inline:Nn
130       \g_@@_option_layers_seq
131       {
132         \prop_get:cnNT
133           { g_@@_ ##1 _option_types_prop }
134           { #1 }
135         \l_tmpa_tl
136         {
137           \bool_set_true:N
138             \l_tmpa_bool
139           \seq_map_break:
140         }
141       }
142     \bool_if:nF
143       \l_tmpa_bool
144     {
145       \msg_error:nnn
146         { @@ }
147         { undefined-option }
148         { #1 }
149     }
150     \seq_if_in:NVF
151       \g_@@_option_types_seq
152       \l_tmpa_tl
153     {
154       \msg_error:nnnV
155         { @@ }
156         { unknown-option-type }
157         { #1 }
158       \l_tmpa_tl
159     }
160     \tl_set_eq:NN
161       #2

```

```

162     \l_tmpa_tl
163 }
164 \msg_new:nnn
165 { @@ }
166 { unknown-option-type }
167 {
168     Option~#1~has~unknown~type~#2.
169 }
170 \msg_new:nnn
171 { @@ }
172 { undefined-option }
173 {
174     Option~#1~is~undefined.
175 }
176 \cs_new:Nn
177 \@@_get_default_option_value:nN
178 {
179     \bool_set_false:N
180         \l_tmpa_bool
181     \seq_map_inline:Nn
182         \g_@@_option_layers_seq
183         {
184             \prop_get:cnNT
185                 { g_@@_default_ ##1 _options_prop }
186                 { #1 }
187             #2
188             {
189                 \bool_set_true:N
190                     \l_tmpa_bool
191                 \seq_map_break:
192             }
193         }
194     \bool_if:nF
195         \l_tmpa_bool
196     {
197         \msg_error:nnn
198             { @@ }
199             { undefined-option }
200             { #1 }
201     }
202 }
203 \cs_new:Nn
204 \@@_get_option_value:nN
205 {
206     \@@_option_tl_to_csname:nN
207         { #1 }
208     \l_tmpa_tl

```

```

209 \cs_if_free:cTF
210 { \l_tmpa_tl }
211 {
212   \@@_get_default_option_value:nN
213   { #1 }
214   #2
215 }
216 {
217   \@@_get_option_type:nN
218   { #1 }
219   \l_tmpa_tl
220   \str_if_eq:NNTF
221   \c_@@_option_type_counter_tl
222   \l_tmpa_tl
223   {
224     \@@_option_tl_to_csname:nN
225     { #1 }
226     \l_tmpa_tl
227     \tl_set:Nx
228     #2
229     { \the \cs:w \l_tmpa_tl \cs_end: }
230   }
231   {
232     \@@_option_tl_to_csname:nN
233     { #1 }
234     \l_tmpa_tl
235     \tl_set:Nv
236     #2
237     { \l_tmpa_tl }
238   }
239 }
240 }
241 \cs_new:Nn \@@_option_tl_to_csname:nN
242 {
243   \tl_set:Nn
244   \l_tmpa_tl
245   % TODO: Replace with \str_uppercase:n in TeX Live 2020.
246   { \str_upper_case:n { #1 } }
247   \tl_set:Nx
248   #2
249   {
250     markdownOption
251     \tl_head:f { \l_tmpa_tl }
252     \tl_tail:n { #1 }
253   }
254 }

```

2.1.3 File and Directory Names

`cacheDir`= $\langle path \rangle$ 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 T_EX 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.

```
255 \@@_add_lua_option:nnn
256   { cacheDir }
257   { path }
258   { \markdownOptionOutputDir / _markdown_\jobname }
259 defaultOptions.cacheDir = "."
```

`frozenCacheFileName`= $\langle path \rangle$ 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 T_EX document that contains markdown documents without invoking Lua using the `\markdownOptionFrozenCache` plain T_EX option. As a result, the plain T_EX document becomes more portable, but further changes in the order and the content of markdown documents will not be reflected.

```
260 \@@_add_lua_option:nnn
261   { frozenCacheFileName }
262   { path }
263   { \markdownOptionCacheDir / frozenCache.tex }
264 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.

```
265 \@@_add_lua_option:nnn
266 { blankBeforeBlockquote }
267 { boolean }
268 { false }
269 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.

```
270 \@@_add_lua_option:nnn
271 { blankBeforeCodeFence }
272 { boolean }
273 { false }
274 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.

```
275 \@@_add_lua_option:nnn
276 { blankBeforeHeading }
277 { boolean }
278 { false }
279 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.

```
280 \@@_add_lua_option:nnn
281   { breakableBlockquotes }
282   { boolean }
283   { false }
284 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.

```
285 \@@_add_lua_option:nnn
286   { citationNbsps }
287   { boolean }
288   { true }
289 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.

```
290 \@@_add_lua_option:nnn
291   { citations }
292   { boolean }
293   { false }
294 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.``
```

```
295 \@@_add_lua_option:nnn
296   { codeSpans }
297   { boolean }
298   { true }
299 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.

```
300 \@@_add_lua_option:nnn
301   { contentBlocks }
302   { boolean }
303   { false }
304 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.

```
305 \@@_add_lua_option:nnn
306   { contentBlocksLanguageMap }
307   { path }
308   { markdown-languages.json }
309 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.

```
310 \@@_add_lua_option:nnn
311   { definitionLists }
312   { boolean }
313   { false }
314 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.

```
315 \@@_add_lua_option:nnn
316   { eagerCache }
317   { boolean }
318   { true }
319 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}

```

```

320 \@@_add_lua_option:nnn
321   { expectJekyllData }
322   { boolean }
323   { false }
324 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.

```

325 \@@_add_lua_option:nnn
326 { fencedCode }
327 { boolean }
328 { false }
329 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 T<sub>E</sub>X document that contains markdown documents without invoking Lua using the **\markdownOptionFrozenCache** plain T<sub>E</sub>X option. As a result, the plain T<sub>E</sub>X document becomes more portable, but further changes in the order and the content of markdown documents will not be reflected.

```

330 \@@_add_lua_option:nnn
331 { finalizeCache }
332 { boolean }
333 { false }
334 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.

```
335 \@@_add_lua_option:nnn
336 { footnotes }
337 { boolean }
338 { false }
339 defaultOptions.footnotes = false
```

**frozenCacheCounter**= $\langle number \rangle$  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 T<sub>E</sub>X macro **\markdownFrozenCache** $\langle number \rangle$  that will typeset markdown document number  $\langle number \rangle$ .

```
340 \@@_add_lua_option:nnn
341 { frozenCacheCounter }
342 { counter }
343 { 0 }
344 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.

```
345 \@@_add_lua_option:nnn
346 { hardLineBreaks }
347 { boolean }
348 { false }
349 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.

```
350 \@@_add_lua_option:nnn
351 { hashEnumerators }
352 { boolean }
353 { false }

354 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.

```
355 \@@_add_lua_option:nnn
356 { headerAttributes }
357 { boolean }
358 { false }

359 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.

```
360 \@@_add_lua_option:nnn
361 { html }
362 { boolean }
363 { false }
364 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 interpreted by  $\TeX$ . This is encouraged when typesetting automatically generated content or markdown documents that were not prepared with this package in mind.

```
365 \@@_add_lua_option:nnn
366 { hybrid }
367 { boolean }
368 { false }
369 defaultOptions.hybrid = false
```

`inlineFootnotes=true, false` default: false

- true** Enable the Pandoc inline footnote syntax extension:

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

`false`      Disable the Pandoc inline footnote syntax extension.

```
370 \@@_add_lua_option:nnn
371 { inlineFootnotes }
372 { boolean }
373 { false }

374 defaultOptions.inlineFootnotes = false
```

`jeekyllData=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.

```
375 \@@_add_lua_option:nnn
376 { jeekyllData }
377 { boolean }
378 { false }

379 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.

```
380 \@@_add_lua_option:nnn
381 { pipeTables }
382 { boolean }
383 { false }
384 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.

```
385 \@@_add_lua_option:nnn
386 { preserveTabs }
387 { boolean }
388 { false }
389 defaultOptions.preserveTabs = false
```

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

`true`      Enable relative references<sup>6</sup> 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.

```
390 \@@_add_lua_option:nnn
391 { relativeReferences }
392 { boolean }
393 { false }
394 defaultOptions.relativeReferences = false
```

---

<sup>6</sup>See <https://datatracker.ietf.org/doc/html/rfc3986#section-4.2>.

`shiftHeadings`= $\langle shift\ amount \rangle$  default: 0

All headings will be shifted by  $\langle shift\ amount \rangle$ , 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  $\langle shift\ amount \rangle$  is positive, and to level 1, when  $\langle shift\ amount \rangle$  is negative.

```
395 \@@_add_lua_option:nnn
396 { shiftHeadings }
397 { number }
398 { 0 }
399 defaultOptions.shiftHeadings = 0
```

`slice`= $\langle the\ beginning\ and\ the\ end\ of\ a\ slice \rangle$  default:  $\wedge$  \$

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 ( $\wedge$ ) selects the beginning of a document.
- The dollar sign (\$) selects the end of a document.
- $\wedge\langle identifier \rangle$  selects the beginning of a section with the HTML attribute  $\# \langle identifier \rangle$  (see the `headerAttributes` option).
- $\$ \langle identifier \rangle$  selects the end of a section with the HTML attribute  $\# \langle identifier \rangle$ .
- $\langle identifier \rangle$  corresponds to  $\wedge\langle identifier \rangle$  for the first selector and to  $\$ \langle identifier \rangle$  for the second selector.

Specifying only a single selector,  $\langle identifier \rangle$ , is equivalent to specifying the two selectors  $\langle identifier \rangle \langle identifier \rangle$ , which is equivalent to  $\wedge\langle identifier \rangle \$ \langle identifier \rangle$ , i.e. the entire section with the HTML attribute  $\# \langle identifier \rangle$  will be selected.

```
400 \@@_add_lua_option:nnn
401 { slice }
402 { slice }
403 { \wedge ~$ }
404 defaultOptions.slice = " \wedge $"
```

`smartEllipses`=true, false default: false

- |                    |                                                                                            |
|--------------------|--------------------------------------------------------------------------------------------|
| <code>true</code>  | Convert any ellipses in the input to the <code>\markdownRendererEllipsis</code> TeX macro. |
| <code>false</code> | Preserve all ellipses in the input.                                                        |

```

405 \@@_add_lua_option:nnn
406 { smartEllipses }
407 { boolean }
408 { false }
409 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 `\markdownRendererOliItemWithNumber` T<sub>E</sub>X macro.
- false**     Ignore the numbers in the ordered list items. Each item will only produce a `\markdownRendererOliItem` T<sub>E</sub>X macro.

```

410 \@@_add_lua_option:nnn
411 { startNumber }
412 { boolean }
413 { true }
414 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.

```

415 \@@_add_lua_option:nnn
416 { stripIndent }
417 { boolean }
418 { false }
419 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.

```
420 \@@_add_lua_option:nnn
421 { tableCaptions }
422 { boolean }
423 { false }
424 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.

```
425 \@@_add_lua_option:nnn
426 { taskLists }
427 { boolean }
428 { false }
429 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.

```
430 \@@_add_lua_option:nnn
431 { texComments }
432 { boolean }
433 { false }
434 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.

```
435 \@@_add_lua_option:nnn
436 { tightLists }
437 { boolean }
438 { true }
```

```
439 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.

```
440 \@@_add_lua_option:nnn
441 { underscores }
442 { boolean }
443 { true }
444 \ExplSyntaxOff
445 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  $\text{\TeX}$  layer hands markdown documents to the Lua layer. Lua converts the documents to  $\text{\TeX}$ , and hands the converted documents back to plain  $\text{\TeX}$  layer for typesetting, see Figure 2.

This procedure has the advantage of being fully automated. However, it also has several important disadvantages: The converted  $\text{\TeX}$  documents are cached on the file system, taking up increasing amount of space. Unless the  $\text{\TeX}$  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  $\text{\TeX}$  is also provided, see Figure 3.

```
446
447 HELP_STRING = [[
448 Usage: texlua]] .. arg[0] .. [[[OPTIONS] -- [INPUT_FILE] [OUTPUT_FILE]
449 where OPTIONS are documented in the Lua interface section of the
450 technical Markdown package documentation.
```

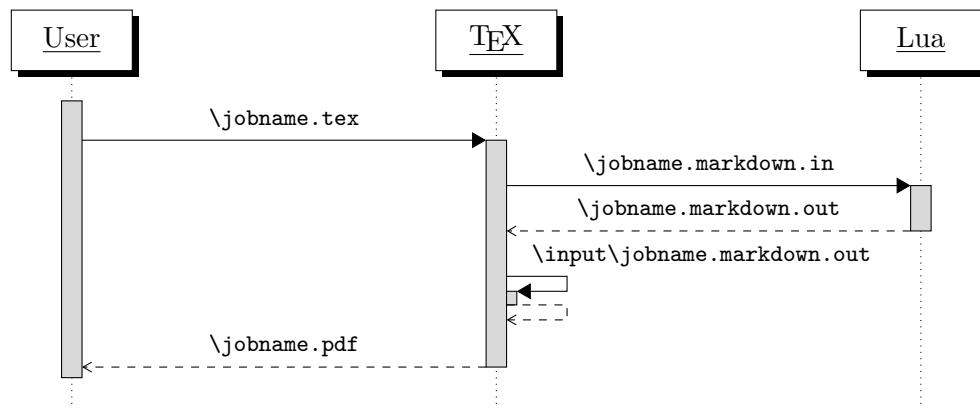


Figure 2: A sequence diagram of the Markdown package typesetting a markdown document using the TeX interface

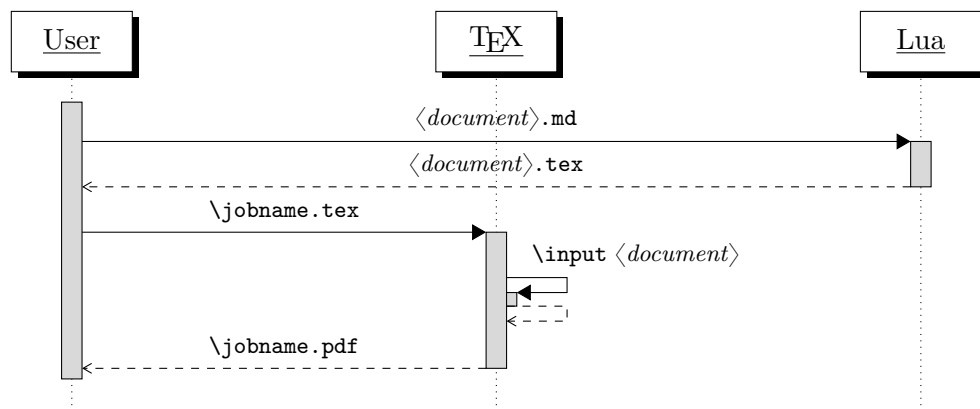


Figure 3: A sequence diagram of the Markdown package typesetting a markdown document using the Lua command-line interface

```

451
452 When OUTPUT_FILE is unspecified, the result of the conversion will be
453 written to the standard output. When INPUT_FILE is also unspecified, the
454 result of the conversion will be read from the standard input.
455
456 Report bugs to: witiko@mail.muni.cz
457 Markdown package home page: <https://github.com/witiko/markdown>]]
458
459 VERSION_STRING = [[
460 markdown-cli.lua (Markdown)]] .. metadata.version .. [[
461
462 Copyright (C)]] .. table.concat(metadata.copyright,
463 "\nCopyright (C) ") .. [[
464
465 License:]] .. metadata.license
466
467 local function warn(s)
468 io.stderr:write("Warning: " .. s .. "\n") end
469
470 local function error(s)
471 io.stderr:write("Error: " .. s .. "\n")
472 os.exit(1) end
473
474 local process_options = true
475 local options = {}
476 local input_filename
477 local output_filename
478 for i = 1, #arg do
479 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.

```

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

```

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

```

The `defaultOptions` table is consulted to identify whether `<value>` should be parsed as a string or as a boolean.

```

485 default_type = type(defaultOptions[key])
486 if default_type == "boolean" then

```



```

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

```

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

```

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

```

The first argument that matches none of the above patterns is assumed to be the input filename. The input filename should correspond to the Markdown document that is going to be converted to a  $\text{\TeX}$  document.

```

512 if input_filename == nil then
513 input_filename = arg[i]

```

The first argument that matches none of the above patterns is assumed to be the output filename. The output filename should correspond to the  $\text{\TeX}$  document that will result from the conversion.

```

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

```

```
519 ::continue::
520 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 T<sub>E</sub>X document `hello.tex`. After the Markdown package for our T<sub>E</sub>X format has been loaded, the converted document can be typeset as follows:

```
\input hello
```

## 2.2 Plain T<sub>E</sub>X Interface

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

```
521 \def\markdownLastModified{(((LASTMODIFIED)))}%
522 \def\markdownVersion{(((VERSION)))}%
```

The plain T<sub>E</sub>X interface is implemented by the `markdown.tex` file that can be loaded as follows:

```
\input markdown
```

It is expected that the special plain T<sub>E</sub>X 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.

```
523 \let\markdownBegin\relax
524 \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 corollary, 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 T<sub>E</sub>X [4, p. 46]. As a corollary, 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 T<sub>E</sub>X 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 T<sub>E</sub>X 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 T<sub>E</sub>X.

525 `\let\markdownInput\relax`

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

The following example plain T<sub>E</sub>X 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.

```
526 \ExplSyntaxOn
527 \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.

```
528 \prop_new:N \g_@@_plain_tex_option_types_prop
529 \prop_new:N \g_@@_default_plain_tex_options_prop
530 \tl_const:Nn \c_@@_option_layer_plain_tex_tl { plain_tex }
531 \seq_put_right:NV \g_@@_option_layers_seq \c_@@_option_layer_plain_tex_tl
532 \cs_new:Nn
533 \@@_add_plain_tex_option:nnn
534 {
535 \@@_add_option:Vnnn
536 \c_@@_option_layer_plain_tex_tl
537 { #1 }
538 { #2 }
539 { #3 }
540 }
```

**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`.

```
541 \@@_add_plain_tex_option:nnn
542 { frozenCache }
543 { boolean }
544 { 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 T<sub>E</sub>X document to populate and finalize the cache.
5. Enable the `\markdownOptionFrozenCache` option.
6. Publish the source code of the plain T<sub>E</sub>X 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 T<sub>E</sub>X in T<sub>E</sub>X 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 T<sub>E</sub>X engines tend to put quotation marks around `\jobname`, when it contains spaces.

```
545 \@@_add_plain_tex_option:nnn
546 { helperScriptFileName }
547 { path }
548 { \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.

```
549 \str_new:N
550 \g_luabridge_helper_script_filename_str
551 \tl_gset:Nn
552 \g_luabridge_helper_script_filename_str
553 { \markdownOptionHelperScriptFileName }
```

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

```
554 \@@_add_plain_tex_option:nnn
555 { inputTempFileName }
556 { path }
557 { \jobname.markdown.in }
```

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

```
558 \@@_add_plain_tex_option:nnn
559 { outputTempFileName }
560 { path }
561 { \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.

```
562 \str_new:N
563 \g_luabridge_standard_output_filename_str
564 \tl_gset:Nn
565 \g_luabridge_standard_output_filename_str
566 { \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 T<sub>E</sub>X in `\markdownMode` other than 2. It defaults to `\jobname.markdown.err`. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
567 \@@_add_plain_tex_option:nnn
568 { errorTempFileName }
569 { path }
570 { \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.

```
571 \str_new:N
572 \g_luabridge_error_output_filename_str
573 \tl_gset:Nn
574 \g_luabridge_error_output_filename_str
575 { \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 T<sub>E</sub>X implementation. The option defaults to `..`.

The path must be set to the same value as the `-output-directory` option of your T<sub>E</sub>X 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.

```
576 \@@_add_plain_tex_option:nnn
577 { outputDir }
578 { path }
579 { . }
```

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

```
580 \str_new:N
581 \g_luabridge_output_dirname_str
```

```

582 \tl_gset:Nn
583 \g_luabridge_output_dirname_str
584 { \markdownOptionOutputDir }

```

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

Furthermore, 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.

```

585 \cs_new:Nn \@@_plain_tex_define_option_commands:
586 {
587 \seq_map_inline:Nn
588 \g_@@_option_layers_seq
589 {
590 \seq_map_inline:cn
591 { g_@@_ ##1 _options_seq }
592 {
593 \@@_plain_tex_define_option_command:n
594 { ####1 }
595 }
596 }
597 }
598 \cs_new:Nn \@@_plain_tex_define_option_command:n
599 {
600 \@@_get_default_option_value:nN
601 { #1 }
602 \l_tmpa_tl
603 \@@_set_option_value:nV
604 { #1 }
605 \l_tmpa_tl
606 }
607 \cs_new:Nn
608 \@@_set_option_value:nn
609 {
610 \@@_define_option:n
611 { #1 }
612 \@@_get_option_type:nN
613 { #1 }
614 \l_tmpa_tl
615 \str_if_eq:NNTF
616 \c_@@_option_type_counter_tl
617 \l_tmpa_tl
618 {
619 \@@_option_tl_to_csname:nN

```

```

620 { #1 }
621 \l_tmpa_tl
622 \int_gset:cn
623 { \l_tmpa_tl }
624 { #2 }
625 }
626 {
627 \@@_option_tl_to_csname:nN
628 { #1 }
629 \l_tmpa_tl
630 \cs_set:cpn
631 { \l_tmpa_tl }
632 { #2 }
633 }
634 }
635 \cs_generate_variant:Nn
636 \@@_set_option_value:nn
637 { nV }
638 \cs_new:Nn
639 \@@_define_option:n
640 {
641 \@@_option_tl_to_csname:nN
642 { #1 }
643 \l_tmpa_tl
644 \cs_if_free:cT
645 { \l_tmpa_tl }
646 {
647 \@@_get_option_type:nN
648 { #1 }
649 \l_tmpb_tl
650 \str_if_eq:NNT
651 \c_@@_option_type_counter_tl
652 \l_tmpb_tl
653 {
654 \@@_option_tl_to_csname:nN
655 { #1 }
656 \l_tmpa_tl
657 \int_new:c
658 { \l_tmpa_tl }
659 }
660 }
661 }
662 \@@_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 T<sub>E</sub>X package documentation using the Doc L<sup>A</sup>T<sub>E</sub>X package [5] or similar. The recognized values of the macro are `true` (discard) and `false` (retain). It defaults to `false`.

```

663 \seq_put_right:Nn
664 \g_@@_plain_tex_options_seq
665 { stripPercentSigns }
666 \prop_put:Nnn
667 \g_@@_plain_tex_option_types_prop
668 { stripPercentSigns }
669 { boolean }
670 \prop_put:Nnx
671 \g_@@_default_plain_tex_options_prop
672 { stripPercentSigns }
673 { false }
674 \ExplSyntaxOff

```

## 2.2.3 Token Renderers

The following T<sub>E</sub>X 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.

```

675 \ExplSyntaxOn
676 \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.

```

677 \prop_new:N \g_@@_renderer_arities_prop
678 \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.

```

679 \def\markdownRendererTickedBox{%
680 \markdownRendererTickedBoxPrototype}%
681 \ExplSyntaxOn
682 \seq_put_right:Nn
683 \g_@@_renderers_seq
684 { tickedBox }

```

```

685 \prop_put:Nnn
686 \g_@@_renderer_arities_prop
687 { tickedBox }
688 { 0 }
689 \ExplSyntaxOff
690 \def\markdownRendererHalfTickedBox{%
691 \markdownRendererHalfTickedBoxPrototype}%
692 \ExplSyntaxOn
693 \seq_put_right:Nn
694 \g_@@_renderers_seq
695 { halfTickedBox }
696 \prop_put:Nnn
697 \g_@@_renderer_arities_prop
698 { halfTickedBox }
699 { 0 }
700 \ExplSyntaxOff
701 \def\markdownRendererUntickedBox{%
702 \markdownRendererUntickedBoxPrototype}%
703 \ExplSyntaxOn
704 \seq_put_right:Nn
705 \g_@@_renderers_seq
706 { untickedBox }
707 \prop_put:Nnn
708 \g_@@_renderer_arities_prop
709 { untickedBox }
710 { 0 }
711 \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  $\text{\TeX}$  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.

```

712 \def\markdownRendererDocumentBegin{%
713 \markdownRendererDocumentBeginPrototype}%
714 \ExplSyntaxOn
715 \seq_put_right:Nn
716 \g_@@_renderers_seq
717 { documentBegin }
718 \prop_put:Nnn
719 \g_@@_renderer_arities_prop
720 { documentBegin }
721 { 0 }

```

```

722 \ExplSyntaxOff
723 \def\markdownRendererDocumentEnd{%
724 \markdownRendererDocumentEndPrototype}%
725 \ExplSyntaxOn
726 \seq_put_right:Nn
727 \g_@@_renderers_seq
728 { documentEnd }
729 \prop_put:Nnn
730 \g_@@_renderer_arities_prop
731 { documentEnd }
732 { 0 }
733 \ExplSyntaxOff

```

**2.2.3.3 Interblock Separator Renderer** The `\markdownRendererInterblockSeparator` macro represents a separator between two markdown block elements. The macro receives no arguments.

```

734 \def\markdownRendererInterblockSeparator{%
735 \markdownRendererInterblockSeparatorPrototype}%
736 \ExplSyntaxOn
737 \seq_put_right:Nn
738 \g_@@_renderers_seq
739 { interblockSeparator }
740 \prop_put:Nnn
741 \g_@@_renderer_arities_prop
742 { interblockSeparator }
743 { 0 }
744 \ExplSyntaxOff

```

**2.2.3.4 Line Break Renderer** The `\markdownRendererLineBreak` macro represents a forced line break. The macro receives no arguments.

```

745 \def\markdownRendererLineBreak{%
746 \markdownRendererLineBreakPrototype}%
747 \ExplSyntaxOn
748 \seq_put_right:Nn
749 \g_@@_renderers_seq
750 { lineBreak }
751 \prop_put:Nnn
752 \g_@@_renderer_arities_prop
753 { lineBreak }
754 { 0 }
755 \ExplSyntaxOff

```

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

```

756 \def\markdownRendererEllipsis{%
757 \markdownRendererEllipsisPrototype}%
758 \ExplSyntaxOn
759 \seq_put_right:Nn
760 \g_@@_renderers_seq
761 { ellipsis }
762 \prop_put:Nnn
763 \g_@@_renderer_arities_prop
764 { ellipsis }
765 { 0 }
766 \ExplSyntaxOff

```

**2.2.3.6 Non-Breaking Space Renderer** The `\markdownRendererNbsp` macro represents a non-breaking space.

```

767 \def\markdownRendererNbsp{%
768 \markdownRendererNbspPrototype}%
769 \ExplSyntaxOn
770 \seq_put_right:Nn
771 \g_@@_renderers_seq
772 { nbsp }
773 \prop_put:Nnn
774 \g_@@_renderer_arities_prop
775 { nbsp }
776 { 0 }
777 \ExplSyntaxOff

```

**2.2.3.7 Special Character Renderers** The following macros replace any special plain  $\text{\TeX}$  characters, including the active pipe character (`|`) of `Con $\text{\TeX}$ t`, in the input text. These macros will only be produced, when the `hybrid` option is `false`.

```

778 \def\markdownRendererLeftBrace{%
779 \markdownRendererLeftBracePrototype}%
780 \ExplSyntaxOn
781 \seq_put_right:Nn
782 \g_@@_renderers_seq
783 { leftBrace }
784 \prop_put:Nnn
785 \g_@@_renderer_arities_prop
786 { leftBrace }
787 { 0 }
788 \ExplSyntaxOff
789 \def\markdownRendererRightBrace{%

```

```

790 \markdownRendererRightBracePrototype}%
791 \ExplSyntaxOn
792 \seq_put_right:Nn
793 \g_@@_renderers_seq
794 { rightBrace }
795 \prop_put:Nnn
796 \g_@@_renderer_arities_prop
797 { rightBrace }
798 { 0 }
799 \ExplSyntaxOff
800 \def\markdownRendererDollarSign{%
801 \markdownRendererDollarSignPrototype}%
802 \ExplSyntaxOn
803 \seq_put_right:Nn
804 \g_@@_renderers_seq
805 { dollarSign }
806 \prop_put:Nnn
807 \g_@@_renderer_arities_prop
808 { dollarSign }
809 { 0 }
810 \ExplSyntaxOff
811 \def\markdownRendererPercentSign{%
812 \markdownRendererPercentSignPrototype}%
813 \ExplSyntaxOn
814 \seq_put_right:Nn
815 \g_@@_renderers_seq
816 { percentSign }
817 \prop_put:Nnn
818 \g_@@_renderer_arities_prop
819 { percentSign }
820 { 0 }
821 \ExplSyntaxOff
822 \def\markdownRendererAmpersand{%
823 \markdownRendererAmpersandPrototype}%
824 \ExplSyntaxOn
825 \seq_put_right:Nn
826 \g_@@_renderers_seq
827 { ampersand }
828 \prop_put:Nnn
829 \g_@@_renderer_arities_prop
830 { ampersand }
831 { 0 }
832 \ExplSyntaxOff
833 \def\markdownRendererUnderscore{%
834 \markdownRendererUnderscorePrototype}%
835 \ExplSyntaxOn
836 \seq_put_right:Nn

```

```

837 \g_@@_renderers_seq
838 { underscore }
839 \prop_put:Nnn
840 \g_@@_renderer_arities_prop
841 { underscore }
842 { 0 }
843 \ExplSyntaxOff
844 \def\markdownRendererHash{%
845 \markdownRendererHashPrototype}%
846 \ExplSyntaxOn
847 \seq_put_right:Nn
848 \g_@@_renderers_seq
849 { hash }
850 \prop_put:Nnn
851 \g_@@_renderer_arities_prop
852 { hash }
853 { 0 }
854 \ExplSyntaxOff
855 \def\markdownRendererCircumflex{%
856 \markdownRendererCircumflexPrototype}%
857 \ExplSyntaxOn
858 \seq_put_right:Nn
859 \g_@@_renderers_seq
860 { circumflex }
861 \prop_put:Nnn
862 \g_@@_renderer_arities_prop
863 { circumflex }
864 { 0 }
865 \ExplSyntaxOff
866 \def\markdownRendererBackslash{%
867 \markdownRendererBackslashPrototype}%
868 \ExplSyntaxOn
869 \seq_put_right:Nn
870 \g_@@_renderers_seq
871 { backslash }
872 \prop_put:Nnn
873 \g_@@_renderer_arities_prop
874 { backslash }
875 { 0 }
876 \ExplSyntaxOff
877 \def\markdownRendererTilde{%
878 \markdownRendererTildePrototype}%
879 \ExplSyntaxOn
880 \seq_put_right:Nn
881 \g_@@_renderers_seq
882 { tilde }
883 \prop_put:Nnn

```

```

884 \g_@@_renderer_arities_prop
885 { tilde }
886 { 0 }
887 \ExplSyntaxOff
888 \def\markdownRendererPipe{%
889 \markdownRendererPipePrototype}%
890 \ExplSyntaxOn
891 \seq_put_right:Nn
892 \g_@@_renderers_seq
893 { pipe }
894 \prop_put:Nnn
895 \g_@@_renderer_arities_prop
896 { pipe }
897 { 0 }
898 \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.

```

899 \def\markdownRendererCodeSpan{%
900 \markdownRendererCodeSpanPrototype}%
901 \ExplSyntaxOn
902 \seq_put_right:Nn
903 \g_@@_renderers_seq
904 { codeSpan }
905 \prop_put:Nnn
906 \g_@@_renderer_arities_prop
907 { codeSpan }
908 { 1 }
909 \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.

```

910 \def\markdownRendererLink{%
911 \markdownRendererLinkPrototype}%
912 \ExplSyntaxOn
913 \seq_put_right:Nn
914 \g_@@_renderers_seq
915 { link }
916 \prop_put:Nnn
917 \g_@@_renderer_arities_prop
918 { link }
919 { 4 }
920 \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.

```

921 \def\markdownRendererImage{%
922 \markdownRendererImagePrototype}%
923 \ExplSyntaxOn
924 \seq_put_right:Nn
925 \g_@@_renderers_seq
926 { image }
927 \prop_put:Nnn
928 \g_@@_renderer_arities_prop
929 { image }
930 { 4 }
931 \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.

```

932 \def\markdownRendererContentBlock{%
933 \markdownRendererContentBlockPrototype}%
934 \ExplSyntaxOn
935 \seq_put_right:Nn
936 \g_@@_renderers_seq
937 { contentBlock }
938 \prop_put:Nnn
939 \g_@@_renderer_arities_prop
940 { contentBlock }
941 { 4 }
942 \ExplSyntaxOff

```

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

```

943 \def\markdownRendererContentBlockOnlineImage{%
944 \markdownRendererContentBlockOnlineImagePrototype}%
945 \ExplSyntaxOn
946 \seq_put_right:Nn
947 \g_@@_renderers_seq
948 { contentBlockOnlineImage }
949 \prop_put:Nnn
950 \g_@@_renderer_arities_prop
951 { contentBlockOnlineImage }

```



```

952 { 4 }
953 \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<sup>7</sup> contains a record  $(k, v)$ , then a non-online-image content block with the filename extension  $s$ ,  $s:\text{lower}() = k$  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 a `markdown-languages.json` file inside your working directory or inside your local T<sub>E</sub>X 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.

```

954 \def\markdownRendererContentBlockCode{%
955 \markdownRendererContentBlockCodePrototype}%
956 \ExplSyntaxOn
957 \seq_put_right:Nn
958 \g_@@_renderers_seq
959 { contentBlockCode }
960 \prop_put:Nnn
961 \g_@@_renderer_arities_prop
962 { contentBlockCode }
963 { 5 }
964 \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.

```

965 \def\markdownRendererUlBegin{%
966 \markdownRendererUlBeginPrototype}%
967 \ExplSyntaxOn
968 \seq_put_right:Nn
969 \g_@@_renderers_seq
970 { ulBegin }
971 \prop_put:Nnn
972 \g_@@_renderer_arities_prop
973 { ulBegin }
974 { 0 }

```

---

<sup>7</sup>Local files take precedence. Filenames other than `markdown-languages.json` may be specified using the `contentBlocksLanguageMap` Lua option.

975 \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.

```
976 \def\markdownRendererUlBeginTight{%
977 \markdownRendererUlBeginTightPrototype}%
978 \ExplSyntaxOn
979 \seq_put_right:Nn
980 \g_@@_renderers_seq
981 { ulBeginTight }
982 \prop_put:Nnn
983 \g_@@_renderer_arities_prop
984 { ulBeginTight }
985 { 0 }
986 \ExplSyntaxOff
```

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

```
987 \def\markdownRendererUlItem{%
988 \markdownRendererUlItemPrototype}%
989 \ExplSyntaxOn
990 \seq_put_right:Nn
991 \g_@@_renderers_seq
992 { ulItem }
993 \prop_put:Nnn
994 \g_@@_renderer_arities_prop
995 { ulItem }
996 { 0 }
997 \ExplSyntaxOff
```

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

```
998 \def\markdownRendererUlItemEnd{%
999 \markdownRendererUlItemEndPrototype}%
1000 \ExplSyntaxOn
1001 \seq_put_right:Nn
1002 \g_@@_renderers_seq
1003 { ulItemEnd }
1004 \prop_put:Nnn
1005 \g_@@_renderer_arities_prop
1006 { ulItemEnd }
1007 { 0 }
1008 \ExplSyntaxOff
```

The `\markdownRendererUEnd` 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.

```

1009 \def\markdownRendererUEnd{%
1010 \markdownRendererUEndPrototype}%
1011 \ExplSyntaxOn
1012 \seq_put_right:Nn
1013 \g_@@_renderers_seq
1014 { ulEnd }
1015 \prop_put:Nnn
1016 \g_@@_renderer_arities_prop
1017 { ulEnd }
1018 { 0 }
1019 \ExplSyntaxOff

```

The `\markdownRendererUEndTight` 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.

```

1020 \def\markdownRendererUEndTight{%
1021 \markdownRendererUEndTightPrototype}%
1022 \ExplSyntaxOn
1023 \seq_put_right:Nn
1024 \g_@@_renderers_seq
1025 { ulEndTight }
1026 \prop_put:Nnn
1027 \g_@@_renderer_arities_prop
1028 { ulEndTight }
1029 { 0 }
1030 \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.

```

1031 \def\markdownRendererOlBegin{%
1032 \markdownRendererOlBeginPrototype}%
1033 \ExplSyntaxOn
1034 \seq_put_right:Nn
1035 \g_@@_renderers_seq
1036 { olBegin }
1037 \prop_put:Nnn
1038 \g_@@_renderer_arities_prop
1039 { olBegin }
1040 { 0 }
1041 \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.

```

1042 \def\markdownRendererOlBeginTight{%
1043 \markdownRendererOlBeginTightPrototype}%
1044 \ExplSyntaxOn
1045 \seq_put_right:Nn
1046 \g_@@_renderers_seq
1047 { olBeginTight }
1048 \prop_put:Nnn
1049 \g_@@_renderer_arities_prop
1050 { olBeginTight }
1051 { 0 }
1052 \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.

```

1053 \def\markdownRendererOlItem{%
1054 \markdownRendererOlItemPrototype}%
1055 \ExplSyntaxOn
1056 \seq_put_right:Nn
1057 \g_@@_renderers_seq
1058 { olItem }
1059 \prop_put:Nnn
1060 \g_@@_renderer_arities_prop
1061 { olItem }
1062 { 0 }
1063 \ExplSyntaxOff

```

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

```

1064 \def\markdownRendererOlItemEnd{%
1065 \markdownRendererOlItemEndPrototype}%
1066 \ExplSyntaxOn
1067 \seq_put_right:Nn
1068 \g_@@_renderers_seq
1069 { olItemEnd }
1070 \prop_put:Nnn
1071 \g_@@_renderer_arities_prop
1072 { olItemEnd }
1073 { 0 }
1074 \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.

```

1075 \def\markdownRendererOlItemWithNumber{%
1076 \markdownRendererOlItemWithNumberPrototype}%
1077 \ExplSyntaxOn
1078 \seq_put_right:Nn
1079 \g_@@_renderers_seq
1080 { olItemWithNumber }
1081 \prop_put:Nnn
1082 \g_@@_renderer_arities_prop
1083 { olItemWithNumber }
1084 { 1 }
1085 \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.

```

1086 \def\markdownRendererOlEnd{%
1087 \markdownRendererOlEndPrototype}%
1088 \ExplSyntaxOn
1089 \seq_put_right:Nn
1090 \g_@@_renderers_seq
1091 { olEnd }
1092 \prop_put:Nnn
1093 \g_@@_renderer_arities_prop
1094 { olEnd }
1095 { 0 }
1096 \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.

```

1097 \def\markdownRendererOlEndTight{%
1098 \markdownRendererOlEndTightPrototype}%
1099 \ExplSyntaxOn
1100 \seq_put_right:Nn
1101 \g_@@_renderers_seq
1102 { olEndTight }
1103 \prop_put:Nnn
1104 \g_@@_renderer_arities_prop
1105 { olEndTight }
1106 { 0 }
1107 \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.

```

1108 \def\markdownRendererDlBegin{%
1109 \markdownRendererDlBeginPrototype}%
1110 \ExplSyntaxOn
1111 \seq_put_right:Nn
1112 \g_@@_renderers_seq
1113 { dlBegin }
1114 \prop_put:Nnn
1115 \g_@@_renderer_arities_prop
1116 { dlBegin }
1117 { 0 }
1118 \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.

```

1119 \def\markdownRendererDlBeginTight{%
1120 \markdownRendererDlBeginTightPrototype}%
1121 \ExplSyntaxOn
1122 \seq_put_right:Nn
1123 \g_@@_renderers_seq
1124 { dlBeginTight }
1125 \prop_put:Nnn
1126 \g_@@_renderer_arities_prop
1127 { dlBeginTight }
1128 { 0 }
1129 \ExplSyntaxOff

```

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

```

1130 \def\markdownRendererDlItem{%
1131 \markdownRendererDlItemPrototype}%
1132 \ExplSyntaxOn
1133 \seq_put_right:Nn
1134 \g_@@_renderers_seq
1135 { dlItem }
1136 \prop_put:Nnn
1137 \g_@@_renderer_arities_prop
1138 { dlItem }
1139 { 1 }
1140 \ExplSyntaxOff

```

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

```

1141 \def\markdownRendererDlItemEnd{%
1142 \markdownRendererDlItemEndPrototype}%
1143 \ExplSyntaxOn
1144 \seq_put_right:Nn
1145 \g_@@_renderers_seq
1146 { dlItemEnd }
1147 \prop_put:Nnn
1148 \g_@@_renderer_arities_prop
1149 { dlItemEnd }
1150 { 0 }
1151 \ExplSyntaxOff

```

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

```

1152 \def\markdownRendererDlDefinitionBegin{%
1153 \markdownRendererDlDefinitionBeginPrototype}%
1154 \ExplSyntaxOn
1155 \seq_put_right:Nn
1156 \g_@@_renderers_seq
1157 { dlDefinitionBegin }
1158 \prop_put:Nnn
1159 \g_@@_renderer_arities_prop
1160 { dlDefinitionBegin }
1161 { 0 }
1162 \ExplSyntaxOff

```

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

```

1163 \def\markdownRendererDlDefinitionEnd{%
1164 \markdownRendererDlDefinitionEndPrototype}%
1165 \ExplSyntaxOn
1166 \seq_put_right:Nn
1167 \g_@@_renderers_seq
1168 { dlDefinitionEnd }
1169 \prop_put:Nnn
1170 \g_@@_renderer_arities_prop
1171 { dlDefinitionEnd }
1172 { 0 }
1173 \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.

```

1174 \def\markdownRendererDlEnd{%

```

```

1175 \markdownRendererDlEndPrototype}%
1176 \ExplSyntaxOn
1177 \seq_put_right:Nn
1178 \g_@@_renderers_seq
1179 { dlEnd }
1180 \prop_put:Nnn
1181 \g_@@_renderer_arities_prop
1182 { dlEnd }
1183 { 0 }
1184 \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.

```

1185 \def\markdownRendererDlEndTight{%
1186 \markdownRendererDlEndTightPrototype}%
1187 \ExplSyntaxOn
1188 \seq_put_right:Nn
1189 \g_@@_renderers_seq
1190 { dlEndTight }
1191 \prop_put:Nnn
1192 \g_@@_renderer_arities_prop
1193 { dlEndTight }
1194 { 0 }
1195 \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.

```

1196 \def\markdownRendererEmphasis{%
1197 \markdownRendererEmphasisPrototype}%
1198 \ExplSyntaxOn
1199 \seq_put_right:Nn
1200 \g_@@_renderers_seq
1201 { emphasis }
1202 \prop_put:Nnn
1203 \g_@@_renderer_arities_prop
1204 { emphasis }
1205 { 1 }
1206 \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.



```

1207 \def\markdownRendererStrongEmphasis{%
1208 \markdownRendererStrongEmphasisPrototype}%
1209 \ExplSyntaxOn
1210 \seq_put_right:Nn
1211 \g_@@_renderers_seq
1212 { strongEmphasis }
1213 \prop_put:Nnn
1214 \g_@@_renderer_arities_prop
1215 { strongEmphasis }
1216 { 1 }
1217 \ExplSyntaxOff

```

**2.2.3.16 Block Quote Renderers** The `\markdownRendererBlockQuoteBegin` macro represents the beginning of a block quote. The macro receives no arguments.

