

A Markdown Interpreter for T_EX

Vít Novotný
witiko@mail.muni.cz

Version 2.12.0-1-g976f074
2021/12/30

Contents

1	Introduction	1	3	Implementation	59
1.1	Requirements	2	3.1	Lua Implementation	60
1.2	Feedback	5	3.2	Plain T _E X Implementation	161
1.3	Acknowledgements	5	3.3	L ^A T _E X Implementation	173
2	Interfaces	6	3.4	ConT _E Xt Implementation	193
2.1	Lua Interface	6			
2.2	Plain T _E X Interface	23			
2.3	L ^A T _E X Interface	41			
2.4	ConT _E Xt Interface	58			
				References	198
				Index	199

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	20
3	A sequence diagram of typesetting a document using the Lua CLI	21
4	Various formats of mathematical formulae	47
5	The banner of the Markdown package	48
6	A pushdown automaton that recognizes T _E X comments	125

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 on page 6 describes the interfaces exposed by the package. Section 3 on page 59 describes the implementation of the package. The technical documentation

¹See <https://ctan.org/pkg/markdown>.

²See <https://daringfireball.net/projects/markdown/basics>.

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               "2016-2021 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 on the preceding page), and 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 on page 20), the plain TeX part of the package will require that either the LuaTeX `\directlua` primitive or the shell access file stream 18 is available in your TeX engine. If only the shell access file stream is available in your TeX engine (as is the case with pdfTeX and XeTeX) or if you enforce the use of shell using the `\markdownMode` macro, then unless your TeX engine is globally configured to enable shell access, you will need to provide the `-shell-escape` parameter to your engine when typesetting a document.

1.1.3 L^ATeX Requirements

The L^ATeX part of the package requires that the L^ATeX 2 ϵ format is loaded,

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

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

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

19 `\RequirePackage{keyval}`

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

20 `\RequirePackage{xstring}`

The following packages are soft prerequisites. They are only used to provide default token renderer prototypes (see sections [2.2.4 on page 38](#) and [3.3.4 on page 177](#)) or L^AT_EX themes (see Section [2.3.2.2 on page 44](#)) and will not be loaded if the `plain` package option has been enabled (see Section [2.3.2.1 on page 43](#)):

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 to determine whether or not the `paralist` package should be loaded based on the user options, in the `witiko/dot` L^AT_EX theme (see Section [2.3.2.2 on page 44](#)), 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` T_EX command that is used in the default renderer prototype for citations. The package is included in T_EXLive \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 on page 44](#).

grffile A package that extends the name processing of package `graphics` to support a larger range of file names in $2006 \leq \text{T\TeX Live} \leq 2019$. Since $\text{T\TeX Live} \geq 2020$, the functionality of the package has been integrated in the $\text{\LaTeX} 2_{\epsilon}$ kernel. It is used in the `witiko/dot` and `witiko/graphicx/http` \LaTeX themes, see Section 2.3.2.2 on page 44.

etoolbox A package that is used to polyfill the general hook management system in the default renderer prototypes for YAML metadata, see Section 3.3.4.5.

expl3 A package that enables the `expl3` language from the $\text{\LaTeX} 3$ kernel in $\text{T\TeX Live} \leq 2019$. It is used in the default renderer prototypes for YAML metadata, see Section 3.3.4.5.

1.1.4 ConT \TeX t Prerequisites

The ConT \TeX t part of the package requires that either the Mark II or the Mark IV format is loaded, all the plain T\TeX prerequisites (see Section 1.1.2 on page 3), and the following ConT \TeX t 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 on page 38).

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 in need of assistance, you might want to consider posting your question to the T\TeX - \LaTeX Stack Exchange.⁵ community question answering web site under the `markdown` tag.

1.3 Acknowledgements

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

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.

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

⁵See <https://tex.stackexchange.com>.

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

The \TeX implementation of the package draws inspiration from several sources including the source code of $\LaTeX 2_{\epsilon}$, the minted package by Geoffrey M. Poore, which likewise tackles the issue of interfacing with an external interpreter from \TeX , the filecontents package by Scott Pakin and others.

2 Interfaces

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

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

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

2.1 Lua Interface

The Lua interface provides the conversion from UTF-8 encoded markdown to plain \TeX . This interface is used by the plain \TeX implementation (see Section 3.2 on page 161) and will be of interest to the developers of other packages and Lua modules.

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

```
21 local M = {metadata = metadata}
```