```

1218 \def\markdownRendererBlockQuoteBegin{%
1219 \markdownRendererBlockQuoteBeginPrototype}%
1220 \ExplSyntaxOn
1221 \seq_put_right:Nn
1222 \g_@@_renderers_seq
1223 { blockQuoteBegin }
1224 \prop_put:Nnn
1225 \g_@@_renderer_arities_prop
1226 { blockQuoteBegin }
1227 { 0 }
1228 \ExplSyntaxOff

```

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

```

1229 \def\markdownRendererBlockQuoteEnd{%
1230 \markdownRendererBlockQuoteEndPrototype}%
1231 \ExplSyntaxOn
1232 \seq_put_right:Nn
1233 \g_@@_renderers_seq
1234 { blockQuoteEnd }
1235 \prop_put:Nnn
1236 \g_@@_renderer_arities_prop
1237 { blockQuoteEnd }
1238 { 0 }
1239 \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 containing the code block contents.

```

1240 \def\markdownRendererInputVerbatim{%
1241 \markdownRendererInputVerbatimPrototype}%

```

```

1242 \ExplSyntaxOn
1243 \seq_put_right:Nn
1244 \g_@@_renderers_seq
1245 { inputVerbatim }
1246 \prop_put:Nnn
1247 \g_@@_renderer_arities_prop
1248 { inputVerbatim }
1249 { 1 }
1250 \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 containing the code block contents and to the code fence infostring.

```

1251 \def\markdownRendererInputFencedCode{%
1252 \markdownRendererInputFencedCodePrototype}%
1253 \ExplSyntaxOn
1254 \seq_put_right:Nn
1255 \g_@@_renderers_seq
1256 { inputFencedCode }
1257 \prop_put:Nnn
1258 \g_@@_renderer_arities_prop
1259 { inputFencedCode }
1260 { 2 }
1261 \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.

```

1262 \def\markdownRendererJekyllDataBegin{%
1263 \markdownRendererJekyllDataBeginPrototype}%
1264 \ExplSyntaxOn
1265 \seq_put_right:Nn
1266 \g_@@_renderers_seq
1267 { jekyllDataBegin }
1268 \prop_put:Nnn
1269 \g_@@_renderer_arities_prop
1270 { jekyllDataBegin }
1271 { 0 }
1272 \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.

```

1273 \def\markdownRendererJekyllDataEnd{%

```

```

1274 \markdownRendererJekyllDataEndPrototype}%
1275 \ExplSyntaxOn
1276 \seq_put_right:Nn
1277 \g_@@_renderers_seq
1278 { jekyllDataEnd }
1279 \prop_put:Nnn
1280 \g_@@_renderer_arities_prop
1281 { jekyllDataEnd }
1282 { 0 }
1283 \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.

```

1284 \def\markdownRendererJekyllDataMappingBegin{%
1285 \markdownRendererJekyllDataMappingBeginPrototype}%
1286 \ExplSyntaxOn
1287 \seq_put_right:Nn
1288 \g_@@_renderers_seq
1289 { jekyllDataMappingBegin }
1290 \prop_put:Nnn
1291 \g_@@_renderer_arities_prop
1292 { jekyllDataMappingBegin }
1293 { 2 }
1294 \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.

```

1295 \def\markdownRendererJekyllDataMappingEnd{%
1296 \markdownRendererJekyllDataMappingEndPrototype}%
1297 \ExplSyntaxOn
1298 \seq_put_right:Nn
1299 \g_@@_renderers_seq
1300 { jekyllDataMappingEnd }
1301 \prop_put:Nnn
1302 \g_@@_renderer_arities_prop
1303 { jekyllDataMappingEnd }
1304 { 0 }
1305 \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.

```

1306 \def\markdownRendererJekyllDataSequenceBegin{%
1307 \markdownRendererJekyllDataSequenceBeginPrototype}%
1308 \ExplSyntaxOn
1309 \seq_put_right:Nn
1310 \g_@@_renderers_seq
1311 { jekyllDataSequenceBegin }
1312 \prop_put:Nnn
1313 \g_@@_renderer_arities_prop
1314 { jekyllDataSequenceBegin }
1315 { 2 }
1316 \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.

```

1317 \def\markdownRendererJekyllDataSequenceEnd{%
1318 \markdownRendererJekyllDataSequenceEndPrototype}%
1319 \ExplSyntaxOn
1320 \seq_put_right:Nn
1321 \g_@@_renderers_seq
1322 { jekyllDataSequenceEnd }
1323 \prop_put:Nnn
1324 \g_@@_renderer_arities_prop
1325 { jekyllDataSequenceEnd }
1326 { 0 }
1327 \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.

```

1328 \def\markdownRendererJekyllDataBoolean{%
1329 \markdownRendererJekyllDataBooleanPrototype}%
1330 \ExplSyntaxOn
1331 \seq_put_right:Nn
1332 \g_@@_renderers_seq
1333 { jekyllDataBoolean }
1334 \prop_put:Nnn
1335 \g_@@_renderer_arities_prop
1336 { jekyllDataBoolean }
1337 { 2 }
1338 \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.

```

1339 \def\markdownRendererJekyllDataNumber{%
1340 \markdownRendererJekyllDataNumberPrototype}%
1341 \ExplSyntaxOn
1342 \seq_put_right:Nn
1343 \g_@@_renderers_seq
1344 { jekyllDataNumber }
1345 \prop_put:Nnn
1346 \g_@@_renderer_arities_prop
1347 { jekyllDataNumber }
1348 { 2 }
1349 \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.

```

1350 \def\markdownRendererJekyllDataString{%
1351 \markdownRendererJekyllDataStringPrototype}%
1352 \ExplSyntaxOn
1353 \seq_put_right:Nn
1354 \g_@@_renderers_seq
1355 { jekyllDataString }
1356 \prop_put:Nnn
1357 \g_@@_renderer_arities_prop
1358 { jekyllDataString }
1359 { 2 }
1360 \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.

```

1361 \def\markdownRendererJekyllDataEmpty{%
1362 \markdownRendererJekyllDataEmptyPrototype}%
1363 \ExplSyntaxOn
1364 \seq_put_right:Nn
1365 \g_@@_renderers_seq
1366 { jekyllDataEmpty }

```

```

1367 \prop_put:Nnn
1368 \g_@@_renderer_arities_prop
1369 { jekyllDataEmpty }
1370 { 1 }
1371 \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.

```

1372 \def\markdownRendererHeadingOne{%
1373 \markdownRendererHeadingOnePrototype}%
1374 \ExplSyntaxOn
1375 \seq_put_right:Nn
1376 \g_@@_renderers_seq
1377 { headingOne }
1378 \prop_put:Nnn
1379 \g_@@_renderer_arities_prop
1380 { headingOne }
1381 { 1 }
1382 \ExplSyntaxOff

```

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

```

1383 \def\markdownRendererHeadingTwo{%
1384 \markdownRendererHeadingTwoPrototype}%
1385 \ExplSyntaxOn
1386 \seq_put_right:Nn
1387 \g_@@_renderers_seq
1388 { headingTwo }
1389 \prop_put:Nnn
1390 \g_@@_renderer_arities_prop
1391 { headingTwo }
1392 { 1 }
1393 \ExplSyntaxOff

```

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

```

1394 \def\markdownRendererHeadingThree{%
1395 \markdownRendererHeadingThreePrototype}%
1396 \ExplSyntaxOn
1397 \seq_put_right:Nn
1398 \g_@@_renderers_seq
1399 { headingThree }
1400 \prop_put:Nnn
1401 \g_@@_renderer_arities_prop
1402 { headingThree }

```

```

1403 { 1 }
1404 \ExplSyntaxOff

```

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

```

1405 \def\markdownRendererHeadingFour{%
1406 \markdownRendererHeadingFourPrototype}%
1407 \ExplSyntaxOn
1408 \seq_put_right:Nn
1409 \g_@@_renderers_seq
1410 { headingFour }
1411 \prop_put:Nnn
1412 \g_@@_renderer_arities_prop
1413 { headingFour }
1414 { 1 }
1415 \ExplSyntaxOff

```

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

```

1416 \def\markdownRendererHeadingFive{%
1417 \markdownRendererHeadingFivePrototype}%
1418 \ExplSyntaxOn
1419 \seq_put_right:Nn
1420 \g_@@_renderers_seq
1421 { headingFive }
1422 \prop_put:Nnn
1423 \g_@@_renderer_arities_prop
1424 { headingFive }
1425 { 1 }
1426 \ExplSyntaxOff

```

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

```

1427 \def\markdownRendererHeadingSix{%
1428 \markdownRendererHeadingSixPrototype}%
1429 \ExplSyntaxOn
1430 \seq_put_right:Nn
1431 \g_@@_renderers_seq
1432 { headingSix }
1433 \prop_put:Nnn
1434 \g_@@_renderer_arities_prop
1435 { headingSix }
1436 { 1 }
1437 \ExplSyntaxOff

```

**2.2.3.20 Horizontal Rule Renderer** The `\markdownRendererHorizontalRule` macro represents a horizontal rule. The macro receives no arguments.

```

1438 \def\markdownRendererHorizontalRule{%
1439 \markdownRendererHorizontalRulePrototype}%
1440 \ExplSyntaxOn
1441 \seq_put_right:Nn
1442 \g_@@_renderers_seq
1443 { horizontalRule }
1444 \prop_put:Nnn
1445 \g_@@_renderer_arities_prop
1446 { horizontalRule }
1447 { 0 }
1448 \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.

```

1449 \def\markdownRendererFootnote{%
1450 \markdownRendererFootnotePrototype}%
1451 \ExplSyntaxOn
1452 \seq_put_right:Nn
1453 \g_@@_renderers_seq
1454 { footnote }
1455 \prop_put:Nnn
1456 \g_@@_renderer_arities_prop
1457 { footnote }
1458 { 1 }
1459 \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.

```

1460 \def\markdownRendererCite{%
1461 \markdownRendererCitePrototype}%
1462 \ExplSyntaxOn
1463 \seq_put_right:Nn
1464 \g_@@_renderers_seq
1465 { cite }
1466 \prop_put:Nnn
1467 \g_@@_renderer_arities_prop

```



```

1468 { cite }
1469 { 1 }
1470 \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.

```

1471 \def\markdownRendererTextCite{%
1472 \markdownRendererTextCitePrototype}%
1473 \ExplSyntaxOn
1474 \seq_put_right:Nn
1475 \g_@@_renderers_seq
1476 { textCite }
1477 \prop_put:Nnn
1478 \g_@@_renderer_arities_prop
1479 { textCite }
1480 { 1 }
1481 \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.

```

1482 \def\markdownRendererTable{%
1483 \markdownRendererTablePrototype}%
1484 \ExplSyntaxOn
1485 \seq_put_right:Nn
1486 \g_@@_renderers_seq
1487 { table }
1488 \prop_put:Nnn
1489 \g_@@_renderer_arities_prop
1490 { table }
1491 { 3 }
1492 \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 `\markdownRendererBlockHtmlCommentEnd` macros represent the beginning and the end of a block HTML comment. The macros receive no arguments.

```

1493 \def\markdownRendererInlineHtmlComment{%
1494 \markdownRendererInlineHtmlCommentPrototype}%
1495 \ExplSyntaxOn
1496 \seq_put_right:Nn
1497 \g_@@_renderers_seq
1498 { inlineHtmlComment }
1499 \prop_put:Nnn
1500 \g_@@_renderer_arities_prop
1501 { inlineHtmlComment }
1502 { 1 }
1503 \ExplSyntaxOff
1504 \def\markdownRendererBlockHtmlCommentBegin{%
1505 \markdownRendererBlockHtmlCommentBeginPrototype}%
1506 \ExplSyntaxOn
1507 \seq_put_right:Nn
1508 \g_@@_renderers_seq
1509 { blockHtmlCommentBegin }
1510 \prop_put:Nnn
1511 \g_@@_renderer_arities_prop
1512 { blockHtmlCommentBegin }
1513 { 0 }
1514 \ExplSyntaxOff
1515 \def\markdownRendererBlockHtmlCommentEnd{%
1516 \markdownRendererBlockHtmlCommentEndPrototype}%
1517 \ExplSyntaxOn
1518 \seq_put_right:Nn
1519 \g_@@_renderers_seq
1520 { blockHtmlCommentEnd }
1521 \prop_put:Nnn
1522 \g_@@_renderer_arities_prop
1523 { blockHtmlCommentEnd }
1524 { 0 }
1525 \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.

```

1526 \def\markdownRendererInlineHtmlTag{%
1527 \markdownRendererInlineHtmlTagPrototype}%
1528 \ExplSyntaxOn
1529 \seq_put_right:Nn
1530 \g_@@_renderers_seq
1531 { inlineHtmlTag }
1532 \prop_put:Nnn
1533 \g_@@_renderer_arities_prop
1534 { inlineHtmlTag }
1535 { 1 }
1536 \ExplSyntaxOff
1537 \def\markdownRendererInputBlockHtmlElement{%
1538 \markdownRendererInputBlockHtmlElementPrototype}%
1539 \ExplSyntaxOn
1540 \seq_put_right:Nn
1541 \g_@@_renderers_seq
1542 { inputBlockHtmlElement }
1543 \prop_put:Nnn
1544 \g_@@_renderer_arities_prop
1545 { inputBlockHtmlElement }
1546 { 1 }
1547 \ExplSyntaxOff

```

**2.2.3.27 Attribute Renderers** The following macros are only produced, when the `headerAttributes` option is enabled.

`\markdownRendererAttributeIdentifier` represents the  $\langle identifier \rangle$  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  $\langle identifier \rangle$ .

`\markdownRendererAttributeClassName` represents the  $\langle class name \rangle$  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  $\langle class name \rangle$ .

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

```

1548 \def\markdownRendererAttributeIdentifier{%
1549 \markdownRendererAttributeIdentifierPrototype}%
1550 \ExplSyntaxOn
1551 \seq_put_right:Nn

```

```

1552 \g_@@_renderers_seq
1553 { attributeIdentifier }
1554 \prop_put:Nnn
1555 \g_@@_renderer_arities_prop
1556 { attributeIdentifier }
1557 { 1 }
1558 \ExplSyntaxOff
1559 \def\markdownRendererAttributeClassName{%
1560 \markdownRendererAttributeClassNamePrototype}%
1561 \ExplSyntaxOn
1562 \seq_put_right:Nn
1563 \g_@@_renderers_seq
1564 { attributeClassName }
1565 \prop_put:Nnn
1566 \g_@@_renderer_arities_prop
1567 { attributeClassName }
1568 { 1 }
1569 \ExplSyntaxOff
1570 \def\markdownRendererAttributeKeyValue{%
1571 \markdownRendererAttributeKeyValuePrototype}%
1572 \ExplSyntaxOn
1573 \seq_put_right:Nn
1574 \g_@@_renderers_seq
1575 { attributeKeyValue }
1576 \prop_put:Nnn
1577 \g_@@_renderer_arities_prop
1578 { attributeKeyValue }
1579 { 2 }
1580 \ExplSyntaxOff

```

**2.2.3.28 Header Attribute Context Renderers** The following macros are only produced, when the `headerAttributes` option is enabled.

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

```

1581 \def\markdownRendererHeaderAttributeContextBegin{%
1582 \markdownRendererHeaderAttributeContextBeginPrototype}%
1583 \ExplSyntaxOn
1584 \seq_put_right:Nn
1585 \g_@@_renderers_seq
1586 { headerAttributeContextBegin }
1587 \prop_put:Nnn
1588 \g_@@_renderer_arities_prop
1589 { headerAttributeContextBegin }
1590 { 0 }
1591 \ExplSyntaxOff

```

```

1592 \def\markdownRendererHeaderAttributeContextEnd{%
1593 \markdownRendererHeaderAttributeContextEndPrototype}%
1594 \ExplSyntaxOn
1595 \seq_put_right:Nn
1596 \g_@@_renderers_seq
1597 { headerAttributeContextEnd }
1598 \prop_put:Nnn
1599 \g_@@_renderer_arities_prop
1600 { headerAttributeContextEnd }
1601 { 0 }
1602 \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  $\text{\LaTeX}$  kernel.

```

1603 \ExplSyntaxOn
1604 \keys_define:nn
1605 { markdown/jekyllData }
1606 { }
1607 \ExplSyntaxOff

```

The following  $\text{\TeX}$  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  $\text{\LaTeX}$  and  $\text{ConTeXt}$  implementations (see sections 3.3 and 3.4).

```

1608 \ExplSyntaxOn
1609 \cs_new:Nn \@@_plaintex_define_renderer_prototypes:
1610 {
1611 \seq_map_function:NN
1612 \g_@@_renderers_seq
1613 \@@_plaintex_define_renderer_prototype:n
1614 \let\markdownRendererBlockHtmlCommentBeginPrototype=\iffalse
1615 \let\markdownRendererBlockHtmlCommentBegin=\iffalse
1616 \let\markdownRendererBlockHtmlCommentEndPrototype=\fi
1617 \let\markdownRendererBlockHtmlCommentEnd=\fi
1618 }
1619 \cs_new:Nn \@@_plaintex_define_renderer_prototype:n
1620 {
1621 \@@_renderer_prototype_tl_to_csname:nN
1622 { #1 }
1623 \l_tmpa_tl
1624 \prop_get:NnN
1625 \g_@@_renderer_arities_prop

```

```

1626 { #1 }
1627 \l_tmpb_tl
1628 \@@_plaintex_define_renderer_prototype:cV
1629 { \l_tmpa_tl }
1630 \l_tmpb_tl
1631 }
1632 \cs_new:Nn \@@_renderer_prototype_tl_to_csname:nN
1633 {
1634 \tl_set:Nn
1635 \l_tmpa_tl
1636 % TODO: Replace with \str_uppercase:n in TeX Live 2020.
1637 { \str_upper_case:n { #1 } }
1638 \tl_set:Nx
1639 #2
1640 {
1641 markdownRenderer
1642 \tl_head:f { \l_tmpa_tl }
1643 \tl_tail:n { #1 }
1644 Prototype
1645 }
1646 }
1647 \cs_new:Nn \@@_plaintex_define_renderer_prototype:Nn
1648 {
1649 \cs_generate_from_arg_count:NNnn
1650 #1
1651 \cs_set:Npn
1652 { #2 }
1653 { }
1654 }
1655 \cs_generate_variant:Nn
1656 \@@_plaintex_define_renderer_prototype:Nn
1657 { cV }
1658 \@@_plaintex_define_renderer_prototypes:
1659 \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 T<sub>E</sub>X 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*.

```
1660 \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.

```
1661 \let\markdownReadAndConvert\relax
```

```
1662 \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*.

```
1663 \catcode`\|=0\catcode`\=12%
1664 |gdef\markdownBegin{%
1665 \markdownReadAndConvert{\markdownEnd}%
1666 {\markdownEnd}}%
1667 |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.

```
1668 \ExplSyntaxOn
1669 \cs_if_exist:NF
1670 \markdownMode
1671 {
```

```

1672 \file_if_exist:nTF
1673 { lt3luabridge.tex }
1674 {
1675 \cs_new:Npn
1676 \markdownMode
1677 { 3 }
1678 }
1679 {
1680 \cs_if_exist:NTF
1681 \directlua
1682 {
1683 \cs_new:Npn
1684 \markdownMode
1685 { 2 }
1686 }
1687 {
1688 \cs_new:Npn
1689 \markdownMode
1690 { 0 }
1691 }
1692 }
1693 }
1694 \int_compare:nF
1695 { \markdownMode = 3 }
1696 {
1697 \int_new:N
1698 \g_luabridge_method_int
1699 \int_gset:Nn
1700 \g_luabridge_method_int
1701 { \markdownMode }
1702 }
1703 \ExplSyntaxOff

```

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

```

1704 \def\markdownLuaRegisterIBCallback#1{\relax}%
1705 \def\markdownLuaUnregisterIBCallback#1{\relax}%

```

## 2.3 L<sup>A</sup>T<sub>E</sub>X Interface

The L<sup>A</sup>T<sub>E</sub>X interface provides L<sup>A</sup>T<sub>E</sub>X environments for the typesetting of markdown input from within L<sup>A</sup>T<sub>E</sub>X, facilities for setting Lua interface options (see Section 2.1.2) used during the conversion from markdown to plain T<sub>E</sub>X, and facilities for changing the way markdown tokens are rendered. The rest of the interface is inherited from the plain T<sub>E</sub>X interface (see Section 2.2).



The L<sup>A</sup>T<sub>E</sub>X implementation redefines the plain T<sub>E</sub>X logging macros (see Section 3.2.1) to use the L<sup>A</sup>T<sub>E</sub>X `\PackageInfo`, `\PackageWarning`, and `\PackageError` macros.

```
1706 \newcommand\markdownInfo[1]{\PackageInfo{markdown}{#1}}%
1707 \newcommand\markdownWarning[1]{\PackageWarning{markdown}{#1}}%
1708 \newcommand\markdownError[2]{\PackageError{markdown}{#1}{#2.}}%
1709 \input markdown/markdown
```

The L<sup>A</sup>T<sub>E</sub>X interface is implemented by the `markdown.sty` file, which can be loaded from the L<sup>A</sup>T<sub>E</sub>X document preamble as follows:

```
\usepackage[<options>]{markdown}
```

where *<options>* are the L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> parses package options.

### 2.3.1 Typesetting Markdown

The interface exposes the `markdown` and `markdown*` L<sup>A</sup>T<sub>E</sub>X environments, and redefines the `\markdownInput` command.

The `markdown` and `markdown*` L<sup>A</sup>T<sub>E</sub>X environments are used to typeset markdown document fragments. The starred version of the `markdown` environment accepts L<sup>A</sup>T<sub>E</sub>X interface options (see Section 2.3.2) as its only argument. These options will only influence this markdown document fragment.

```
1710 \newenvironment{markdown}\relax\relax
1711 \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` L<sup>A</sup>T<sub>E</sub>X environment (and likewise for the starred version).

Note that the `markdown` and `markdown*` L<sup>A</sup>T<sub>E</sub>X environments are subject to the same limitations as the `\markdownBegin` and `\markdownEnd` macros exposed by the plain T<sub>E</sub>X interface.

The following example L<sup>A</sup>T<sub>E</sub>X code showcases the usage of the `markdown` and `markdown*` environments:

<pre>\documentclass{article} \usepackage{markdown} \begin{document} % ... \begin{markdown} _Hello_ **world** ... \end{markdown}</pre>	<pre>\documentclass{article} \usepackage{markdown} \begin{document} % ... \begin{markdown*}{smartEllipses} _Hello_ **world** ... \end{markdown*}</pre>
---------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------

<code>% ... \end{document}</code>	<code>% ... \end{document}</code>
---------------------------------------	---------------------------------------

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  $\text{\TeX}$ . Unlike the `\markdownInput` macro provided by the plain  $\text{\TeX}$  interface, this macro also accepts  $\text{\LaTeX}$  interface options (see Section 2.3.2) as its optional argument. These options will only influence this markdown document.

The following example  $\text{\LaTeX}$  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  $\text{\LaTeX}$  options are represented by a comma-delimited list of  $\langle key \rangle = \langle value \rangle$  pairs. For boolean options, the  $= \langle value \rangle$  part is optional, and  $\langle key \rangle$  will be interpreted as  $\langle key \rangle = \text{true}$  if the  $= \langle value \rangle$  part has been omitted.

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

The  $\text{\LaTeX}$  options may be specified when loading the  $\text{\LaTeX}$  package, when using the `markdown*`  $\text{\LaTeX}$  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:

```
1712 \newcommand\markdownSetup[1]{%
1713 \setkeys{markdownOptions}{#1}}%
```

We may also store  $\text{\LaTeX}$  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:

```
1714 \newcommand\markdownSetupSnippet[2]{%
1715 \markdownIfSnippetExists{#1}%
1716 {%
1717 \markdownWarning
1718 {Redefined setup snippet \markdownLaTeXThemeName#1}}%
```

```

1719 \csname markdownLaTeXSetupSnippet%
1720 \markdownLaTeXThemeName#1\endcsname={#2}%
1721 }{%
1722 \newtoks\next
1723 \next={#2}%
1724 \expandafter\let\csname markdownLaTeXSetupSnippet%
1725 \markdownLaTeXThemeName#1\endcsname=\next
1726 }}%

```

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

```

1727 \newcommand\markdownIfSnippetExists[3]{%
1728 \@ifundefined
1729 {markdownLaTeXSetupSnippet\markdownLaTeXThemeName#1}%
1730 {#3}{#2}}%

```

See Section 2.3.2.2 for information on interactions between setup snippets and L<sup>A</sup>T<sub>E</sub>X themes. See Section 2.3.2.3 for information about invoking the stored setup snippets.

To enable the enumeration of L<sup>A</sup>T<sub>E</sub>X options, we will maintain the `\g_@@_latex_options_seq` sequence.

```

1731 \ExplSyntaxOn
1732 \seq_new:N \g_@@_latex_options_seq

```

To enable the reflection of default L<sup>A</sup>T<sub>E</sub>X options and their types, we will maintain the `\g_@@_default_latex_options_prop` and `\g_@@_latex_option_types_prop` property lists, respectively.

```

1733 \prop_new:N \g_@@_latex_option_types_prop
1734 \prop_new:N \g_@@_default_latex_options_prop
1735 \tl_const:Nn \c_@@_option_layer_latex_tl { latex }
1736 \seq_put_right:NV \g_@@_option_layers_seq \c_@@_option_layer_latex_tl
1737 \cs_new:Nn
1738 \@@_add_latex_option:nnn
1739 {
1740 \@@_add_option:Vnnn
1741 \c_@@_option_layer_latex_tl
1742 { #1 }
1743 { #2 }
1744 { #3 }
1745 }

```

**2.3.2.1 No default token renderer prototypes** Default token renderer prototypes require L<sup>A</sup>T<sub>E</sub>X 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 T<sub>E</sub>X implementation (see Section 3.2.2) and prevent the soft L<sup>A</sup>T<sub>E</sub>X 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}
```

```
1746 \@@_add_latex_option:nnn
1747 { plain }
1748 { boolean }
1749 { false }
1750 \ExplSyntaxOff
```

**2.3.2.2 L<sup>A</sup>T<sub>E</sub>X themes** User-contributed L<sup>A</sup>T<sub>E</sub>X themes for the Markdown package provide a domain-specific interpretation of some Markdown tokens. Similarly to L<sup>A</sup>T<sub>E</sub>X packages, themes allow the authors to achieve a specific look and other high-level goals without low-level programming.

The L<sup>A</sup>T<sub>E</sub>X option with key `theme` loads a L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X document class or for a single L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X package named `markdownthemewitiko_beamer_MU.sty`.

If the L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X 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` L<sup>A</sup>T<sub>E</sub>X package, and finally the `markdownthemewitiko_dot.sty` L<sup>A</sup>T<sub>E</sub>X package:

```

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

```

```

1751 \newif\ifmarkdownLaTeXLoaded
1752 \markdownLaTeXLoadedfalse
1753 \AtEndOfPackage{\markdownLaTeXLoadedtrue}
1754 \define@key{markdownOptions}{theme}{%
1755 \IfSubStr{#1}{/}{%}{%
1756 \markdownError
1757 {Won't load theme with unqualified name #1}%
1758 {Theme names must contain at least one forward slash}}%
1759 \StrSubstitute{#1}{/}{_}{\markdownLaTeXThemePackageName}%
1760 \edef\markdownLaTeXThemePackageName{%
1761 markdowntheme\markdownLaTeXThemePackageName}%
1762 \expandafter\markdownLaTeXThemeLoad\expandafter{%
1763 \markdownLaTeXThemePackageName}{#1/}%

```

The  $\text{\LaTeX}$  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  $\langle\textit{theme name}\rangle/$  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  $\text{\LaTeX}$ , themes may not be loaded after the beginning of a  $\text{\LaTeX}$  document.

```

1764 \@onlypreamble\KV@markdownOptions@theme

```

Example themes provided with the Markdown package include:

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

```

\documentclass{article}
\usepackage[theme=witiko/dot]{markdown}
\setkeys{Gin}{
 width = \columnwidth,
 height = 0.65\paperheight,
 keepaspectratio}
\begin{document}
\begin{markdown}
``` dot Various formats of mathematical formulae
digraph tree {

```

```

margin = 0;
rankdir = "LR";

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

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

```

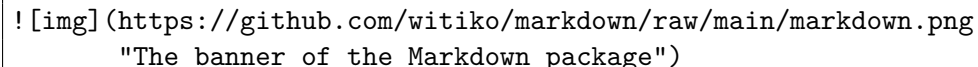
Typesetting the above document produces the output shown in Figure 4.

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

1765 \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}


```

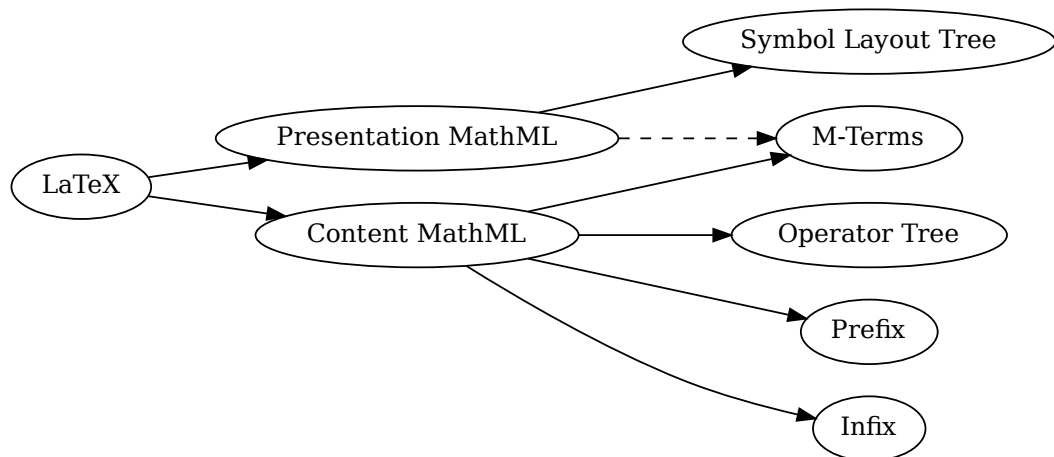


Figure 4: Various formats of mathematical formulae

```

\end{markdown}
\end{document}

```

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

1766 `\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”.

1767 `\ProvidesPackage{markdownthemewitiko_tilde}[2021/03/22]%`

```

\documentclass{book}
\usepackage{markdown}
\markdownSetup{pipeTables,tableCaptions}
\begin{document}
\begin{markdown}
Introduction
=====
## Section
### Subsection
Hello *Markdown*!

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

: Table
\end{markdown}
\end{document}

```



Chapter 1

Introduction

1.1 Section

1.1.1 Subsection

Hello *Markdown*!

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

Table 1.1: Table

Figure 5: The banner of the Markdown package

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 $\langle value \rangle$:

```

1768 \define@key{markdownOptions}{snippet}{%
1769   \markdownIfSnippetExists{#1}%
1770   {%
1771     \expandafter\markdownSetup\expandafter{%
1772       \the\cname markdownLaTeXSetupSnippet%
1773       \markdownLaTeXThemeName#1\endcsname}%
1774   }{%
1775     \markdownError
1776     {Can't invoke setup snippet #1}%
1777     {The setup snippet is undefined}%
1778   }%
1779 }%

```

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 T_EX Interface Options Here, we automatically define plain T_EX macros and the $\langle key \rangle = \langle value \rangle$ interface for the above L^AT_EX options.

```

1780 \ExplSyntaxOn
1781 \cs_new:Nn \@@_latex_define_option_commands_and_keyvals:
1782 {
1783   \seq_map_inline:Nn
1784     \g_@@_latex_options_seq
1785     {
1786       \@@_plain_tex_define_option_command:n
1787         { ##1 }
1788     }

```

Furthermore, we also define the $\langle key \rangle = \langle value \rangle$ interface for all option macros recognized by the Lua plain T_EX interfaces.

```

1789   \seq_map_inline:Nn
1790     \g_@@_option_layers_seq
1791     {
1792       \seq_map_inline:cn
1793         { g_@@_ ##1 _options_seq }
1794         {
1795           \@@_latex_define_option_keyval:nn
1796             { ##1 }
1797             { #####1 }
1798         }
1799     }

```

```

1800 }
1801 \cs_new:Nn \@@_latex_define_option_keyval:nn
1802 {
1803   \prop_get:cnN
1804     { g_@@_ #1 _option_types_prop }
1805     { #2 }
1806     \l_tmpa_tl
1807   \str_if_eq:VVTF
1808     \l_tmpa_tl
1809     \c_@@_option_type_boolean_tl
1810     {
1811       \define@key
1812         { markdownOptions }
1813         { #2 }
1814         [ true ]
1815         {
1816           \@@_set_option_value:nn
1817             { #2 }
1818             { ##1 }
1819         }
1820     }
1821   {
1822     \define@key
1823       { markdownOptions }
1824       { #2 }
1825       {
1826         \@@_set_option_value:nn
1827           { #2 }
1828           { ##1 }
1829       }
1830   }
1831 }
1832 \@@_latex_define_option_commands_and_keyvals:
1833 \ExplSyntaxOff

```

The `\markdownOptionFinalizeCache` and `\markdownOptionFrozenCache` plain TeX options are exposed through L^AT_EX options with keys `finalizeCache` and `frozenCache`.

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

```

\documentclass[frozenscache]{article}
\usepackage{markdown,minted}

```

```
\begin{document}
\end{document}
```

We hope that other packages will support the `finalizcache` and `frozenscache` package options in the future, so that they can become a standard interface for preparing L^AT_EX document sources for distribution.

```
1834 \DeclareOption{finalizcache}{\markdownSetup{finalizeCache}}
1835 \DeclareOption{frozenscache}{\markdownSetup{frozenCache}}
```

The following example L^AT_EX code showcases a possible configuration of plain T_EX interface options `\markdownOptionHybrid`, `\markdownOptionSmartEllipses`, and `\markdownOptionCacheDir`.

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

2.3.2.5 Plain T_EX Markdown Token Renderers The L^AT_EX 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 T_EX interface (see Section 2.2.3).

```
1836 \ExplSyntaxOn
1837 \cs_new:Nn \@@_latex_define_renderers:
1838 {
1839   \seq_map_function:NN
1840     \g_@@_renderers_seq
1841     \@@_latex_define_renderer:n
1842 }
1843 \cs_new:Nn \@@_latex_define_renderer:n
1844 {
1845   \@@_renderer_tl_to_csname:nN
1846     { #1 }
1847     \l_tmpa_tl
1848   \prop_get:NnN
1849     \g_@@_renderer_arities_prop
1850     { #1 }
1851     \l_tmpb_tl
1852   \@@_latex_define_renderer:ncV
1853     { #1 }
1854     { \l_tmpa_tl }
1855     \l_tmpb_tl
1856 }
```

```

1857 \cs_new:Nn \@@_renderer_tl_to_csname:nN
1858 {
1859   \tl_set:Nn
1860     \l_tmpa_tl
1861     % TODO: Replace with \str_uppercase:n in TeX Live 2020.
1862     { \str_upper_case:n { #1 } }
1863   \tl_set:Nx
1864     #2
1865     {
1866       markdownRenderer
1867       \tl_head:f { \l_tmpa_tl }
1868       \tl_tail:n { #1 }
1869     }
1870 }
1871 \cs_new:Nn \@@_latex_define_renderer:nNn
1872 {
1873   \define@key
1874     { markdownRenderers }
1875     { #1 }
1876     {
1877       \cs_generate_from_arg_count:NNnn
1878         #2
1879         \cs_set:Npn
1880           { #3 }
1881           { ##1 }
1882     }
1883 }
1884 \cs_generate_variant:Nn
1885   \@@_latex_define_renderer:nNn
1886   { ncV }
1887 \ExplSyntaxOff

```

The following example L^AT_EX 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 T_EX Markdown Token Renderer Prototypes The L^AT_EX 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 T_EX interface (see Section 2.2.4).

```

1888 \ExplSyntaxOn
1889 \cs_new:Nn \@@_latex_define_renderer_prototypes:
1890 {
1891   \seq_map_function:NN
1892     \g_@@_renderers_seq
1893     \@@_latex_define_renderer_prototype:n
1894 }
1895 \cs_new:Nn \@@_latex_define_renderer_prototype:n
1896 {
1897   \@@_renderer_prototype_tl_to_csname:nN
1898     { #1 }
1899     \l_tmpa_tl
1900   \prop_get:NnN
1901     \g_@@_renderer_arities_prop
1902     { #1 }
1903     \l_tmpb_tl
1904   \@@_latex_define_renderer_prototype:ncV
1905     { #1 }
1906     { \l_tmpa_tl }
1907     \l_tmpb_tl
1908 }
1909 \cs_new:Nn \@@_latex_define_renderer_prototype:nNn
1910 {
1911   \define@key
1912     { markdownRendererPrototypes }
1913     { #1 }
1914     {
1915       \cs_generate_from_arg_count:NNnn
1916         #2
1917         \cs_set:Npn
1918           { #3 }
1919           { ##1 }
1920     }
1921 }
1922 \cs_generate_variant:Nn
1923   \@@_latex_define_renderer_prototype:nNn
1924   { ncV }
1925 \ExplSyntaxOff

```

The following example L^AT_EX 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 ConT_EXt Interface

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

```

1926 \writestatus{loading}{ConTeXt User Module / markdown}%
1927 \startmodule[markdown]
1928 \unprotect

```

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

```

\usemodule[t][markdown]

```

It is expected that the special plain T_EX characters have the expected category codes, when `\inputting` the file.

2.4.1 Typesetting Markdown

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

```

1929 \let\startmarkdown\relax
1930 \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 T_EX interface.

The following example ConT_EXt 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 T_EX *token renderers* is performed by the Lua layer. The plain T_EX layer provides default definitions for the token renderers. The L^AT_EX and ConT_EXt layers correct idiosyncrasies of the respective T_EX 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 T_EX, 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 T_EX writer were hidden behind the converter functions exposed by the Lua interface (see Section 2.1).

```
1931 local upper, gsub, format, length =
1932   string.upper, string.gsub, string.format, string.len
1933 local concat = table.concat
1934 local P, R, S, V, C, Cg, Cb, Cmt, Cc, Ct, B, Cs, any =
1935   lpeg.P, lpeg.R, lpeg.S, lpeg.V, lpeg.C, lpeg.Cg, lpeg.Cb,
1936   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 T_EX 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.

```
1937 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.

```
1938 function util.err(msg, exit_code)
1939   io.stderr:write("markdown.lua: " .. msg .. "\n")
1940   os.exit(exit_code or 1)
1941 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.

```
1942 function util.cache(dir, string, salt, transform, suffix)
1943   local digest = md5.sumhexa(string .. (salt or ""))
1944   local name = util.pathname(dir, digest .. suffix)
1945   local file = io.open(name, "r")
1946   if file == nil then -- If no cache entry exists, then create a new one.
1947     local file = assert(io.open(name, "w"),
1948       [[could not open file ]] .. name .. [[ for writing]])
1949     local result = string
1950     if transform ~= nil then
1951       result = transform(result)
1952     end
1953     assert(file:write(result))
1954     assert(file:close())
1955   end
1956   return name
1957 end
```

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

```
1958 function util.table_copy(t)
1959   local u = { }
1960   for k, v in pairs(t) do u[k] = v end
1961   return setmetatable(u, getmetatable(t))
1962 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 Ierusalimsky [9, Chapter 21].

```
1963 function util.expand_tabs_in_line(s, tabstop)
1964   local tab = tabstop or 4
1965   local corr = 0
1966   return (s:gsub("\t", function(p)
1967     local sp = tab - (p - 1 + corr) % tab
1968     corr = corr - 1 + sp
1969     return string.rep(" ", sp)
1970   end))
1971 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.

```
1972 function util.walk(t, f)
1973   local typ = type(t)
1974   if typ == "string" then
```



```

1975     f(t)
1976 elseif typ == "table" then
1977     local i = 1
1978     local n
1979     n = t[i]
1980     while n do
1981         util.walk(n, f)
1982         i = i + 1
1983         n = t[i]
1984     end
1985 elseif typ == "function" then
1986     local ok, val = pcall(t)
1987     if ok then
1988         util.walk(val, f)
1989     end
1990 else
1991     f(tostring(t))
1992 end
1993 end

```

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

```

1994 function util.flatten(ary)
1995     local new = {}
1996     for _,v in ipairs(ary) do
1997         if type(v) == "table" then
1998             for _,w in ipairs(util.flatten(v)) do
1999                 new[#new + 1] = w
2000             end
2001         else
2002             new[#new + 1] = v
2003         end
2004     end
2005     return new
2006 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.

```

2007 function util.rope_to_string(rope)
2008     local buffer = {}
2009     util.walk(rope, function(x) buffer[#buffer + 1] = x end)
2010     return table.concat(buffer)
2011 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.

```

2012 function util.rope_last(rope)
2013     if #rope == 0 then

```

```

2014     return nil
2015 else
2016     local l = rope[#rope]
2017     if type(l) == "table" then
2018         return util.rope_last(l)
2019     else
2020         return l
2021     end
2022 end
2023 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 $1 \leq i \leq \#ary$.

```

2024 function util.intersperse(ary, x)
2025     local new = {}
2026     local l = #ary
2027     for i,v in ipairs(ary) do
2028         local n = #new
2029         new[n + 1] = v
2030         if i ~= l then
2031             new[n + 2] = x
2032         end
2033     end
2034     return new
2035 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 $1 \leq i \leq \#ary$.

```

2036 function util.map(ary, f)
2037     local new = {}
2038     for i,v in ipairs(ary) do
2039         new[i] = f(v)
2040     end
2041     return new
2042 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 occurrences of escapable strings and characters (in this order).

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

```

2043 function util.escaper(char_escapes, string_escapes)

```

Build a string of escapable characters.

```

2044     local char_escapes_list = ""
2045     for i,_ in pairs(char_escapes) do
2046         char_escapes_list = char_escapes_list .. i

```

2047 end

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

2048 local escapable = S(char_escapes_list) / char_escapes

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

$$\sum_{(k,v) \in \text{string_escapes}} P(k) / v + \text{escapable}$$

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

```
2049 if string_escapes then
2050   for k,v in pairs(string_escapes) do
2051     escapable = P(k) / v + escapable
2052   end
2053 end
```

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

2054 local escape_string = Cs((escapable + any)^0)

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

```
2055 return function(s)
2056   return lpeg.match(escape_string, s)
2057 end
2058 end
```

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

```
2059 function util.pathname(dir, file)
2060   if #dir == 0 then
2061     return file
2062   else
2063     return dir .. "/" .. file
2064   end
2065 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.

2066 local entities = {}

```

2067
2068 local character_entities = {
2069     ["Tab"] = 9,
2070     ["NewLine"] = 10,
2071     ["excl"] = 33,
2072     ["quot"] = 34,
2073     ["QUOT"] = 34,
2074     ["num"] = 35,
2075     ["dollar"] = 36,
2076     ["percent"] = 37,
2077     ["amp"] = 38,
2078     ["AMP"] = 38,
2079     ["apos"] = 39,
2080     ["lpar"] = 40,
2081     ["rpar"] = 41,
2082     ["ast"] = 42,
2083     ["midast"] = 42,
2084     ["plus"] = 43,
2085     ["comma"] = 44,
2086     ["period"] = 46,
2087     ["sol"] = 47,
2088     ["colon"] = 58,
2089     ["semi"] = 59,
2090     ["lt"] = 60,
2091     ["LT"] = 60,
2092     ["equals"] = 61,
2093     ["gt"] = 62,
2094     ["GT"] = 62,
2095     ["quest"] = 63,
2096     ["commat"] = 64,
2097     ["lsqb"] = 91,
2098     ["lbrack"] = 91,
2099     ["bsol"] = 92,
2100     ["rsqb"] = 93,
2101     ["rbrack"] = 93,
2102     ["Hat"] = 94,
2103     ["lowbar"] = 95,
2104     ["grave"] = 96,
2105     ["DiacriticalGrave"] = 96,
2106     ["lcub"] = 123,
2107     ["lbrace"] = 123,
2108     ["verbar"] = 124,
2109     ["vert"] = 124,
2110     ["VerticalLine"] = 124,
2111     ["rcub"] = 125,
2112     ["rbrace"] = 125,
2113     ["nbsp"] = 160,

```

```

2114 ["NonBreakingSpace"] = 160,
2115 ["iexcl"] = 161,
2116 ["cent"] = 162,
2117 ["pound"] = 163,
2118 ["curren"] = 164,
2119 ["yen"] = 165,
2120 ["brvbar"] = 166,
2121 ["sect"] = 167,
2122 ["Dot"] = 168,
2123 ["die"] = 168,
2124 ["DoubleDot"] = 168,
2125 ["uml"] = 168,
2126 ["copy"] = 169,
2127 ["COPY"] = 169,
2128 ["ordf"] = 170,
2129 ["laquo"] = 171,
2130 ["not"] = 172,
2131 ["shy"] = 173,
2132 ["reg"] = 174,
2133 ["circledR"] = 174,
2134 ["REG"] = 174,
2135 ["macr"] = 175,
2136 ["OverBar"] = 175,
2137 ["strns"] = 175,
2138 ["deg"] = 176,
2139 ["plusmn"] = 177,
2140 ["pm"] = 177,
2141 ["PlusMinus"] = 177,
2142 ["sup2"] = 178,
2143 ["sup3"] = 179,
2144 ["acute"] = 180,
2145 ["DiacriticalAcute"] = 180,
2146 ["micro"] = 181,
2147 ["para"] = 182,
2148 ["middot"] = 183,
2149 ["centerdot"] = 183,
2150 ["CenterDot"] = 183,
2151 ["cedil"] = 184,
2152 ["Cedilla"] = 184,
2153 ["sup1"] = 185,
2154 ["ordm"] = 186,
2155 ["raquo"] = 187,
2156 ["frac14"] = 188,
2157 ["frac12"] = 189,
2158 ["half"] = 189,
2159 ["frac34"] = 190,
2160 ["iquest"] = 191,

```

2161 ["Agrave"] = 192,
 2162 ["Aacute"] = 193,
 2163 ["Acirc"] = 194,
 2164 ["Atilde"] = 195,
 2165 ["Auml"] = 196,
 2166 ["Aring"] = 197,
 2167 ["AElig"] = 198,
 2168 ["Ccedil"] = 199,
 2169 ["Egrave"] = 200,
 2170 ["Eacute"] = 201,
 2171 ["Ecirc"] = 202,
 2172 ["Euml"] = 203,
 2173 ["Igrave"] = 204,
 2174 ["Iacute"] = 205,
 2175 ["Icirc"] = 206,
 2176 ["Iuml"] = 207,
 2177 ["ETH"] = 208,
 2178 ["Ntilde"] = 209,
 2179 ["Ograve"] = 210,
 2180 ["Oacute"] = 211,
 2181 ["Ocirc"] = 212,
 2182 ["Otilde"] = 213,
 2183 ["Ouml"] = 214,
 2184 ["times"] = 215,
 2185 ["Oslash"] = 216,
 2186 ["Ugrave"] = 217,
 2187 ["Uacute"] = 218,
 2188 ["Ucirc"] = 219,
 2189 ["Uuml"] = 220,
 2190 ["Yacute"] = 221,
 2191 ["THORN"] = 222,
 2192 ["szlig"] = 223,
 2193 ["agrave"] = 224,
 2194 ["aacute"] = 225,
 2195 ["acirc"] = 226,
 2196 ["atilde"] = 227,
 2197 ["auml"] = 228,
 2198 ["aring"] = 229,
 2199 ["aelig"] = 230,
 2200 ["ccedil"] = 231,
 2201 ["egrave"] = 232,
 2202 ["eacute"] = 233,
 2203 ["ecirc"] = 234,
 2204 ["euml"] = 235,
 2205 ["igrave"] = 236,
 2206 ["iacute"] = 237,
 2207 ["icirc"] = 238,

2208 ["iuml"] = 239,
 2209 ["eth"] = 240,
 2210 ["ntilde"] = 241,
 2211 ["ograve"] = 242,
 2212 ["oacute"] = 243,
 2213 ["ocirc"] = 244,
 2214 ["otilde"] = 245,
 2215 ["ouml"] = 246,
 2216 ["divide"] = 247,
 2217 ["div"] = 247,
 2218 ["oslash"] = 248,
 2219 ["ugrave"] = 249,
 2220 ["uacute"] = 250,
 2221 ["ucirc"] = 251,
 2222 ["uuml"] = 252,
 2223 ["yacute"] = 253,
 2224 ["thorn"] = 254,
 2225 ["yuml"] = 255,
 2226 ["Amacr"] = 256,
 2227 ["amacr"] = 257,
 2228 ["Abreve"] = 258,
 2229 ["abreve"] = 259,
 2230 ["Aogon"] = 260,
 2231 ["aogon"] = 261,
 2232 ["Cacute"] = 262,
 2233 ["cacute"] = 263,
 2234 ["Ccirc"] = 264,
 2235 ["ccirc"] = 265,
 2236 ["Cdot"] = 266,
 2237 ["cdot"] = 267,
 2238 ["Ccaron"] = 268,
 2239 ["ccaron"] = 269,
 2240 ["Dcaron"] = 270,
 2241 ["dcaron"] = 271,
 2242 ["Dstrok"] = 272,
 2243 ["dstrok"] = 273,
 2244 ["Emacr"] = 274,
 2245 ["emacr"] = 275,
 2246 ["Edot"] = 278,
 2247 ["edot"] = 279,
 2248 ["Eogon"] = 280,
 2249 ["eogon"] = 281,
 2250 ["Ecaron"] = 282,
 2251 ["ecaron"] = 283,
 2252 ["Gcirc"] = 284,
 2253 ["gcirc"] = 285,
 2254 ["Gbreve"] = 286,

2255 ["gbreve"] = 287,
 2256 ["Gdot"] = 288,
 2257 ["gdot"] = 289,
 2258 ["Gcedil"] = 290,
 2259 ["Hcirc"] = 292,
 2260 ["hcirc"] = 293,
 2261 ["Hstrok"] = 294,
 2262 ["hstrok"] = 295,
 2263 ["Itilde"] = 296,
 2264 ["itilde"] = 297,
 2265 ["Imacr"] = 298,
 2266 ["imacr"] = 299,
 2267 ["Iogon"] = 302,
 2268 ["iogon"] = 303,
 2269 ["Idot"] = 304,
 2270 ["imath"] = 305,
 2271 ["inodot"] = 305,
 2272 ["IJlig"] = 306,
 2273 ["ijlig"] = 307,
 2274 ["Jcirc"] = 308,
 2275 ["jcirc"] = 309,
 2276 ["Kcedil"] = 310,
 2277 ["kcedil"] = 311,
 2278 ["kgreen"] = 312,
 2279 ["Lacute"] = 313,
 2280 ["lacute"] = 314,
 2281 ["Lcedil"] = 315,
 2282 ["lcedil"] = 316,
 2283 ["Lcaron"] = 317,
 2284 ["lcaron"] = 318,
 2285 ["Lmidot"] = 319,
 2286 ["lmidot"] = 320,
 2287 ["Lstrok"] = 321,
 2288 ["lstrok"] = 322,
 2289 ["Nacute"] = 323,
 2290 ["nacute"] = 324,
 2291 ["Ncedil"] = 325,
 2292 ["ncedil"] = 326,
 2293 ["Ncaron"] = 327,
 2294 ["ncaron"] = 328,
 2295 ["napos"] = 329,
 2296 ["ENG"] = 330,
 2297 ["eng"] = 331,
 2298 ["Omacr"] = 332,
 2299 ["omacr"] = 333,
 2300 ["Odblac"] = 336,
 2301 ["odblac"] = 337,

2302 ["OElig"] = 338,
 2303 ["oelig"] = 339,
 2304 ["Racute"] = 340,
 2305 ["racute"] = 341,
 2306 ["Rcedil"] = 342,
 2307 ["rcedil"] = 343,
 2308 ["Rcaron"] = 344,
 2309 ["rcaron"] = 345,
 2310 ["Sacute"] = 346,
 2311 ["sacute"] = 347,
 2312 ["Scirc"] = 348,
 2313 ["scirc"] = 349,
 2314 ["Scedil"] = 350,
 2315 ["scedil"] = 351,
 2316 ["Scaron"] = 352,
 2317 ["scaron"] = 353,
 2318 ["Tcedil"] = 354,
 2319 ["tcedil"] = 355,
 2320 ["Tcaron"] = 356,
 2321 ["tcaron"] = 357,
 2322 ["Tstrok"] = 358,
 2323 ["tstrok"] = 359,
 2324 ["Utilde"] = 360,
 2325 ["utilde"] = 361,
 2326 ["Umacr"] = 362,
 2327 ["umacr"] = 363,
 2328 ["Ubreve"] = 364,
 2329 ["ubreve"] = 365,
 2330 ["Uring"] = 366,
 2331 ["uring"] = 367,
 2332 ["Udblac"] = 368,
 2333 ["udblac"] = 369,
 2334 ["Uogon"] = 370,
 2335 ["uogon"] = 371,
 2336 ["Wcirc"] = 372,
 2337 ["wcirc"] = 373,
 2338 ["Ycirc"] = 374,
 2339 ["ycirc"] = 375,
 2340 ["Yuml"] = 376,
 2341 ["Zacute"] = 377,
 2342 ["zacute"] = 378,
 2343 ["Zdot"] = 379,
 2344 ["zdot"] = 380,
 2345 ["Zcaron"] = 381,
 2346 ["zcaron"] = 382,
 2347 ["fnof"] = 402,
 2348 ["imped"] = 437,

2349 ["gacute"] = 501,
 2350 ["jmath"] = 567,
 2351 ["circ"] = 710,
 2352 ["caron"] = 711,
 2353 ["Hacek"] = 711,
 2354 ["breve"] = 728,
 2355 ["Breve"] = 728,
 2356 ["dot"] = 729,
 2357 ["DiacriticalDot"] = 729,
 2358 ["ring"] = 730,
 2359 ["ogon"] = 731,
 2360 ["tilde"] = 732,
 2361 ["DiacriticalTilde"] = 732,
 2362 ["dblac"] = 733,
 2363 ["DiacriticalDoubleAcute"] = 733,
 2364 ["DownBreve"] = 785,
 2365 ["UnderBar"] = 818,
 2366 ["Alpha"] = 913,
 2367 ["Beta"] = 914,
 2368 ["Gamma"] = 915,
 2369 ["Delta"] = 916,
 2370 ["Epsilon"] = 917,
 2371 ["Zeta"] = 918,
 2372 ["Eta"] = 919,
 2373 ["Theta"] = 920,
 2374 ["Iota"] = 921,
 2375 ["Kappa"] = 922,
 2376 ["Lambda"] = 923,
 2377 ["Mu"] = 924,
 2378 ["Nu"] = 925,
 2379 ["Xi"] = 926,
 2380 ["Omicron"] = 927,
 2381 ["Pi"] = 928,
 2382 ["Rho"] = 929,
 2383 ["Sigma"] = 931,
 2384 ["Tau"] = 932,
 2385 ["Upsilon"] = 933,
 2386 ["Phi"] = 934,
 2387 ["Chi"] = 935,
 2388 ["Psi"] = 936,
 2389 ["Omega"] = 937,
 2390 ["alpha"] = 945,
 2391 ["beta"] = 946,
 2392 ["gamma"] = 947,
 2393 ["delta"] = 948,
 2394 ["epsiv"] = 949,
 2395 ["varepsilon"] = 949,

```

2396 ["epsilon"] = 949,
2397 ["zeta"] = 950,
2398 ["eta"] = 951,
2399 ["theta"] = 952,
2400 ["iota"] = 953,
2401 ["kappa"] = 954,
2402 ["lambda"] = 955,
2403 ["mu"] = 956,
2404 ["nu"] = 957,
2405 ["xi"] = 958,
2406 ["omicron"] = 959,
2407 ["pi"] = 960,
2408 ["rho"] = 961,
2409 ["sigmav"] = 962,
2410 ["varsigma"] = 962,
2411 ["sigmaf"] = 962,
2412 ["sigma"] = 963,
2413 ["tau"] = 964,
2414 ["upsi"] = 965,
2415 ["upsilon"] = 965,
2416 ["phi"] = 966,
2417 ["phiv"] = 966,
2418 ["varphi"] = 966,
2419 ["chi"] = 967,
2420 ["psi"] = 968,
2421 ["omega"] = 969,
2422 ["thetav"] = 977,
2423 ["vartheta"] = 977,
2424 ["thetasym"] = 977,
2425 ["Upsi"] = 978,
2426 ["upsih"] = 978,
2427 ["straightphi"] = 981,
2428 ["piv"] = 982,
2429 ["varpi"] = 982,
2430 ["Gammad"] = 988,
2431 ["gammad"] = 989,
2432 ["digamma"] = 989,
2433 ["kappav"] = 1008,
2434 ["varkappa"] = 1008,
2435 ["rhov"] = 1009,
2436 ["varrho"] = 1009,
2437 ["epsi"] = 1013,
2438 ["straightepsilon"] = 1013,
2439 ["bepsi"] = 1014,
2440 ["backepsilon"] = 1014,
2441 ["IOcy"] = 1025,
2442 ["DJcy"] = 1026,

```

```
2443 ["GJcy"] = 1027,
2444 ["Jukcy"] = 1028,
2445 ["DScy"] = 1029,
2446 ["Iukcy"] = 1030,
2447 ["YIcy"] = 1031,
2448 ["Jsercy"] = 1032,
2449 ["LJcy"] = 1033,
2450 ["NJcy"] = 1034,
2451 ["TSHcy"] = 1035,
2452 ["KJcy"] = 1036,
2453 ["Ubrcy"] = 1038,
2454 ["DZcy"] = 1039,
2455 ["Acy"] = 1040,
2456 ["Bcy"] = 1041,
2457 ["Vcy"] = 1042,
2458 ["Gcy"] = 1043,
2459 ["Dcy"] = 1044,
2460 ["IEcy"] = 1045,
2461 ["ZHcy"] = 1046,
2462 ["Zcy"] = 1047,
2463 ["Icy"] = 1048,
2464 ["Jcy"] = 1049,
2465 ["Kcy"] = 1050,
2466 ["Lcy"] = 1051,
2467 ["Mcy"] = 1052,
2468 ["Ncy"] = 1053,
2469 ["Ocy"] = 1054,
2470 ["Pcy"] = 1055,
2471 ["Rcy"] = 1056,
2472 ["Scy"] = 1057,
2473 ["Tcy"] = 1058,
2474 ["Ucy"] = 1059,
2475 ["Fcy"] = 1060,
2476 ["KHcy"] = 1061,
2477 ["TScy"] = 1062,
2478 ["CHcy"] = 1063,
2479 ["SHcy"] = 1064,
2480 ["SHCHcy"] = 1065,
2481 ["HARDcy"] = 1066,
2482 ["Ycy"] = 1067,
2483 ["SOFTcy"] = 1068,
2484 ["Ecy"] = 1069,
2485 ["YUcy"] = 1070,
2486 ["YAcy"] = 1071,
2487 ["acy"] = 1072,
2488 ["bcy"] = 1073,
2489 ["vcy"] = 1074,
```

```
2490 ["gcy"] = 1075,  
2491 ["dcy"] = 1076,  
2492 ["iecy"] = 1077,  
2493 ["zhcy"] = 1078,  
2494 ["zcy"] = 1079,  
2495 ["icy"] = 1080,  
2496 ["jcy"] = 1081,  
2497 ["kcy"] = 1082,  
2498 ["lcy"] = 1083,  
2499 ["mcy"] = 1084,  
2500 ["ncy"] = 1085,  
2501 ["ocy"] = 1086,  
2502 ["pcy"] = 1087,  
2503 ["rcy"] = 1088,  
2504 ["scy"] = 1089,  
2505 ["tcy"] = 1090,  
2506 ["ucy"] = 1091,  
2507 ["fcy"] = 1092,  
2508 ["khcy"] = 1093,  
2509 ["tscy"] = 1094,  
2510 ["chcy"] = 1095,  
2511 ["shcy"] = 1096,  
2512 ["shchcy"] = 1097,  
2513 ["hardcy"] = 1098,  
2514 ["ycy"] = 1099,  
2515 ["softcy"] = 1100,  
2516 ["ecy"] = 1101,  
2517 ["yucy"] = 1102,  
2518 ["yacy"] = 1103,  
2519 ["iocy"] = 1105,  
2520 ["djcy"] = 1106,  
2521 ["gjcy"] = 1107,  
2522 ["jukcy"] = 1108,  
2523 ["dscy"] = 1109,  
2524 ["iukcy"] = 1110,  
2525 ["yicy"] = 1111,  
2526 ["jsercy"] = 1112,  
2527 ["ljcy"] = 1113,  
2528 ["njcy"] = 1114,  
2529 ["tshcy"] = 1115,  
2530 ["kjcy"] = 1116,  
2531 ["ubrcy"] = 1118,  
2532 ["dzcy"] = 1119,  
2533 ["ensp"] = 8194,  
2534 ["emsp"] = 8195,  
2535 ["emsp13"] = 8196,  
2536 ["emsp14"] = 8197,
```

2537 ["numsp"] = 8199,
 2538 ["puncsp"] = 8200,
 2539 ["thinsp"] = 8201,
 2540 ["ThinSpace"] = 8201,
 2541 ["hairsp"] = 8202,
 2542 ["VeryThinSpace"] = 8202,
 2543 ["ZeroWidthSpace"] = 8203,
 2544 ["NegativeVeryThinSpace"] = 8203,
 2545 ["NegativeThinSpace"] = 8203,
 2546 ["NegativeMediumSpace"] = 8203,
 2547 ["NegativeThickSpace"] = 8203,
 2548 ["zwnj"] = 8204,
 2549 ["zwj"] = 8205,
 2550 ["lrm"] = 8206,
 2551 ["rlm"] = 8207,
 2552 ["hyphen"] = 8208,
 2553 ["dash"] = 8208,
 2554 ["ndash"] = 8211,
 2555 ["mdash"] = 8212,
 2556 ["horbar"] = 8213,
 2557 ["Verbar"] = 8214,
 2558 ["Vert"] = 8214,
 2559 ["lsquo"] = 8216,
 2560 ["OpenCurlyQuote"] = 8216,
 2561 ["rsquo"] = 8217,
 2562 ["rsquor"] = 8217,
 2563 ["CloseCurlyQuote"] = 8217,
 2564 ["lsquor"] = 8218,
 2565 ["sbquo"] = 8218,
 2566 ["ldquo"] = 8220,
 2567 ["OpenCurlyDoubleQuote"] = 8220,
 2568 ["rdquo"] = 8221,
 2569 ["rdquor"] = 8221,
 2570 ["CloseCurlyDoubleQuote"] = 8221,
 2571 ["ldquor"] = 8222,
 2572 ["bdquo"] = 8222,
 2573 ["dagger"] = 8224,
 2574 ["Dagger"] = 8225,
 2575 ["ddagger"] = 8225,
 2576 ["bull"] = 8226,
 2577 ["bullet"] = 8226,
 2578 ["nldr"] = 8229,
 2579 ["hellip"] = 8230,
 2580 ["mldr"] = 8230,
 2581 ["permil"] = 8240,
 2582 ["pertenk"] = 8241,
 2583 ["prime"] = 8242,

```

2584 ["Prime"] = 8243,
2585 ["tprime"] = 8244,
2586 ["bprime"] = 8245,
2587 ["backprime"] = 8245,
2588 ["lsaquo"] = 8249,
2589 ["rsaquo"] = 8250,
2590 ["oline"] = 8254,
2591 ["caret"] = 8257,
2592 ["hybull"] = 8259,
2593 ["frasl"] = 8260,
2594 ["bsemi"] = 8271,
2595 ["qprime"] = 8279,
2596 ["MediumSpace"] = 8287,
2597 ["NoBreak"] = 8288,
2598 ["ApplyFunction"] = 8289,
2599 ["af"] = 8289,
2600 ["InvisibleTimes"] = 8290,
2601 ["it"] = 8290,
2602 ["InvisibleComma"] = 8291,
2603 ["ic"] = 8291,
2604 ["euro"] = 8364,
2605 ["tdot"] = 8411,
2606 ["TripleDot"] = 8411,
2607 ["DotDot"] = 8412,
2608 ["Copf"] = 8450,
2609 ["complexes"] = 8450,
2610 ["incare"] = 8453,
2611 ["gscr"] = 8458,
2612 ["hamilt"] = 8459,
2613 ["HilbertSpace"] = 8459,
2614 ["Hscr"] = 8459,
2615 ["Hfr"] = 8460,
2616 ["Poincareplane"] = 8460,
2617 ["quaternions"] = 8461,
2618 ["Hopf"] = 8461,
2619 ["planckh"] = 8462,
2620 ["planck"] = 8463,
2621 ["hbar"] = 8463,
2622 ["plankv"] = 8463,
2623 ["hslash"] = 8463,
2624 ["Iscr"] = 8464,
2625 ["imagline"] = 8464,
2626 ["image"] = 8465,
2627 ["Im"] = 8465,
2628 ["imagpart"] = 8465,
2629 ["Ifr"] = 8465,
2630 ["Lscr"] = 8466,

```

```

2631 ["lagran"] = 8466,
2632 ["Laplacetrif"] = 8466,
2633 ["ell"] = 8467,
2634 ["Nopf"] = 8469,
2635 ["naturals"] = 8469,
2636 ["numero"] = 8470,
2637 ["copysr"] = 8471,
2638 ["weierp"] = 8472,
2639 ["wp"] = 8472,
2640 ["Popf"] = 8473,
2641 ["primes"] = 8473,
2642 ["rationals"] = 8474,
2643 ["Qopf"] = 8474,
2644 ["Rscr"] = 8475,
2645 ["realine"] = 8475,
2646 ["real"] = 8476,
2647 ["Re"] = 8476,
2648 ["realpart"] = 8476,
2649 ["Rfr"] = 8476,
2650 ["reals"] = 8477,
2651 ["Ropf"] = 8477,
2652 ["rx"] = 8478,
2653 ["trade"] = 8482,
2654 ["TRADE"] = 8482,
2655 ["integers"] = 8484,
2656 ["Zopf"] = 8484,
2657 ["ohm"] = 8486,
2658 ["mho"] = 8487,
2659 ["Zfr"] = 8488,
2660 ["zeetrif"] = 8488,
2661 ["iiota"] = 8489,
2662 ["angst"] = 8491,
2663 ["bernou"] = 8492,
2664 ["Bernoullis"] = 8492,
2665 ["Bscr"] = 8492,
2666 ["Cfr"] = 8493,
2667 ["Cayleys"] = 8493,
2668 ["eser"] = 8495,
2669 ["Escr"] = 8496,
2670 ["expectation"] = 8496,
2671 ["Fscr"] = 8497,
2672 ["Fouriertrif"] = 8497,
2673 ["phmmat"] = 8499,
2674 ["Mellintrif"] = 8499,
2675 ["Mscr"] = 8499,
2676 ["order"] = 8500,
2677 ["orderof"] = 8500,

```



```

2678 ["oscr"] = 8500,
2679 ["alefsym"] = 8501,
2680 ["aleph"] = 8501,
2681 ["beth"] = 8502,
2682 ["gimel"] = 8503,
2683 ["daleth"] = 8504,
2684 ["CapitalDifferentialD"] = 8517,
2685 ["DD"] = 8517,
2686 ["DifferentialD"] = 8518,
2687 ["dd"] = 8518,
2688 ["ExponentialE"] = 8519,
2689 ["exponentiale"] = 8519,
2690 ["ee"] = 8519,
2691 ["ImaginaryI"] = 8520,
2692 ["ii"] = 8520,
2693 ["frac13"] = 8531,
2694 ["frac23"] = 8532,
2695 ["frac15"] = 8533,
2696 ["frac25"] = 8534,
2697 ["frac35"] = 8535,
2698 ["frac45"] = 8536,
2699 ["frac16"] = 8537,
2700 ["frac56"] = 8538,
2701 ["frac18"] = 8539,
2702 ["frac38"] = 8540,
2703 ["frac58"] = 8541,
2704 ["frac78"] = 8542,
2705 ["larr"] = 8592,
2706 ["leftarrow"] = 8592,
2707 ["LeftArrow"] = 8592,
2708 ["slarr"] = 8592,
2709 ["ShortLeftArrow"] = 8592,
2710 ["uarr"] = 8593,
2711 ["uparrow"] = 8593,
2712 ["UpArrow"] = 8593,
2713 ["ShortUpArrow"] = 8593,
2714 ["rarr"] = 8594,
2715 ["rightarrow"] = 8594,
2716 ["RightArrow"] = 8594,
2717 ["srarr"] = 8594,
2718 ["ShortRightArrow"] = 8594,
2719 ["darr"] = 8595,
2720 ["downarrow"] = 8595,
2721 ["DownArrow"] = 8595,
2722 ["ShortDownArrow"] = 8595,
2723 ["harr"] = 8596,
2724 ["leftrightarrow"] = 8596,

```

```

2725 ["LeftRightArrow"] = 8596,
2726 ["varr"] = 8597,
2727 ["updownarrow"] = 8597,
2728 ["UpDownArrow"] = 8597,
2729 ["nwarr"] = 8598,
2730 ["UpperLeftArrow"] = 8598,
2731 ["nwarrow"] = 8598,
2732 ["nearr"] = 8599,
2733 ["UpperRightArrow"] = 8599,
2734 ["nearrow"] = 8599,
2735 ["searr"] = 8600,
2736 ["searrow"] = 8600,
2737 ["LowerRightArrow"] = 8600,
2738 ["swarr"] = 8601,
2739 ["swarrow"] = 8601,
2740 ["LowerLeftArrow"] = 8601,
2741 ["nlarr"] = 8602,
2742 ["nleftarrow"] = 8602,
2743 ["nrarr"] = 8603,
2744 ["nrightarrow"] = 8603,
2745 ["rarrw"] = 8605,
2746 ["rightsquigarrow"] = 8605,
2747 ["Larr"] = 8606,
2748 ["twoheadleftarrow"] = 8606,
2749 ["Uarr"] = 8607,
2750 ["Rarr"] = 8608,
2751 ["twoheadrightarrow"] = 8608,
2752 ["Darr"] = 8609,
2753 ["larrtl"] = 8610,
2754 ["leftarrowtail"] = 8610,
2755 ["rarrtl"] = 8611,
2756 ["rightarrowtail"] = 8611,
2757 ["LeftTeeArrow"] = 8612,
2758 ["mapstoleft"] = 8612,
2759 ["UpTeeArrow"] = 8613,
2760 ["mapstoup"] = 8613,
2761 ["map"] = 8614,
2762 ["RightTeeArrow"] = 8614,
2763 ["mapsto"] = 8614,
2764 ["DownTeeArrow"] = 8615,
2765 ["mapstodown"] = 8615,
2766 ["larrhk"] = 8617,
2767 ["hookleftarrow"] = 8617,
2768 ["rarrhk"] = 8618,
2769 ["hookrightarrow"] = 8618,
2770 ["larrlp"] = 8619,
2771 ["looparrowleft"] = 8619,

```

```

2772 ["rarrlp"] = 8620,
2773 ["looparrowright"] = 8620,
2774 ["harrw"] = 8621,
2775 ["leftrightsquigarrow"] = 8621,
2776 ["nharr"] = 8622,
2777 ["nletrightarrow"] = 8622,
2778 ["lsh"] = 8624,
2779 ["Lsh"] = 8624,
2780 ["rsh"] = 8625,
2781 ["Rsh"] = 8625,
2782 ["ldsh"] = 8626,
2783 ["rdsh"] = 8627,
2784 ["crarr"] = 8629,
2785 ["cularr"] = 8630,
2786 ["curvearrowleft"] = 8630,
2787 ["curarr"] = 8631,
2788 ["curvearrowright"] = 8631,
2789 ["olarr"] = 8634,
2790 ["circlearrowleft"] = 8634,
2791 ["orarr"] = 8635,
2792 ["circlearrowright"] = 8635,
2793 ["lharu"] = 8636,
2794 ["LeftVector"] = 8636,
2795 ["leftharpoonup"] = 8636,
2796 ["lhard"] = 8637,
2797 ["leftharpoondown"] = 8637,
2798 ["DownLeftVector"] = 8637,
2799 ["uharr"] = 8638,
2800 ["upharpoonright"] = 8638,
2801 ["RightUpVector"] = 8638,
2802 ["uharl"] = 8639,
2803 ["upharpoonleft"] = 8639,
2804 ["LeftUpVector"] = 8639,
2805 ["rharu"] = 8640,
2806 ["RightVector"] = 8640,
2807 ["rightharpoonup"] = 8640,
2808 ["rhard"] = 8641,
2809 ["rightharpoondown"] = 8641,
2810 ["DownRightVector"] = 8641,
2811 ["dharr"] = 8642,
2812 ["RightDownVector"] = 8642,
2813 ["downharpoonright"] = 8642,
2814 ["dharl"] = 8643,
2815 ["LeftDownVector"] = 8643,
2816 ["downharpoonleft"] = 8643,
2817 ["rlarr"] = 8644,
2818 ["rightleftarrows"] = 8644,

```

```

2819 ["RightArrowLeftArrow"] = 8644,
2820 ["udarr"] = 8645,
2821 ["UpArrowDownArrow"] = 8645,
2822 ["lrarr"] = 8646,
2823 ["leftrightarrows"] = 8646,
2824 ["LeftArrowRightArrow"] = 8646,
2825 ["llarr"] = 8647,
2826 ["leftleftarrows"] = 8647,
2827 ["uuarr"] = 8648,
2828 ["upuparrows"] = 8648,
2829 ["rrarr"] = 8649,
2830 ["rightrightarrows"] = 8649,
2831 ["ddarr"] = 8650,
2832 ["downdownarrows"] = 8650,
2833 ["lrhar"] = 8651,
2834 ["ReverseEquilibrium"] = 8651,
2835 ["leftrightharpoons"] = 8651,
2836 ["rlhar"] = 8652,
2837 ["rightleftharpoons"] = 8652,
2838 ["Equilibrium"] = 8652,
2839 ["nlArr"] = 8653,
2840 ["nLeftarrow"] = 8653,
2841 ["nhArr"] = 8654,
2842 ["nLeftrightarrow"] = 8654,
2843 ["nrArr"] = 8655,
2844 ["nRightarrow"] = 8655,
2845 ["lArr"] = 8656,
2846 ["Leftarrow"] = 8656,
2847 ["DoubleLeftArrow"] = 8656,
2848 ["uArr"] = 8657,
2849 ["Uparrow"] = 8657,
2850 ["DoubleUpArrow"] = 8657,
2851 ["rArr"] = 8658,
2852 ["Rightarrow"] = 8658,
2853 ["Implies"] = 8658,
2854 ["DoubleRightArrow"] = 8658,
2855 ["dArr"] = 8659,
2856 ["Downarrow"] = 8659,
2857 ["DoubleDownArrow"] = 8659,
2858 ["hArr"] = 8660,
2859 ["Leftrightarrow"] = 8660,
2860 ["DoubleLeftRightArrow"] = 8660,
2861 ["iff"] = 8660,
2862 ["vArr"] = 8661,
2863 ["Updownarrow"] = 8661,
2864 ["DoubleUpDownArrow"] = 8661,
2865 ["nwArr"] = 8662,

```

```

2866 ["neArr"] = 8663,
2867 ["seArr"] = 8664,
2868 ["swArr"] = 8665,
2869 ["lAarr"] = 8666,
2870 ["Lleftarrow"] = 8666,
2871 ["rAarr"] = 8667,
2872 ["Rrightarrow"] = 8667,
2873 ["zigrarr"] = 8669,
2874 ["larrb"] = 8676,
2875 ["LeftArrowBar"] = 8676,
2876 ["rarrb"] = 8677,
2877 ["RightArrowBar"] = 8677,
2878 ["duarr"] = 8693,
2879 ["DownArrowUpArrow"] = 8693,
2880 ["loarr"] = 8701,
2881 ["roarr"] = 8702,
2882 ["hoarr"] = 8703,
2883 ["forall"] = 8704,
2884 ["ForAll"] = 8704,
2885 ["comp"] = 8705,
2886 ["complement"] = 8705,
2887 ["part"] = 8706,
2888 ["PartialD"] = 8706,
2889 ["exist"] = 8707,
2890 ["Exists"] = 8707,
2891 ["nexist"] = 8708,
2892 ["NotExists"] = 8708,
2893 ["nexists"] = 8708,
2894 ["empty"] = 8709,
2895 ["emptyset"] = 8709,
2896 ["emptyv"] = 8709,
2897 ["varnothing"] = 8709,
2898 ["nabla"] = 8711,
2899 ["Del"] = 8711,
2900 ["isin"] = 8712,
2901 ["isinv"] = 8712,
2902 ["Element"] = 8712,
2903 ["in"] = 8712,
2904 ["notin"] = 8713,
2905 ["NotElement"] = 8713,
2906 ["notinva"] = 8713,
2907 ["niv"] = 8715,
2908 ["ReverseElement"] = 8715,
2909 ["ni"] = 8715,
2910 ["SuchThat"] = 8715,
2911 ["notni"] = 8716,
2912 ["notniva"] = 8716,

```

```

2913 ["NotReverseElement"] = 8716,
2914 ["prod"] = 8719,
2915 ["Product"] = 8719,
2916 ["coprod"] = 8720,
2917 ["Coproduct"] = 8720,
2918 ["sum"] = 8721,
2919 ["Sum"] = 8721,
2920 ["minus"] = 8722,
2921 ["mnplus"] = 8723,
2922 ["mp"] = 8723,
2923 ["MinusPlus"] = 8723,
2924 ["plusdo"] = 8724,
2925 ["dotplus"] = 8724,
2926 ["setmn"] = 8726,
2927 ["setminus"] = 8726,
2928 ["Backslash"] = 8726,
2929 ["ssetmn"] = 8726,
2930 ["smallsetminus"] = 8726,
2931 ["lowast"] = 8727,
2932 ["compfn"] = 8728,
2933 ["SmallCircle"] = 8728,
2934 ["radic"] = 8730,
2935 ["Sqrt"] = 8730,
2936 ["prop"] = 8733,
2937 ["propto"] = 8733,
2938 ["Proportional"] = 8733,
2939 ["vprop"] = 8733,
2940 ["varpropto"] = 8733,
2941 ["infin"] = 8734,
2942 ["angrt"] = 8735,
2943 ["ang"] = 8736,
2944 ["angle"] = 8736,
2945 ["angmsd"] = 8737,
2946 ["measuredangle"] = 8737,
2947 ["angsph"] = 8738,
2948 ["mid"] = 8739,
2949 ["VerticalBar"] = 8739,
2950 ["smid"] = 8739,
2951 ["shortmid"] = 8739,
2952 ["nmid"] = 8740,
2953 ["NotVerticalBar"] = 8740,
2954 ["nsmid"] = 8740,
2955 ["nshortmid"] = 8740,
2956 ["par"] = 8741,
2957 ["parallel"] = 8741,
2958 ["DoubleVerticalBar"] = 8741,
2959 ["spar"] = 8741,

```

```

2960 ["shortparallel"] = 8741,
2961 ["npar"] = 8742,
2962 ["nparallel"] = 8742,
2963 ["NotDoubleVerticalBar"] = 8742,
2964 ["nspar"] = 8742,
2965 ["nshortparallel"] = 8742,
2966 ["and"] = 8743,
2967 ["wedge"] = 8743,
2968 ["or"] = 8744,
2969 ["vee"] = 8744,
2970 ["cap"] = 8745,
2971 ["cup"] = 8746,
2972 ["int"] = 8747,
2973 ["Integral"] = 8747,
2974 ["Int"] = 8748,
2975 ["tint"] = 8749,
2976 ["iiint"] = 8749,
2977 ["conint"] = 8750,
2978 ["oint"] = 8750,
2979 ["ContourIntegral"] = 8750,
2980 ["Conint"] = 8751,
2981 ["DoubleContourIntegral"] = 8751,
2982 ["Cconint"] = 8752,
2983 ["cwint"] = 8753,
2984 ["cwconint"] = 8754,
2985 ["ClockwiseContourIntegral"] = 8754,
2986 ["awconint"] = 8755,
2987 ["CounterClockwiseContourIntegral"] = 8755,
2988 ["there4"] = 8756,
2989 ["therefore"] = 8756,
2990 ["Therefore"] = 8756,
2991 ["because"] = 8757,
2992 ["because"] = 8757,
2993 ["Because"] = 8757,
2994 ["ratio"] = 8758,
2995 ["Colon"] = 8759,
2996 ["Proportion"] = 8759,
2997 ["minusd"] = 8760,
2998 ["dotminus"] = 8760,
2999 ["mDDot"] = 8762,
3000 ["homtht"] = 8763,
3001 ["sim"] = 8764,
3002 ["Tilde"] = 8764,
3003 ["thksim"] = 8764,
3004 ["thicksim"] = 8764,
3005 ["bsim"] = 8765,
3006 ["backsim"] = 8765,

```

```

3007 ["ac"] = 8766,
3008 ["mstpos"] = 8766,
3009 ["acd"] = 8767,
3010 ["wreath"] = 8768,
3011 ["VerticalTilde"] = 8768,
3012 ["wr"] = 8768,
3013 ["nsim"] = 8769,
3014 ["NotTilde"] = 8769,
3015 ["esim"] = 8770,
3016 ["EqualTilde"] = 8770,
3017 ["eqsim"] = 8770,
3018 ["sime"] = 8771,
3019 ["TildeEqual"] = 8771,
3020 ["simeq"] = 8771,
3021 ["nsime"] = 8772,
3022 ["nsimeq"] = 8772,
3023 ["NotTildeEqual"] = 8772,
3024 ["cong"] = 8773,
3025 ["TildeFullEqual"] = 8773,
3026 ["simne"] = 8774,
3027 ["ncong"] = 8775,
3028 ["NotTildeFullEqual"] = 8775,
3029 ["asymp"] = 8776,
3030 ["ap"] = 8776,
3031 ["TildeTilde"] = 8776,
3032 ["approx"] = 8776,
3033 ["thkap"] = 8776,
3034 ["thickapprox"] = 8776,
3035 ["nap"] = 8777,
3036 ["NotTildeTilde"] = 8777,
3037 ["napprox"] = 8777,
3038 ["ape"] = 8778,
3039 ["approxeq"] = 8778,
3040 ["apid"] = 8779,
3041 ["bcong"] = 8780,
3042 ["backcong"] = 8780,
3043 ["asympeq"] = 8781,
3044 ["CupCap"] = 8781,
3045 ["bump"] = 8782,
3046 ["HumpDownHump"] = 8782,
3047 ["Bumpeq"] = 8782,
3048 ["bumpe"] = 8783,
3049 ["HumpEqual"] = 8783,
3050 ["bumpeq"] = 8783,
3051 ["esdot"] = 8784,
3052 ["DotEqual"] = 8784,
3053 ["doteq"] = 8784,

```



```

3054 ["eDot"] = 8785,
3055 ["doteqdot"] = 8785,
3056 ["efDot"] = 8786,
3057 ["fallingdotseq"] = 8786,
3058 ["erDot"] = 8787,
3059 ["risingdotseq"] = 8787,
3060 ["colone"] = 8788,
3061 ["coloneq"] = 8788,
3062 ["Assign"] = 8788,
3063 ["ecolon"] = 8789,
3064 ["eqcolon"] = 8789,
3065 ["ecir"] = 8790,
3066 ["eqcirc"] = 8790,
3067 ["cire"] = 8791,
3068 ["circeq"] = 8791,
3069 ["wedgeq"] = 8793,
3070 ["veeeq"] = 8794,
3071 ["trie"] = 8796,
3072 ["triangleq"] = 8796,
3073 ["equest"] = 8799,
3074 ["questeq"] = 8799,
3075 ["ne"] = 8800,
3076 ["NotEqual"] = 8800,
3077 ["equiv"] = 8801,
3078 ["Congruent"] = 8801,
3079 ["nequiv"] = 8802,
3080 ["NotCongruent"] = 8802,
3081 ["le"] = 8804,
3082 ["leq"] = 8804,
3083 ["ge"] = 8805,
3084 ["GreaterEqual"] = 8805,
3085 ["geq"] = 8805,
3086 ["lE"] = 8806,
3087 ["LessFullEqual"] = 8806,
3088 ["leqq"] = 8806,
3089 ["gE"] = 8807,
3090 ["GreaterFullEqual"] = 8807,
3091 ["geqq"] = 8807,
3092 ["lnE"] = 8808,
3093 ["lneqq"] = 8808,
3094 ["gnE"] = 8809,
3095 ["gneqq"] = 8809,
3096 ["Lt"] = 8810,
3097 ["NestedLessLess"] = 8810,
3098 ["ll"] = 8810,
3099 ["Gt"] = 8811,
3100 ["NestedGreaterGreater"] = 8811,

```

```

3101 ["gg"] = 8811,
3102 ["twixt"] = 8812,
3103 ["between"] = 8812,
3104 ["NotCupCap"] = 8813,
3105 ["nlt"] = 8814,
3106 ["NotLess"] = 8814,
3107 ["nless"] = 8814,
3108 ["ngt"] = 8815,
3109 ["NotGreater"] = 8815,
3110 ["ngtr"] = 8815,
3111 ["nle"] = 8816,
3112 ["NotLessEqual"] = 8816,
3113 ["nleq"] = 8816,
3114 ["nge"] = 8817,
3115 ["NotGreaterEqual"] = 8817,
3116 ["ngeq"] = 8817,
3117 ["lsim"] = 8818,
3118 ["LessTilde"] = 8818,
3119 ["lesssim"] = 8818,
3120 ["gsim"] = 8819,
3121 ["gtrsim"] = 8819,
3122 ["GreaterTilde"] = 8819,
3123 ["nlsim"] = 8820,
3124 ["NotLessTilde"] = 8820,
3125 ["ngsim"] = 8821,
3126 ["NotGreaterTilde"] = 8821,
3127 ["lg"] = 8822,
3128 ["lessgtr"] = 8822,
3129 ["LessGreater"] = 8822,
3130 ["gl"] = 8823,
3131 ["gtrless"] = 8823,
3132 ["GreaterLess"] = 8823,
3133 ["ntlg"] = 8824,
3134 ["NotLessGreater"] = 8824,
3135 ["ntgl"] = 8825,
3136 ["NotGreaterLess"] = 8825,
3137 ["pr"] = 8826,
3138 ["Precedes"] = 8826,
3139 ["prec"] = 8826,
3140 ["sc"] = 8827,
3141 ["Succeeds"] = 8827,
3142 ["succ"] = 8827,
3143 ["prcue"] = 8828,
3144 ["PrecedesSlantEqual"] = 8828,
3145 ["preccurlyeq"] = 8828,
3146 ["sccue"] = 8829,
3147 ["SucceedsSlantEqual"] = 8829,

```

```

3148 ["succcurlyeq"] = 8829,
3149 ["prsim"] = 8830,
3150 ["precsim"] = 8830,
3151 ["PrecedesTilde"] = 8830,
3152 ["scsim"] = 8831,
3153 ["succsim"] = 8831,
3154 ["SucceedsTilde"] = 8831,
3155 ["npr"] = 8832,
3156 ["nprec"] = 8832,
3157 ["NotPrecedes"] = 8832,
3158 ["nsc"] = 8833,
3159 ["nsucc"] = 8833,
3160 ["NotSucceeds"] = 8833,
3161 ["sub"] = 8834,
3162 ["subset"] = 8834,
3163 ["sup"] = 8835,
3164 ["supset"] = 8835,
3165 ["Superset"] = 8835,
3166 ["nsub"] = 8836,
3167 ["nsup"] = 8837,
3168 ["sube"] = 8838,
3169 ["SubsetEqual"] = 8838,
3170 ["subseteq"] = 8838,
3171 ["supe"] = 8839,
3172 ["supseteq"] = 8839,
3173 ["SupersetEqual"] = 8839,
3174 ["nsube"] = 8840,
3175 ["nsubseteq"] = 8840,
3176 ["NotSubsetEqual"] = 8840,
3177 ["nsupe"] = 8841,
3178 ["nsupseteq"] = 8841,
3179 ["NotSupersetEqual"] = 8841,
3180 ["subne"] = 8842,
3181 ["subsetneq"] = 8842,
3182 ["supne"] = 8843,
3183 ["supsetneq"] = 8843,
3184 ["cupdot"] = 8845,
3185 ["uplus"] = 8846,
3186 ["UnionPlus"] = 8846,
3187 ["sqsub"] = 8847,
3188 ["SquareSubset"] = 8847,
3189 ["sqsubset"] = 8847,
3190 ["sqsup"] = 8848,
3191 ["SquareSuperset"] = 8848,
3192 ["sqsupset"] = 8848,
3193 ["sqsube"] = 8849,
3194 ["SquareSubsetEqual"] = 8849,

```

```

3195 ["sqsubseteq"] = 8849,
3196 ["sqsupe"] = 8850,
3197 ["SquareSupersetEqual"] = 8850,
3198 ["sqsupseteq"] = 8850,
3199 ["sqcap"] = 8851,
3200 ["SquareIntersection"] = 8851,
3201 ["sqcup"] = 8852,
3202 ["SquareUnion"] = 8852,
3203 ["oplus"] = 8853,
3204 ["CirclePlus"] = 8853,
3205 ["ominus"] = 8854,
3206 ["CircleMinus"] = 8854,
3207 ["otimes"] = 8855,
3208 ["CircleTimes"] = 8855,
3209 ["osol"] = 8856,
3210 ["odot"] = 8857,
3211 ["CircleDot"] = 8857,
3212 ["ocir"] = 8858,
3213 ["circledcirc"] = 8858,
3214 ["oast"] = 8859,
3215 ["circledast"] = 8859,
3216 ["odash"] = 8861,
3217 ["circleddash"] = 8861,
3218 ["plusb"] = 8862,
3219 ["boxplus"] = 8862,
3220 ["minusb"] = 8863,
3221 ["boxminus"] = 8863,
3222 ["timesb"] = 8864,
3223 ["boxtimes"] = 8864,
3224 ["sdotb"] = 8865,
3225 ["dotsquare"] = 8865,
3226 ["vdash"] = 8866,
3227 ["RightTee"] = 8866,
3228 ["dashv"] = 8867,
3229 ["LeftTee"] = 8867,
3230 ["top"] = 8868,
3231 ["DownTee"] = 8868,
3232 ["bottom"] = 8869,
3233 ["bot"] = 8869,
3234 ["perp"] = 8869,
3235 ["UpTee"] = 8869,
3236 ["models"] = 8871,
3237 ["vDash"] = 8872,
3238 ["DoubleRightTee"] = 8872,
3239 ["Vdash"] = 8873,
3240 ["Vvdash"] = 8874,
3241 ["VDash"] = 8875,

```

```

3242 ["nvdash"] = 8876,
3243 ["nvDash"] = 8877,
3244 ["nVdash"] = 8878,
3245 ["nVDash"] = 8879,
3246 ["prurel"] = 8880,
3247 ["vltri"] = 8882,
3248 ["vartriangleleft"] = 8882,
3249 ["LeftTriangle"] = 8882,
3250 ["vrtri"] = 8883,
3251 ["vartriangleright"] = 8883,
3252 ["RightTriangle"] = 8883,
3253 ["ltrie"] = 8884,
3254 ["trianglelefteq"] = 8884,
3255 ["LeftTriangleEqual"] = 8884,
3256 ["rtrie"] = 8885,
3257 ["trianglerighteq"] = 8885,
3258 ["RightTriangleEqual"] = 8885,
3259 ["origof"] = 8886,
3260 ["imof"] = 8887,
3261 ["mumap"] = 8888,
3262 ["multimap"] = 8888,
3263 ["hercon"] = 8889,
3264 ["intcal"] = 8890,
3265 ["intercal"] = 8890,
3266 ["veebar"] = 8891,
3267 ["barvee"] = 8893,
3268 ["angrtvb"] = 8894,
3269 ["lrtri"] = 8895,
3270 ["xwedge"] = 8896,
3271 ["Wedge"] = 8896,
3272 ["bigwedge"] = 8896,
3273 ["xvee"] = 8897,
3274 ["Vee"] = 8897,
3275 ["bigvee"] = 8897,
3276 ["xcap"] = 8898,
3277 ["Intersection"] = 8898,
3278 ["bigcap"] = 8898,
3279 ["xcup"] = 8899,
3280 ["Union"] = 8899,
3281 ["bigcup"] = 8899,
3282 ["diam"] = 8900,
3283 ["diamond"] = 8900,
3284 ["Diamond"] = 8900,
3285 ["sdot"] = 8901,
3286 ["sstarf"] = 8902,
3287 ["Star"] = 8902,
3288 ["divonx"] = 8903,

```

```

3289 ["divideontimes"] = 8903,
3290 ["bowtie"] = 8904,
3291 ["ltimes"] = 8905,
3292 ["rtimes"] = 8906,
3293 ["lthree"] = 8907,
3294 ["leftthreetimes"] = 8907,
3295 ["rthree"] = 8908,
3296 ["rightthreetimes"] = 8908,
3297 ["bsime"] = 8909,
3298 ["backsimeq"] = 8909,
3299 ["cuvee"] = 8910,
3300 ["curlyvee"] = 8910,
3301 ["cuwed"] = 8911,
3302 ["curlywedge"] = 8911,
3303 ["Sub"] = 8912,
3304 ["Subset"] = 8912,
3305 ["Sup"] = 8913,
3306 ["Supset"] = 8913,
3307 ["Cap"] = 8914,
3308 ["Cup"] = 8915,
3309 ["fork"] = 8916,
3310 ["pitchfork"] = 8916,
3311 ["epar"] = 8917,
3312 ["ltdot"] = 8918,
3313 ["lessdot"] = 8918,
3314 ["gtdot"] = 8919,
3315 ["gtrdot"] = 8919,
3316 ["Ll"] = 8920,
3317 ["Gg"] = 8921,
3318 ["ggg"] = 8921,
3319 ["leg"] = 8922,
3320 ["LessEqualGreater"] = 8922,
3321 ["lesseqgtr"] = 8922,
3322 ["gel"] = 8923,
3323 ["gtreqless"] = 8923,
3324 ["GreaterEqualLess"] = 8923,
3325 ["cuepr"] = 8926,
3326 ["curlyeqprec"] = 8926,
3327 ["cuesc"] = 8927,
3328 ["curlyeqsucc"] = 8927,
3329 ["nprcue"] = 8928,
3330 ["NotPrecedesSlantEqual"] = 8928,
3331 ["nsccue"] = 8929,
3332 ["NotSucceedsSlantEqual"] = 8929,
3333 ["nsqsube"] = 8930,
3334 ["NotSquareSubsetEqual"] = 8930,
3335 ["nsqsupe"] = 8931,

```

```

3336 ["NotSquareSupersetEqual"] = 8931,
3337 ["lnsim"] = 8934,
3338 ["gnsim"] = 8935,
3339 ["prnsim"] = 8936,
3340 ["precnsim"] = 8936,
3341 ["scnsim"] = 8937,
3342 ["succnsim"] = 8937,
3343 ["nltri"] = 8938,
3344 ["ntriangleleft"] = 8938,
3345 ["NotLeftTriangle"] = 8938,
3346 ["nrtri"] = 8939,
3347 ["ntriangleright"] = 8939,
3348 ["NotRightTriangle"] = 8939,
3349 ["nltrie"] = 8940,
3350 ["ntrianglelefteq"] = 8940,
3351 ["NotLeftTriangleEqual"] = 8940,
3352 ["nrtrie"] = 8941,
3353 ["ntrianglerighteq"] = 8941,
3354 ["NotRightTriangleEqual"] = 8941,
3355 ["vellip"] = 8942,
3356 ["ctdot"] = 8943,
3357 ["utdot"] = 8944,
3358 ["dtdot"] = 8945,
3359 ["disin"] = 8946,
3360 ["isinsv"] = 8947,
3361 ["isins"] = 8948,
3362 ["isindot"] = 8949,
3363 ["notinvc"] = 8950,
3364 ["notinvb"] = 8951,
3365 ["isinE"] = 8953,
3366 ["nisd"] = 8954,
3367 ["xnis"] = 8955,
3368 ["nis"] = 8956,
3369 ["notnivc"] = 8957,
3370 ["notnivb"] = 8958,
3371 ["barwed"] = 8965,
3372 ["barwedge"] = 8965,
3373 ["Barwed"] = 8966,
3374 ["doublebarwedge"] = 8966,
3375 ["lceil"] = 8968,
3376 ["LeftCeiling"] = 8968,
3377 ["rceil"] = 8969,
3378 ["RightCeiling"] = 8969,
3379 ["lfloor"] = 8970,
3380 ["LeftFloor"] = 8970,
3381 ["rfloor"] = 8971,
3382 ["RightFloor"] = 8971,

```

```

3383 ["drcrop"] = 8972,
3384 ["dlcrop"] = 8973,
3385 ["urcrop"] = 8974,
3386 ["ulcrop"] = 8975,
3387 ["bnot"] = 8976,
3388 ["proflin"] = 8978,
3389 ["profsurf"] = 8979,
3390 ["telrec"] = 8981,
3391 ["target"] = 8982,
3392 ["ulcorn"] = 8988,
3393 ["ulcorner"] = 8988,
3394 ["urcorn"] = 8989,
3395 ["urcorner"] = 8989,
3396 ["dlcorn"] = 8990,
3397 ["llcorner"] = 8990,
3398 ["drcorn"] = 8991,
3399 ["lrcorner"] = 8991,
3400 ["frown"] = 8994,
3401 ["sfrown"] = 8994,
3402 ["smile"] = 8995,
3403 ["ssmile"] = 8995,
3404 ["cylcty"] = 9005,
3405 ["profalar"] = 9006,
3406 ["topbot"] = 9014,
3407 ["ovbar"] = 9021,
3408 ["solbar"] = 9023,
3409 ["angzarr"] = 9084,
3410 ["lmoust"] = 9136,
3411 ["lmoustache"] = 9136,
3412 ["rmoust"] = 9137,
3413 ["rmoustache"] = 9137,
3414 ["tbrk"] = 9140,
3415 ["OverBracket"] = 9140,
3416 ["bbrk"] = 9141,
3417 ["UnderBracket"] = 9141,
3418 ["bbrktbrk"] = 9142,
3419 ["OverParenthesis"] = 9180,
3420 ["UnderParenthesis"] = 9181,
3421 ["OverBrace"] = 9182,
3422 ["UnderBrace"] = 9183,
3423 ["trpezium"] = 9186,
3424 ["elinters"] = 9191,
3425 ["blank"] = 9251,
3426 ["oS"] = 9416,
3427 ["circledS"] = 9416,
3428 ["boxh"] = 9472,
3429 ["HorizontalLine"] = 9472,

```



```

3430 ["boxv"] = 9474,
3431 ["boxdr"] = 9484,
3432 ["boxdl"] = 9488,
3433 ["boxur"] = 9492,
3434 ["boxul"] = 9496,
3435 ["boxvr"] = 9500,
3436 ["boxvl"] = 9508,
3437 ["boxhd"] = 9516,
3438 ["boxhu"] = 9524,
3439 ["boxvh"] = 9532,
3440 ["boxH"] = 9552,
3441 ["boxV"] = 9553,
3442 ["boxdR"] = 9554,
3443 ["boxDr"] = 9555,
3444 ["boxDR"] = 9556,
3445 ["boxdL"] = 9557,
3446 ["boxDL"] = 9558,
3447 ["boxDL"] = 9559,
3448 ["boxuR"] = 9560,
3449 ["boxUr"] = 9561,
3450 ["boxUR"] = 9562,
3451 ["boxuL"] = 9563,
3452 ["boxUL"] = 9564,
3453 ["boxUL"] = 9565,
3454 ["boxvR"] = 9566,
3455 ["boxVr"] = 9567,
3456 ["boxVR"] = 9568,
3457 ["boxvL"] = 9569,
3458 ["boxVL"] = 9570,
3459 ["boxVL"] = 9571,
3460 ["boxHd"] = 9572,
3461 ["boxhD"] = 9573,
3462 ["boxHD"] = 9574,
3463 ["boxHu"] = 9575,
3464 ["boxhU"] = 9576,
3465 ["boxHU"] = 9577,
3466 ["boxvH"] = 9578,
3467 ["boxVh"] = 9579,
3468 ["boxVH"] = 9580,
3469 ["uhblk"] = 9600,
3470 ["lhblk"] = 9604,
3471 ["block"] = 9608,
3472 ["blk14"] = 9617,
3473 ["blk12"] = 9618,
3474 ["blk34"] = 9619,
3475 ["squ"] = 9633,
3476 ["square"] = 9633,

```

```

3477 ["Square"] = 9633,
3478 ["squf"] = 9642,
3479 ["squarf"] = 9642,
3480 ["blacksquare"] = 9642,
3481 ["FilledVerySmallSquare"] = 9642,
3482 ["EmptyVerySmallSquare"] = 9643,
3483 ["rect"] = 9645,
3484 ["marker"] = 9646,
3485 ["fltns"] = 9649,
3486 ["xutri"] = 9651,
3487 ["bigtriangleup"] = 9651,
3488 ["utrif"] = 9652,
3489 ["blacktriangle"] = 9652,
3490 ["utri"] = 9653,
3491 ["triangle"] = 9653,
3492 ["rtrif"] = 9656,
3493 ["blacktriangleright"] = 9656,
3494 ["rtri"] = 9657,
3495 ["triangleright"] = 9657,
3496 ["xdtri"] = 9661,
3497 ["bigtriangledown"] = 9661,
3498 ["dtrif"] = 9662,
3499 ["blacktriangledown"] = 9662,
3500 ["dtri"] = 9663,
3501 ["triangledown"] = 9663,
3502 ["ltrif"] = 9666,
3503 ["blacktriangleleft"] = 9666,
3504 ["ltri"] = 9667,
3505 ["triangleleft"] = 9667,
3506 ["loz"] = 9674,
3507 ["lozenge"] = 9674,
3508 ["cir"] = 9675,
3509 ["tridot"] = 9708,
3510 ["xcirc"] = 9711,
3511 ["bigcirc"] = 9711,
3512 ["ultri"] = 9720,
3513 ["urtri"] = 9721,
3514 ["lltri"] = 9722,
3515 ["EmptySmallSquare"] = 9723,
3516 ["FilledSmallSquare"] = 9724,
3517 ["starf"] = 9733,
3518 ["bigstar"] = 9733,
3519 ["star"] = 9734,
3520 ["phone"] = 9742,
3521 ["female"] = 9792,
3522 ["male"] = 9794,
3523 ["spades"] = 9824,

```

```

3524 ["spadesuit"] = 9824,
3525 ["clubs"] = 9827,
3526 ["clubsuit"] = 9827,
3527 ["hearts"] = 9829,
3528 ["heartsuit"] = 9829,
3529 ["diams"] = 9830,
3530 ["diamondsuit"] = 9830,
3531 ["sung"] = 9834,
3532 ["flat"] = 9837,
3533 ["natur"] = 9838,
3534 ["natural"] = 9838,
3535 ["sharp"] = 9839,
3536 ["check"] = 10003,
3537 ["checkmark"] = 10003,
3538 ["cross"] = 10007,
3539 ["malt"] = 10016,
3540 ["maltese"] = 10016,
3541 ["sext"] = 10038,
3542 ["VerticalSeparator"] = 10072,
3543 ["lbrk"] = 10098,
3544 ["rbrk"] = 10099,
3545 ["lobrk"] = 10214,
3546 ["LeftDoubleBracket"] = 10214,
3547 ["robrk"] = 10215,
3548 ["RightDoubleBracket"] = 10215,
3549 ["lang"] = 10216,
3550 ["LeftAngleBracket"] = 10216,
3551 ["langle"] = 10216,
3552 ["rang"] = 10217,
3553 ["RightAngleBracket"] = 10217,
3554 ["rangle"] = 10217,
3555 ["Lang"] = 10218,
3556 ["Rang"] = 10219,
3557 ["loang"] = 10220,
3558 ["roang"] = 10221,
3559 ["xlarr"] = 10229,
3560 ["longleftarrow"] = 10229,
3561 ["LongLeftArrow"] = 10229,
3562 ["xrarr"] = 10230,
3563 ["longrightarrow"] = 10230,
3564 ["LongRightArrow"] = 10230,
3565 ["xharr"] = 10231,
3566 ["longlefttrightarrow"] = 10231,
3567 ["LongLeftRightArrow"] = 10231,
3568 ["xlArr"] = 10232,
3569 ["Longleftarrow"] = 10232,
3570 ["DoubleLongLeftArrow"] = 10232,

```

```

3571 ["xrArr"] = 10233,
3572 ["Longrightarrow"] = 10233,
3573 ["DoubleLongRightArrow"] = 10233,
3574 ["xhArr"] = 10234,
3575 ["Longlefttrightharpoon"] = 10234,
3576 ["DoubleLongLeftRightArrow"] = 10234,
3577 ["xmap"] = 10236,
3578 ["longmapsto"] = 10236,
3579 ["dzigrarr"] = 10239,
3580 ["nvlArr"] = 10498,
3581 ["nvrArr"] = 10499,
3582 ["nvHarr"] = 10500,
3583 ["Map"] = 10501,
3584 ["lbarr"] = 10508,
3585 ["rbarr"] = 10509,
3586 ["bkarow"] = 10509,
3587 ["lBarr"] = 10510,
3588 ["rBarr"] = 10511,
3589 ["dbkarow"] = 10511,
3590 ["RBarr"] = 10512,
3591 ["drbkarow"] = 10512,
3592 ["DDottrahd"] = 10513,
3593 ["UpArrowBar"] = 10514,
3594 ["DownArrowBar"] = 10515,
3595 ["Rarrtl"] = 10518,
3596 ["latail"] = 10521,
3597 ["ratail"] = 10522,
3598 ["lAtail"] = 10523,
3599 ["rAtail"] = 10524,
3600 ["larrfs"] = 10525,
3601 ["rarrfs"] = 10526,
3602 ["larrbfs"] = 10527,
3603 ["rarrbfs"] = 10528,
3604 ["nwarhk"] = 10531,
3605 ["nearhk"] = 10532,
3606 ["searhk"] = 10533,
3607 ["hksearow"] = 10533,
3608 ["swarhk"] = 10534,
3609 ["hkswarow"] = 10534,
3610 ["nwnear"] = 10535,
3611 ["nesear"] = 10536,
3612 ["toea"] = 10536,
3613 ["seswar"] = 10537,
3614 ["tosa"] = 10537,
3615 ["swnwar"] = 10538,
3616 ["rarrc"] = 10547,
3617 ["cudarrr"] = 10549,

```

```

3618 ["ldca"] = 10550,
3619 ["rdca"] = 10551,
3620 ["cudarrl"] = 10552,
3621 ["larrpl"] = 10553,
3622 ["curarrm"] = 10556,
3623 ["cularrp"] = 10557,
3624 ["rarrpl"] = 10565,
3625 ["harrcir"] = 10568,
3626 ["Uarroccir"] = 10569,
3627 ["lurdshar"] = 10570,
3628 ["ldrushar"] = 10571,
3629 ["LeftRightVector"] = 10574,
3630 ["RightUpDownVector"] = 10575,
3631 ["DownLeftRightVector"] = 10576,
3632 ["LeftUpDownVector"] = 10577,
3633 ["LeftVectorBar"] = 10578,
3634 ["RightVectorBar"] = 10579,
3635 ["RightUpVectorBar"] = 10580,
3636 ["RightDownVectorBar"] = 10581,
3637 ["DownLeftVectorBar"] = 10582,
3638 ["DownRightVectorBar"] = 10583,
3639 ["LeftUpVectorBar"] = 10584,
3640 ["LeftDownVectorBar"] = 10585,
3641 ["LeftTeeVector"] = 10586,
3642 ["RightTeeVector"] = 10587,
3643 ["RightUpTeeVector"] = 10588,
3644 ["RightDownTeeVector"] = 10589,
3645 ["DownLeftTeeVector"] = 10590,
3646 ["DownRightTeeVector"] = 10591,
3647 ["LeftUpTeeVector"] = 10592,
3648 ["LeftDownTeeVector"] = 10593,
3649 ["lHar"] = 10594,
3650 ["uHar"] = 10595,
3651 ["rHar"] = 10596,
3652 ["dHar"] = 10597,
3653 ["luruhar"] = 10598,
3654 ["ldrdhar"] = 10599,
3655 ["ruluhar"] = 10600,
3656 ["rdldhar"] = 10601,
3657 ["lharul"] = 10602,
3658 ["llhard"] = 10603,
3659 ["rharul"] = 10604,
3660 ["lrhard"] = 10605,
3661 ["udhar"] = 10606,
3662 ["UpEquilibrium"] = 10606,
3663 ["duhar"] = 10607,
3664 ["ReverseUpEquilibrium"] = 10607,

```

```

3665 ["RoundImplies"] = 10608,
3666 ["erarr"] = 10609,
3667 ["simrarr"] = 10610,
3668 ["larrsim"] = 10611,
3669 ["rarrsim"] = 10612,
3670 ["rarrap"] = 10613,
3671 ["ltlarr"] = 10614,
3672 ["gtrarr"] = 10616,
3673 ["subrarr"] = 10617,
3674 ["suplarr"] = 10619,
3675 ["lfisht"] = 10620,
3676 ["rfisht"] = 10621,
3677 ["ufisht"] = 10622,
3678 ["dfisht"] = 10623,
3679 ["lopar"] = 10629,
3680 ["ropar"] = 10630,
3681 ["lbrke"] = 10635,
3682 ["rbrke"] = 10636,
3683 ["lbrkslu"] = 10637,
3684 ["rbrksld"] = 10638,
3685 ["lbrksld"] = 10639,
3686 ["rbrkslu"] = 10640,
3687 ["langd"] = 10641,
3688 ["rangd"] = 10642,
3689 ["lparlt"] = 10643,
3690 ["rpargt"] = 10644,
3691 ["gtlPar"] = 10645,
3692 ["ltrPar"] = 10646,
3693 ["vzigzag"] = 10650,
3694 ["vangrt"] = 10652,
3695 ["angrtvbd"] = 10653,
3696 ["ange"] = 10660,
3697 ["range"] = 10661,
3698 ["dwangle"] = 10662,
3699 ["uwangle"] = 10663,
3700 ["angmsdaa"] = 10664,
3701 ["angmsdab"] = 10665,
3702 ["angmsdac"] = 10666,
3703 ["angmsdad"] = 10667,
3704 ["angmsdae"] = 10668,
3705 ["angmsdaf"] = 10669,
3706 ["angmsdag"] = 10670,
3707 ["angmsdah"] = 10671,
3708 ["bemptyv"] = 10672,
3709 ["demptyv"] = 10673,
3710 ["cemptyv"] = 10674,
3711 ["raemptyv"] = 10675,

```

```

3712 ["laemptyv"] = 10676,
3713 ["ohbar"] = 10677,
3714 ["omid"] = 10678,
3715 ["opar"] = 10679,
3716 ["operp"] = 10681,
3717 ["olcross"] = 10683,
3718 ["odsold"] = 10684,
3719 ["olcir"] = 10686,
3720 ["ofcir"] = 10687,
3721 ["olt"] = 10688,
3722 ["ogt"] = 10689,
3723 ["cirscir"] = 10690,
3724 ["cirE"] = 10691,
3725 ["solb"] = 10692,
3726 ["bsolb"] = 10693,
3727 ["boxbox"] = 10697,
3728 ["trish"] = 10701,
3729 ["rtriltri"] = 10702,
3730 ["LeftTriangleBar"] = 10703,
3731 ["RightTriangleBar"] = 10704,
3732 ["race"] = 10714,
3733 ["iinfin"] = 10716,
3734 ["infintie"] = 10717,
3735 ["nvinfin"] = 10718,
3736 ["eparsl"] = 10723,
3737 ["smeparsl"] = 10724,
3738 ["eqvparsl"] = 10725,
3739 ["lozf"] = 10731,
3740 ["blacklozenge"] = 10731,
3741 ["RuleDelayed"] = 10740,
3742 ["dsol"] = 10742,
3743 ["xodot"] = 10752,
3744 ["bigodot"] = 10752,
3745 ["xoplus"] = 10753,
3746 ["bigoplus"] = 10753,
3747 ["xotime"] = 10754,
3748 ["bigotimes"] = 10754,
3749 ["xuplus"] = 10756,
3750 ["biguplus"] = 10756,
3751 ["xsqcup"] = 10758,
3752 ["bigsqcup"] = 10758,
3753 ["qint"] = 10764,
3754 ["iiiint"] = 10764,
3755 ["fpartint"] = 10765,
3756 ["cirfnint"] = 10768,
3757 ["awint"] = 10769,
3758 ["rppolint"] = 10770,

```

```

3759 ["scpolint"] = 10771,
3760 ["npolint"] = 10772,
3761 ["pointint"] = 10773,
3762 ["quatint"] = 10774,
3763 ["intlarhk"] = 10775,
3764 ["pluscir"] = 10786,
3765 ["plusacir"] = 10787,
3766 ["simplus"] = 10788,
3767 ["plusdu"] = 10789,
3768 ["plussim"] = 10790,
3769 ["plustwo"] = 10791,
3770 ["mcomma"] = 10793,
3771 ["minusdu"] = 10794,
3772 ["loplus"] = 10797,
3773 ["roplus"] = 10798,
3774 ["Cross"] = 10799,
3775 ["timesd"] = 10800,
3776 ["timesbar"] = 10801,
3777 ["smashp"] = 10803,
3778 ["lotimes"] = 10804,
3779 ["rotimes"] = 10805,
3780 ["otimesas"] = 10806,
3781 ["Otimes"] = 10807,
3782 ["odiv"] = 10808,
3783 ["triplus"] = 10809,
3784 ["triminus"] = 10810,
3785 ["tritime"] = 10811,
3786 ["iprod"] = 10812,
3787 ["intprod"] = 10812,
3788 ["amalg"] = 10815,
3789 ["capdot"] = 10816,
3790 ["ncup"] = 10818,
3791 ["ncap"] = 10819,
3792 ["capand"] = 10820,
3793 ["cupor"] = 10821,
3794 ["cupcap"] = 10822,
3795 ["capcup"] = 10823,
3796 ["cupbrcap"] = 10824,
3797 ["capbrcup"] = 10825,
3798 ["cupcup"] = 10826,
3799 ["capcap"] = 10827,
3800 ["ccups"] = 10828,
3801 ["ccaps"] = 10829,
3802 ["ccupssm"] = 10832,
3803 ["And"] = 10835,
3804 ["Or"] = 10836,
3805 ["andand"] = 10837,

```



```

3806 ["oror"] = 10838,
3807 ["orslope"] = 10839,
3808 ["andslope"] = 10840,
3809 ["andv"] = 10842,
3810 ["orv"] = 10843,
3811 ["andd"] = 10844,
3812 ["ord"] = 10845,
3813 ["wedbar"] = 10847,
3814 ["sdote"] = 10854,
3815 ["simdot"] = 10858,
3816 ["congdot"] = 10861,
3817 ["easter"] = 10862,
3818 ["apacir"] = 10863,
3819 ["apE"] = 10864,
3820 ["eplus"] = 10865,
3821 ["pluse"] = 10866,
3822 ["Esim"] = 10867,
3823 ["Colone"] = 10868,
3824 ["Equal"] = 10869,
3825 ["eDDot"] = 10871,
3826 ["ddotseq"] = 10871,
3827 ["equivDD"] = 10872,
3828 ["ltcir"] = 10873,
3829 ["gtcir"] = 10874,
3830 ["ltquest"] = 10875,
3831 ["gtquest"] = 10876,
3832 ["les"] = 10877,
3833 ["LessSlantEqual"] = 10877,
3834 ["leqslant"] = 10877,
3835 ["ges"] = 10878,
3836 ["GreaterSlantEqual"] = 10878,
3837 ["geqslant"] = 10878,
3838 ["lesdot"] = 10879,
3839 ["gesdot"] = 10880,
3840 ["lesdoto"] = 10881,
3841 ["gesdoto"] = 10882,
3842 ["lesdotor"] = 10883,
3843 ["gesdoto1"] = 10884,
3844 ["lap"] = 10885,
3845 ["lessapprox"] = 10885,
3846 ["gap"] = 10886,
3847 ["gtrapprox"] = 10886,
3848 ["lne"] = 10887,
3849 ["lneq"] = 10887,
3850 ["gne"] = 10888,
3851 ["gneq"] = 10888,
3852 ["lnap"] = 10889,

```

```

3853 ["lnapprox"] = 10889,
3854 ["gnap"] = 10890,
3855 ["gnapprox"] = 10890,
3856 ["lEg"] = 10891,
3857 ["lesseqqgtr"] = 10891,
3858 ["gEl"] = 10892,
3859 ["gtreqqless"] = 10892,
3860 ["lsime"] = 10893,
3861 ["gsime"] = 10894,
3862 ["lsimg"] = 10895,
3863 ["gsiml"] = 10896,
3864 ["lgE"] = 10897,
3865 ["glE"] = 10898,
3866 ["lesges"] = 10899,
3867 ["gesles"] = 10900,
3868 ["els"] = 10901,
3869 ["eqslantless"] = 10901,
3870 ["egs"] = 10902,
3871 ["eqslantgtr"] = 10902,
3872 ["elsdot"] = 10903,
3873 ["egsdot"] = 10904,
3874 ["el"] = 10905,
3875 ["eg"] = 10906,
3876 ["siml"] = 10909,
3877 ["sing"] = 10910,
3878 ["simlE"] = 10911,
3879 ["singE"] = 10912,
3880 ["LessLess"] = 10913,
3881 ["GreaterGreater"] = 10914,
3882 ["glj"] = 10916,
3883 ["gla"] = 10917,
3884 ["ltcc"] = 10918,
3885 ["gtcc"] = 10919,
3886 ["lescc"] = 10920,
3887 ["gescc"] = 10921,
3888 ["smt"] = 10922,
3889 ["lat"] = 10923,
3890 ["smte"] = 10924,
3891 ["late"] = 10925,
3892 ["bumpE"] = 10926,
3893 ["pre"] = 10927,
3894 ["preceq"] = 10927,
3895 ["PrecedesEqual"] = 10927,
3896 ["sce"] = 10928,
3897 ["succeq"] = 10928,
3898 ["SucceedsEqual"] = 10928,
3899 ["prE"] = 10931,

```

```

3900 ["scE"] = 10932,
3901 ["prnE"] = 10933,
3902 ["precneqq"] = 10933,
3903 ["scnE"] = 10934,
3904 ["succneqq"] = 10934,
3905 ["prap"] = 10935,
3906 ["precapprox"] = 10935,
3907 ["scap"] = 10936,
3908 ["succapprox"] = 10936,
3909 ["prnap"] = 10937,
3910 ["precnapprox"] = 10937,
3911 ["scnap"] = 10938,
3912 ["succnapprox"] = 10938,
3913 ["Pr"] = 10939,
3914 ["Sc"] = 10940,
3915 ["subdot"] = 10941,
3916 ["supdot"] = 10942,
3917 ["subplus"] = 10943,
3918 ["supplus"] = 10944,
3919 ["submult"] = 10945,
3920 ["supmult"] = 10946,
3921 ["subedot"] = 10947,
3922 ["supedot"] = 10948,
3923 ["subE"] = 10949,
3924 ["subseteqq"] = 10949,
3925 ["supE"] = 10950,
3926 ["supseteqq"] = 10950,
3927 ["subsim"] = 10951,
3928 ["supsim"] = 10952,
3929 ["subnE"] = 10955,
3930 ["subsetneqq"] = 10955,
3931 ["supnE"] = 10956,
3932 ["supsetneqq"] = 10956,
3933 ["csub"] = 10959,
3934 ["csup"] = 10960,
3935 ["csube"] = 10961,
3936 ["csupe"] = 10962,
3937 ["subsup"] = 10963,
3938 ["supsub"] = 10964,
3939 ["subsub"] = 10965,
3940 ["supsup"] = 10966,
3941 ["suphsub"] = 10967,
3942 ["supdsub"] = 10968,
3943 ["forkv"] = 10969,
3944 ["topfork"] = 10970,
3945 ["mlcp"] = 10971,
3946 ["Dashv"] = 10980,

```

```

3947 ["DoubleLeftTee"] = 10980,
3948 ["Vdashl"] = 10982,
3949 ["Barv"] = 10983,
3950 ["vBar"] = 10984,
3951 ["vBarv"] = 10985,
3952 ["Vbar"] = 10987,
3953 ["Not"] = 10988,
3954 ["bNot"] = 10989,
3955 ["rnmid"] = 10990,
3956 ["cirmid"] = 10991,
3957 ["midcir"] = 10992,
3958 ["topcir"] = 10993,
3959 ["nhpar"] = 10994,
3960 ["parsim"] = 10995,
3961 ["parsl"] = 11005,
3962 ["fflig"] = 64256,
3963 ["filig"] = 64257,
3964 ["fllig"] = 64258,
3965 ["ffilig"] = 64259,
3966 ["ffllig"] = 64260,
3967 ["Ascr"] = 119964,
3968 ["Cscr"] = 119966,
3969 ["Dscr"] = 119967,
3970 ["Gscr"] = 119970,
3971 ["Jscr"] = 119973,
3972 ["Kscr"] = 119974,
3973 ["Nscr"] = 119977,
3974 ["Oscr"] = 119978,
3975 ["Pscr"] = 119979,
3976 ["Qscr"] = 119980,
3977 ["Sscr"] = 119982,
3978 ["Tscr"] = 119983,
3979 ["Uscr"] = 119984,
3980 ["Vscr"] = 119985,
3981 ["Wscr"] = 119986,
3982 ["Xscr"] = 119987,
3983 ["Yscr"] = 119988,
3984 ["Zscr"] = 119989,
3985 ["ascr"] = 119990,
3986 ["bscr"] = 119991,
3987 ["cscr"] = 119992,
3988 ["dscr"] = 119993,
3989 ["fscr"] = 119995,
3990 ["hscr"] = 119997,
3991 ["iscr"] = 119998,
3992 ["jscr"] = 119999,
3993 ["kscr"] = 120000,

```

```

3994 ["lscr"] = 120001,
3995 ["mscr"] = 120002,
3996 ["nscr"] = 120003,
3997 ["pscr"] = 120005,
3998 ["qscr"] = 120006,
3999 ["rscr"] = 120007,
4000 ["sscr"] = 120008,
4001 ["tscr"] = 120009,
4002 ["uscr"] = 120010,
4003 ["vscr"] = 120011,
4004 ["wscr"] = 120012,
4005 ["xscr"] = 120013,
4006 ["yscr"] = 120014,
4007 ["zscr"] = 120015,
4008 ["Afr"] = 120068,
4009 ["Bfr"] = 120069,
4010 ["Dfr"] = 120071,
4011 ["Efr"] = 120072,
4012 ["Ffr"] = 120073,
4013 ["Gfr"] = 120074,
4014 ["Jfr"] = 120077,
4015 ["Kfr"] = 120078,
4016 ["Lfr"] = 120079,
4017 ["Mfr"] = 120080,
4018 ["Nfr"] = 120081,
4019 ["Ofr"] = 120082,
4020 ["Pfr"] = 120083,
4021 ["Qfr"] = 120084,
4022 ["Sfr"] = 120086,
4023 ["Tfr"] = 120087,
4024 ["Ufr"] = 120088,
4025 ["Vfr"] = 120089,
4026 ["Wfr"] = 120090,
4027 ["Xfr"] = 120091,
4028 ["Yfr"] = 120092,
4029 ["afr"] = 120094,
4030 ["bfr"] = 120095,
4031 ["cfr"] = 120096,
4032 ["dfr"] = 120097,
4033 ["efr"] = 120098,
4034 ["ffr"] = 120099,
4035 ["gfr"] = 120100,
4036 ["hfr"] = 120101,
4037 ["ifr"] = 120102,
4038 ["jfr"] = 120103,
4039 ["kfr"] = 120104,
4040 ["lfr"] = 120105,

```

```

4041 ["mfr"] = 120106,
4042 ["nfr"] = 120107,
4043 ["ofr"] = 120108,
4044 ["pfr"] = 120109,
4045 ["qfr"] = 120110,
4046 ["rfr"] = 120111,
4047 ["sfr"] = 120112,
4048 ["tfr"] = 120113,
4049 ["ufr"] = 120114,
4050 ["vfr"] = 120115,
4051 ["wfr"] = 120116,
4052 ["xfr"] = 120117,
4053 ["yfr"] = 120118,
4054 ["zfr"] = 120119,
4055 ["Aopf"] = 120120,
4056 ["Bopf"] = 120121,
4057 ["Dopf"] = 120123,
4058 ["Eopf"] = 120124,
4059 ["Fopf"] = 120125,
4060 ["Gopf"] = 120126,
4061 ["Iopf"] = 120128,
4062 ["Jopf"] = 120129,
4063 ["Kopf"] = 120130,
4064 ["Lopf"] = 120131,
4065 ["Mopf"] = 120132,
4066 ["Oopf"] = 120134,
4067 ["Sopf"] = 120138,
4068 ["Topf"] = 120139,
4069 ["Uopf"] = 120140,
4070 ["Vopf"] = 120141,
4071 ["Wopf"] = 120142,
4072 ["Xopf"] = 120143,
4073 ["Yopf"] = 120144,
4074 ["aopf"] = 120146,
4075 ["bopf"] = 120147,
4076 ["copf"] = 120148,
4077 ["dopf"] = 120149,
4078 ["eopf"] = 120150,
4079 ["fopf"] = 120151,
4080 ["gopf"] = 120152,
4081 ["hopf"] = 120153,
4082 ["iopf"] = 120154,
4083 ["jopf"] = 120155,
4084 ["kopf"] = 120156,
4085 ["lopf"] = 120157,
4086 ["mopf"] = 120158,
4087 ["nopf"] = 120159,

```

```

4088  ["oopf"] = 120160,
4089  ["popf"] = 120161,
4090  ["qopf"] = 120162,
4091  ["ropf"] = 120163,
4092  ["sopf"] = 120164,
4093  ["topf"] = 120165,
4094  ["uopf"] = 120166,
4095  ["vopf"] = 120167,
4096  ["wopf"] = 120168,
4097  ["xopf"] = 120169,
4098  ["yopf"] = 120170,
4099  ["zopf"] = 120171,
4100 }

```

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

```

4101 function entities.dec_entity(s)
4102     return unicode.utf8.char(tonumber(s))
4103 end

```

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

```

4104 function entities.hex_entity(s)
4105     return unicode.utf8.char(tonumber("0x"..s))
4106 end

```

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

```

4107 function entities.char_entity(s)
4108     local n = character_entities[s]
4109     if n == nil then
4110         return "&" .. s .. ";"
4111     end
4112     return unicode.utf8.char(n)
4113 end

```

3.1.3 Plain T_EX Writer

This section documents the `writer` object, which implements the routines for producing the T_EX output. The object is an amalgamate of the generic, T_EX, L^AT_EX writer objects that were located in the `lunamark/writer/generic.lua`, `lunamark/writer/tex.lua`, and `lunamark/writer/latex.lua` files in the Luna-mark 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.

```
4114 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.

```
4115 function M.writer.new(options)
4116   local self = {}
```

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

```
4117   self.cacheDir = options.cacheDir
```

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

```
4118   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.

```
4119   local slice_specifiers = {}
4120   for specifier in options.slice:gmatch("[^%s]+") do
4121     table.insert(slice_specifiers, specifier)
4122   end
4123
4124   if #slice_specifiers == 2 then
4125     self.slice_begin, self.slice_end = table.unpack(slice_specifiers)
4126     local slice_begin_type = self.slice_begin:sub(1, 1)
4127     if slice_begin_type ~= "^" and slice_begin_type ~= "$" then
4128       self.slice_begin = "^" .. self.slice_begin
4129     end
4130     local slice_end_type = self.slice_end:sub(1, 1)
4131     if slice_end_type ~= "^" and slice_end_type ~= "$" then
4132       self.slice_end = "$" .. self.slice_end
4133     end
4134   elseif #slice_specifiers == 1 then
4135     self.slice_begin = "^" .. slice_specifiers[1]
4136     self.slice_end = "$" .. slice_specifiers[1]
4137   end
4138
4139   if self.slice_begin == "^" and self.slice_end ~= "^" then
4140     self.is_writing = true
4141   else
4142     self.is_writing = false
4143   end
```


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

```
4144 self.suffix = ".tex"
```

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

```
4145 self.space = " "
```

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

```
4146 self.nbsp = "\\markdownRendererNbsp{}"
```

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

```
4147 function self.plain(s)
4148   return s
4149 end
```

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

```
4150 function self.paragraph(s)
4151   if not self.is_writing then return "" end
4152   return s
4153 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.

```
4154 function self.pack(name)
4155   return "[\\input ]] .. name .. [[\\relax]]
4156 end
```

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

```
4157 function self.interblocksep()
4158   if not self.is_writing then return "" end
4159   return "\\markdownRendererInterblockSeparator\\n{}"
4160 end
```

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

```
4161 self.linebreak = "\\markdownRendererLineBreak\\n{}"
```

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

```
4162 self.ellipsis = "\\markdownRendererEllipsis{}"
```

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

```
4163 function self.hrule()
4164   if not self.is_writing then return "" end
4165   return "\\markdownRendererHorizontalRule{}"
4166 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.

```
4167 self.escaped_uri_chars = {
```

```

4168     ["{"] = "\\markdownRendererLeftBrace{}",
4169     ["}"] = "\\markdownRendererRightBrace{}",
4170     ["%"] = "\\markdownRendererPercentSign{}",
4171     ["\\"] = "\\markdownRendererBackslash{}",
4172   }
4173   self.escaped_minimal_strings = {
4174     ["^^"] = "\\markdownRendererCircumflex\\markdownRendererCircumflex ",
4175     ["☒"] = "\\markdownRendererTickedBox{}",
4176     ["☐"] = "\\markdownRendererHalfTickedBox{}",
4177     ["□"] = "\\markdownRendererUntickedBox{}",
4178   }

```

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.

```

4179   self.escaped_chars = {
4180     ["{"] = "\\markdownRendererLeftBrace{}",
4181     ["}"] = "\\markdownRendererRightBrace{}",
4182     ["%"] = "\\markdownRendererPercentSign{}",
4183     ["\\"] = "\\markdownRendererBackslash{}",
4184     ["#"] = "\\markdownRendererHash{}",
4185     ["$"] = "\\markdownRendererDollarSign{}",
4186     ["&"] = "\\markdownRendererAmpersand{}",
4187     ["_"] = "\\markdownRendererUnderscore{}",
4188     ["^"] = "\\markdownRendererCircumflex{}",
4189     ["~"] = "\\markdownRendererTilde{}",
4190     ["|"] = "\\markdownRendererPipe{}",
4191   }

```

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

```

4192   self.escape = util.escaper(self.escaped_chars, self.escaped_minimal_strings)
4193   self.escape_uri = util.escaper(self.escaped_uri_chars, self.escaped_minimal_strings)
4194   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.

```

4195   if options.hybrid then
4196     self.string = self.escape_minimal
4197     self.uri = self.escape_minimal
4198   else
4199     self.string = self.escape
4200     self.uri = self.escape_uri

```

4201 end

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

```
4202 function self.code(s)
4203     return {"\\markdownRendererCodeSpan{",self.escape(s),"}"}
4204 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.

```
4205 function self.link(lab,src,tit)
4206     return {"\\markdownRendererLink{",lab,"}",
4207             "{",self.escape(src),"}",
4208             "{",self.uri(src),"}",
4209             "{",self.string(tit or ""),"}"}
4210 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.

```
4211 function self.image(lab,src,tit)
4212     return {"\\markdownRendererImage{",lab,"}",
4213             "{",self.string(src),"}",
4214             "{",self.uri(src),"}",
4215             "{",self.string(tit or ""),"}"}
4216 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.

```
4217 local function ulitem(s)
4218     return {"\\markdownRendererULItem ",s,
4219             "\\markdownRendererULItemEnd "}
4220 end
4221
4222 function self.bulletlist(items,tight)
4223     if not self.is_writing then return "" end
4224     local buffer = {}
4225     for _,item in ipairs(items) do
4226         buffer[#buffer + 1] = ulitem(item)
4227     end
4228     local contents = util.intersperse(buffer,"\n")
4229     if tight and options.tightLists then
4230         return {"\\markdownRendererULBeginTight\n",contents,
4231                 "\n\\markdownRendererULEndTight "}
4232     else
4233         return {"\\markdownRendererULBegin\n",contents,
```

```

4234         "\n\\markdownRendererUlEnd "}
4235     end
4236 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.

```

4237 local function olitem(s,num)
4238     if num ~= nil then
4239         return {"\\markdownRendererOlItemWithNumber{" ,num,"} ",s,
4240             "\\markdownRendererOlItemEnd "}
4241     else
4242         return {"\\markdownRendererOlItem ",s,
4243             "\\markdownRendererOlItemEnd "}
4244     end
4245 end
4246
4247 function self.orderedlist(items,tight,startnum)
4248     if not self.is_writing then return "" end
4249     local buffer = {}
4250     local num = startnum
4251     for _,item in ipairs(items) do
4252         buffer[#buffer + 1] = olitem(item,num)
4253         if num ~= nil then
4254             num = num + 1
4255         end
4256     end
4257     local contents = util.intersperse(buffer,"\n")
4258     if tight and options.tightLists then
4259         return {"\\markdownRendererOlBeginTight\n",contents,
4260             "\n\\markdownRendererOlEndTight "}
4261     else
4262         return {"\\markdownRendererOlBegin\n",contents,
4263             "\n\\markdownRendererOlEnd "}
4264     end
4265 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.

```

4266 function self.inline_html_comment(contents)
4267     return {"\\markdownRendererInlineHtmlComment{" ,contents,""}
4268 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.

```

4269 function self.block_html_comment(contents)
4270     if not self.is_writing then return "" end
4271     return {"\\markdownRendererBlockHtmlCommentBegin\\n",contents,
4272           "\\n\\markdownRendererBlockHtmlCommentEnd "}
4273 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.

```

4274 function self.inline_html_tag(contents)
4275     return {"\\markdownRendererInlineHtmlTag{" ,self.string(contents),"}"}
4276 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.

```

4277 function self.block_html_element(s)
4278     if not self.is_writing then return "" end
4279     local name = util.cache(self.cacheDir, s, nil, nil, ".verbatim")
4280     return {"\\markdownRendererInputBlockHtmlElement{" ,name,""}
4281 end

```

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

```

4282 function self.emphasis(s)
4283     return {"\\markdownRendererEmphasis{" ,s,""}
4284 end

```

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

```

4285 function self.checkbox(f)
4286     if f == 1.0 then
4287         return "☑ "
4288     elseif f == 0.0 then
4289         return "☐ "
4290     else
4291         return "☐ "
4292     end
4293 end

```

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

```

4294 function self.strong(s)
4295     return {"\\markdownRendererStrongEmphasis{" ,s,""}
4296 end

```

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

```

4297 function self.blockquote(s)
4298   if #util.ropetostring(s) == 0 then return "" end
4299   return {"\\markdownRendererBlockQuoteBegin\\n",s,
4300     "\\n\\markdownRendererBlockQuoteEnd "}
4301 end

```

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

```

4302 function self.verbatim(s)
4303   if not self.is_writing then return "" end
4304   s = string.gsub(s, '[\\r\\n%s]*$', '')
4305   local name = util.cache(self.cacheDir, s, nil, nil, ".verbatim")
4306   return {"\\markdownRendererInputVerbatim{" ,name,"}"}
4307 end

```

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

```

4308 function self.document(d)
4309   local active_attributes = self.active_attributes
4310   local buf = {"\\markdownRendererDocumentBegin\\n", d}
4311
4312   -- pop attributes for sections that have ended
4313   if options.headerAttributes and self.is_writing then
4314     while #active_attributes > 0 do
4315       local attributes = active_attributes[#active_attributes]
4316       if #attributes > 0 then
4317         table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
4318       end
4319       table.remove(active_attributes, #active_attributes)
4320     end
4321   end
4322
4323   table.insert(buf, "\\markdownRendererDocumentEnd")
4324
4325   return buf
4326 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.

```

4327 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.

```

4328 function self.heading(s, level, attributes)
4329   attributes = attributes or {}
4330   for i = 1, #attributes do
4331     attributes[attributes[i]] = true
4332   end

```

```

4333
4334     local active_attributes = self.active_attributes
4335     local slice_begin_type = self.slice_begin:sub(1, 1)
4336     local slice_begin_identifier = self.slice_begin:sub(2) or ""
4337     local slice_end_type = self.slice_end:sub(1, 1)
4338     local slice_end_identifier = self.slice_end:sub(2) or ""
4339
4340     local buf = {}
4341
4342     -- push empty attributes for implied sections
4343     while #active_attributes < level-1 do
4344         table.insert(active_attributes, {})
4345     end
4346
4347     -- pop attributes for sections that have ended
4348     while #active_attributes >= level do
4349         local active_identifiers = active_attributes[#active_attributes]
4350         -- tear down all active attributes at slice end
4351         if active_identifiers["#" .. slice_end_identifier] ~= nil
4352             and slice_end_type == "$" then
4353             for header_level = #active_attributes, 1, -1 do
4354                 if options.headerAttributes and #active_attributes[header_level] > 0 then
4355                     table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
4356                 end
4357             end
4358             self.is_writing = false
4359         end
4360         table.remove(active_attributes, #active_attributes)
4361         if self.is_writing and options.headerAttributes and #active_identifiers > 0 then
4362             table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
4363         end
4364         -- apply all active attributes at slice beginning
4365         if active_identifiers["#" .. slice_begin_identifier] ~= nil
4366             and slice_begin_type == "$" then
4367             for header_level = 1, #active_attributes do
4368                 if options.headerAttributes and #active_attributes[header_level] > 0 then
4369                     table.insert(buf, "\\markdownRendererHeaderAttributeContextBegin")
4370                 end
4371             end
4372             self.is_writing = true
4373         end
4374     end
4375
4376     -- tear down all active attributes at slice end
4377     if attributes["#" .. slice_end_identifier] ~= nil
4378         and slice_end_type == "^" then
4379         for header_level = #active_attributes, 1, -1 do

```

```

4380         if options.headerAttributes and #active_attributes[header_level] > 0 then
4381             table.insert(buf, "\\markdownRendererHeaderAttributeContextEnd")
4382         end
4383     end
4384     self.is_writing = false
4385 end
4386
4387 -- push attributes for the new section
4388 table.insert(active_attributes, attributes)
4389 if self.is_writing and options.headerAttributes and #attributes > 0 then
4390     table.insert(buf, "\\markdownRendererHeaderAttributeContextBegin")
4391 end
4392
4393 -- apply all active attributes at slice beginning
4394 if attributes["#" .. slice_begin_identfier] ~= nil
4395     and slice_begin_type == "^" then
4396     for header_level = 1, #active_attributes do
4397         if options.headerAttributes and #active_attributes[header_level] > 0 then
4398             table.insert(buf, "\\markdownRendererHeaderAttributeContextBegin")
4399         end
4400     end
4401     self.is_writing = true
4402 end
4403
4404 if self.is_writing then
4405     table.sort(attributes)
4406     local key, value
4407     for i = 1, #attributes do
4408         if attributes[i]:sub(1, 1) == "#" then
4409             table.insert(buf, {"\\markdownRendererAttributeIdentifier{",
4410                             attributes[i]:sub(2), "}"}))
4411         elseif attributes[i]:sub(1, 1) == "." then
4412             table.insert(buf, {"\\markdownRendererAttributeClassName{",
4413                             attributes[i]:sub(2), "}"}))
4414         else
4415             key, value = attributes[i]:match("(%w+)=(%w+)")
4416             table.insert(buf, {"\\markdownRendererAttributeKeyValue{",
4417                             key, "{", value, "}"}))
4418         end
4419     end
4420 end
4421
4422 local cmd
4423 level = level + options.shiftHeadings
4424 if level <= 1 then
4425     cmd = "\\markdownRendererHeadingOne"
4426 elseif level == 2 then

```



```

4427     cmd = "\\markdownRendererHeadingTwo"
4428 elseif level == 3 then
4429     cmd = "\\markdownRendererHeadingThree"
4430 elseif level == 4 then
4431     cmd = "\\markdownRendererHeadingFour"
4432 elseif level == 5 then
4433     cmd = "\\markdownRendererHeadingFive"
4434 elseif level >= 6 then
4435     cmd = "\\markdownRendererHeadingSix"
4436 else
4437     cmd = ""
4438 end
4439 if self.is_writing then
4440     table.insert(buf, {cmd, "{", s, "}"})
4441 end
4442
4443 return buf
4444 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.

```

4445 function self.get_state()
4446     return {
4447         is_writing=self.is_writing,
4448         active_attributes={table.unpack(self.active_attributes)},
4449     }
4450 end

```

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

```

4451 function self.set_state(s)
4452     local previous_state = self.get_state()
4453     for key, value in pairs(s) do
4454         self[key] = value
4455     end
4456     return previous_state
4457 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`.

```

4458 function self.defer_call(f)
4459     local previous_state = self.get_state()
4460     return function(...)
4461         local state = self.set_state(previous_state)
4462         local return_value = f(...)
4463         self.set_state(state)
4464         return return_value

```

```

4465     end
4466 end
4467
4468 return self
4469 end

```

3.1.4 Parsers

The `parsers` hash table stores PEG patterns that are static and can be reused between different `reader` objects.

```

4470 local parsers = {}

```

3.1.4.1 Basic Parsers

```

4471 parsers.percent = P("%")
4472 parsers.at = P("@")
4473 parsers.comma = P(",")
4474 parsers.asterisk = P("*")
4475 parsers.dash = P("-")
4476 parsers.plus = P("+")
4477 parsers.underscore = P("_")
4478 parsers.period = P(".")
4479 parsers.hash = P("#")
4480 parsers.ampersand = P("&")
4481 parsers.backtick = P("`")
4482 parsers.less = P("<")
4483 parsers.more = P(">")
4484 parsers.space = P(" ")
4485 parsers.squote = P("'")
4486 parsers.dquote = P('"')
4487 parsers.lparent = P("(")
4488 parsers.rparent = P(")")
4489 parsers.lbracket = P("[")
4490 parsers.rbracket = P("]")
4491 parsers.lbrace = P("{")
4492 parsers.rbrace = P("}")
4493 parsers.circumflex = P("^")
4494 parsers.slash = P("/")
4495 parsers.equal = P("=")
4496 parsers.colon = P(":")
4497 parsers.semicolon = P(";")
4498 parsers.exclamation = P("!")
4499 parsers.pipe = P("|")
4500 parsers.tilde = P("~")
4501 parsers.backslash = P("\\")
4502 parsers.tab = P("\t")
4503 parsers.newline = P("\n")

```

```

4504 parsers.tightblocksep      = P("\001")
4505
4506 parsers.digit                = R("09")
4507 parsers.hexdigit             = R("09","af","AF")
4508 parsers.letter               = R("AZ","az")
4509 parsers.alphanumeric         = R("AZ","az","09")
4510 parsers.keyword               = parsers.letter
4511                             * parsers.alphanumeric^0
4512 parsers.internal_punctuation = S(":;,?.")
4513
4514 parsers.doubleasterisks      = P("**")
4515 parsers.doubleunderscores    = P("__")
4516 parsers.fourspace           = P("    ")
4517
4518 parsers.any                  = P(1)
4519 parsers.fail                  = parsers.any - 1
4520
4521 parsers.escapable            = S("\\`*_{}[]()+.!<>#-~:~^@;")
4522 parsers.anyescaped           = parsers.backslash / "\"" * parsers.escapable
4523 + parsers.any
4524
4525 parsers.spacechar            = S("\t ")
4526 parsers.spacing              = S(" \n\r\t")
4527 parsers.nonspacechar         = parsers.any - parsers.spacing
4528 parsers.optionalspace        = parsers.spacechar^0
4529
4530 parsers.specialchar          = S("*_`&[]<!\. @-~")
4531
4532 parsers.normalchar           = parsers.any - (parsers.specialchar
4533                                             + parsers.spacing
4534                                             + parsers.tightblocksep)
4535
4536 parsers.eof                   = -parsers.any
4537 parsers.nonindentpace        = parsers.space^3 * - parsers.spacechar
4538 parsers.indent                = parsers.space^3 * parsers.tab
4539 parsers.linechar              = parsers.fourspace / "\""
4540
4541 parsers.blankline            = P(1 - parsers.newline)
4542
4543 parsers.blanklines           = parsers.optionalspace
4544 parsers.skipblanklines        = parsers.newline / "\n"
4545 parsers.indentedline          = parsers.blankline^0
4546                             = (parsers.optionalspace * parsers.newline)^0
4547                             + parsers.indent / "\""
4548                             * C(parsers.linechar^1 * parsers.newline^-
1)
4547 parsers.optionallyindentedline = parsers.indent^-1 / "\""
4548                             * C(parsers.linechar^1 * parsers.newline^-
1)

```

```

4549 parsers.sp                = parsers.spacing^0
4550 parsers.spnl               = parsers.optionalspace
4551                             * (parsers.newline * parsers.optionalspace)^-
1
4552 parsers.line                = parsers.linechar^0 * parsers.newline
4553 parsers.nonemptyline        = parsers.line - parsers.blankline

```

The `parsers.commented_line^1` parser recognizes the regular language of T_EX comments, see an equivalent finite automaton in Figure 6.

```

4554 parsers.commented_line_letter = parsers.linechar
4555                               + parsers.newline
4556                               - parsers.backslash
4557                               - parsers.percent
4558 parsers.commented_line        = Cg(Cc(""), "backslashes")
4559                               * ((#(parsers.commented_line_letter
4560                                   - parsers.newline)
4561                                   * Cb("backslashes")
4562                                   * Cs(parsers.commented_line_letter
4563                                       - parsers.newline)^1 -- initial
4564                                   * Cg(Cc(""), "backslashes")))
4565                               + #(parsers.backslash * parsers.backslash)
4566                               * Cg((parsers.backslash -- even backslash
4567                                   * parsers.backslash)^1, "backslashes")
4568                               + (parsers.backslash
4569                                   * (#parsers.percent
4570                                       * Cb("backslashes")
4571                                       / function(backslashes)
4572                                           return string.rep("\\", #backslashes / 2)
4573                                       end
4574                                       * C(parsers.percent)
4575                                       + #parsers.commented_line_letter
4576                                       * Cb("backslashes")
4577                                       * Cc("\\")
4578                                       * C(parsers.commented_line_letter))
4579                                   * Cg(Cc(""), "backslashes")))^0
4580                               * (#parsers.percent
4581                                   * Cb("backslashes")
4582                                   / function(backslashes)
4583                                       return string.rep("\\", #backslashes / 2)
4584                                       end
4585                                   * ((parsers.percent -- comment
4586                                       * parsers.line
4587                                       * #parsers.blankline) -- blank line
4588                                       / "\\n"
4589                                       + parsers.percent -- comment
4590                                       * parsers.line
4591                                       * parsers.optionalspace) -- leading tabs and spaces

```



```

4592             + #(parsers.newline)
4593             * Cb("backslashes")
4594             * C(parsers.newline))
4595
4596 parsers.chunk = parsers.line * (parsers.optionallyindentedline
4597                               - parsers.blankline)^0
4598
4599 parsers.css_identifier_char = R("AZ", "az", "09") + S("-_")
4600 parsers.css_identifier = (parsers.hash + parsers.period)
4601                        * (((parsers.css_identifier_char
4602                          - parsers.dash - parsers.digit)
4603                          * parsers.css_identifier_char^1)
4604                          + (parsers.dash
4605                            * (parsers.css_identifier_char
4606                              - parsers.digit)
4607                              * parsers.css_identifier_char^0))
4608 parsers.attribute_key_char = parsers.any - parsers.space
4609                          - parsers.squote - parsers.dquote
4610                          - parsers.more - parsers.slash
4611                          - parsers.equal
4612 parsers.attribute_value_char = parsers.any - parsers.space
4613                          - parsers.dquote - parsers.more
4614
4615 -- block followed by 0 or more optionally
4616 -- indented blocks with first line indented.
4617 parsers.indented_blocks = function(bl)
4618   return Cs( bl
4619             * (parsers.blankline^1 * parsers.indent * -parsers.blankline * bl)^0
4620             * (parsers.blankline^1 + parsers.eof) )
4621 end

```

3.1.4.2 Parsers Used for Markdown Lists

```

4622 parsers.bulletchar = C(parsers.plus + parsers.asterisk + parsers.dash)
4623
4624 parsers.bullet = ( parsers.bulletchar * #parsers.spacing
4625                  * (parsers.tab + parsers.space^-3)
4626                  + parsers.space * parsers.bulletchar * #parsers.spacing
4627                  * (parsers.tab + parsers.space^-2)
4628                  + parsers.space * parsers.space * parsers.bulletchar
4629                  * #parsers.spacing
4630                  * (parsers.tab + parsers.space^-1)
4631                  + parsers.space * parsers.space * parsers.space
4632                  * parsers.bulletchar * #parsers.spacing
4633                  )
4634
4635 local function tickbox(interior)

```

```

4636 return parsers.optionalspace * parsers.lbracket
4637       * interior * parsers.rbracket * parsers.spacechar^1
4638 end
4639
4640 parsers.ticked_box = tickbox(S("xX")) * Cc(1.0)
4641 parsers.halfticked_box = tickbox(S("./")) * Cc(0.5)
4642 parsers.unticked_box = tickbox(parsers.spacechar^1) * Cc(0.0)
4643

```

3.1.4.3 Parsers Used for Markdown Code Spans

```

4644 parsers.openticks = Cg(parsers.backtick^1, "ticks")
4645
4646 local function captures_equal_length(s,i,a,b)
4647   return #a == #b and i
4648 end
4649
4650 parsers.closeticks = parsers.space^-1
4651                   * Cmt(C(parsers.backtick^1)
4652                       * Cb("ticks"), captures_equal_length)
4653
4654 parsers.intickschar = (parsers.any - S(" \n\r`"))
4655                   + (parsers.newline * -parsers.blankline)
4656                   + (parsers.space - parsers.closeticks)
4657                   + (parsers.backtick^1 - parsers.closeticks)
4658
4659 parsers.inticks = parsers.openticks * parsers.space^-1
4660                 * C(parsers.intickschar^0) * parsers.closeticks

```

3.1.4.4 Parsers Used for Fenced Code Blocks

```

4661 local function captures_geq_length(s,i,a,b)
4662   return #a >= #b and i
4663 end
4664
4665 parsers.infostring = (parsers.linechar - (parsers.backtick
4666                                     + parsers.space^1 * (parsers.newline + parsers.eof)))^0
4667
4668 local fenceindent
4669 parsers.fencehead = function(char)
4670   return C(parsers.nonindentospace) / function(s) fenceindent = #s end
4671   * Cg(char^3, "fencelength")
4672   * parsers.optionalspace * C(parsers.infostring)
4673   * parsers.optionalspace * (parsers.newline + parsers.eof)
4674 end
4675
4676 parsers.fencetail = function(char)
4677   return parsers.nonindentospace

```

```

4678             * Cmt(C(char^3) * Cb("fencelength"), captures_geq_length)
4679             * parsers.optionalspace * (parsers.newline + parsers.eof)
4680             + parsers.eof
4681         end
4682
4683     parsers.fencedline = function(char)
4684         return      C(parsers.line - parsers.fencetail(char))
4685         / function(s)
4686             i = 1
4687             remaining = fenceindent
4688             while true do
4689                 c = s:sub(i, i)
4690                 if c == " " and remaining > 0 then
4691                     remaining = remaining - 1
4692                     i = i + 1
4693                 elseif c == "\t" and remaining > 3 then
4694                     remaining = remaining - 4
4695                     i = i + 1
4696                 else
4697                     break
4698                 end
4699             end
4700             return s:sub(i)
4701         end
4702     end

```

3.1.4.5 Parsers Used for Markdown Tags and Links

```

4703     parsers.leader      = parsers.space^-3
4704
4705     -- content in balanced brackets, parentheses, or quotes:
4706     parsers.bracketed   = P{ parsers.lbracket
4707         * ((parsers.anyescaped - (parsers.lbracket
4708             + parsers.rbracket
4709             + parsers.blankline^2)
4710         ) + V(1))^0
4711         * parsers.rbracket }
4712
4713     parsers.inparens    = P{ parsers.lparent
4714         * ((parsers.anyescaped - (parsers.lparent
4715             + parsers.rparent
4716             + parsers.blankline^2)
4717         ) + V(1))^0
4718         * parsers.rparent }
4719
4720     parsers.squoted     = P{ parsers.squote * parsers.alphanumeric
4721         * ((parsers.anyescaped - (parsers.squote

```



```

4722                                     + parsers.blankline^2)
4723                                     ) + V(1))^0
4724                                     * parsers.squote }
4725
4726 parsers.dquoted      = P{ parsers.dquote * parsers.alphanumeric
4727                             * ((parsers.anyescaped - (parsers.dquote
4728                                     + parsers.blankline^2)
4729                                     ) + V(1))^0
4730                             * parsers.dquote }
4731
4732 -- bracketed tag for markdown links, allowing nested brackets:
4733 parsers.tag          = parsers.lbracket
4734                             * Cs((parsers.alphanumeric^1
4735                                     + parsers.bracketed
4736                                     + parsers.inticks
4737                                     + (parsers.anyescaped
4738                                         - (parsers.rbracket + parsers.blankline^2)))^0)
4739                             * parsers.rbracket
4740
4741 -- url for markdown links, allowing nested brackets:
4742 parsers.url          = parsers.less * Cs((parsers.anyescaped
4743                                     - parsers.more)^0)
4744                                     * parsers.more
4745                                     + Cs((parsers.inparens + (parsers.anyescaped
4746                                         - parsers.spacing
4747                                         - parsers.rparent))^1)
4748
4749 -- quoted text, possibly with nested quotes:
4750 parsers.title_s      = parsers.squote * Cs(((parsers.anyescaped-parsers.squote)
4751                                     + parsers.squoted)^0)
4752                                     * parsers.squote
4753
4754 parsers.title_d      = parsers.dquote * Cs(((parsers.anyescaped-parsers.dquote)
4755                                     + parsers.dquoted)^0)
4756                                     * parsers.dquote
4757
4758 parsers.title_p      = parsers.lparent
4759                             * Cs((parsers.inparens + (parsers.anyescaped-parsers.rparent))^0)
4760                             * parsers.rparent
4761
4762 parsers.title        = parsers.title_d + parsers.title_s + parsers.title_p
4763
4764 parsers.optionaltitle
4765                             = parsers.spnl * parsers.title * parsers.spacechar^0
4766                             + Cc("")

```

3.1.4.6 Parsers Used for HTML

```
4767 -- case-insensitive match (we assume s is lowercase). must be single byte encoding
4768 parsers.keyword_exact = function(s)
4769   local parser = P(0)
4770   for i=1,#s do
4771     local c = s:sub(i,i)
4772     local m = c .. upper(c)
4773     parser = parser * S(m)
4774   end
4775   return parser
4776 end
4777
4778 parsers.block_keyword =
4779   parsers.keyword_exact("address") + parsers.keyword_exact("blockquote") +
4780   parsers.keyword_exact("center") + parsers.keyword_exact("del") +
4781   parsers.keyword_exact("dir") + parsers.keyword_exact("div") +
4782   parsers.keyword_exact("p") + parsers.keyword_exact("pre") +
4783   parsers.keyword_exact("li") + parsers.keyword_exact("ol") +
4784   parsers.keyword_exact("ul") + parsers.keyword_exact("dl") +
4785   parsers.keyword_exact("dd") + parsers.keyword_exact("form") +
4786   parsers.keyword_exact("fieldset") + parsers.keyword_exact("isindex") +
4787   parsers.keyword_exact("ins") + parsers.keyword_exact("menu") +
4788   parsers.keyword_exact("noframes") + parsers.keyword_exact("frameset") +
4789   parsers.keyword_exact("h1") + parsers.keyword_exact("h2") +
4790   parsers.keyword_exact("h3") + parsers.keyword_exact("h4") +
4791   parsers.keyword_exact("h5") + parsers.keyword_exact("h6") +
4792   parsers.keyword_exact("hr") + parsers.keyword_exact("script") +
4793   parsers.keyword_exact("noscript") + parsers.keyword_exact("table") +
4794   parsers.keyword_exact("tbody") + parsers.keyword_exact("tfoot") +
4795   parsers.keyword_exact("thead") + parsers.keyword_exact("th") +
4796   parsers.keyword_exact("td") + parsers.keyword_exact("tr")
4797
4798 -- There is no reason to support bad html, so we expect quoted attributes
4799 parsers.htmlattributevalue
4800   = parsers.squote * (parsers.any - (parsers.blankline
4801   + parsers.squote))^0
4802   * parsers.squote
4803   + parsers.dquote * (parsers.any - (parsers.blankline
4804   + parsers.dquote))^0
4805   * parsers.dquote
4806
4807 parsers.htmlattribute = parsers.spacing^1
4808   * (parsers.alphanumeric + S("_-"))^1
4809   * parsers.sp * parsers.equal * parsers.sp
4810   * parsers.htmlattributevalue
4811
4812 parsers.htmlcomment = P("<!--")
```

```

4813             * parsers.optionalspace
4814             * Cs((parsers.any - parsers.optionalspace * P("-->"))^0)
4815             * parsers.optionalspace
4816             * P("-->")
4817
4818 parsers.htmlinstruction = P("<?") * (parsers.any - P("?>"))^0 * P("?>")
4819
4820 parsers.openelt_any = parsers.less * parsers.keyword * parsers.htmlattribute^0
4821             * parsers.sp * parsers.more
4822
4823 parsers.openelt_exact = function(s)
4824     return parsers.less * parsers.sp * parsers.keyword_exact(s)
4825             * parsers.htmlattribute^0 * parsers.sp * parsers.more
4826 end
4827
4828 parsers.openelt_block = parsers.sp * parsers.block_keyword
4829             * parsers.htmlattribute^0 * parsers.sp * parsers.more
4830
4831 parsers.closeelt_any = parsers.less * parsers.sp * parsers.slash
4832             * parsers.keyword * parsers.sp * parsers.more
4833
4834 parsers.closeelt_exact = function(s)
4835     return parsers.less * parsers.sp * parsers.slash * parsers.keyword_exact(s)
4836             * parsers.sp * parsers.more
4837 end
4838
4839 parsers.emptyelt_any = parsers.less * parsers.sp * parsers.keyword
4840             * parsers.htmlattribute^0 * parsers.sp * parsers.slash
4841             * parsers.more
4842
4843 parsers.emptyelt_block = parsers.less * parsers.sp * parsers.block_keyword
4844             * parsers.htmlattribute^0 * parsers.sp * parsers.slash
4845             * parsers.more
4846
4847 parsers.displaytext = (parsers.any - parsers.less)^1
4848
4849 -- return content between two matched HTML tags
4850 parsers.in_matched = function(s)
4851     return { parsers.openelt_exact(s)
4852             * (V(1) + parsers.displaytext
4853             + (parsers.less - parsers.closeelt_exact(s)))^0
4854             * parsers.closeelt_exact(s) }
4855 end
4856
4857 local function parse_matched_tags(s,pos)
4858     local t = string.lower(lpeg.match(C(parsers.keyword),s,pos))
4859     return lpeg.match(parsers.in_matched(t),s,pos-1)

```

```

4860 end
4861
4862 parsers.in_matched_block_tags = parsers.less
4863                               * Cmt(#parsers.openelt_block, parse_matched_tags)
4864

```

3.1.4.7 Parsers Used for HTML Entities

```

4865 parsers.hexentity = parsers.ampersand * parsers.hash * S("Xx")
4866                  * C(parsers.hexdigit^1) * parsers.semicolon
4867 parsers.decentity = parsers.ampersand * parsers.hash
4868                  * C(parsers.digit^1) * parsers.semicolon
4869 parsers.tagentity = parsers.ampersand * C(parsers.alphanumeric^1)
4870                  * parsers.semicolon

```

3.1.4.8 Helpers for References

```

4871 -- parse a reference definition: [foo]: /bar "title"
4872 parsers.define_reference_parser = parsers.leader * parsers.tag * parsers.colon
4873                               * parsers.spacechar^0 * parsers.url
4874                               * parsers.optionaltitle * parsers.blankline^1

```

3.1.4.9 Inline Elements

```

4875 parsers.Inline          = V("Inline")
4876 parsers.IndentedInline = V("IndentedInline")
4877
4878 -- parse many p between starter and ender
4879 parsers.between = function(p, starter, ender)
4880   local ender2 = B(parsers.nonspacechar) * ender
4881   return (starter * #parsers.nonspacechar * Ct(p * (p - ender2)^0) * ender2)
4882 end
4883
4884 parsers.urlchar          = parsers.anyescaped - parsers.newline - parsers.more

```

3.1.4.10 Block Elements

```

4885 parsers.lineof = function(c)
4886   return (parsers.leader * (P(c) * parsers.optionalspace)^3
4887         * (parsers.newline * parsers.blankline^1
4888           + parsers.newline^-1 * parsers.eof))
4889 end

```

3.1.4.11 Lists

3.1.4.12 Headings

```
4890 parsers.heading_attribute = C(parsers.css_identifier)
4891                               + C((parsers.attribute_key_char
4892                                   - parsers.rbrace)^1
4893                                   * parsers.equal
4894                                   * (parsers.attribute_value_char
4895                                       - parsers.rbrace)^1)
4896 parsers.HeadingAttributes = parsers.lbrace
4897                               * parsers.heading_attribute
4898                               * (parsers.spacechar^1
4899                                   * parsers.heading_attribute)^0
4900                               * parsers.rbrace
4901
4902 -- parse Atx heading start and return level
4903 parsers.HeadingStart = #parsers.hash * C(parsers.hash^-6)
4904                       * -parsers.hash / length
4905
4906 -- parse setext header ending and return level
4907 parsers.HeadingLevel = parsers.equal^1 * Cc(1) + parsers.dash^1 * Cc(2)
4908
4909 local function strip_atx_end(s)
4910   return s:gsub("#%s*\n$", "")
4911 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 $\langle member \rangle$ s as `reader-> $\langle member \rangle$` .

```
4912 M.reader = {}
4913 function M.reader.new(writer, options, extensions)
4914   local self = {}
```

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

```
4915 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.

```
4916 self.parsers = {}
4917 (function(parsers)
4918     setmetatable(self.parsers, {
4919         __index = function (_, key)
4920             return parsers[key]
4921         end
4922     })
4923 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.

```
4924 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.

```
4925 function self.normalize_tag(tag)
4926     return string.lower(
4927         gsub(util.rope_to_string(tag), "[ \\n\\r\\t]+", " "))
4928 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.

```
4929 local function iterlines(s, f)
4930     rope = lpeg.match(Ct((parsers.line / f)^1), s)
4931     return util.rope_to_string(rope)
4932 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.

```
4933 if options.preserveTabs then
4934     self.expandtabs = function(s) return s end
4935 else
4936     self.expandtabs = function(s)
4937         if s:find("\\t") then
4938             return iterlines(s, util.expand_tabs_in_line)
4939         else
4940             return s
4941         end
4942     end
4943 end
```

```

4942                                     end
4943     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.

```

4944     self.parser_functions = {}
4945     self.create_parser = function(name, grammar, toplevel)
4946         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.

```

4947         if toplevel and options.stripIndent then
4948             local min_prefix_length, min_prefix = nil, ''
4949             str = iterlines(str, function(line)
4950                 if lpeg.match(parsers.nonemptyline, line) == nil then
4951                     return line
4952                 end
4953                 line = util.expand_tabs_in_line(line)
4954                 prefix = lpeg.match(C(parsers.optionalspace), line)
4955                 local prefix_length = #prefix
4956                 local is_shorter = min_prefix_length == nil
4957                 is_shorter = is_shorter or prefix_length < min_prefix_length
4958                 if is_shorter then
4959                     min_prefix_length, min_prefix = prefix_length, prefix
4960                 end
4961                 return line
4962             end)
4963             str = str:gsub('^' .. min_prefix, '')
4964         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.

```

4965         if toplevel and (options.texComments or options.hybrid) then
4966             str = lpeg.match(Ct(parsers.commented_line^1), str)
4967             str = util.rope_to_string(str)
4968         end
4969         local res = lpeg.match(grammar(), str)
4970         if res == nil then
4971             error(format("%s failed on:\n%s", name, str:sub(1,20)))

```

```

4972         else
4973             return res
4974         end
4975     end
4976 end
4977
4978 self.create_parser("parse_blocks",
4979                   function()
4980                       return parsers.blocks
4981                   end, true)
4982
4983 self.create_parser("parse_blocks_nested",
4984                   function()
4985                       return parsers.blocks_nested
4986                   end, false)
4987
4988 self.create_parser("parse_inlines",
4989                   function()
4990                       return parsers.inlines
4991                   end, false)
4992
4993 self.create_parser("parse_inlines_no_link",
4994                   function()
4995                       return parsers.inlines_no_link
4996                   end, false)
4997
4998 self.create_parser("parse_inlines_no_inline_note",
4999                   function()
5000                       return parsers.inlines_no_inline_note
5001                   end, false)
5002
5003 self.create_parser("parse_inlines_no_html",
5004                   function()
5005                       return parsers.inlines_no_html
5006                   end, false)
5007
5008 self.create_parser("parse_inlines_nbsp",
5009                   function()
5010                       return parsers.inlines_nbsp
5011                   end, false)

```

3.1.5.3 Parsers Used for Markdown Lists (local)

```

5012 if options.hashEnumerators then
5013     parsers.dig = parsers.digit + parsers.hash
5014 else
5015     parsers.dig = parsers.digit

```



```

5016 end
5017
5018 parsers.enumerator = C(parsers.dig^3 * parsers.period) * #parsers.spacing
5019 + C(parsers.dig^2 * parsers.period) * #parsers.spacing
5020 + (parsers.tab + parsers.space^1)
5021 + C(parsers.dig * parsers.period) * #parsers.spacing
5022 + (parsers.tab + parsers.space^-2)
5023 + parsers.space * C(parsers.dig^2 * parsers.period)
5024 + #parsers.spacing
5025 + parsers.space * C(parsers.dig * parsers.period)
5026 + #parsers.spacing
5027 + (parsers.tab + parsers.space^-1)
5028 + parsers.space * parsers.space * C(parsers.dig^1
5029 + parsers.period) * #parsers.spacing

```

3.1.5.4 Parsers Used for Blockquotes (local)

```

5030 -- strip off leading > and indents, and run through blocks
5031 parsers.blockquote_body = ((parsers.leader * parsers.more * parsers.space^-
5032 1)/""
5033 + parsers.linechar^0 * parsers.newline)^1
5034 + (-(parsers.leader * parsers.more
5035 + parsers.blankline) * parsers.linechar^1
5036 + parsers.newline)^0
5037 if not options.breakableBlockquotes then
5038 parsers.blockquote_body = parsers.blockquote_body
5039 + (parsers.blankline^0 / "")
5040 end

```

3.1.5.5 Parsers Used for Footnotes (local)

3.1.5.6 Helpers for Links and References (local)

```

5041 -- List of references defined in the document
5042 local references
5043
5044 -- add a reference to the list
5045 local function register_link(tag,url,title)
5046 references[self.normalize_tag(tag)] = { url = url, title = title }
5047 return ""
5048 end
5049
5050 -- lookup link reference and return either
5051 -- the link or nil and fallback text.
5052 local function lookup_reference(label,sps,tag)
5053 local tagpart

```

```

5054     if not tag then
5055         tag = label
5056         tagpart = ""
5057     elseif tag == "" then
5058         tag = label
5059         tagpart = "[]"
5060     else
5061         tagpart = {"[",
5062             self.parser_functions.parse_inlines(tag),
5063             "]" }
5064     end
5065     if sps then
5066         tagpart = {sps, tagpart}
5067     end
5068     local r = references[self.normalize_tag(tag)]
5069     if r then
5070         return r
5071     else
5072         return nil, {"[",
5073             self.parser_functions.parse_inlines(label),
5074             "]", tagpart}
5075     end
5076 end
5077
5078 -- lookup link reference and return a link, if the reference is found,
5079 -- or a bracketed label otherwise.
5080 local function indirect_link(label,sps,tag)
5081     return writer.defer_call(function()
5082         local r,fallback = lookup_reference(label,sps,tag)
5083         if r then
5084             return writer.link(
5085                 self.parser_functions.parse_inlines_no_link(label),
5086                 r.url, r.title)
5087         else
5088             return fallback
5089         end
5090     end)
5091 end
5092
5093 -- lookup image reference and return an image, if the reference is found,
5094 -- or a bracketed label otherwise.
5095 local function indirect_image(label,sps,tag)
5096     return writer.defer_call(function()
5097         local r,fallback = lookup_reference(label,sps,tag)
5098         if r then
5099             return writer.image(writer.string(label), r.url, r.title)
5100         else

```

```

5101         return {"!", fallback}
5102     end
5103 end)
5104 end

```

3.1.5.7 Inline Elements (local)

```

5105 parsers.Str      = (parsers.normalchar * (parsers.normalchar + parsers.at)^0)
5106                  / writer.string
5107
5108 parsers.Symbol    = (parsers.specialchar - parsers.tightblocksep)
5109                  / writer.string
5110
5111 parsers.Ellipsis  = P("...") / writer.ellipsis
5112
5113 parsers.Smart     = parsers.Ellipsis
5114
5115 parsers.Code      = parsers.inticks / writer.code
5116
5117 if options.blankBeforeBlockquote then
5118     parsers.bqstart = parsers.fail
5119 else
5120     parsers.bqstart = parsers.more
5121 end
5122
5123 if options.blankBeforeHeading then
5124     parsers.headerstart = parsers.fail
5125 else
5126     parsers.headerstart = parsers.hash
5127                          + (parsers.line * (parsers.equal^1 + parsers.dash^1)
5128                             * parsers.optionalspace * parsers.newline)
5129 end
5130
5131 parsers.EndlineExceptions
5132     = parsers.blankline -- paragraph break
5133     + parsers.tightblocksep -- nested list
5134     + parsers.eof         -- end of document
5135     + parsers.bqstart
5136     + parsers.headerstart
5137
5138 parsers.Endline    = parsers.newline
5139                     * -V("EndlineExceptions")
5140                     * parsers.spacechar^0
5141                     / (options.hardLineBreaks and writer.linebreak
5142                        or writer.space)
5143
5144 parsers.OptionalIndent

```

```

5145         = parsers.spacechar~1 / writer.space
5146
5147 parsers.Space = parsers.spacechar~2 * parsers.Endline / writer.linebreak
5148               + parsers.spacechar~1 * parsers.Endline~-1 * parsers.eof / ""
5149               + parsers.spacechar~1 * parsers.Endline
5150                   * parsers.optionalspace
5151                   / (options.hardLineBreaks
5152                      and writer.linebreak
5153                      or writer.space)
5154               + parsers.spacechar~1 * parsers.optionalspace
5155                   / writer.space
5156
5157 parsers.NonbreakingEndline
5158     = parsers.newline
5159     * -V("EndlineExceptions")
5160     * parsers.spacechar~0
5161     / (options.hardLineBreaks and writer.linebreak
5162        or writer.nbsp)
5163
5164 parsers.NonbreakingSpace
5165     = parsers.spacechar~2 * parsers.Endline / writer.linebreak
5166     + parsers.spacechar~1 * parsers.Endline~-1 * parsers.eof / ""
5167     + parsers.spacechar~1 * parsers.Endline
5168         * parsers.optionalspace
5169         / (options.hardLineBreaks
5170            and writer.linebreak
5171            or writer.nbsp)
5172     + parsers.spacechar~1 * parsers.optionalspace
5173         / writer.nbsp
5174
5175 if options.underscores then
5176   parsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
5177                                     parsers.doubleasterisks)
5178                     + parsers.between(parsers.Inline, parsers.doubleunderscores,
5179                                     parsers.doubleunderscores)
5180                     ) / writer.strong
5181
5182   parsers.Emph   = ( parsers.between(parsers.Inline, parsers.asterisk,
5183                                     parsers.asterisk)
5184                     + parsers.between(parsers.Inline, parsers.underscore,
5185                                     parsers.underscore)
5186                     ) / writer.emphasis
5187 else
5188   parsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
5189                                     parsers.doubleasterisks)
5190                     ) / writer.strong
5191

```

```

5192     parsers.Emph    = ( parsers.between(parsers.Inline, parsers.asterisk,
5193                                         parsers.asterisk)
5194                         ) / writer.emphasis
5195 end
5196
5197 parsers.AutoLinkUrl  = parsers.less
5198                     * C(parsers.alphanumeric^1 * P("://") * parsers.urlchar^1)
5199                     * parsers.more
5200                     / function(url)
5201                         return writer.link(writer.escape(url), url)
5202                     end
5203
5204 parsers.AutoLinkEmail = parsers.less
5205                     * C((parsers.alphanumeric + S("-._+"))^1
5206                     * P("@") * parsers.urlchar^1)
5207                     * parsers.more
5208                     / function(email)
5209                         return writer.link(writer.escape(email),
5210                                             "mailto:.."email)
5211                     end
5212
5213 parsers.AutoLinkRelativeReference
5214                     = parsers.less
5215                     * C(parsers.urlchar^1)
5216                     * parsers.more
5217                     / function(url)
5218                         return writer.link(writer.escape(url), url)
5219                     end
5220
5221 parsers.DirectLink   = (parsers.tag / self.parser_functions.parse_inlines_no_link)
5222                     * parsers.spnl
5223                     * parsers.lparent
5224                     * (parsers.url + C("")) -- link can be empty [foo]()
5225                     * parsers.optionaltitle
5226                     * parsers.rparent
5227                     / writer.link
5228
5229 parsers.IndirectLink = parsers.tag * (C(parsers.spnl) * parsers.tag)^-
5230
5231 / indirect_link
5232
5233 -- parse a link or image (direct or indirect)
5234 parsers.Link          = parsers.DirectLink + parsers.IndirectLink
5235
5236 parsers.DirectImage   = parsers.exclamation
5237                     * (parsers.tag / self.parser_functions.parse_inlines)
5238                     * parsers.spnl

```

```

5238         * parsers.lparent
5239         * (parsers.url + Cc("")) -- link can be empty [foo]()
5240         * parsers.optionalttitle
5241         * parsers.rparent
5242         / writer.image
5243
5244     parsers.IndirectImage = parsers.exclamation * parsers.tag
5245                          * (C(parsers.spnl) * parsers.tag)^-1 / indirect_image
5246
5247     parsers.Image         = parsers.DirectImage + parsers.IndirectImage
5248
5249     -- avoid parsing long strings of * or _ as emph/strong
5250     parsers.UlOrStarLine = parsers.asterisk^4 + parsers.underscore^4
5251                          / writer.string
5252
5253     parsers.EscapedChar   = parsers.backslash * C(parsers.escapable) / writer.string
5254
5255     parsers.InlineHtml    = parsers.emptyelt_any / writer.inline_html_tag
5256                          + (parsers.htmlcomment / self.parser_functions.parse_inlines_no
5257                          / writer.inline_html_comment
5258                          + parsers.htmlinstruction
5259                          + parsers.openelt_any / writer.inline_html_tag
5260                          + parsers.closeelt_any / writer.inline_html_tag
5261
5262     parsers.HtmlEntity     = parsers.hexentity / entities.hex_entity / writer.string
5263                          + parsers.decentity / entities.dec_entity / writer.string
5264                          + parsers.tagentity / entities.char_entity / writer.string

```

3.1.5.8 Block Elements (local)

```

5265     parsers.DisplayHtml   = (parsers.htmlcomment / self.parser_functions.parse_blocks_nested
5266                          / writer.block_html_comment
5267                          + parsers.emptyelt_block / writer.block_html_element
5268                          + parsers.openelt_exact("hr") / writer.block_html_element
5269                          + parsers.in_matched_block_tags / writer.block_html_element
5270                          + parsers.htmlinstruction
5271
5272     parsers.Verbatim      = Cs( (parsers.blanklines
5273                          * ((parsers.indentedline - parsers.blankline))^1)^1
5274                          ) / self.expandtabs / writer.verbatim
5275
5276     parsers.Blockquote    = Cs(parsers.blockquote_body^1)
5277                          / self.parser_functions.parse_blocks_nested
5278                          / writer.blockquote
5279
5280     parsers.HorizontalRule = ( parsers.lineof(parsers.asterisk)
5281                          + parsers.lineof(parsers.dash)

```

```

5282             + parsers.lineof(parsers.underscore)
5283             ) / writer.hrulerule
5284
5285 parsers.Reference      = parsers.define_reference_parser / register_link
5286
5287 parsers.Paragraph      = parsers.nonindentspace * Ct(parsers.Inline~1)
5288                       * ( parsers.newline
5289                       * ( parsers.blankline~1
5290                       + #parsers.hash
5291                       + #(parsers.leader * parsers.more * parsers.space~1)
5292                       + parsers.eof
5293                       )
5294                       + parsers.eof )
5295                       / writer.paragraph
5296
5297 parsers.Plain          = parsers.nonindentspace * Ct(parsers.Inline~1)
5298                       / writer.plain

```

3.1.5.9 Lists (local)

```

5299 parsers.starter = parsers.bullet + parsers.enumerator
5300
5301 if options.taskLists then
5302   parsers.tickbox = ( parsers.ticked_box
5303                     + parsers.halfticked_box
5304                     + parsers.unticked_box
5305                     ) / writer.tickbox
5306 else
5307   parsers.tickbox = parsers.fail
5308 end
5309
5310 -- we use \001 as a separator between a tight list item and a
5311 -- nested list under it.
5312 parsers.NestedList      = Cs((parsers.optionallyindentedline
5313                             - parsers.starter)^1)
5314                             / function(a) return "\001"..a end
5315
5316 parsers.ListBlockLine   = parsers.optionallyindentedline
5317                             - parsers.blankline - (parsers.indent~1
5318                             * parsers.starter)
5319
5320 parsers.ListBlock       = parsers.line * parsers.ListBlockLine~0
5321
5322 parsers.ListContinuationBlock = parsers.blanklines * (parsers.indent / "")
5323                             * parsers.ListBlock
5324

```

```

5325 parsers.TightListItem = function(starter)
5326     return -parsers.HorizontalRule
5327     * (Cs(starter / "" * parsers.tickbox^-1 * parsers.ListBlock * parsers.Nest
1)
5328         / self.parser_functions.parse_blocks_nested)
5329     * -(parsers.blanklines * parsers.indent)
5330 end
5331
5332 parsers.LooseListItem = function(starter)
5333     return -parsers.HorizontalRule
5334     * Cs( starter / "" * parsers.tickbox^-1 * parsers.ListBlock * Cc("\n")
5335     * (parsers.NestedList + parsers.ListContinuationBlock^0)
5336     * (parsers.blanklines / "\n\n")
5337     ) / self.parser_functions.parse_blocks_nested
5338 end
5339
5340 parsers.BulletList = ( Ct(parsers.TightListItem(parsers.bullet)^1) * Cc(true)
5341     * parsers.skipblanklines * -parsers.bullet
5342     + Ct(parsers.LooseListItem(parsers.bullet)^1) * Cc(false)
5343     * parsers.skipblanklines )
5344     / writer.bulletlist
5345
5346 local function ordered_list(items,tight,startNumber)
5347     if options.startNumber then
5348         startNumber = tonumber(startNumber) or 1 -- fallback for '#'
5349         if startNumber ~= nil then
5350             startNumber = math.floor(startNumber)
5351         end
5352     else
5353         startNumber = nil
5354     end
5355     return writer.orderedlist(items,tight,startNumber)
5356 end
5357
5358 parsers.OrderedList = Cg(parsers.enumerator, "listtype") *
5359     ( Ct(parsers.TightListItem(Cb("listtype")))
5360     * parsers.TightListItem(parsers.enumerator)^0)
5361     * Cc(true) * parsers.skipblanklines * -parsers.enumerator
5362     + Ct(parsers.LooseListItem(Cb("listtype")))
5363     * parsers.LooseListItem(parsers.enumerator)^0)
5364     * Cc(false) * parsers.skipblanklines
5365     ) * Cb("listtype") / ordered_list

```

3.1.5.10 Blank (local)

```

5366 parsers.Blank           = parsers.blankline / ""
5367                          + parsers.Reference

```



```
5368                                     + (parsers.tightblocksep / "\n")
```

3.1.5.11 Headings (local)

```
5369 -- parse atx header
5370 parsers.AtHeading = Cg(parsers.HeadingStart,"level")
5371                     * parsers.optionalspace
5372                     * (C(parsers.line)
5373                       / strip_atx_end
5374                       / self.parser_functions.parse_inlines)
5375                     * Cb("level")
5376                     / writer.heading
5377
5378 parsers.SettextHeading = #(parsers.line * S("=-"))
5379                          * Ct(parsers.linechar^1
5380                            / self.parser_functions.parse_inlines)
5381                          * parsers.newline
5382                          * parsers.HeadingLevel
5383                          * parsers.optionalspace
5384                          * parsers.newline
5385                          / writer.heading
5386
5387 parsers.Heading = parsers.AtHeading + parsers.SettextHeading
```

3.1.5.12 Syntax Specification Create a [reader->syntax](#) hash table that stores the PEG grammar.

```
5388 self.syntax =
5389     { "Blocks",
5390
5391       Blocks = ( V("ExpectedJekyllData")
5392                 * (V("Blank")^0 / writer.interblocksep)
5393                 )^1
5394                 * V("Blank")^0
5395                 * V("Block")^1
5396                 * (V("Blank")^0 / writer.interblocksep
5397                   * V("Block"))^0
5398                 * V("Blank")^0 * parsers.eof,
5399
5400       Blank = parsers.Blank,
5401
5402       UnexpectedJekyllData = parsers.fail,
5403       ExpectedJekyllData = parsers.fail,
5404
5405       Block = V("ContentBlock")
5406              + V("UnexpectedJekyllData")
5407              + V("Blockquote")
5408              + V("PipeTable")
```

5409		+ V("Verbatim")
5410		+ V("FencedCode")
5411		+ V("HorizontalRule")
5412		+ V("BulletList")
5413		+ V("OrderedList")
5414		+ V("Heading")
5415		+ V("DefinitionList")
5416		+ V("DisplayHtml")
5417		+ V("Paragraph")
5418		+ V("Plain"),
5419		
5420	ContentBlock	= parsers.fail,
5421	Blockquote	= parsers.Blockquote,
5422	Verbatim	= parsers.Verbatim,
5423	FencedCode	= parsers.fail,
5424	HorizontalRule	= parsers.HorizontalRule,
5425	BulletList	= parsers.BulletList,
5426	OrderedList	= parsers.OrderedList,
5427	Heading	= parsers.Heading,
5428	DefinitionList	= parsers.fail,
5429	DisplayHtml	= parsers.DisplayHtml,
5430	Paragraph	= parsers.Paragraph,
5431	PipeTable	= parsers.fail,
5432	Plain	= parsers.Plain,
5433	EndlineExceptions	= parsers.EndlineExceptions,
5434		
5435	Inline	= V("Str")
5436		+ V("Space")
5437		+ V("Endline")
5438		+ V("ULOrStarLine")
5439		+ V("Strong")
5440		+ V("Emph")
5441		+ V("InlineNote")
5442		+ V("NoteRef")
5443		+ V("Citations")
5444		+ V("Link")
5445		+ V("Image")
5446		+ V("Code")
5447		+ V("AutoLinkUrl")
5448		+ V("AutoLinkEmail")
5449		+ V("AutoLinkRelativeReference")
5450		+ V("InlineHtml")
5451		+ V("HtmlEntity")
5452		+ V("EscapedChar")
5453		+ V("Smart")
5454		+ V("Symbol"),
5455		

```

5456     IndentedInline      = V("Str")
5457                           + V("OptionalIndent")
5458                           + V("Endline")
5459                           + V("U1OrStarLine")
5460                           + V("Strong")
5461                           + V("Emph")
5462                           + V("InlineNote")
5463                           + V("NoteRef")
5464                           + V("Citations")
5465                           + V("Link")
5466                           + V("Image")
5467                           + V("Code")
5468                           + V("AutoLinkUrl")
5469                           + V("AutoLinkEmail")
5470                           + V("AutoLinkRelativeReference")
5471                           + V("InlineHtml")
5472                           + V("HtmlEntity")
5473                           + V("EscapedChar")
5474                           + V("Smart")
5475                           + V("Symbol"),
5476
5477     Str                    = parsers.Str,
5478     Space                  = parsers.Space,
5479     OptionalIndent        = parsers.OptionalIndent,
5480     Endline                = parsers.Endline,
5481     U1OrStarLine          = parsers.U1OrStarLine,
5482     Strong                 = parsers.Strong,
5483     Emph                   = parsers.Emph,
5484     InlineNote             = parsers.fail,
5485     NoteRef                = parsers.fail,
5486     Citations              = parsers.fail,
5487     Link                   = parsers.Link,
5488     Image                  = parsers.Image,
5489     Code                   = parsers.Code,
5490     AutoLinkUrl            = parsers.AutoLinkUrl,
5491     AutoLinkEmail          = parsers.AutoLinkEmail,
5492     AutoLinkRelativeReference
5493                           = parsers.AutoLinkRelativeReference,
5494     InlineHtml             = parsers.InlineHtml,
5495     HtmlEntity             = parsers.HtmlEntity,
5496     EscapedChar            = parsers.EscapedChar,
5497     Smart                  = parsers.Smart,
5498     Symbol                 = parsers.Symbol,
5499 }
5500

```

Apply syntax extensions.

```

5501   for _, extension in ipairs(extensions) do

```

```

5502     extension.extend_writer(writer)
5503     extension.extend_reader(self)
5504 end
5505
5506 if not options.codeSpans then
5507     self.syntax.Code = parsers.fail
5508 end
5509
5510 if not options.html then
5511     self.syntax.DisplayHtml = parsers.fail
5512     self.syntax.InlineHtml = parsers.fail
5513     self.syntax.HtmlEntity = parsers.fail
5514 end
5515
5516 if options.preserveTabs then
5517     options.stripIndent = false
5518 end
5519
5520 if not options.smartEllipses then
5521     self.syntax.Smart = parsers.fail
5522 end
5523
5524 if not options.relativeReferences then
5525     self.syntax.AutoLinkRelativeReference = parsers.fail
5526 end
5527
5528 local blocks_nested_t = util.table_copy(self.syntax)
5529 blocks_nested_t.ExpectedJekyllData = parsers.fail
5530 parsers.blocks_nested = Ct(blocks_nested_t)
5531
5532 parsers.blocks = Ct(self.syntax)
5533
5534 local inlines_t = util.table_copy(self.syntax)
5535 inlines_t[1] = "Inlines"
5536 inlines_t.Inlines = parsers.Inline^0 * (parsers.spacing^0 * parsers.eof / "")
5537 parsers.inlines = Ct(inlines_t)
5538
5539 local inlines_no_link_t = util.table_copy(inlines_t)
5540 inlines_no_link_t.Link = parsers.fail
5541 parsers.inlines_no_link = Ct(inlines_no_link_t)
5542
5543 local inlines_no_inline_note_t = util.table_copy(inlines_t)
5544 inlines_no_inline_note_t.InlineNote = parsers.fail
5545 parsers.inlines_no_inline_note = Ct(inlines_no_inline_note_t)
5546
5547 local inlines_no_html_t = util.table_copy(inlines_t)
5548 inlines_no_html_t.DisplayHtml = parsers.fail

```

```

5549 inlines_no_html_t.InlineHtml = parsers.fail
5550 inlines_no_html_t.HtmlEntity = parsers.fail
5551 parsers.inlines_no_html = Ct(inlines_no_html_t)
5552
5553 local inlines_nbsp_t = util.table_copy(inlines_t)
5554 inlines_nbsp_t.Endline = parsers.NonbreakingEndline
5555 inlines_nbsp_t.Space = parsers.NonbreakingSpace
5556 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 T_EX output and returns it. Note that the converter assumes that the input has UNIX line endings.

```

5557 function self.convert(input)
5558     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.

```

5559     local opt_string = {}
5560     for k,_ in pairs(defaultOptions) do
5561         local v = options[k]
5562         if k ~= "cacheDir" then
5563             opt_string[#opt_string+1] = k .. "=" .. tostring(v)
5564         end
5565     end
5566     table.sort(opt_string)
5567     local salt = table.concat(opt_string, ",") .. "," .. metadata.version
5568     local output

```

If we cache markdown documents, produce the cache file and transform its filename to plain T_EX output via the `writer->pack` method.

```

5569     local function convert(input)
5570         local document = self.parser_functions.parse_blocks(input)
5571         return util.rope_to_string(writer.document(document))
5572     end
5573     if options.eagerCache or options.finalizeCache then
5574         local name = util.cache(options.cacheDir, input, salt, convert, ".md" .. writer.s
5575         output = writer.pack(name)

```

Otherwise, return the result of the conversion directly.

```

5576     else
5577         output = convert(input)
5578     end

```

If the `finalizeCache` option is enabled, populate the frozen cache in the file `frozenCacheFileName` with an entry for markdown document number `frozenCacheCounter`.

```

5579     if options.finalizeCache then
5580         local file, mode
5581         if options.frozenCacheCounter > 0 then
5582             mode = "a"
5583         else
5584             mode = "w"
5585         end
5586         file = assert(io.open(options.frozenCacheFileName, mode),
5587             [[could not open file ]] .. options.frozenCacheFileName
5588             .. [[ for writing]])
5589         assert(file:write([[\\expandafter\\global\\expandafter\\def\\csname ]]
5590             .. [[markdownFrozenCache]] .. options.frozenCacheCounter
5591             .. [[\\endcsname{]] .. output .. [[]]] .. "\\n"))
5592         assert(file:close())
5593     end
5594     return output
5595 end
5596 return self
5597 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.

```

5598 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.

```

5599 M.extensions.citations = function(citation_nbsps)

```

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

```

5600     local escaped_citation_chars = {
5601         [{""] = "\\markdownRendererLeftBrace{]",
5602         ["}"] = "\\markdownRendererRightBrace{]",
5603         [%"] = "\\markdownRendererPercentSign{]",
5604         [\\"] = "\\markdownRendererBackslash{]",
5605         [#"] = "\\markdownRendererHash{]",
5606     }
5607     return {
5608         extend_writer = function(self)

```

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

```
5609     local escape_citation = util.escaper(
5610         escaped_citation_chars,
5611         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.

```
5612     if self.hybrid then
5613         self.citation = self.escape_minimal
5614     else
5615         self.citation = escape_citation
5616     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.

```
5617     function self.citations(text_cites, cites)
5618         local buffer = {"\\markdownRenderer", text_cites and "TextCite" or "Cite",
5619             "{", #cites, "}"}
5620         for _,cite in ipairs(cites) do
5621             buffer[#buffer+1] = {cite.suppress_author and "-" or "+", "{",
5622                 cite.prenote or "", "}{" , cite.postnote or "", "}{" , cite.name, "}"}
5623         end
5624         return buffer
5625     end
5626 end, extend_reader = function(self)
5627     local parsers = self.parsers
5628     local syntax = self.syntax
5629     local writer = self.writer
5630
5631     local citation_chars
5632         = parsers.alphanumeric
5633         + S("#$%&-+<>~/_")
```

```

5634
5635     local citation_name
5636         = Cs(parsers.dash~-1) * parsers.at
5637         * Cs(citation_chars
5638             * (((citation_chars + parsers.internal_punctuation
5639                 - parsers.comma - parsers.semicolon)
5640                 * -#((parsers.internal_punctuation - parsers.comma
5641                     - parsers.semicolon)^0
5642                     * -(citation_chars + parsers.internal_punctuation
5643                         - parsers.comma - parsers.semicolon)))^0
5644                 * citation_chars)^-1)
5645
5646     local citation_body_prenote
5647         = Cs((parsers.alphanumeric^1
5648             + parsers.bracketed
5649             + parsers.inticks
5650             + (parsers.anyescaped
5651                 - (parsers.rbracket + parsers.blankline^2))
5652             - (parsers.spnl * parsers.dash~-1 * parsers.at))^0)
5653
5654     local citation_body_postnote
5655         = Cs((parsers.alphanumeric^1
5656             + parsers.bracketed
5657             + parsers.inticks
5658             + (parsers.anyescaped
5659                 - (parsers.rbracket + parsers.semicolon
5660                     + parsers.blankline^2))
5661             - (parsers.spnl * parsers.rbracket))^0)
5662
5663     local citation_body_chunk
5664         = citation_body_prenote
5665         * parsers.spnl * citation_name
5666         * (parsers.internal_punctuation - parsers.semicolon)^-
5667         1
5668         * parsers.spnl * citation_body_postnote
5669
5670     local citation_body
5671         = citation_body_chunk
5672         * (parsers.semicolon * parsers.spnl
5673             * citation_body_chunk)^0
5674
5675     local citation_headless_body_postnote
5676         = Cs((parsers.alphanumeric^1
5677             + parsers.bracketed
5678             + parsers.inticks
5679             + (parsers.anyescaped

```



```

5680         + parsers.semicolon + parsers.blankline^2))
5681         - (parsers.spnl * parsers.rbracket))^0)
5682
5683     local citation_headless_body
5684         = citation_headless_body_postnote
5685         * (parsers.sp * parsers.semicolon * parsers.spnl
5686         * citation_body_chunk)^0
5687
5688     local citations
5689         = function(text_cites, raw_cites)
5690         local function normalize(str)
5691             if str == "" then
5692                 str = nil
5693             else
5694                 str = (citation_nbsps and
5695                     self.parser_functions.parse_inlines_nbsp or
5696                     self.parser_functions.parse_inlines)(str)
5697             end
5698             return str
5699         end
5700
5701         local cites = {}
5702         for i = 1,#raw_cites,4 do
5703             cites[#cites+1] = {
5704                 prenote = normalize(raw_cites[i]),
5705                 suppress_author = raw_cites[i+1] == "-",
5706                 name = writer.citation(raw_cites[i+2]),
5707                 postnote = normalize(raw_cites[i+3]),
5708             }
5709         end
5710         return writer.citations(text_cites, cites)
5711     end
5712
5713     local TextCitations
5714         = Ct((parsers.spnl
5715             * Cc("")
5716             * citation_name
5717             * ((parsers.spnl
5718                 * parsers.lbracket
5719                 * citation_headless_body
5720                 * parsers.rbracket) + Cc("")))^1)
5721         / function(raw_cites)
5722             return citations(true, raw_cites)
5723         end
5724
5725     local ParenthesizedCitations
5726         = Ct((parsers.spnl

```

```

5727         * parsers.lbracket
5728         * citation_body
5729         * parsers.rbracket)^1)
5730     / function(raw_cites)
5731         return citations(false, raw_cites)
5732     end
5733
5734     local Citations = TextCitations + ParenthesizedCitations
5735
5736     syntax.Citations = Citations
5737 end
5738 }
5739 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.

```

5740 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.

```

5741     local languages_json = (function()
5742         local ran_ok, kpse = pcall(require, "kpse")
5743         if ran_ok then
5744             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:

```

5745     else
5746         kpse = {lookup=function(filename, options) return filename end}
5747     end
5748     local base, prev, curr
5749     for _, filename in ipairs{kpse.lookup(language_map, { all=true })} do
5750         local file = io.open(filename, "r")
5751         if not file then goto continue end
5752         json = file:read("*all"):gsub('"([^\n]-)":', '[%1]=')
5753         curr = (function()
5754             local _ENV={ json=json, load=load } -- run in sandbox
5755             return load("return "..json)()
5756         end)()
5757         if type(curr) == "table" then
5758             if base == nil then

```

```

5759         base = curr
5760     else
5761         setmetatable(prev, { __index = curr })
5762     end
5763     prev = curr
5764 end
5765 ::continue::
5766 end
5767 return base or {}
5768 end()
5769
5770 return {
5771     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.

```

5772     function self.contentblock(src,suf,type,tit)
5773         if not self.is_writing then return "" end
5774         src = src.." "..suf
5775         suf = suf:lower()
5776         if type == "onlineimage" then
5777             return {"\\markdownRendererContentBlockOnlineImage{" ,suf,"} ",
5778                 {"",self.string(src),""},
5779                 {"",self.uri(src),""},
5780                 {"",self.string(tit or ""),""} }
5781         elseif languages_json[suf] then
5782             return {"\\markdownRendererContentBlockCode{" ,suf,"} ",
5783                 {"",self.string(languages_json[suf]),""},
5784                 {"",self.string(src),""},
5785                 {"",self.uri(src),""},
5786                 {"",self.string(tit or ""),""} }
5787         else
5788             return {"\\markdownRendererContentBlock{" ,suf,"} ",
5789                 {"",self.string(src),""},
5790                 {"",self.uri(src),""},
5791                 {"",self.string(tit or ""),""} }
5792         end
5793     end
5794 end, extend_reader = function(self)
5795     local parsers = self.parsers
5796     local syntax = self.syntax
5797     local writer = self.writer
5798
5799     local contentblock_tail
5800         = parsers.optionaltitle

```

```

5801             * (parsers.newline + parsers.eof)
5802
5803 -- case insensitive online image suffix:
5804 local onlineimagesuffix
5805     = (function(...)
5806         local parser = nil
5807         for _, suffix in ipairs({...}) do
5808             local pattern=nil
5809             for i=1,#suffix do
5810                 local char=suffix:sub(i,i)
5811                 char = S(char:lower()..char:upper())
5812                 if pattern == nil then
5813                     pattern = char
5814                 else
5815                     pattern = pattern * char
5816                 end
5817             end
5818             if parser == nil then
5819                 parser = pattern
5820             else
5821                 parser = parser+ pattern
5822             end
5823         end
5824         return parser
5825     end)("png", "jpg", "jpeg", "gif", "tif", "tiff")
5826
5827 -- online image url for iA Writer content blocks with mandatory suffix,
5828 -- allowing nested brackets:
5829 local onlineimageurl
5830     = (parsers.less
5831         * Cs((parsers.anyescaped
5832             - parsers.more
5833             - #(parsers.period
5834                 * onlineimagesuffix
5835                 * parsers.more
5836                 * contentblock_tail))^0)
5837         * parsers.period
5838         * Cs(onlineimagesuffix)
5839         * parsers.more
5840         + (Cs((parsers.inparens
5841             + (parsers.anyescaped
5842                 - parsers.spacing
5843                 - parsers.rparent
5844                 - #(parsers.period
5845                     * onlineimagesuffix
5846                     * contentblock_tail))))^0)
5847         * parsers.period

```

```

5848             * Cs(onlineimagesuffix))
5849         ) * Cc("onlineimage")
5850
5851     -- filename for iA Writer content blocks with mandatory suffix:
5852     local localfilepath
5853         = parsers.slash
5854         * Cs((parsers.anyescaped
5855             - parsers.tab
5856             - parsers.newline
5857             - #(parsers.period
5858                 * parsers.alphanumeric~1
5859                 * contentblock_tail))^1)
5860         * parsers.period
5861         * Cs(parsers.alphanumeric~1)
5862         * Cc("localfile")
5863
5864     local ContentBlock
5865         = parsers.leader
5866         * (localfilepath + onlineimageurl)
5867         * contentblock_tail
5868         / writer.contentblock
5869
5870     syntax.ContentBlock = ContentBlock
5871 end
5872 }
5873 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.

```

5874 M.extensions.definition_lists = function(tight_lists)
5875     return {
5876         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.

```

5877     local function dlitem(term, defs)
5878         local retVal = {"\\markdownRendererDlItem{",term,""}
5879         for _, def in ipairs(defs) do
5880             retVal[#retVal+1] = {"\\markdownRendererDlDefinitionBegin ",def,
5881                 "\\markdownRendererDlDefinitionEnd "}
5882         end
5883         retVal[#retVal+1] = "\\markdownRendererDlItemEnd "
5884         return retVal

```

```

5885     end
5886
5887     function self.definitionlist(items,tight)
5888         if not self.is_writing then return "" end
5889         local buffer = {}
5890         for _,item in ipairs(items) do
5891             buffer[#buffer + 1] = dlitem(item.term, item.definitions)
5892         end
5893         if tight and tight_lists then
5894             return {"\\markdownRendererDlBeginTight\n", buffer,
5895                 "\n\\markdownRendererDlEndTight"}
5896         else
5897             return {"\\markdownRendererDlBegin\n", buffer,
5898                 "\n\\markdownRendererDlEnd"}
5899         end
5900     end
5901 end, extend_reader = function(self)
5902     local parsers = self.parsers
5903     local syntax = self.syntax
5904     local writer = self.writer
5905
5906     local defstartchar = S("~:")
5907
5908     local defstart = ( defstartchar * #parsers.spacing
5909                       * (parsers.tab + parsers.space^-3)
5910                       + parsers.space * defstartchar * #parsers.spacing
5911                       * (parsers.tab + parsers.space^-2)
5912                       + parsers.space * parsers.space * defstartchar
5913                       * #parsers.spacing
5914                       * (parsers.tab + parsers.space^-1)
5915                       + parsers.space * parsers.space * parsers.space
5916                       * defstartchar * #parsers.spacing
5917                       )
5918
5919     local dlchunk = Cs(parsers.line * (parsers.indentedline - parsers.blankline)^0)
5920
5921     local function definition_list_item(term, defs, tight)
5922         return { term = self.parser_functions.parse_inlines(term),
5923                 definitions = defs }
5924     end
5925
5926     local DefinitionListItemLoose
5927         = C(parsers.line) * parsers.skipblanklines
5928         * Ct((defstart
5929             * parsers.indented_blocks(dlchunk)
5930             / self.parser_functions.parse_blocks_nested)^1)
5931         * Cc(false) / definition_list_item

```

```

5932
5933     local DefinitionListItemTight
5934         = C(parsers.line)
5935         * Ct((defstart * dlchunk
5936             / self.parser_functions.parse_blocks_nested)^1)
5937         * Cc(true) / definition_list_item
5938
5939     local DefinitionList
5940         = ( Ct(DefinitionListItemLoose^1) * Cc(false)
5941           + Ct(DefinitionListItemTight^1)
5942           * (parsers.skipblanklines
5943             * -DefinitionListItemLoose * Cc(true))
5944           ) / writer.definitionlist
5945
5946     syntax.DefinitionList = DefinitionList
5947 end
5948 }
5949 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.

```

5950 M.extensions.fenced_code = function(blank_before_code_fence)
5951     return {
5952         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.

```

5953         function self.fencedCode(i, s)
5954             if not self.is_writing then return "" end
5955             s = string.gsub(s, '[\r\n%s]*$', '')
5956             local name = util.cache(self.cacheDir, s, nil, nil, ".verbatim")
5957             return {"\\markdownRendererInputFencedCode{"..name.."}{"..i.."}"}
5958         end
5959     end, extend_reader = function(self)
5960         local parsers = self.parsers
5961         local syntax = self.syntax
5962         local writer = self.writer

```

```

5963
5964         local function captures_geq_length(s,i,a,b)
5965             return #a >= #b and i
5966         end
5967
5968         local infostring = (parsers.linechar - (parsers.backtick
5969             + parsers.space^1 * (parsers.newline + parsers.eof)))^0

```

```

5970
5971     local fenceindent
5972     local fencehead      = function(char)
5973         return           C(parsers.nonindentSPACE) / function(s) fenceindent = #s end
5974                         * Cg(char^3, "fencelength")
5975                         * parsers.optionalspace * C(infostring)
5976                         * parsers.optionalspace * (parsers.newline + parsers.eof)
5977     end
5978
5979     local fencetail      = function(char)
5980         return           parsers.nonindentSPACE
5981                         * Cmt(C(char^3) * Cb("fencelength"), captures_geq_length)
5982                         * parsers.optionalspace * (parsers.newline + parsers.eof)
5983                         + parsers.eof
5984     end
5985
5986     local fencedline     = function(char)
5987         return           C(parsers.line - fencetail(char))
5988                         / function(s)
5989                             i = 1
5990                             remaining = fenceindent
5991                             while true do
5992                                 c = s:sub(i, i)
5993                                 if c == " " and remaining > 0 then
5994                                     remaining = remaining - 1
5995                                     i = i + 1
5996                                 elseif c == "\t" and remaining > 3 then
5997                                     remaining = remaining - 4
5998                                     i = i + 1
5999                                 else
6000                                     break
6001                                 end
6002                             end
6003                             return s:sub(i)
6004                         end
6005     end
6006
6007     local TildeFencedCode
6008         = fencehead(parsers.tilde)
6009         * Cs(fencedline(parsers.tilde)^0)
6010         * fencetail(parsers.tilde)
6011
6012     local BacktickFencedCode
6013         = fencehead(parsers.backtick)
6014         * Cs(fencedline(parsers.backtick)^0)
6015         * fencetail(parsers.backtick)
6016

```



```

6017     local FencedCode = (TildeFencedCode
6018                           + BacktickFencedCode)
6019     / function(infostring, code)
6020         return writer.fencedCode(writer.string(infostring),
6021                                   self.expandtabs(code))
6022     end
6023
6024     syntax.FencedCode = FencedCode
6025
6026     if blank_before_code_fence then
6027         fencestart = parsers.fail
6028     else
6029         fencestart = fencehead(parsers.backtick)
6030                     + fencehead(parsers.tilde)
6031     end
6032
6033     parsers.EndlineExceptions = parsers.EndlineExceptions + fencestart
6034     syntax.EndlineExceptions = parsers.EndlineExceptions
6035 end
6036 }
6037 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.

```

6038 M.extensions.footnotes = function(footnotes, inline_footnotes)
6039     assert(footnotes or inline_footnotes)
6040     return {
6041         extend_writer = function(self)

```

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

```

6042         function self.note(s)
6043             return {"\\markdownRendererFootnote{",s,""}
6044         end
6045     end, extend_reader = function(self)
6046         local parsers = self.parsers
6047         local syntax = self.syntax
6048         local writer = self.writer
6049
6050         if footnotes then
6051             local function strip_first_char(s)
6052                 return s:sub(2)
6053             end

```

```

6054
6055     local RawNoteRef
6056         = #(parsers.lbracket * parsers.circumflex)
6057         * parsers.tag / strip_first_char
6058
6059     local rawnotes = {}
6060
6061     -- like indirect_link
6062     local function lookup_note(ref)
6063         return writer.defer_call(function()
6064             local found = rawnotes[self.normalize_tag(ref)]
6065             if found then
6066                 return writer.note(
6067                     self.parser_functions.parse_blocks_nested(found))
6068             else
6069                 return {"[",
6070                     self.parser_functions.parse_inlines("^" .. ref), "]" }
6071             end
6072         end)
6073     end
6074
6075     local function register_note(ref, rawnote)
6076         rawnotes[self.normalize_tag(ref)] = rawnote
6077         return ""
6078     end
6079
6080     local NoteRef = RawNoteRef / lookup_note
6081
6082     local NoteBlock
6083         = parsers.leader * RawNoteRef * parsers.colon
6084         * parsers.spnl * parsers.indented_blocks(parsers.chunk)
6085         / register_note
6086
6087     parsers.Blank = NoteBlock + parsers.Blank
6088     syntax.Blank = parsers.Blank
6089
6090     syntax.NoteRef = NoteRef
6091 end
6092 if inline_footnotes then
6093     local InlineNote
6094         = parsers.circumflex
6095         * (parsers.tag / self.parser_functions.parse_inlines_no_inline_note)
6096         / writer.note
6097     syntax.InlineNote = InlineNote
6098 end
6099 end
6100 }

```

6101 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.

```
6102 M.extensions.header_attributes = function()
6103   return {
6104     extend_writer = function(self)
6105     end, extend_reader = function(self)
6106       local parsers = self.parsers
6107       local syntax = self.syntax
6108       local writer = self.writer
6109
6110       parsers.AtxHeading = Cg(parsers.HeadingStart,"level")
6111         * parsers.optionalspace
6112         * (C(((parsers.linechar
6113           - ((parsers.hash^1
6114             * parsers.optionalspace
6115             * parsers.HeadingAttributes~-1
6116             + parsers.HeadingAttributes)
6117             * parsers.optionalspace
6118             * parsers.newline))
6119           * (parsers.linechar
6120             - parsers.hash
6121             - parsers.lbrace)^0)^1)
6122           / self.parser_functions.parse_inlines)
6123         * Cg(Ct(parsers.newline
6124           + (parsers.hash^1
6125             * parsers.optionalspace
6126             * parsers.HeadingAttributes~-1
6127             + parsers.HeadingAttributes)
6128             * parsers.optionalspace
6129             * parsers.newline), "attributes")
6130         * Cb("level")
6131         * Cb("attributes")
6132         / writer.heading
6133
6134       parsers.SetextHeading = #(parsers.line * S("--"))
6135         * (C(((parsers.linechar
6136           - (parsers.HeadingAttributes
6137             * parsers.optionalspace
6138             * parsers.newline))
6139           * (parsers.linechar
6140             - parsers.lbrace)^0)^1)
6141           / self.parser_functions.parse_inlines)
6142         * Cg(Ct(parsers.newline
6143           + (parsers.HeadingAttributes
```

```

6144             * parsers.optionalspace
6145             * parsers.newline)), "attributes")
6146         * parsers.HeadingLevel
6147         * Cb("attributes")
6148         * parsers.optionalspace
6149         * parsers.newline
6150         / writer.heading
6151
6152     parsers.Heading = parsers.AtxHeading + parsers.SettextHeading
6153     syntax.Heading = parsers.Heading
6154 end
6155 }
6156 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

```

6157 M.extensions.jekyll_data = function(expect_jekyll_data)
6158   return {
6159     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`.

```

6160     function self.jekyllData(d, t, p)
6161       if not self.is_writing then return "" end
6162
6163       local buf = {}
6164
6165       local keys = {}
6166       for k, _ in pairs(d) do
6167         table.insert(keys, k)
6168       end
6169       table.sort(keys)
6170
6171       if not p then
6172         table.insert(buf, "\\markdownRendererJekyllDataBegin")
6173       end
6174
6175       if #d > 0 then
6176         table.insert(buf, "\\markdownRendererJekyllDataSequenceBegin{")
6177         table.insert(buf, self.uri(p or "null"))
6178         table.insert(buf, "}{")

```

```

6179         table.insert(buf, #keys)
6180         table.insert(buf, "}")
6181     else
6182         table.insert(buf, "\\markdownRendererJekyllDataMappingBegin{")
6183         table.insert(buf, self.uri(p or "null"))
6184         table.insert(buf, "{")
6185         table.insert(buf, #keys)
6186         table.insert(buf, "}")
6187     end
6188
6189     for _, k in ipairs(keys) do
6190         local v = d[k]
6191         local typ = type(v)
6192         k = tostring(k or "null")
6193         if typ == "table" and next(v) ~= nil then
6194             table.insert(
6195                 buf,
6196                 self.jekyllData(v, t, k)
6197             )
6198         else
6199             k = self.uri(k)
6200             v = tostring(v)
6201             if typ == "boolean" then
6202                 table.insert(buf, "\\markdownRendererJekyllDataBoolean{")
6203                 table.insert(buf, k)
6204                 table.insert(buf, "{")
6205                 table.insert(buf, v)
6206                 table.insert(buf, "}")
6207             elseif typ == "number" then
6208                 table.insert(buf, "\\markdownRendererJekyllDataNumber{")
6209                 table.insert(buf, k)
6210                 table.insert(buf, "{")
6211                 table.insert(buf, v)
6212                 table.insert(buf, "}")
6213             elseif typ == "string" then
6214                 table.insert(buf, "\\markdownRendererJekyllDataString{")
6215                 table.insert(buf, k)
6216                 table.insert(buf, "{")
6217                 table.insert(buf, t(v))
6218                 table.insert(buf, "}")
6219             elseif typ == "table" then
6220                 table.insert(buf, "\\markdownRendererJekyllDataEmpty{")
6221                 table.insert(buf, k)
6222                 table.insert(buf, "}")
6223             else
6224                 error(format("Unexpected type %s for value of " ..
6225                     "YAML key %s", typ, k))

```

```

6226         end
6227     end
6228 end
6229
6230     if #d > 0 then
6231         table.insert(buf, "\\markdownRendererJekyllDataSequenceEnd")
6232     else
6233         table.insert(buf, "\\markdownRendererJekyllDataMappingEnd")
6234     end
6235
6236     if not p then
6237         table.insert(buf, "\\markdownRendererJekyllDataEnd")
6238     end
6239
6240     return buf
6241 end
6242 end, extend_reader = function(self)
6243     local parsers = self.parsers
6244     local syntax = self.syntax
6245     local writer = self.writer
6246
6247     local JekyllData
6248         = Cmt( C((parsers.line - P("---") - P("..."))^0)
6249             , function(s, i, text)
6250                 local data
6251                 local ran_ok, error = pcall(function()
6252                     local tinyyaml = require("markdown-tinyyaml")
6253                     data = tinyyaml.parse(text, {timestamps=false})
6254                 end)
6255                 if ran_ok and data ~= nil then
6256                     return true, writer.jekyllData(data, function(s)
6257                         return self.parser_functions.parse_blocks_nested(s)
6258                     end, nil)
6259                 else
6260                     return false
6261                 end
6262             end
6263         )
6264
6265     local UnexpectedJekyllData
6266         = P("---")
6267         * parsers.blankline / 0
6268         * #(-parsers.blankline) -- if followed by blank, it's an hrule
6269         * JekyllData
6270         * (P("---") + P("..."))
6271
6272     local ExpectedJekyllData

```

```

6273             = ( P("----")
6274               * parsers.blankline / 0
6275               * #(-parsers.blankline) -- if followed by blank, it's an hrule
6276               )^-1
6277             * JekyllData
6278             * (P("----") + P("..."))^-1
6279
6280     syntax.UnexpectedJekyllData = UnexpectedJekyllData
6281     if expect_jekyll_data then
6282       syntax.ExpectedJekyllData = ExpectedJekyllData
6283     end
6284   end
6285 }
6286 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.

```

6287 M.extensions.pipe_tables = function(table_captions)
6288
6289   local function make_pipe_table_rectangular(rows)
6290     local num_columns = #rows[2]
6291     local rectangular_rows = {}
6292     for i = 1, #rows do
6293       local row = rows[i]
6294       local rectangular_row = {}
6295       for j = 1, num_columns do
6296         rectangular_row[j] = row[j] or ""
6297       end
6298       table.insert(rectangular_rows, rectangular_row)
6299     end
6300     return rectangular_rows
6301   end
6302
6303   local function pipe_table_row(allow_empty_first_column
6304                                , nonempty_column
6305                                , column_separator
6306                                , column)
6307     local row_beginning
6308     if allow_empty_first_column then
6309       row_beginning = -- empty first column
6310                       #(parsers.spacechar^4
6311                         * column_separator)
6312     * parsers.optionalspace
6313     * column

```

```

6314             * parsers.optionalspace
6315             -- non-empty first column
6316             + parsers.nonindentspace
6317             * nonempty_column~-1
6318             * parsers.optionalspace
6319         else
6320             row_beginning = parsers.nonindentspace
6321             * nonempty_column~-1
6322             * parsers.optionalspace
6323         end
6324
6325         return Ct(row_beginning
6326                 * (-- single column with no leading pipes
6327                   #(column_separator
6328                     * parsers.optionalspace
6329                     * parsers.newline)
6330                   * column_separator
6331                   * parsers.optionalspace
6332                   -- single column with leading pipes or
6333                   -- more than a single column
6334                   + (column_separator
6335                     * parsers.optionalspace
6336                     * column
6337                     * parsers.optionalspace)^1
6338                   * (column_separator
6339                     * parsers.optionalspace)^-1))
6340         end
6341
6342         return {
6343             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.

```

6344         function self.table(rows, caption)
6345             if not self.is_writing then return "" end
6346             local buffer = {"\\markdownRendererTable{",
6347                 caption or "", "{", #rows - 1, "{", #rows[1], "}"
6348             }
6349             local temp = rows[2] -- put alignments on the first row
6350             rows[2] = rows[1]
6351             rows[1] = temp
6352             for i, row in ipairs(rows) do
6353                 table.insert(buffer, "{")
6354                 for _, column in ipairs(row) do
6355                     if i > 1 then -- do not use braces for alignments
6356                         table.insert(buffer, "{")

```



```

6357         table.insert(buffer, column)
6358         if i > 1 then
6359             table.insert(buffer, "|")
6360         end
6361     end
6362     table.insert(buffer, "|")
6363 end
6364 return buffer
6365 end
6366 end, extend_reader = function(self)
6367     local parsers = self.parsers
6368     local syntax = self.syntax
6369     local writer = self.writer
6370
6371     local table_hline_separator = parsers.pipe + parsers.plus
6372
6373     local table_hline_column = (parsers.dash
6374         - #(parsers.dash
6375             * (parsers.spacechar
6376                 + table_hline_separator
6377                 + parsers.newline)))^1
6378         * (parsers.colon * Cc("r")
6379             + parsers.dash * Cc("d"))
6380         + parsers.colon
6381         * (parsers.dash
6382             - #(parsers.dash
6383                 * (parsers.spacechar
6384                     + table_hline_separator
6385                     + parsers.newline)))^1
6386         * (parsers.colon * Cc("c")
6387             + parsers.dash * Cc("l")))
6388
6389     local table_hline = pipe_table_row(false
6390         , table_hline_column
6391         , table_hline_separator
6392         , table_hline_column)
6393
6394     local table_caption_beginning = parsers.skipblanklines
6395         * parsers.nonindentspace
6396         * (P("Table")^-1 * parsers.colon)
6397         * parsers.optionalspace
6398
6399     local table_row = pipe_table_row(true
6400         , (C((parsers.linechar - parsers.pipe)^1)
6401             / self.parser_functions.parse_inlines)
6402         , parsers.pipe
6403         , (C((parsers.linechar - parsers.pipe)^0)

```

```

6404                                     / self.parser_functions.parse_inlines))
6405
6406     local table_caption
6407     if table_captions then
6408         table_caption = #table_caption_beginning
6409                         * table_caption_beginning
6410                         * Ct(parsers.IndentedInline~1)
6411                         * parsers.newline
6412     else
6413         table_caption = parsers.fail
6414     end
6415
6416     local PipeTable = Ct(table_row * parsers.newline
6417                         * table_hline
6418                         * (parsers.newline * table_row)^0)
6419                         / make_pipe_table_rectangular
6420                         * table_caption^-1
6421                         / writer.table
6422
6423     syntax.PipeTable = PipeTable
6424 end
6425 }
6426 end

```

3.1.7 Conversion from Markdown to Plain T_EX

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`.

```

6427 function M.new(options)
    Make the options table inherit from the defaultOptions table.
6428     options = options or {}
6429     setmetatable(options, { __index = function (_, key)
6430         return defaultOptions[key] end })
6431 % \par
6432 % \begin{markdown}
6433 %
6434 % Apply syntax extensions based on `options`.
6435 %
6436 % \end{markdown}
6437 % \begin{macrocode}
6438     extensions = {}
6439
6440     if options.citations then
6441         citations_extension = M.extensions.citations(options.citationNbsps)
6442         table.insert(extensions, citations_extension)

```

```

6443 end
6444
6445 if options.contentBlocks then
6446     content_blocks_extension = M.extensions.content_blocks(
6447         options.contentBlocksLanguageMap)
6448     table.insert(extensions, content_blocks_extension)
6449 end
6450
6451 if options.definitionLists then
6452     definition_lists_extension = M.extensions.definition_lists(
6453         options.tightLists)
6454     table.insert(extensions, definition_lists_extension)
6455 end
6456
6457 if options.fencedCode then
6458     fenced_code_extension = M.extensions.fenced_code(
6459         options.blankBeforeCodeFence)
6460     table.insert(extensions, fenced_code_extension)
6461 end
6462
6463 if options.footnotes or options.inlineFootnotes then
6464     footnotes_extension = M.extensions.footnotes(
6465         options.footnotes, options.inlineFootnotes)
6466     table.insert(extensions, footnotes_extension)
6467 end
6468
6469 if options.headerAttributes then
6470     header_attributes_extension = M.extensions.header_attributes()
6471     table.insert(extensions, header_attributes_extension)
6472 end
6473
6474 if options.jekyllData then
6475     jekyll_data_extension = M.extensions.jekyll_data(
6476         options.expectJekyllData)
6477     table.insert(extensions, jekyll_data_extension)
6478 end
6479
6480 if options.pipeTables then
6481     pipe_tables_extension = M.extensions.pipe_tables(
6482         options.tableCaptions)
6483     table.insert(extensions, pipe_tables_extension)
6484 end
6485
6486 local writer = M.writer.new(options)
6487 local reader = M.reader.new(writer, options, extensions)
6488
6489 return reader.convert

```

```

6490 end
6491
6492 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.

```

6493
6494 local input
6495 if input_filename then
6496     local input_file = assert(io.open(input_filename, "r"),
6497         [[could not open file ]] .. input_filename .. [[ for reading]])
6498     input = assert(input_file:read("*a"))
6499     assert(input_file:close())
6500 else
6501     input = assert(io.read("*a"))
6502 end
6503

```

First, ensure that the `options.cacheDir` directory exists.

```

6504 local lfs = require("lfs")
6505 if options.cacheDir and not lfs.isdir(options.cacheDir) then
6506     assert(lfs.mkdir(options["cacheDir"]))
6507 end
6508
6509 local ran_ok, kpse = pcall(require, "kpse")
6510 if ran_ok then kpse.set_program_name("luatex") end
6511 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.

```

6512 if metadata.version ~= md.metadata.version then
6513     warn("markdown-cli.lua " .. metadata.version .. " used with " ..
6514         "markdown.lua " .. md.metadata.version .. ".")
6515 end
6516 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.

```

6517 local output = convert(input:gsub("\r\n?", "\n") .. "\n")
6518
6519 if output_filename then
6520     local output_file = assert(io.open(output_filename, "w"),
6521         [[could not open file ]] .. output_filename .. [[ for writing]])
6522     assert(output_file:write(output))
6523     assert(output_file:close())
6524 else

```

```

6525  assert(io.write(output))
6526  end

```

3.2 Plain T_EX Implementation

The plain T_EX implementation provides macros for the interfacing between T_EX 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 T_EX exposed by the plain T_EX interface (see Section 2.2).

3.2.1 Logging Facilities

```

6527 \ifx\markdownInfo\undefined
6528   \def\markdownInfo#1{%
6529     \immediate\write-1{(1.\the\inputlineno) markdown.tex info: #1.}}%
6530 \fi
6531 \ifx\markdownWarning\undefined
6532   \def\markdownWarning#1{%
6533     \immediate\write16{(1.\the\inputlineno) markdown.tex warning: #1}}%
6534 \fi
6535 \ifx\markdownError\undefined
6536   \def\markdownError#1#2{%
6537     \errhelp{#2.}%
6538     \errmessage{(1.\the\inputlineno) markdown.tex error: #1}}%
6539 \fi

```

3.2.2 Token Renderer Prototypes

The following definitions should be considered placeholder.

```

6540 \def\markdownRendererInterblockSeparatorPrototype{\par}%
6541 \def\markdownRendererLineBreakPrototype{\hfil\break}%
6542 \let\markdownRendererEllipsisPrototype\dots
6543 \def\markdownRendererNbspPrototype{~}%
6544 \def\markdownRendererLeftBracePrototype{\char`\{}%
6545 \def\markdownRendererRightBracePrototype{\char`\}%
6546 \def\markdownRendererDollarSignPrototype{\char`$}%
6547 \def\markdownRendererPercentSignPrototype{\char`\}%
6548 \def\markdownRendererAmpersandPrototype{\&%
6549 \def\markdownRendererUnderscorePrototype{\char`_%
6550 \def\markdownRendererHashPrototype{\char`\#}%
6551 \def\markdownRendererCircumflexPrototype{\char`^}%
6552 \def\markdownRendererBackslashPrototype{\char`\}%
6553 \def\markdownRendererTildePrototype{\char`~}%
6554 \def\markdownRendererPipePrototype{|}%
6555 \def\markdownRendererCodeSpanPrototype#1{{\tt#1}}%
6556 \def\markdownRendererLinkPrototype#1#2#3#4{#2}%
6557 \def\markdownRendererContentBlockPrototype#1#2#3#4{%

```

```

6558 \markdownInput{#3}}%
6559 \def\markdownRendererContentBlockOnlineImagePrototype{%
6560 \markdownRendererImage}%
6561 \def\markdownRendererContentBlockCodePrototype#1#2#3#4#5{%
6562 \markdownRendererInputFencedCode{#3}{#2}}%
6563 \def\markdownRendererImagePrototype#1#2#3#4{#2}%
6564 \def\markdownRendererUlBeginPrototype{}%
6565 \def\markdownRendererUlBeginTightPrototype{}%
6566 \def\markdownRendererUlItemPrototype{}%
6567 \def\markdownRendererUlItemEndPrototype{}%
6568 \def\markdownRendererUlEndPrototype{}%
6569 \def\markdownRendererUlEndTightPrototype{}%
6570 \def\markdownRendererOlBeginPrototype{}%
6571 \def\markdownRendererOlBeginTightPrototype{}%
6572 \def\markdownRendererOlItemPrototype{}%
6573 \def\markdownRendererOlItemWithNumberPrototype#1{}%
6574 \def\markdownRendererOlItemEndPrototype{}%
6575 \def\markdownRendererOlEndPrototype{}%
6576 \def\markdownRendererOlEndTightPrototype{}%
6577 \def\markdownRendererDlBeginPrototype{}%
6578 \def\markdownRendererDlBeginTightPrototype{}%
6579 \def\markdownRendererDlItemPrototype#1{#1}%
6580 \def\markdownRendererDlItemEndPrototype{}%
6581 \def\markdownRendererDlDefinitionBeginPrototype{}%
6582 \def\markdownRendererDlDefinitionEndPrototype{\par}%
6583 \def\markdownRendererDlEndPrototype{}%
6584 \def\markdownRendererDlEndTightPrototype{}%
6585 \def\markdownRendererEmphasisPrototype#1{{\it#1}}%
6586 \def\markdownRendererStrongEmphasisPrototype#1{{\bf#1}}%
6587 \def\markdownRendererBlockQuoteBeginPrototype{\par\begingroup\it}%
6588 \def\markdownRendererBlockQuoteEndPrototype{\endgroup\par}%
6589 \def\markdownRendererInputVerbatimPrototype#1{%
6590 \par{\tt\input#1\relax{}}\par}%
6591 \def\markdownRendererInputFencedCodePrototype#1#2{%
6592 \markdownRendererInputVerbatimPrototype{#1}}%
6593 \def\markdownRendererHeadingOnePrototype#1{#1}%
6594 \def\markdownRendererHeadingTwoPrototype#1{#1}%
6595 \def\markdownRendererHeadingThreePrototype#1{#1}%
6596 \def\markdownRendererHeadingFourPrototype#1{#1}%
6597 \def\markdownRendererHeadingFivePrototype#1{#1}%
6598 \def\markdownRendererHeadingSixPrototype#1{#1}%
6599 \def\markdownRendererHorizontalRulePrototype{}%
6600 \def\markdownRendererFootnotePrototype#1{#1}%
6601 \def\markdownRendererCitePrototype#1{}%
6602 \def\markdownRendererTextCitePrototype#1{}%
6603 \def\markdownRendererTickedBoxPrototype{[X]}%
6604 \def\markdownRendererHalfTickedBoxPrototype{[/]}%

```

```
6605 \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 .

```
6606 \ExplSyntaxOn
6607 \seq_new:N \g_@@_jekyll_data_datatypes_seq
6608 \tl_const:Nn \c_@@_jekyll_data_sequence_tl { sequence }
6609 \tl_const:Nn \c_@@_jekyll_data_mapping_tl { mapping }
6610 \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.

```
6611 \seq_new:N \g_@@_jekyll_data_wildcard_absolute_address_seq
6612 \cs_new:Nn \markdown_jekyll_data_push_address_segment:n
6613 {
6614   \seq_if_empty:NF
6615     \g_@@_jekyll_data_datatypes_seq
6616     {
6617       \seq_get_right:NN
6618       \g_@@_jekyll_data_datatypes_seq
6619       \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.

```
6620   \str_if_eq:NNTF
6621     \l_tmpa_tl
6622     \c_@@_jekyll_data_sequence_tl
6623     {
6624       \seq_put_right:Nn
6625       \g_@@_jekyll_data_wildcard_absolute_address_seq
```

```

6626         { * }
6627     }
6628     {
6629         \seq_put_right:Nn
6630         \g_@@_jekyll_data_wildcard_absolute_address_seq
6631         { #1 }
6632     }
6633 }
6634 }

```

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.

```

6635 \tl_new:N \g_@@_jekyll_data_wildcard_absolute_address_tl
6636 \tl_new:N \g_@@_jekyll_data_wildcard_relative_address_tl
6637 \cs_new:Nn \markdown_jekyll_data_concatenate_address:NN
6638 {
6639     \seq_pop_left:NN #1 \l_tmpa_tl
6640     \tl_set:Nx #2 { / \seq_use:Nn #1 { / } }
6641     \seq_put_left:NV #1 \l_tmpa_tl
6642 }
6643 \cs_new:Nn \markdown_jekyll_data_update_address_tls:
6644 {

```



```

6645 \markdown_jekyll_data_concatenate_address:NN
6646 \g_@@_jekyll_data_wildcard_absolute_address_seq
6647 \g_@@_jekyll_data_wildcard_absolute_address_tl
6648 \seq_get_right:NN
6649 \g_@@_jekyll_data_wildcard_absolute_address_seq
6650 \g_@@_jekyll_data_wildcard_relative_address_tl
6651 }

```

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.

```

6652 \cs_new:Nn \markdown_jekyll_data_push:nN
6653 {
6654   \markdown_jekyll_data_push_address_segment:n
6655   { #1 }
6656   \seq_put_right:NV
6657   \g_@@_jekyll_data_datatypes_seq
6658   #2
6659   \markdown_jekyll_data_update_address_tls:
6660 }
6661 \cs_new:Nn \markdown_jekyll_data_pop:
6662 {
6663   \seq_pop_right:NN
6664   \g_@@_jekyll_data_wildcard_absolute_address_seq
6665   \l_tmpa_tl
6666   \seq_pop_right:NN
6667   \g_@@_jekyll_data_datatypes_seq
6668   \l_tmpa_tl
6669   \markdown_jekyll_data_update_address_tls:
6670 }

```

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.

```

6671 \cs_new:Nn \markdown_jekyll_data_set_keyval:nn
6672 {
6673   \keys_set_known:nn
6674   { markdown/jekyllData }
6675   { { #1 } = { #2 } }
6676 }
6677 \cs_generate_variant:Nn
6678 \markdown_jekyll_data_set_keyval:nn
6679 { Vn }
6680 \cs_new:Nn \markdown_jekyll_data_set_keyvals:nn
6681 {
6682   \markdown_jekyll_data_push:nN
6683   { #1 }
6684   \c_@@_jekyll_data_scalar_tl
6685   \markdown_jekyll_data_set_keyval:Vn

```

```

6686     \g_@@_jekyll_data_wildcard_absolute_address_tl
6687     { #2 }
6688     \markdown_jekyll_data_set_keyval:Vn
6689     \g_@@_jekyll_data_wildcard_relative_address_tl
6690     { #2 }
6691     \markdown_jekyll_data_pop:
6692 }

```

Finally, we will register our macros as token renderer prototypes to be able to react to the traversal of a YAML document.

```

6693 \def\markdownRendererJekyllDataSequenceBeginPrototype#1#2{
6694     \markdown_jekyll_data_push:nN
6695     { #1 }
6696     \c_@@_jekyll_data_sequence_tl
6697 }
6698 \def\markdownRendererJekyllDataMappingBeginPrototype#1#2{
6699     \markdown_jekyll_data_push:nN
6700     { #1 }
6701     \c_@@_jekyll_data_mapping_tl
6702 }
6703 \def\markdownRendererJekyllDataSequenceEndPrototype{
6704     \markdown_jekyll_data_pop:
6705 }
6706 \def\markdownRendererJekyllDataMappingEndPrototype{
6707     \markdown_jekyll_data_pop:
6708 }
6709 \def\markdownRendererJekyllDataBooleanPrototype#1#2{
6710     \markdown_jekyll_data_set_keyvals:nn
6711     { #1 }
6712     { #2 }
6713 }
6714 \def\markdownRendererJekyllDataEmptyPrototype#1{}
6715 \def\markdownRendererJekyllDataNumberPrototype#1#2{
6716     \markdown_jekyll_data_set_keyvals:nn
6717     { #1 }
6718     { #2 }
6719 }
6720 \def\markdownRendererJekyllDataStringPrototype#1#2{
6721     \markdown_jekyll_data_set_keyvals:nn
6722     { #1 }
6723     { #2 }
6724 }
6725 \ExplSyntaxOff

```

3.2.3 Lua Snippets

After the `\markdownPrepareLuaOptions` macro has been fully expanded, the

`\markdownLuaOptions` macro will expand 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).

```

6726 \ExplSyntaxOn
6727 \tl_new:N \g_@@_formatted_lua_options_tl
6728 \cs_new:Nn \@@_format_lua_options:
6729 {
6730   \tl_gclear:N
6731   \g_@@_formatted_lua_options_tl
6732   \seq_map_function:NN
6733   \g_@@_lua_options_seq
6734   \@@_format_lua_option:n
6735 }
6736 \cs_new:Nn \@@_format_lua_option:n
6737 {
6738   \@@_typecheck_option:n
6739   { #1 }
6740   \@@_get_option_type:nN
6741   { #1 }
6742   \l_tmpa_tl
6743   \bool_if:nTF
6744   {
6745     \str_if_eq_p:VV
6746     \l_tmpa_tl
6747     \c_@@_option_type_boolean_tl ||
6748     \str_if_eq_p:VV
6749     \l_tmpa_tl
6750     \c_@@_option_type_number_tl ||
6751     \str_if_eq_p:VV
6752     \l_tmpa_tl
6753     \c_@@_option_type_counter_tl
6754   }
6755   {
6756     \@@_get_option_value:nN
6757     { #1 }
6758     \l_tmpa_tl
6759     \tl_gput_right:Nx
6760     \g_@@_formatted_lua_options_tl
6761     { #1~== \l_tmpa_tl ,~ }
6762   }
6763   {
6764     \@@_get_option_value:nN
6765     { #1 }
6766     \l_tmpa_tl
6767     \tl_gput_right:Nx
6768     \g_@@_formatted_lua_options_tl
6769     { #1~== " \l_tmpa_tl " ,~ }
6770   }

```

```

6771 }
6772 \let\markdownPrepareLuaOptions=\@@_format_lua_options:
6773 \def\markdownLuaOptions{\g_@@_formatted_lua_options_tl }
6774 \ExplSyntaxOff

```

The `\markdownPrepare` macro contains the Lua code that is executed prior to any conversion from markdown to plain T_EX. It exposes the `convert` function for the use by any further Lua code.

```

6775 \def\markdownPrepare{%

```

First, ensure that the `\markdownOptionCacheDir` directory exists.

```

6776 local lfs = require("lfs")
6777 local cacheDir = "\markdownOptionCacheDir"
6778 if not lfs.isdir(cacheDir) then
6779     assert(lfs.mkdir(cacheDir))
6780 end

```

Next, load the `markdown` module and create a converter function using the plain T_EX options, which were serialized to a Lua table via the `\markdownLuaOptions` macro.

```

6781 local md = require("markdown")
6782 local convert = md.new(\markdownLuaOptions)
6783 }%

```

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.

```

6784 \ExplSyntaxOn
6785 \cs_new:Nn
6786   \@@_if_option:nTF
6787 {
6788   \@@_get_option_type:nN
6789   { #1 }
6790   \l_tmpa_tl
6791   \str_if_eq:NNF
6792   \l_tmpa_tl
6793   \c_@@_option_type_boolean_tl
6794   {
6795     \msg_error:nnxx
6796     { @@ }
6797     { expected-boolean-option }
6798     { #1 }
6799     { \l_tmpa_tl }
6800   }
6801   \@@_get_option_value:nN
6802   { #1 }

```

```

6803     \l_tmpa_tl
6804     \str_if_eq:NNTF
6805     \l_tmpa_tl
6806     \c_@@_option_value_true_tl
6807     { #2 }
6808     { #3 }
6809   }
6810 \msg_new:nnn
6811 { @@ }
6812 { expected-boolean-option }
6813 {
6814   Option~#1~has~type~#2,~
6815   but~a~boolean~was~expected.
6816 }
6817 \let\markdownIfOption=\@@_if_option:nTF
6818 \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.

```

6819 \csname newread\endcsname\markdownInputFileStream
6820 \csname newwrite\endcsname\markdownOutputFileStream

```

The `\markdownReadAndConvertTab` macro contains the tab character literal.

```

6821 \begingroup
6822 \catcode`\^^I=12%
6823 \gdef\markdownReadAndConvertTab{^^I}%
6824 \endgroup

```

The `\markdownReadAndConvert` macro is largely a rewrite of the $\text{\LaTeX 2}\epsilon$ `\filecontents` macro to plain \TeX .

```

6825 \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.

```

6826 \catcode`\^^M=13%
6827 \catcode`\^^I=13%
6828 \catcode`\|=0%
6829 \catcode`\=12%
6830 |catcode`@=14%
6831 |catcode`|=12@
6832 |gdef|markdownReadAndConvert#1#2{
6833   |begingroup@

```

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

```

6834     |markdownIfOption{frozenCache}{@}{@
6835         |immediate|openout|markdownOutputFileStream@
6836         |markdownOptionInputTempFileName|relax@
6837         |markdownInfo{Buffering markdown input into the temporary @
6838             input file "|markdownOptionInputTempFileName" and scanning @
6839             for the closing token sequence "#1"}@
6840     }@

```

Locally change the category of the special plain T_EX 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.

```

6841     |def|do##1{|catcode`##1=12}|dospecials@
6842     |catcode`| =12@
6843     |markdownMakeOther@

```

The `\markdownReadAndConvertStripPercentSigns` macro will process the individual lines of output, stripping 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 (\sim) are produced.

```

6844     |def|markdownReadAndConvertStripPercentSign##1{@
6845         |markdownIfOption{stripPercentSigns}{@
6846             |if##1%@
6847                 |expandafter|expandafter|expandafter@
6848                 |markdownReadAndConvertProcessLine@
6849             |else@
6850                 |expandafter|expandafter|expandafter@
6851                 |markdownReadAndConvertProcessLine@
6852                 |expandafter|expandafter|expandafter##1@
6853             |fi@
6854         }{@
6855             |expandafter@
6856             |markdownReadAndConvertProcessLine@
6857             |expandafter##1@
6858         }@
6859     }@

```

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 (\sim) are produced.

```

6860     |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.

```
6861      |ifx|relax##3|relax@
6862      |markdownIfOption{frozenCache}{-}{@
6863      |immediate|write|markdownOutputFileStream{##1}@
6864      }@
6865      |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 T_EX, `\input` the result of the conversion, and expand the ending control sequence.

```
6866      |def^^M{@
6867      |markdownInfo{The ending token sequence was found}@
6868      |markdownIfOption{frozenCache}{-}{@
6869      |immediate|closeout|markdownOutputFileStream@
6870      }@
6871      |endgroup@
6872      |markdownInput{@
6873      |markdownOptionOutputDir@
6874      /|markdownOptionInputTempFileName@
6875      }@
6876      #2}@
6877      |fi@
```

Repeat with the next line.

```
6878      ^^M}@
```

Make the tab character active at expansion time and make it expand to a literal tab character.

```
6879      |catcode`|^I=13@
6880      |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.

```
6881      |catcode`|^M=13@
6882      |def^^M##1^^M{@
6883      |def^^M###1^^M{@
6884      |markdownReadAndConvertStripPercentSign####1#1#1|relax}@
6885      ^^M}@
6886      ^^M}@
```

Reset the character categories back to the former state.

```
6887 |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.

```

6888 \ExplSyntaxOn
6889 \int_compare:nT
6890   { \markdownMode = 3 }
6891   {
6892     \markdownInfo{Using mode 3: ~The~lt3luabridge~package}
6893     \file_input:n { lt3luabridge.tex }
6894     \cs_new:Npn
6895       \markdownLuaExecute
6896       { \luabridgeExecute }
6897   }
6898 \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 Lua \TeX engine, their \TeX distribution contains it, and uses shell access to produce and execute Lua scripts using the \TeX Lua interpreter [1, Section 3.1.1].

```

6899 \ifnum\markdownMode<2\relax
6900 \ifnum\markdownMode=0\relax
6901   \markdownWarning{Using mode 0: Shell escape via write18
6902                     (deprecated, to be removed in Markdown 3.0.0)}%
6903 \else
6904   \markdownWarning{Using mode 1: Shell escape via os.execute
6905                     (deprecated, to be removed in Markdown 3.0.0)}%
6906 \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` (Lua \TeX , Pdf \TeX) or the `\shellescape` (X \TeX) 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.

```

6907 \ifx\pdfshellescape\undefined
6908   \ifx\shellescape\undefined
6909     \ifnum\markdownMode=0\relax
6910       \def\markdownExecuteShellEscape{1}%
6911     \else

```



```

6912     \def\markdownExecuteShellEscape{%
6913         \directlua{tex.sprint(status.shell_escape or "1")}}%
6914     \fi
6915 \else
6916     \let\markdownExecuteShellEscape\shellescape
6917 \fi
6918 \else
6919     \let\markdownExecuteShellEscape\pdfshellescape
6920 \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.

```

6921 \ifnum\markdownMode=0\relax
6922     \def\markdownExecuteDirect#1{\immediate\write18{#1}}%
6923 \else
6924     \def\markdownExecuteDirect#1{%
6925         \directlua{os.execute("\luaescapestring{#1}")}}%
6926 \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.

```

6927 \def\markdownExecute#1{%
6928     \ifnum\markdownExecuteShellEscape=1\relax
6929         \markdownExecuteDirect{#1}%
6930     \else
6931         \markdownError{I can not access the shell}{Either run the TeX
6932             compiler with the --shell-escape or the --enable-write18 flag,
6933             or set shell_escape=t in the texmf.cnf file}%
6934     \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.

```

6935 \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.

```

6936     \catcode`\|=0%
6937     \catcode`\|=12%
6938     \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.

```

6939         \immediate\openout\markdownOutputFileStream=%
6940         \markdownOptionHelperScriptFileName

```

```

6941 |markdownInfo{Writing a helper Lua script to the file
6942   "|markdownOptionHelperScriptFileName"}%
6943 |immediate|write|markdownOutputFileStream{%
6944   local ran_ok, error = pcall(function()
6945     local ran_ok, kpse = pcall(require, "kpse")
6946     if ran_ok then kpse.set_program_name("luatex") end
6947     #1
6948   end)

```

If there was an error, use the file `\markdownOptionErrorTempFileName` to store the error message.

```

6949   if not ran_ok then
6950     local file = io.open("%
6951       |markdownOptionOutputDir
6952       /|markdownOptionErrorTempFileName", "w")
6953     if file then
6954       file:write(error .. "\n")
6955       file:close()
6956     end
6957     print('\|markdownError{An error was encountered while executing
6958       Lua code}{For further clues, examine the file
6959       "|markdownOptionOutputDir
6960       /|markdownOptionErrorTempFileName"}')
6961   end}%
6962 |immediate|closeout|markdownOutputFileStream

```

Execute the generated `\markdownOptionHelperScriptFileName` Lua script using the `TeXLua` binary and store the output in the `\markdownOptionOutputTempFileName` file.

```

6963 |markdownInfo{Executing a helper Lua script from the file
6964   "|markdownOptionHelperScriptFileName" and storing the result in the
6965   file "|markdownOptionOutputTempFileName"}%
6966 |markdownExecute{texlua "|markdownOptionOutputDir
6967   /|markdownOptionHelperScriptFileName" > %
6968   "|markdownOptionOutputDir
6969   /|markdownOptionOutputTempFileName"}%

```

`\input` the generated `\markdownOptionOutputTempFileName` file.

```

6970 |input|markdownOptionOutputTempFileName|relax}%
6971 |endgroup

```

3.2.6 Direct Lua Access

The following `TeX` code is intended for `TeX` engines that provide direct access to Lua (Lua`TeX`). 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.

```

6972 \fi
6973 \ifnum\markdownMode=2\relax
6974   \markdownWarning{Using mode 2: Direct Lua access
6975                   (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,

```

6976 \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.

```

6977   \catcode`\|=0%
6978   \catcode`\|=12%
6979   \gdef\markdownLuaExecute#1{%
6980     \directlua{%
6981       local function print(input)
6982         local output = {}
6983         for line in input:gmatch("[^\r\n]+") do
6984           table.insert(output, line)
6985         end
6986         tex.print(output)
6987       end
6988       #1
6989     }%
6990   }%
6991 \endgroup
6992 \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.

```

6993 \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.

```

6994   \catcode`\|=0%
6995   \catcode`\|=12%
6996   \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.

```

6997     \begingroup
6998     \catcode`\|=12

```

If we are reading from the frozen cache, input it, expand the corresponding `\markdownFrozenCache⟨number⟩` macro, and increment `frozenCacheCounter`.

```

6999 |markdownIfOption{frozenCache}{%
7000 |ifnum|markdownOptionFrozenCacheCounter=0|relax
7001 |markdownInfo{Reading frozen cache from
7002 |"|markdownOptionFrozenCacheFileName"|}%
7003 |input|markdownOptionFrozenCacheFileName|relax
7004 |fi
7005 |markdownInfo{Including markdown document number
7006 |"|the|markdownOptionFrozenCacheCounter" from frozen cache}%
7007 |csname markdownFrozenCache|the|markdownOptionFrozenCacheCounter|endcsname
7008 |global|advance|markdownOptionFrozenCacheCounter by 1|relax
7009 }{%
7010 |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 L^AT_EX_Mk to track changes to the markdown document.

```

7011 |openin|markdownInputFileStream#1
7012 |closein|markdownInputFileStream
7013 |markdownPrepareLuaOptions
7014 |markdownLuaExecute{%
7015 |markdownPrepare
7016 |local file = assert(io.open("#1", "r"),
7017 |[[could not open file "#1" for reading]])
7018 |local input = assert(file:read("*a"))
7019 |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.

```

7020 |print(convert(input:gsub("\r\n?", "\n") .. "\n"))}%

```

If we are finalizing the frozen cache, increment `frozenCacheCounter`.

```

7021 |markdownIfOption{finalizeCache}{%
7022 |global|advance|markdownOptionFrozenCacheCounter by 1|relax
7023 }%
7024 }%
7025 |endgroup
7026 }%
7027 |endgroup

```

3.3 L^AT_EX Implementation

The L^AT_EX implemenation makes use of the fact that, apart from some subtle differences, L^AT_EX implements the majority of the plain T_EX format [10, Section 9]. As a consequence, we can directly reuse the existing plain T_EX implementation.

```

7028 \def\markdownVersionSpace{ }%

```

```

7029 \ProvidesPackage{markdown}[\markdownLastModified\markdownVersionSpace v%
7030 \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.

```

7031 \ExplSyntaxOn
7032 \@@_latex_define_renderers:
7033 \@@_latex_define_renderer_prototypes:
7034 \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).

```

7035 \let\markdownInputPlainTeX\markdownInput
7036 \renewcommand\markdownInput[2][]{\%
7037 \begingroup
7038 \markdownSetup{#1}%
7039 \markdownInputPlainTeX{#2}%
7040 \endgroup}%

```

The `markdown`, and `markdown*` \LaTeX environments are implemented using the `\markdownReadAndConvert` macro.

```

7041 \renewenvironment{markdown}{\%
7042 \markdownReadAndConvert@markdown{}}{\%
7043 \markdownEnd}%
7044 \renewenvironment{markdown*}[1]{\%
7045 \markdownSetup{#1}%
7046 \markdownReadAndConvert@markdown*}{\%
7047 \markdownEnd}%
7048 \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*.

```

7049 \catcode`\|=0\catcode`\<=1\catcode`\>=2%
7050 \catcode`\|=12\catcode`\{=12\catcode`\}=12%
7051 \gdef\markdownReadAndConvert@markdown#1<%

```

```

7052 |markdownReadAndConvert<\end{markdown#1}>%
7053 <|end<markdown#1>>>%
7054 |endgroup

```

3.3.2.1 \LaTeX Themes This section implements the theme-loading mechanism and the example themes provided with the Markdown package.

```

7055 \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.

```

7056 \newcommand\markdownLaTeXThemeName{}
7057 \seq_new:N \g_@@_latex_themes_seq
7058 \seq_put_right:NV
7059   \g_@@_latex_themes_seq
7060   \markdownLaTeXThemeName
7061 \newcommand\markdownLaTeXThemeLoad[2]{
7062   \def\@tempa{%
7063     \def\markdownLaTeXThemeName{#2}
7064     \seq_put_right:NV
7065       \g_@@_latex_themes_seq
7066       \markdownLaTeXThemeName
7067     \RequirePackage{#1}
7068     \seq_pop_right:NN
7069       \g_@@_latex_themes_seq
7070     \l_tmpa_tl
7071     \seq_get_right:NN
7072       \g_@@_latex_themes_seq
7073     \l_tmpa_tl
7074     \exp_args:NNV
7075       \def
7076         \markdownLaTeXThemeName
7077         \l_tmpa_tl}
7078 \ifmarkdownLaTeXLoaded
7079   \@tempa
7080 \else
7081   \exp_args:No
7082     \AtEndOfPackage
7083     { \@tempa }
7084 \fi}
7085 \ExplSyntaxOff

```

The `witiko/dot` theme enables the `fencedCode` Lua option:

```

7086 \markdownSetup{fencedCode}%

```

We load the `ifthen` and `grffile` packages, see also Section 1.1.3:

```

7087 \RequirePackage{ifthen,grffile}

```

We store the previous definition of the fenced code token renderer prototype:

```

7088 \let\markdown@witiko@dot@oldRendererInputFencedCodePrototype
7089 \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:

```

7090 \renewcommand\markdownRendererInputFencedCode[2]{%
7091   \def\next##1 ##2\relax{%
7092     \ifthenelse{\equal{##1}{dot}}{%
7093       \markdownIfOption{frozenCache}{}{%
7094         \immediate\write18{%
7095           if ! test -e #1.pdf.source || ! diff #1 #1.pdf.source;
7096           then
7097             dot -Tpdf -o #1.pdf #1;
7098             cp #1 #1.pdf.source;
7099             fi}}%

```

We include the typeset image using the image token renderer:

```

7100   \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:

```

7101   }{%
7102   \markdown@witiko@dot@oldRendererInputFencedCodePrototype{#1}{#2}%
7103   }%
7104 }%
7105 \next#2 \relax}%

```

The `witiko/graphicx/http` theme stores the previous definition of the image token renderer prototype:

```

7106 \let\markdown@witiko@graphicx@http@oldRendererImagePrototype
7107 \markdownRendererImagePrototype

```

We load the catchfile and grffile packages, see also Section 1.1.3:

```

7108 \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.

```

7109 \newcount\markdown@witiko@graphicx@http@counter
7110 \markdown@witiko@graphicx@http@counter=0
7111 \newcommand\markdown@witiko@graphicx@http@filename{%
7112   \markdownOptionCacheDir/witiko_graphicx_http%
7113   .\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 download the online image to the pathname.

```
7114 \newcommand\markdown@witiko@graphicx@http@download[2]{%
7115   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:

```
7116 \begingroup
7117 \catcode`\%=12
7118 \catcode`\^A=14
```

We redefine the image token renderer prototype, so that it tries to download an online image.

```
7119 \global\def\markdownRendererImagePrototype#1#2#3#4{^A
7120   \begingroup
7121   \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:

```
7122   \markdownIfOption{frozenCache}{}{^A
7123     \immediate\write18{^A
7124       mkdir -p "\markdownOptionCacheDir";
7125       if printf '%s' "#3" | grep -q -E '^https?:';
7126       then
```

The image will be downloaded to the pathname `\markdownOptionCacheDir/⟨the MD5 digest of the image URL⟩.⟨the suffix of the image URL⟩`:

```
7127       OUTPUT_PREFIX="\markdownOptionCacheDir";
7128       OUTPUT_BODY="$(printf '%s' '#3' | md5sum | cut -d' ' -f1)";
7129       OUTPUT_SUFFIX="$(printf '%s' '#3' | sed 's/.*[.]/')";
7130       OUTPUT="$OUTPUT_PREFIX/$OUTPUT_BODY.$OUTPUT_SUFFIX";
```

The image will be downloaded only if it has not already been downloaded:

```
7131       if ! [ -e "$OUTPUT" ];
7132       then
7133         \markdown@witiko@graphicx@http@download{'#3'}{"$OUTPUT"};
7134         printf '%s' "$OUTPUT" > "\filename";
7135       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:

```
7136       else
7137         printf '%s' '#3' > "\filename";
7138       fi}}^A
```

We load the pathname of the downloaded image and we typeset the image using the previous definition of the image renderer prototype:

```
7139   \CatchFileDef{\filename}{\filename}{\newlinechar=-1}^A
7140   \markdown@witiko@graphicx@http@oldRendererImagePrototype^A
```



```

7141      {#1}{#2}{\filename}{#4}^^A
7142  \endgroup
7143  \global\advance\markdown@witiko@graphicx@http@counter by 1\relax}^^A
7144  \endgroup

```

The `witiko/tilde` theme redefines the tilde token renderer prototype, so that it expands to a non-breaking space:

```

7145  \renewcommand\markdownRendererTildePrototype{~}%

```

3.3.3 Options

The supplied package options are processed using the `\markdownSetup` macro.

```

7146  \DeclareOption*{%
7147    \expandafter\markdownSetup\expandafter{\CurrentOption}}%
7148  \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 L^AT_EX setup snippets.

```

7149  \define@key{markdownOptions}{renderers}{%
7150    \setkeys{markdownRenderers}{#1}%
7151    \def\KV@prefix{KV@markdownOptions@}}%
7152  \define@key{markdownOptions}{rendererPrototypes}{%
7153    \setkeys{markdownRendererPrototypes}{#1}%
7154    \def\KV@prefix{KV@markdownOptions@}}%
7155  \define@key{markdownOptions}{code}{#1}%

```

3.3.4 Token Renderer Prototypes

The following configuration should be considered placeholder. If the `plain` package option has been enabled (see Section 2.3.2.1), none of it will take effect.

```

7156  \markdownIfOption{plain}{\iffalse}{\iftrue}

```

If the `tightLists` Lua option is disabled or the current document class is beamer, do not load the `paralist` package.

```

7157  \markdownIfOption{tightLists}{
7158    \@ifclassloaded{beamer}{}{\RequirePackage{paralist}}%
7159  }{}

```

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.

```

7160  \@ifpackageloaded{paralist}{
7161    \markdownSetup{rendererPrototypes={
7162      ulBeginTight = {\begin{compactitem}},
7163      ulEndTight = {\end{compactitem}},
7164      olBeginTight = {\begin{compactenum}},

```

```

7165     olEndTight = {\end{compactenum}},
7166     dlBeginTight = {\begin{compactdesc}},
7167     dlEndTight = {\end{compactdesc}}}}
7168 }{
7169   \markdownSetup{rendererPrototypes={
7170     ulBeginTight = {\markdownRendererUlBegin},
7171     ulEndTight = {\markdownRendererUlEnd},
7172     olBeginTight = {\markdownRendererOlBegin},
7173     olEndTight = {\markdownRendererOlEnd},
7174     dlBeginTight = {\markdownRendererDlBegin},
7175     dlEndTight = {\markdownRendererDlEnd}}}}
7176 \RequirePackage{amsmath,ifthen}

```

Unless the unicode-math package has been loaded, load the amssymb package with symbols to be used for tickboxes.

```

7177 \@ifpackageloaded{unicode-math}{
7178   \markdownSetup{rendererPrototypes={
7179     untickedBox = {\$ \mdlgwhtsquare$},
7180   }}
7181 }{
7182   \RequirePackage{amssymb}
7183   \markdownSetup{rendererPrototypes={
7184     untickedBox = {\$ \square$},
7185   }}
7186 }
7187 \RequirePackage{csvsimple}
7188 \RequirePackage{fancyvrb}
7189 \RequirePackage{graphicx}
7190 \markdownSetup{rendererPrototypes={
7191   lineBreak = {\},
7192   leftBrace = {\textbraceleft},
7193   rightBrace = {\textbraceright},
7194   dollarSign = {\textdollar},
7195   underscore = {\textunderscore},
7196   circumflex = {\textasciicircum},
7197   backslash = {\textbackslash},
7198   tilde = {\textasciitilde},
7199   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.

⁸This 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.

Here, we will redefine the code span renderer prototype to typeset upright text in math formulae and typewriter text outside math formulae.

```

7200 codeSpan = {%
7201   \ifmmode
7202     \text{#1}%
7203   \else
7204     \texttt{#1}%
7205   \fi
7206 },
7207 contentBlock = {%
7208   \ifthenelse{\equal{#1}{csv}}{%
7209     \begin{table}%
7210       \begin{center}%
7211         \csvautotabular{#3}%
7212       \end{center}
7213     \ifx\empty#4\empty\else
7214       \caption{#4}%
7215     \fi
7216   \end{table}%
7217 }{%
7218   \ifthenelse{\equal{#1}{tex}}{%
7219     \catcode`\%=14\relax
7220     \input #3\relax
7221     \catcode`\%=12\relax
7222   }{%
7223     \markdownInput{#3}%
7224   }%
7225 }%
7226 },
7227 image = {%
7228   \begin{figure}%
7229     \begin{center}%
7230       \includegraphics{#3}%
7231     \end{center}%
7232     \ifx\empty#4\empty\else
7233       \caption{#4}%
7234     \fi
7235   \end{figure}},
7236 ulBegin = {\begin{itemize}},
7237 ulEnd = {\end{itemize}},
7238 olBegin = {\begin{enumerate}},
7239 olItem = {\item{}},
7240 olItemWithNumber = {\item[#1.]},
7241 olEnd = {\end{enumerate}},
7242 dlBegin = {\begin{description}},
7243 dlItem = {\item[#1]},
7244 dlEnd = {\end{description}},

```

```

7245 emphasis = {\emph{#1}},
7246 tickedBox = {\$\boxtimes$},
7247 halfTickedBox = {\$\boxdot$},

```

If identifier attributes appear at the beginning of a section, we make the next heading produce the `\label` macro.

```

7248 headerAttributeContextBegin = {
7249   \markdownSetup{
7250     rendererPrototypes = {
7251       attributeIdentifier = {%
7252         \beginGroup
7253         \def\next####1{%
7254           \def####1#####1{%
7255             \endGroup
7256             ####1{#####1}%
7257             \label{##1}%
7258           }%
7259         }%
7260         \next\markdownRendererHeadingOne
7261         \next\markdownRendererHeadingTwo
7262         \next\markdownRendererHeadingThree
7263         \next\markdownRendererHeadingFour
7264         \next\markdownRendererHeadingFive
7265         \next\markdownRendererHeadingSix
7266       },
7267     },
7268   }%
7269 },
7270 blockQuoteBegin = {\begin{quotation}},
7271 blockQuoteEnd = {\end{quotation}},
7272 inputVerbatim = {\VerbatimInput{#1}},
7273 inputFencedCode = {%
7274   \ifx\relax#2\relax
7275     \VerbatimInput{#1}%
7276   \else
7277     \@ifundefined{minted@code}{%
7278       \@ifundefined{lst@version}{%
7279         \markdownRendererInputFencedCode{#1}{}%

```

When the listings package is loaded, use it for syntax highlighting.

```

7280   }{%
7281     \lstinputlisting[language=#2]{#1}%
7282   }%

```

When the minted package is loaded, use it for syntax highlighting. The minted package is preferred over listings.

```

7283   }{%
7284     \inputminted{#2}{#1}%

```

```

7285     }%
7286     \fi},
7287     horizontalRule = {\noindent\rule[0.5ex]{\linewidth}{1pt}},
7288     footnote = {\footnote{#1}}}}

```

Support the nesting of strong emphasis.

```

7289 \def\markdownLATEXStrongEmphasis#1{%
7290   \IfSubStr\f@series{b}{\textnormal{#1}}{\textbf{#1}}}
7291 \markdownSetup{rendererPrototypes={strongEmphasis={%
7292   \protect\markdownLATEXStrongEmphasis{#1}}}}

```

Support L^AT_EX document classes that do not provide chapters.

```

7293 \@ifundefined{chapter}{%
7294   \markdownSetup{rendererPrototypes = {
7295     headingOne = {\section{#1}},
7296     headingTwo = {\subsection{#1}},
7297     headingThree = {\subsubsection{#1}},
7298     headingFour = {\paragraph{#1}\leavevmode},
7299     headingFive = {\subparagraph{#1}\leavevmode}}}
7300 }{%
7301   \markdownSetup{rendererPrototypes = {
7302     headingOne = {\chapter{#1}},
7303     headingTwo = {\section{#1}},
7304     headingThree = {\subsection{#1}},
7305     headingFour = {\subsubsection{#1}},
7306     headingFive = {\paragraph{#1}\leavevmode},
7307     headingSix = {\subparagraph{#1}\leavevmode}}}
7308 }%

```

3.3.4.1 Tickboxes If the `taskLists` option is enabled, we will hide bullets in unordered list items with tickboxes.

```

7309 \markdownSetup{
7310   rendererPrototypes = {
7311     ulItem = {%
7312       \futurelet\markdownLaTeXCheckbox\markdownLaTeXUListItem
7313     },
7314   },
7315 }
7316 \def\markdownLaTeXUListItem{%
7317   \ifx\markdownLaTeXCheckbox\markdownRendererTickedBox
7318     \item[\markdownLaTeXCheckbox]%
7319     \expandafter\@gobble
7320   \else
7321     \ifx\markdownLaTeXCheckbox\markdownRendererHalfTickedBox
7322       \item[\markdownLaTeXCheckbox]%
7323       \expandafter\expandafter\expandafter\@gobble
7324     \else

```

```

7325     \ifx\markdownLaTeXCheckbox\markdownRendererUntickedBox
7326     \item[\markdownLaTeXCheckbox]%
7327     \expandafter\expandafter\expandafter\expandafter
7328     \expandafter\expandafter\expandafter\@gobble
7329     \else
7330     \item{}%
7331     \fi
7332 \fi
7333 \fi
7334 }

```

3.3.4.2 HTML elements If the `html` option is enabled and we are using $\text{\TeX}4\text{ht}$ ⁹, we will pass HTML elements to the output HTML document unchanged.

```

7335 \@ifundefined{HCode}{-}{-}
7336 \markdownSetup{
7337   rendererPrototypes = {
7338     inlineHtmlTag = {%
7339       \ifvmode
7340       \IgnorePar
7341       \EndP
7342       \fi
7343       \HCode{#1}%
7344     },
7345     inputBlockHtmlElement = {%
7346       \ifvmode
7347       \IgnorePar
7348       \fi
7349       \EndP
7350       \special{t4ht*<#1}%
7351       \par
7352       \ShowPar
7353     },
7354   },
7355 }
7356 }

```

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 `Bib \LaTeX` `\autocites` and `\textcites` macros. These implementations will be used, when the respective packages are loaded.

```

7357 \newcount\markdownLaTeXCitationsCounter
7358
7359 % Basic implementation
7360 \RequirePackage{gobble}

```

⁹See <https://tug.org/tex4ht/>.

```

7361 \def\markdownLaTeXBasicCitations#1#2#3#4#5#6{%
7362   \advance\markdownLaTeXCitationsCounter by 1\relax
7363   \ifx\relax#4\relax
7364     \ifx\relax#5\relax
7365       \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
7366         \cite{#1#2#6}% Without prenotes and postnotes, just accumulate cites
7367         \expandafter\expandafter\expandafter
7368         \expandafter\expandafter\expandafter\expandafter
7369         \@gobblethree
7370       \fi
7371     \else% Before a postnote (#5), dump the accumulator
7372       \ifx\relax#1\relax\else
7373         \cite{#1}%
7374       \fi
7375       \cite[#5]{#6}%
7376       \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
7377       \else
7378         \expandafter\expandafter\expandafter
7379         \expandafter\expandafter\expandafter\expandafter
7380         \expandafter\expandafter\expandafter
7381         \expandafter\expandafter\expandafter\expandafter
7382         \markdownLaTeXBasicCitations
7383       \fi
7384       \expandafter\expandafter\expandafter
7385       \expandafter\expandafter\expandafter\expandafter{%
7386       \expandafter\expandafter\expandafter
7387       \expandafter\expandafter\expandafter\expandafter}%
7388       \expandafter\expandafter\expandafter
7389       \expandafter\expandafter\expandafter\expandafter{%
7390       \expandafter\expandafter\expandafter
7391       \expandafter\expandafter\expandafter\expandafter}%
7392       \expandafter\expandafter\expandafter
7393       \@gobblethree
7394     \fi
7395   \else% Before a prenote (#4), dump the accumulator
7396     \ifx\relax#1\relax\else
7397       \cite{#1}%
7398     \fi
7399     \ifnum\markdownLaTeXCitationsCounter>1\relax
7400       \space % Insert a space before the prenote in later citations
7401     \fi
7402     #4~\expandafter\cite\ifx\relax#5\relax{#6}\else[#5]{#6}\fi
7403     \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
7404     \else
7405       \expandafter\expandafter\expandafter
7406       \expandafter\expandafter\expandafter\expandafter
7407       \markdownLaTeXBasicCitations

```

```

7408 \fi
7409 \expandafter\expandafter\expandafter{%
7410 \expandafter\expandafter\expandafter}%
7411 \expandafter\expandafter\expandafter{%
7412 \expandafter\expandafter\expandafter}%
7413 \expandafter
7414 \@gobblethree
7415 \fi\markdownLaTeXBasicCitations{#1#2#6},}
7416 \let\markdownLaTeXBasicTextCitations\markdownLaTeXBasicCitations
7417
7418 % Natbib implementation
7419 \def\markdownLaTeXNatbibCitations#1#2#3#4#5{%
7420 \advance\markdownLaTeXCitationsCounter by 1\relax
7421 \ifx\relax#3\relax
7422 \ifx\relax#4\relax
7423 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
7424 \citep{#1,#5}% Without prenotes and postnotes, just accumulate cites
7425 \expandafter\expandafter\expandafter
7426 \expandafter\expandafter\expandafter\expandafter
7427 \@gobbletwo
7428 \fi
7429 \else% Before a postnote (#4), dump the accumulator
7430 \ifx\relax#1\relax\else
7431 \citep{#1}%
7432 \fi
7433 \citep[] [#4]{#5}%
7434 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
7435 \else
7436 \expandafter\expandafter\expandafter
7437 \expandafter\expandafter\expandafter\expandafter
7438 \expandafter\expandafter\expandafter
7439 \expandafter\expandafter\expandafter\expandafter
7440 \markdownLaTeXNatbibCitations
7441 \fi
7442 \expandafter\expandafter\expandafter
7443 \expandafter\expandafter\expandafter\expandafter{%
7444 \expandafter\expandafter\expandafter
7445 \expandafter\expandafter\expandafter\expandafter}%
7446 \expandafter\expandafter\expandafter
7447 \@gobbletwo
7448 \fi
7449 \else% Before a prenote (#3), dump the accumulator
7450 \ifx\relax#1\relax\relax\else
7451 \citep{#1}%
7452 \fi
7453 \citep[#3] [#4]{#5}%
7454 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax

```



```

7455 \else
7456 \expandafter\expandafter\expandafter
7457 \expandafter\expandafter\expandafter\expandafter
7458 \markdownLaTeXNatbibCitations
7459 \fi
7460 \expandafter\expandafter\expandafter{%
7461 \expandafter\expandafter\expandafter}%
7462 \expandafter
7463 \@gobbletwo
7464 \fi\markdownLaTeXNatbibCitations{#1,#5}}
7465 \def\markdownLaTeXNatbibTextCitations#1#2#3#4#5{%
7466 \advance\markdownLaTeXCitationsCounter by 1\relax
7467 \ifx\relax#3\relax
7468 \ifx\relax#4\relax
7469 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
7470 \citet{#1,#5}% Without prenotes and postnotes, just accumulate cites
7471 \expandafter\expandafter\expandafter
7472 \expandafter\expandafter\expandafter\expandafter
7473 \@gobbletwo
7474 \fi
7475 \else% After a prenote or a postnote, dump the accumulator
7476 \ifx\relax#1\relax\else
7477 \citet{#1}%
7478 \fi
7479 , \citet[#3][#4]{#5}%
7480 \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
7481 ,
7482 \else
7483 \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
7484 ,
7485 \fi
7486 \fi
7487 \expandafter\expandafter\expandafter
7488 \expandafter\expandafter\expandafter\expandafter
7489 \markdownLaTeXNatbibTextCitations
7490 \expandafter\expandafter\expandafter
7491 \expandafter\expandafter\expandafter\expandafter{%
7492 \expandafter\expandafter\expandafter
7493 \expandafter\expandafter\expandafter\expandafter}%
7494 \expandafter\expandafter\expandafter
7495 \@gobbletwo
7496 \fi
7497 \else% After a prenote or a postnote, dump the accumulator
7498 \ifx\relax#1\relax\relax\else
7499 \citet{#1}%
7500 \fi
7501 , \citet[#3][#4]{#5}%

```

```

7502 \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
7503 ,
7504 \else
7505 \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
7506 ,
7507 \fi
7508 \fi
7509 \expandafter\expandafter\expandafter
7510 \markdownLaTeXNatbibTextCitations
7511 \expandafter\expandafter\expandafter{%
7512 \expandafter\expandafter\expandafter}%
7513 \expandafter
7514 \@gobbletwo
7515 \fi\markdownLaTeXNatbibTextCitations{#1,#5}}
7516
7517 % BibLaTeX implementation
7518 \def\markdownLaTeXBibLaTeXCitations#1#2#3#4#5{%
7519 \advance\markdownLaTeXCitationsCounter by 1\relax
7520 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
7521 \autocites#1[#3][#4]{#5}%
7522 \expandafter\@gobbletwo
7523 \fi\markdownLaTeXBibLaTeXCitations{#1[#3][#4]{#5}}}
7524 \def\markdownLaTeXBibLaTeXTextCitations#1#2#3#4#5{%
7525 \advance\markdownLaTeXCitationsCounter by 1\relax
7526 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
7527 \textcites#1[#3][#4]{#5}%
7528 \expandafter\@gobbletwo
7529 \fi\markdownLaTeXBibLaTeXTextCitations{#1[#3][#4]{#5}}}
7530
7531 \markdownSetup{rendererPrototypes = {
7532 cite = {%
7533 \markdownLaTeXCitationsCounter=1%
7534 \def\markdownLaTeXCitationsTotal{#1}%
7535 \@ifundefined{autocites}{%
7536 \@ifundefined{citep}{%
7537 \expandafter\expandafter\expandafter
7538 \markdownLaTeXBasicCitations
7539 \expandafter\expandafter\expandafter{%
7540 \expandafter\expandafter\expandafter}%
7541 \expandafter\expandafter\expandafter{%
7542 \expandafter\expandafter\expandafter}%
7543 }{%
7544 \expandafter\expandafter\expandafter
7545 \markdownLaTeXNatbibCitations
7546 \expandafter\expandafter\expandafter{%
7547 \expandafter\expandafter\expandafter}%
7548 }%

```

```

7549 }{%
7550   \expandafter\expandafter\expandafter
7551   \markdownLaTeXBibLaTeXCitations
7552   \expandafter{\expandafter}%
7553 },
7554 textCite = {%
7555   \markdownLaTeXCitationsCounter=1%
7556   \def\markdownLaTeXCitationsTotal{#1}%
7557   \@ifundefined{autocites}{%
7558     \@ifundefined{citep}{%
7559       \expandafter\expandafter\expandafter
7560       \markdownLaTeXBasicTextCitations
7561       \expandafter\expandafter\expandafter{%
7562         \expandafter\expandafter\expandafter}%
7563       \expandafter\expandafter\expandafter{%
7564         \expandafter\expandafter\expandafter}%
7565     }{%
7566       \expandafter\expandafter\expandafter
7567       \markdownLaTeXNatbibTextCitations
7568       \expandafter\expandafter\expandafter{%
7569         \expandafter\expandafter\expandafter}%
7570     }%
7571   }{%
7572     \expandafter\expandafter\expandafter
7573     \markdownLaTeXBibLaTeXTextCitations
7574     \expandafter{\expandafter}%
7575   }}}

```

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.

```

7576 \RequirePackage{url}
7577 \RequirePackage{expl3}
7578 \ExplSyntaxOn
7579 \def\markdownRendererLinkPrototype{
7580   \begingroup
7581   \catcode`\#=12
7582   \def\next##1##2##3##4{
7583     \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.

```

7584   \tl_set:Nn \l_tmpa_tl { ##1 }
7585   \tl_set:Nn \l_tmpb_tl { ##2 }
7586   \bool_set:Nn

```

```

7587     \l_tmpa_bool
7588     {
7589         \tl_if_eq_p:NN
7590         \l_tmpa_tl
7591         \l_tmpb_tl
7592     }
7593     \tl_set:Nn \l_tmpa_tl { ##4 }
7594     \bool_set:Nn
7595         \l_tmpb_bool
7596     {
7597         \tl_if_empty_p:N
7598         \l_tmpa_tl
7599     }
7600     \bool_if:nTF
7601     {
7602         \l_tmpa_bool && \l_tmpb_bool
7603     }
7604     {
7605         \markdownLaTeXRendererAutolink { ##2 } { ##3 }
7606     }{
7607         \markdownLaTeXRendererDirectOrIndirectLink { ##1 } { ##2 } { ##3 } { ##4 }
7608     }
7609 }
7610 \next
7611 }
7612 \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.
7613     \tl_set:Nn
7614         \l_tmpa_tl
7615         { #2 }
7616     \tl_trim_spaces:N
7617         \l_tmpa_tl
7618     \tl_set:Nx
7619         \l_tmpb_tl
7620         {
7621             \tl_range:Nnn
7622                 \l_tmpa_tl
7623                 { 1 }
7624                 { 1 }
7625         }
7626     \str_if_eq:NNTF
7627         \l_tmpb_tl
7628         \c_hash_str
7629         {
7630             \tl_set:Nx

```

```

7631         \l_tmpb_tl
7632     {
7633         \tl_range:Nnn
7634         \l_tmpa_tl
7635         { 2 }
7636         { -1 }
7637     }
7638     \exp_args:NV
7639     \ref
7640     \l_tmpb_tl
7641 }{
7642     \url { #2 }
7643 }
7644 }
7645 \ExplSyntaxOff
7646 \def\markdownLaTeXRendererDirectOrIndirectLink#1#2#3#4{%
7647     #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.

```

7648 \newcount\markdownLaTeXRowCounter
7649 \newcount\markdownLaTeXRowTotal
7650 \newcount\markdownLaTeXColumnCounter
7651 \newcount\markdownLaTeXColumnTotal
7652 \newtoks\markdownLaTeXTable
7653 \newtoks\markdownLaTeXTableAlignment
7654 \newtoks\markdownLaTeXTableEnd
7655 \AtBeginDocument{%
7656     \@ifpackageloaded{booktabs}{%
7657         \def\markdownLaTeXTopRule{\toprule}%
7658         \def\markdownLaTeXMidRule{\midrule}%
7659         \def\markdownLaTeXBottomRule{\bottomrule}%
7660     }{%
7661         \def\markdownLaTeXTopRule{\hline}%
7662         \def\markdownLaTeXMidRule{\hline}%
7663         \def\markdownLaTeXBottomRule{\hline}%
7664     }%
7665 }
7666 \markdownSetup{rendererPrototypes={
7667     table = {%
7668         \markdownLaTeXTable={}%
7669         \markdownLaTeXTableAlignment={}%
7670         \markdownLaTeXTableEnd={%
7671             \markdownLaTeXBottomRule
7672             \end{tabular}}}%
7673     \ifx\empty#1\empty\else

```

```

7674     \addto@hook\markdownLaTeXTable{%
7675         \begin{table}
7676         \centering}%
7677     \addto@hook\markdownLaTeXTableEnd{%
7678         \caption{#1}
7679         \end{table}}}%
7680     \fi
7681     \addto@hook\markdownLaTeXTable{\begin{tabular}}}%
7682     \markdownLaTeXRowCount=0%
7683     \markdownLaTeXRowTotal=#2%
7684     \markdownLaTeXColumnTotal=#3%
7685     \markdownLaTeXRenderTableRow
7686 }
7687 }}
7688 \def\markdownLaTeXRenderTableRow#1{%
7689     \markdownLaTeXColumnCounter=0%
7690     \ifnum\markdownLaTeXRowCount=0\relax
7691         \markdownLaTeXReadAlignments#1%
7692         \markdownLaTeXTable=\expandafter\expandafter\expandafter{%
7693             \expandafter\the\expandafter\markdownLaTeXTable\expandafter{%
7694                 \the\markdownLaTeXTableAlignment}}}%
7695         \addto@hook\markdownLaTeXTable{\markdownLaTeXTopRule}%
7696     \else
7697         \markdownLaTeXRenderTableCell#1%
7698     \fi
7699     \ifnum\markdownLaTeXRowCount=1\relax
7700         \addto@hook\markdownLaTeXTable\markdownLaTeXMidRule
7701     \fi
7702     \advance\markdownLaTeXRowCount by 1\relax
7703     \ifnum\markdownLaTeXRowCount>\markdownLaTeXRowTotal\relax
7704         \the\markdownLaTeXTable
7705         \the\markdownLaTeXTableEnd
7706         \expandafter\@gobble
7707     \fi\markdownLaTeXRenderTableRow}
7708 \def\markdownLaTeXReadAlignments#1{%
7709     \advance\markdownLaTeXColumnCounter by 1\relax
7710     \if#1d%
7711         \addto@hook\markdownLaTeXTableAlignment{1}%
7712     \else
7713         \addto@hook\markdownLaTeXTableAlignment{#1}%
7714     \fi
7715     \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax\else
7716         \expandafter\@gobble
7717     \fi\markdownLaTeXReadAlignments}
7718 \def\markdownLaTeXRenderTableCell#1{%
7719     \advance\markdownLaTeXColumnCounter by 1\relax
7720     \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax

```

```

7721 \addto@hook\markdownLaTeXTable{#1&}%
7722 \else
7723 \addto@hook\markdownLaTeXTable{#1\\}%
7724 \expandafter\@gobble
7725 \fi\markdownLaTeXRenderTableCell}
7726 \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.

```

7727 \ExplSyntaxOn
7728 \keys_define:nn
7729 { markdown/jekyllData }
7730 {
7731   author .code:n = { \author{#1} },
7732   date   .code:n = { \date{#1}   },
7733   title  .code:n = { \title{#1}  },
7734 }

```

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.

```

7735 % TODO: Remove the command definition in TeX Live 2021.
7736 \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
7737 \markdownSetup{
7738   rendererPrototypes = {
7739     jekyllDataEnd = {
7740 %       TODO: Remove the else branch in TeX Live 2021.
7741       \IfFormatAtLeastTF
7742       { 2020-10-01 }
7743       { \AddToHook{begindocument/end}{\maketitle} }
7744       {
7745         \ifx\@onlypreamble\@notprerr
7746         % We are in the document
7747         \maketitle
7748       \else
7749         % We are in the preamble
7750         \RequirePackage{etoolbox}
7751         \AfterEndPreamble{\maketitle}
7752       \fi
7753     }
7754   },
7755 },
7756 }
7757 \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.

```
7758 \newcommand\markdownMakeOther{%
7759   \count0=128\relax
7760   \loop
7761     \catcode\count0=11\relax
7762     \advance\count0 by 1\relax
7763   \ifnum\count0<256\repeat}%
```

3.4 ConT_EXt Implementation

The ConT_EXt implementation makes use of the fact that, apart from some subtle differences, the Mark II and Mark IV ConT_EXt formats *seem* to implement (the documentation is scarce) the majority of the plain T_EX format required by the plain T_EX implementation. As a consequence, we can directly reuse the existing plain T_EX implementation after supplying the missing plain T_EX macros.

The ConT_EXt implementation redefines the plain T_EX logging macros (see Section 3.2.1) to use the ConT_EXt `\writestatus` macro.

```
7764 \def\markdownInfo#1{\writestatus{markdown}{#1.}}%
7765 \def\markdownWarning#1{\writestatus{markdown\space warn}{#1.}}%
7766 \def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
7767   \do\#\do\~\do\_ \do\% \do\~}%
7768 \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 L^AT_EX package.

```
7769 \def\markdownMakeOther{%
7770   \count0=128\relax
7771   \loop
7772     \catcode\count0=11\relax
7773     \advance\count0 by 1\relax
7774   \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 ConT_EXt.

```
7775   \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 Con \TeX t MkIV funnels all input through its own input handler. This makes it possible to suppress the removal of trailing spaces in Con \TeX t MkIV and therefore to insert hard line breaks into markdown text.

```
7776 \ifx\startluacode\undefined % MkII
7777   \begingroup
7778     \catcode`\|=0%
7779     \catcode`\|=12%
7780     |gdef|startmarkdown{%
7781       |markdownReadAndConvert{\stopmarkdown}%
7782                                   {|stopmarkdown}}%
7783     |gdef|stopmarkdown{%
7784       |markdownEnd}%
7785   |endgroup
7786 \else % MkIV
7787   \startluacode
7788     document.markdown_buffering = false
7789     local function preserve_trailing_spaces(line)
7790       if document.markdown_buffering then
7791         line = line:gsub("[ \t][ \t]$", "\t\t")
7792       end
7793       return line
7794     end
7795     resolvers.installinputlinehandler(preserve_trailing_spaces)
7796   \stopluacode
7797   \begingroup
7798     \catcode`\|=0%
7799     \catcode`\|=12%
7800     |gdef|startmarkdown{%
7801       |ctxlua{document.markdown_buffering = true}%
7802       |markdownReadAndConvert{\stopmarkdown}%
7803                                   {|stopmarkdown}}%
7804     |gdef|stopmarkdown{%
7805       |ctxlua{document.markdown_buffering = false}%
7806       |markdownEnd}%
7807   |endgroup
7808 \fi
```

3.4.2 Token Renderer Prototypes

The following configuration should be considered placeholder.

```
7809 \def\markdownRendererLineBreakPrototype{\blank}%
7810 \def\markdownRendererLeftBracePrototype{\textbraceleft}%
7811 \def\markdownRendererRightBracePrototype{\textbraceright}%
7812 \def\markdownRendererDollarSignPrototype{\textdollar}%
7813 \def\markdownRendererPercentSignPrototype{\percent}%
7814 \def\markdownRendererUnderscorePrototype{\textunderscore}%
7815 \def\markdownRendererCircumflexPrototype{\textcircumflex}%
7816 \def\markdownRendererBackslashPrototype{\textbackslash}%
7817 \def\markdownRendererTildePrototype{\textasciitilde}%
7818 \def\markdownRendererPipePrototype{\char`|}%
7819 \def\markdownRendererLinkPrototype#1#2#3#4{%
7820   \useURL[#1][#3][[#4]#1\footnote[#1]{\ifx\empty#4\empty\else#4:
7821     \fi}\tt<\hyphenatedurl{#3}>}}%
7822 \usemodule[database]
7823 \defineseparatedlist
7824   [MarkdownConTeXtCSV]
7825   [separator={,},
7826     before=\bTABLE,after=\eTABLE,
7827     first=\bTR,last=\eTR,
7828     left=\bTD,right=\eTD]
7829 \def\markdownConTeXtCSV{csv}
7830 \def\markdownRendererContentBlockPrototype#1#2#3#4{%
7831   \def\markdownConTeXtCSV@arg{#1}%
7832   \ifx\markdownConTeXtCSV@arg\markdownConTeXtCSV
7833     \placetable[] [tab:#1]{#4}{%
7834       \processseparatedfile[MarkdownConTeXtCSV][#3]}%
7835   \else
7836     \markdownInput{#3}%
7837   \fi}%
7838 \def\markdownRendererImagePrototype#1#2#3#4{%
7839   \placefigure[] []{#4}{\externalfigure[#3]}}%
7840 \def\markdownRendererUlBeginPrototype{\startitemize}%
7841 \def\markdownRendererUlBeginTightPrototype{\startitemize[packed]}%
7842 \def\markdownRendererUlItemPrototype{\item}%
7843 \def\markdownRendererUlEndPrototype{\stopitemize}%
7844 \def\markdownRendererUlEndTightPrototype{\stopitemize}%
7845 \def\markdownRendererOlBeginPrototype{\startitemize[n]}%
7846 \def\markdownRendererOlBeginTightPrototype{\startitemize[packed,n]}%
7847 \def\markdownRendererOlItemPrototype{\item}%
7848 \def\markdownRendererOlItemWithNumberPrototype#1{\sym{#1.}}%
7849 \def\markdownRendererOlEndPrototype{\stopitemize}%
7850 \def\markdownRendererOlEndTightPrototype{\stopitemize}%
7851 \definedescription
7852   [MarkdownConTeXtDlItemPrototype]
```

```

7853 [location=hanging,
7854     margin=standard,
7855     headstyle=bold]%
7856 \definestartstop
7857 [MarkdownConTeXtDlPrototype]
7858 [before=\blank,
7859     after=\blank]%
7860 \definestartstop
7861 [MarkdownConTeXtDlTightPrototype]
7862 [before=\blank\startpacked,
7863     after=\stoppacked\blank]%
7864 \def\markdownRendererDlBeginPrototype{%
7865     \startMarkdownConTeXtDlPrototype}%
7866 \def\markdownRendererDlBeginTightPrototype{%
7867     \startMarkdownConTeXtDlTightPrototype}%
7868 \def\markdownRendererDlItemPrototype#1{%
7869     \startMarkdownConTeXtDlItemPrototype{#1}}%
7870 \def\markdownRendererDlItemEndPrototype{%
7871     \stopMarkdownConTeXtDlItemPrototype}%
7872 \def\markdownRendererDlEndPrototype{%
7873     \stopMarkdownConTeXtDlPrototype}%
7874 \def\markdownRendererDlEndTightPrototype{%
7875     \stopMarkdownConTeXtDlTightPrototype}%
7876 \def\markdownRendererEmphasisPrototype#1{{\em#1}}%
7877 \def\markdownRendererStrongEmphasisPrototype#1{{\bf#1}}%
7878 \def\markdownRendererBlockQuoteBeginPrototype{\startquotation}%
7879 \def\markdownRendererBlockQuoteEndPrototype{\stopquotation}%
7880 \def\markdownRendererInputVerbatimPrototype#1{\typefile{#1}}%
7881 \def\markdownRendererInputFencedCodePrototype#1#2{%
7882     \ifx\relax#2\relax
7883         \typefile{#1}%
7884     \else

```

The code fence infostring is used as a name from the ConT_EXt `\definetying` macro. This allows the user to set up code highlighting mapping as follows:

```

\definetying [latex]
\setuptying [latex] [option=TEX]

\starttext
  \startmarkdown
~~~ latex
\documentclass{article}
\begin{document}
  Hello world!
\end{document}

```

~~~

`\stopmarkdown`  
`\stoptext`

```
7885 \typefile[#2] []{#1}%
7886 \fi}%
7887 \def\markdownRendererHeadingOnePrototype#1{\chapter{#1}}%
7888 \def\markdownRendererHeadingTwoPrototype#1{\section{#1}}%
7889 \def\markdownRendererHeadingThreePrototype#1{\subsection{#1}}%
7890 \def\markdownRendererHeadingFourPrototype#1{\subsubsection{#1}}%
7891 \def\markdownRendererHeadingFivePrototype#1{\subsubsubsection{#1}}%
7892 \def\markdownRendererHeadingSixPrototype#1{\subsubsubsubsection{#1}}%
7893 \def\markdownRendererHorizontalRulePrototype{%
7894 \blackrule[height=1pt, width=\hsize]}%
7895 \def\markdownRendererFootnotePrototype#1{\footnote{#1}}%
7896 \stopmodule\protect
```

There is a basic implementation of tables.

```
7897 \newcount\markdownConTeXtRowCounter
7898 \newcount\markdownConTeXtRowTotal
7899 \newcount\markdownConTeXtColumnCounter
7900 \newcount\markdownConTeXtColumnTotal
7901 \newtoks\markdownConTeXtTable
7902 \newtoks\markdownConTeXtTableFloat
7903 \def\markdownRendererTablePrototype#1#2#3{%
7904 \markdownConTeXtTable={}%
7905 \ifx\empty#1\empty
7906 \markdownConTeXtTableFloat={%
7907 \the\markdownConTeXtTable}%
7908 \else
7909 \markdownConTeXtTableFloat={%
7910 \placetable{#1}{\the\markdownConTeXtTable}}%
7911 \fi
7912 \begingroup
7913 \setupTABLE[r][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]
7914 \setupTABLE[c][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]
7915 \setupTABLE[r][1][topframe=on, bottomframe=on]
7916 \setupTABLE[r][#1][bottomframe=on]
7917 \markdownConTeXtRowCounter=0%
7918 \markdownConTeXtRowTotal=#2%
7919 \markdownConTeXtColumnTotal=#3%
7920 \markdownConTeXtRenderTableRow}
7921 \def\markdownConTeXtRenderTableRow#1{%
7922 \markdownConTeXtColumnCounter=0%
7923 \ifnum\markdownConTeXtRowCounter=0\relax
7924 \markdownConTeXtReadAlignments#1%
7925 \markdownConTeXtTable={\bTABLE}%

```

```

7926 \else
7927   \markdownConTeXtTable=\expandafter{%
7928     \the\markdownConTeXtTable\bTR}%
7929   \markdownConTeXtRenderTableCell#1%
7930   \markdownConTeXtTable=\expandafter{%
7931     \the\markdownConTeXtTable\eTR}%
7932 \fi
7933 \advance\markdownConTeXtRowCounter by 1\relax
7934 \ifnum\markdownConTeXtRowCounter>\markdownConTeXtRowTotal\relax
7935   \markdownConTeXtTable=\expandafter{%
7936     \the\markdownConTeXtTable\eTABLE}%
7937   \the\markdownConTeXtTableFloat
7938   \endgroup
7939   \expandafter\gobbleoneargument
7940 \fi\markdownConTeXtRenderTableRow}
7941 \def\markdownConTeXtReadAlignments#1{%
7942   \advance\markdownConTeXtColumnCounter by 1\relax
7943   \if#1d%
7944     \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
7945   \fi\if#1l%
7946     \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
7947   \fi\if#1c%
7948     \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=middle]
7949   \fi\if#1r%
7950     \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=left]
7951   \fi
7952   \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
7953     \expandafter\gobbleoneargument
7954   \fi\markdownConTeXtReadAlignments}
7955 \def\markdownConTeXtRenderTableCell#1{%
7956   \advance\markdownConTeXtColumnCounter by 1\relax
7957   \markdownConTeXtTable=\expandafter{%
7958     \the\markdownConTeXtTable\bTD#1\eTD}%
7959   \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
7960     \expandafter\gobbleoneargument
7961   \fi\markdownConTeXtRenderTableCell}
7962 \def\markdownRendererTickedBox{$\boxtimes$}
7963 \def\markdownRendererHalfTickedBox{$\boxdot$}
7964 \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 T<sub>E</sub>Xbook*. 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 Miletic. *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. *L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 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 L<sup>A</sup>T<sub>E</sub>X*. July 19, 2017. URL: <https://mirrors.ctan.org/macros/latex/contrib/minted/minted.pdf> (visited on 09/01/2020).
- [9] Roberto Ierusalimsky. *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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> Sources*. Apr. 15, 2017. URL: <https://mirrors.ctan.org/macros/latex/base/source2e.pdf> (visited on 01/08/2018).
- [11] Donald Ervin Knuth. *T<sub>E</sub>X: The Program*. Vol. B. Computers & Typesetting. Reading, MA: Addison-Wesley, 1986. xvi, 594. ISBN: 0-201-13437-7.
- [12] Victor Eijkhout. *T<sub>E</sub>X by Topic. A T<sub>E</sub>Xnician's Reference*. Wokingham, England: Addison-Wesley, Feb. 1, 1992. 307 pp. ISBN: 0-201-56882-0.

## Index

|                                    |                     |
|------------------------------------|---------------------|
| <code>\author</code>               | 231                 |
| <code>\autocites</code>            | 222                 |
| <code>blankBeforeBlockquote</code> | 14                  |
| <code>blankBeforeCodeFence</code>  | 14                  |
| <code>blankBeforeHeading</code>    | 14                  |
| <code>breakableBlockquotes</code>  | 15                  |
| <code>cacheDir</code>              | 13, 17, 36, 37, 173 |

|                              |             |
|------------------------------|-------------|
| citationNbsps                | 15          |
| citations                    | 15, 64, 65  |
| \cite                        | 222         |
| \citep                       | 222         |
| \citet                       | 222         |
| codeSpans                    | 16          |
| compactdesc                  | 4           |
| compactenum                  | 4           |
| compactitem                  | 4           |
| contentBlocks                | 16          |
| contentBlocksLanguageMap     | 17, 178     |
| convert                      | 204         |
| \csvautotabular              | 5           |
| \date                        | 231         |
| defaultOptions               | 7, 32, 194  |
| \definetyping                | 235         |
| definitionLists              | 17, 54      |
| \directlua                   | 4, 37, 211  |
| eagerCache                   | 17, 18      |
| \enableregime                | 232         |
| \endmarkdown                 | 73          |
| entities.char_entity         | 135         |
| entities.dec_entity          | 135         |
| entities.hex_entity          | 135         |
| escape_citation              | 175, 175    |
| escaped_citation_chars       | 174, 175    |
| expandtabs                   | 158         |
| expectJekyllData             | 18          |
| extension                    | 87          |
| extensions                   | 174         |
| extensions.citations         | 174         |
| extensions.content_blocks    | 178         |
| extensions.definition_lists  | 181         |
| extensions.fenced_code       | 183         |
| extensions.footnotes         | 185         |
| extensions.header_attributes | 187         |
| extensions.jekyll_data       | 188         |
| extensions.pipe_table        | 191         |
| fencedCode                   | 19, 58, 214 |
| \filecontents                | 205         |

|                                              |                                       |
|----------------------------------------------|---------------------------------------|
| finalizeCache                                | 13, 17, 18, 20, 21, 36, 173           |
| footnotes                                    | 20, 64                                |
| frozenCacheCounter                           | 21, 173, 212                          |
| frozenCacheFileName                          | 13, 20, 173                           |
| \g_luabridge_error_output_filename_str       | 38                                    |
| \g_luabridge_helper_script_filename_str      | 37                                    |
| \g_luabridge_output_dirname_str              | 38                                    |
| hardLineBreaks                               | 21                                    |
| hashEnumerators                              | 22                                    |
| headerAttributes                             | 22, 26, 67, 68                        |
| html                                         | 23, 66, 67, 222                       |
| hybrid                                       | 23, 29, 30, 39, 44, 79, 138, 159, 211 |
| \includegraphics                             | 4                                     |
| inlineFootnotes                              | 23                                    |
| \input                                       | 34, 86, 207, 210                      |
| isdir                                        | 3                                     |
| iterlines                                    | 158                                   |
| jekyllData                                   | 3, 18, 24, 58–61                      |
| \jobname                                     | 37, 38                                |
| \label                                       | 220                                   |
| languages_json                               | 178, 178                              |
| \maketitle                                   | 231                                   |
| \markdown                                    | 73                                    |
| markdown                                     | 73, 73, 213                           |
| markdown*                                    | 4, 73, 73, 74, 213                    |
| \markdown_jekyll_data_concatenate_address:NN | 200                                   |
| \markdown_jekyll_data_pop:                   | 201                                   |
| \markdown_jekyll_data_push:nN                | 201                                   |
| \markdown_jekyll_data_push_address_segment:n | 199                                   |
| \markdown_jekyll_data_set_keyval:Nn          | 201                                   |
| \markdown_jekyll_data_set_keyvals:nn         | 201                                   |
| \markdown_jekyll_data_update_address_tls:    | 200                                   |
| \markdownBegin                               | 34, 34, 35, 71, 73, 86                |
| \markdownEnd                                 | 34, 34, 35, 71, 73, 86                |
| \markdownError                               | 70, 70                                |
| \markdownExecute                             | 209                                   |
| \markdownExecuteDirect                       | 209, 209                              |



|                                                       |                                      |
|-------------------------------------------------------|--------------------------------------|
| <code>\markdownExecuteShellEscape</code>              | 208, 209                             |
| <code>\markdownIfOption</code>                        | 204                                  |
| <code>\markdownIfSnippetExists</code>                 | 75                                   |
| <code>\markdownInfo</code>                            | 70                                   |
| <code>\markdownInput</code>                           | 34, 35, 73, 74, 211, 213             |
| <code>\markdownInputFileStream</code>                 | 205                                  |
| <code>\markdownInputPlainTeX</code>                   | 213                                  |
| <code>\markdownLuaExecute</code>                      | 208, 209, 210, 211                   |
| <code>\markdownLuaOptions</code>                      | 203, 204                             |
| <code>\markdownLuaRegisterIBCallback</code>           | 72                                   |
| <code>\markdownLuaUnregisterIBCallback</code>         | 72                                   |
| <code>\markdownMakeOther</code>                       | 70, 232                              |
| <code>\markdownMode</code>                            | 4, 37, 38, 71, 71, 208, 210          |
| <code>\markdownOptionCacheDir</code>                  | 3, 83, 204, 216                      |
| <code>\markdownOptionErrorTempFileName</code>         | 38, 38, 210                          |
| <code>\markdownOptionFinalizeCache</code>             | 36, 36, 82                           |
| <code>\markdownOptionFrozenCache</code>               | 13, 20, 36, 37, 78, 79, 82, 215, 216 |
| <code>\markdownOptionFrozenCacheFileName</code>       | 36                                   |
| <code>\markdownOptionHelperScriptFileName</code>      | 37, 37–39, 209, 210                  |
| <code>\markdownOptionHybrid</code>                    | 39, 83                               |
| <code>\markdownOptionInputTempFileName</code>         | 37, 206, 207                         |
| <code>\markdownOptionOutputDir</code>                 | 38, 38                               |
| <code>\markdownOptionOutputTempFileName</code>        | 37, 38, 210                          |
| <code>\markdownOptionSmartEllipses</code>             | 83                                   |
| <code>\markdownOptionStripPercentSigns</code>         | 40, 205, 206                         |
| <code>\markdownOutputFileStream</code>                | 205                                  |
| <code>\markdownPrepare</code>                         | 204                                  |
| <code>\markdownPrepareLuaOptions</code>               | 202                                  |
| <code>\markdownReadAndConvert</code>                  | 71, 205, 213, 233                    |
| <code>\markdownReadAndConvertProcessLine</code>       | 206, 207                             |
| <code>\markdownReadAndConvertStripPercentSigns</code> | 206                                  |
| <code>\markdownReadAndConvertTab</code>               | 205                                  |
| <code>\markdownRendererAttributeClassName</code>      | 67                                   |
| <code>\markdownRendererAttributeIdentifier</code>     | 67                                   |
| <code>\markdownRendererAttributeKeyValue</code>       | 67                                   |
| <code>\markdownRendererBlockHtmlCommentBegin</code>   | 66                                   |
| <code>\markdownRendererBlockHtmlCommentEnd</code>     | 66                                   |
| <code>\markdownRendererBlockQuoteBegin</code>         | 57                                   |
| <code>\markdownRendererBlockQuoteEnd</code>           | 57                                   |
| <code>\markdownRendererCite</code>                    | 64, 65                               |
| <code>\markdownRendererCodeSpan</code>                | 47                                   |
| <code>\markdownRendererCodeSpanPrototype</code>       | 85                                   |

|                                              |        |
|----------------------------------------------|--------|
| \markdownRendererContentBlock                | 48, 48 |
| \markdownRendererContentBlockCode            | 49     |
| \markdownRendererContentBlockOnlineImage     | 48     |
| \markdownRendererDlBegin                     | 54     |
| \markdownRendererDlBeginTight                | 54     |
| \markdownRendererDlDefinitionBegin           | 55     |
| \markdownRendererDlDefinitionEnd             | 55     |
| \markdownRendererDlEnd                       | 55     |
| \markdownRendererDlEndTight                  | 56     |
| \markdownRendererDlItem                      | 54     |
| \markdownRendererDlItemEnd                   | 55     |
| \markdownRendererDocumentBegin               | 42     |
| \markdownRendererDocumentEnd                 | 42     |
| \markdownRendererEllipsis                    | 26, 44 |
| \markdownRendererEmphasis                    | 56, 84 |
| \markdownRendererFootnote                    | 64     |
| \markdownRendererHalfTickedBox               | 41     |
| \markdownRendererHeaderAttributeContextBegin | 68     |
| \markdownRendererHeaderAttributeContextEnd   | 68     |
| \markdownRendererHeadingFive                 | 63     |
| \markdownRendererHeadingFour                 | 63     |
| \markdownRendererHeadingOne                  | 62     |
| \markdownRendererHeadingSix                  | 63     |
| \markdownRendererHeadingThree                | 62     |
| \markdownRendererHeadingTwo                  | 62     |
| \markdownRendererHorizontalRule              | 64     |
| \markdownRendererImage                       | 48     |
| \markdownRendererImagePrototype              | 85     |
| \markdownRendererInlineHtmlComment           | 66     |
| \markdownRendererInlineHtmlTag               | 66     |
| \markdownRendererInputBlockHtmlElement       | 67     |
| \markdownRendererInputFencedCode             | 58     |
| \markdownRendererInputVerbatim               | 57     |
| \markdownRendererInterblockSeparator         | 43     |
| \markdownRendererJekyllDataBegin             | 58     |
| \markdownRendererJekyllDataBoolean           | 60     |
| \markdownRendererJekyllDataEmpty             | 61     |
| \markdownRendererJekyllDataEnd               | 58     |
| \markdownRendererJekyllDataMappingBegin      | 59     |
| \markdownRendererJekyllDataMappingEnd        | 59     |
| \markdownRendererJekyllDataNumber            | 61     |
| \markdownRendererJekyllDataSequenceBegin     | 59     |

|                                                     |                           |
|-----------------------------------------------------|---------------------------|
| <code>\markdownRendererJekyllDataSequenceEnd</code> | 60                        |
| <code>\markdownRendererJekyllDataString</code>      | 61                        |
| <code>\markdownRendererLineBreak</code>             | 43                        |
| <code>\markdownRendererLink</code>                  | 47, 84                    |
| <code>\markdownRendererNbsp</code>                  | 44                        |
| <code>\markdownRendererOlBegin</code>               | 51                        |
| <code>\markdownRendererOlBeginTight</code>          | 52                        |
| <code>\markdownRendererOlEnd</code>                 | 53                        |
| <code>\markdownRendererOlEndTight</code>            | 53                        |
| <code>\markdownRendererOlItem</code>                | 27, 52                    |
| <code>\markdownRendererOlItemEnd</code>             | 52                        |
| <code>\markdownRendererOlItemWithNumber</code>      | 27, 52                    |
| <code>\markdownRendererStrongEmphasis</code>        | 56                        |
| <code>\markdownRendererTable</code>                 | 65                        |
| <code>\markdownRendererTextCite</code>              | 65                        |
| <code>\markdownRendererTickedBox</code>             | 41                        |
| <code>\markdownRendererUlBegin</code>               | 49                        |
| <code>\markdownRendererUlBeginTight</code>          | 50                        |
| <code>\markdownRendererUlEnd</code>                 | 51                        |
| <code>\markdownRendererUlEndTight</code>            | 51                        |
| <code>\markdownRendererUlItem</code>                | 50                        |
| <code>\markdownRendererUlItemEnd</code>             | 50                        |
| <code>\markdownRendererUntickedBox</code>           | 41                        |
| <code>\markdownSetup</code>                         | 4, 74, 74, 213, 217       |
| <code>\markdownSetupSnippet</code>                  | 74, 74                    |
| <code>\markdownWarning</code>                       | 70                        |
| <br><code>new</code>                                | <br>7, 194                |
| <br><code>os.execute</code>                         | <br>71, 209               |
| <br><code>\PackageError</code>                      | <br>73, 213               |
| <code>\PackageInfo</code>                           | 73, 213                   |
| <code>\PackageWarning</code>                        | 73, 213                   |
| <code>parsers</code>                                | 146, 158                  |
| <code>parsers.commented_line</code>                 | 148                       |
| <code>\pdfshellescape</code>                        | 208                       |
| <code>pipeTables</code>                             | 6, 24, 28, 65             |
| <code>preserveTabs</code>                           | 25, 27, 158               |
| <code>print</code>                                  | 209, 211                  |
| <br><code>reader</code>                             | <br>87, 87, 146, 157, 174 |
| <code>reader-&gt;convert</code>                     | 173, 194                  |

|                               |                    |
|-------------------------------|--------------------|
| reader->create_parser         | 159                |
| reader->normalize_tag         | 158                |
| reader->parser_functions      | 159                |
| reader->parser_functions.name | 159                |
| reader->parsers               | 158, 158           |
| reader->syntax                | 169                |
| reader->writer                | 158                |
| reader.new                    | 157, 157           |
| relativeReferences            | 25                 |
|                               |                    |
| \shellescape                  | 208                |
| shiftHeadings                 | 6, 26              |
| slice                         | 6, 22, 26, 136     |
| smartEllipses                 | 26, 44             |
| \startmarkdown                | 86, 86, 233        |
| startNumber                   | 27, 52             |
| status.shell_escape           | 208                |
| \stopmarkdown                 | 86, 86, 233        |
| stripIndent                   | 27, 159            |
|                               |                    |
| tableCaptions                 | 6, 28              |
| taskLists                     | 28, 41, 221        |
| tex.print                     | 209, 211           |
| texComments                   | 29, 159            |
| \textcites                    | 222                |
| tightLists                    | 29, 50–54, 56, 217 |
| \title                        | 231                |
|                               |                    |
| underscores                   | 30                 |
| \url                          | 4                  |
| \usepackage                   | 73, 76             |
| \usetheme                     | 76                 |
| util.cache                    | 87                 |
| util.err                      | 87                 |
| util.escaper                  | 90                 |
| util.expand_tabs_in_line      | 88                 |
| util.flatten                  | 89                 |
| util.intersperse              | 90                 |
| util.map                      | 90                 |
| util.pathname                 | 91                 |
| util.rope_last                | 89                 |
| util.rope_to_string           | 89                 |
| util.table_copy               | 88                 |

|                                 |                  |
|---------------------------------|------------------|
| util.walk                       | 88, 89           |
| \VerbatimInput                  | 4                |
| writer                          | 87, 87, 135, 174 |
| writer->active_attributes       | 142              |
| writer->active_headings         | 142              |
| writer->block_html_comment      | 140              |
| writer->block_html_element      | 141              |
| writer->blockquote              | 141              |
| writer->bulletlist              | 139              |
| writer->cacheDir                | 136              |
| writer->citation                | 175              |
| writer->citations               | 175              |
| writer->code                    | 139              |
| writer->codeFence               | 183              |
| writer->contentblock            | 179              |
| writer->defer_call              | 145, 145         |
| writer->definitionlist          | 181              |
| writer->document                | 142              |
| writer->ellipsis                | 137              |
| writer->emphasis                | 141              |
| writer->escape                  | 138, 138         |
| writer->escape_minimal          | 138, 138, 175    |
| writer->escape_uri              | 138, 138         |
| writer->escaped_chars           | 138, 138         |
| writer->escaped_minimal_strings | 137, 138         |
| writer->escaped_uri_chars       | 137, 138         |
| writer->get_state               | 145              |
| writer->heading                 | 142              |
| writer->hrule                   | 137              |
| writer->hybrid                  | 136, 175         |
| writer->image                   | 139              |
| writer->inline_html_comment     | 140              |
| writer->inline_html_tag         | 141              |
| writer->interblocksep           | 137              |
| writer->is_writing              | 136, 136         |
| writer->jekyllData              | 188              |
| writer->linebreak               | 137              |
| writer->link                    | 139              |
| writer->nbsp                    | 137              |
| writer->note                    | 185              |

|                     |               |
|---------------------|---------------|
| writer->ollist      | 140           |
| writer->pack        | 137, 173      |
| writer->paragraph   | 137           |
| writer->plain       | 137           |
| writer->set_state   | 145           |
| writer->slice_begin | 136           |
| writer->slice_end   | 136           |
| writer->space       | 137           |
| writer->string      | 138           |
| writer->strong      | 141           |
| writer->suffix      | 137           |
| writer->table       | 192           |
| writer->textbox     | 141           |
| writer->uri         | 138           |
| writer->verbatim    | 142           |
| writer.new          | 135, 136, 136 |
| \writestatus        | 232           |