2.1.1 Conversion from Markdown to Plain \TeX

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

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

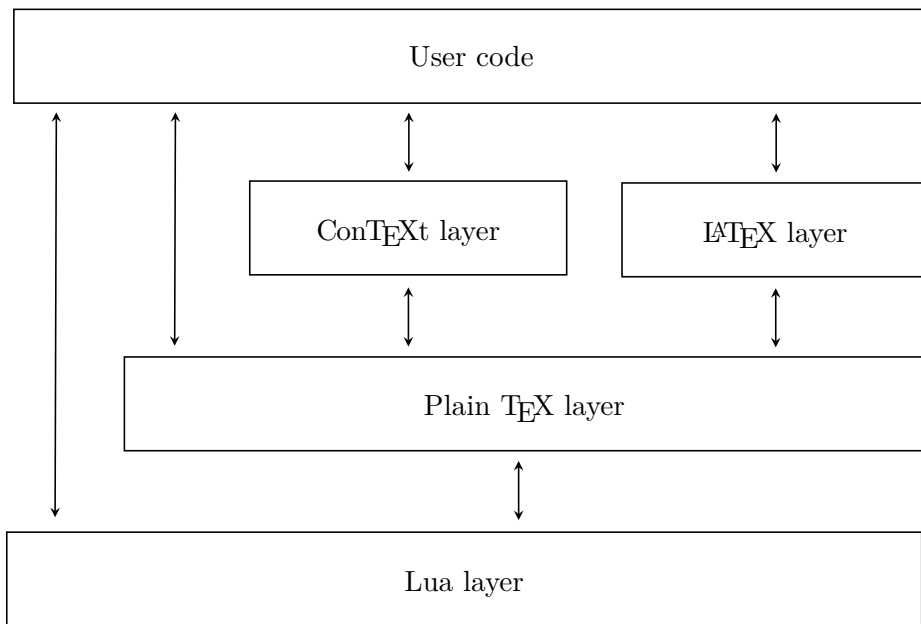


Figure 1: A block diagram of the Markdown package

```

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.

```
22 local defaultOptions = {}
```

2.1.3 File and Directory Names

`cacheDir`=*<path>* default: .

A path to the directory containing auxiliary cache files. If the last segment of the path does not exist, it will be created by the Lua command-line and plain 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.

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

`frozenCacheFileName`=*<path>* default: `frozenCache.tex`

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

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

```
24 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.

```
25 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.

```
26 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.

```
27 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.

28 `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.

29 `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.

30 `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.``
```

```
31 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.

```
32 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 2.2.3.10 on page 30 for more information.

```
33 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.

```
34 defaultOptions.definitionLists = false
```

`eagerCache=true, false`

default: `true`

`true`

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

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

`false`

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

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

```
35 defaultOptions.eagerCache = true
```

`fencedCode=true, false`

default: `false`

`true` Enable the commonmark fenced code block extension:

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

``` html


```

  <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.

```
36 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 $\text{T}_{\text{E}}\text{X}$ document that contains markdown documents without invoking Lua using the `\markdownOptionFrozenCache` plain $\text{T}_{\text{E}}\text{X}$ option. As a result, the plain $\text{T}_{\text{E}}\text{X}$ document becomes more portable, but further changes in the order and the content of markdown documents will not be reflected.

```
37 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.

```
38 defaultOptions.footnotes = false
```

`frozenCacheCounter=<number>`

default: 0

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

Each frozen cache entry will define a T_EX macro `\markdownFrozenCache<number>` that will typeset markdown document number `<number>`.

```
39 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.

```
40 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.

```
41 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.

```
42 defaultOptions.headerAttributes = false
```

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

`true` Enable the recognition of HTML tags, block elements, comments, HTML instructions, and entities in the input. Tags, block elements (along with contents), HTML instructions, and comments 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.

```
43 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.

44 `defaultOptions.hybrid = false`

`inlineFootnotes=true, false` default: false

`true` Enable the pandoc inline footnote syntax extension:

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

`false` Disable the pandoc inline footnote syntax extension.

45 `defaultOptions.inlineFootnotes = false`

`jeekyllData=true, false` default: false

`true` Enable the Pandoc `yml_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.

```
46 defaultOptions.jekyllData = false
```

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

`true` Enable the PHP Markdown table syntax extension:

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

`false` Disable the PHP Markdown table syntax extension.

```
47 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.

```
48 defaultOptions.preserveTabs = false
```

`shiftHeadings=<shift amount>` default: `0`

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

```
49 defaultOptions.shiftHeadings = 0
```


`slice=<the beginning and the end of a slice>` default: ^ \$

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

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

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

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

`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. |

```
51 defaultOptions.smartEllipses = false
```

`startNumber=true, false` default: true

- | | |
|--------------------|--|
| <code>true</code> | Make the number in the first item of an ordered lists significant. The item numbers will be passed to the <code>\markdownRendererOliItemWithNumber</code> TeX macro. |
| <code>false</code> | Ignore the numbers in the ordered list items. Each item will only produce a <code>\markdownRendererOliItem</code> TeX macro. |

```
52 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.

```
53 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.

```
54 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.

```
55 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.

56 `defaultOptions.texComments = false`

`tightLists=true, false`

default: `true`

`true` Lists whose bullets do not consist of multiple paragraphs will be passed to the `\markdownRendererOlBeginTight`, `\markdownRendererOlEndTight`, `\markdownRendererUlBeginTight`, `\markdownRendererUlEndTight`, `\markdownRendererDlBeginTight`, and `\markdownRendererDlEndTight` TeX macros.

`false` Lists whose bullets do not consist of multiple paragraphs will be treated the same way as lists that do consist of multiple paragraphs.

57 `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.

58 `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 $\text{T}_{\text{E}}\text{X}$ layer hands markdown documents to the Lua layer. Lua converts the documents to $\text{T}_{\text{E}}\text{X}$, and hands the converted documents back to plain $\text{T}_{\text{E}}\text{X}$ layer for typesetting, see Figure 2.

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

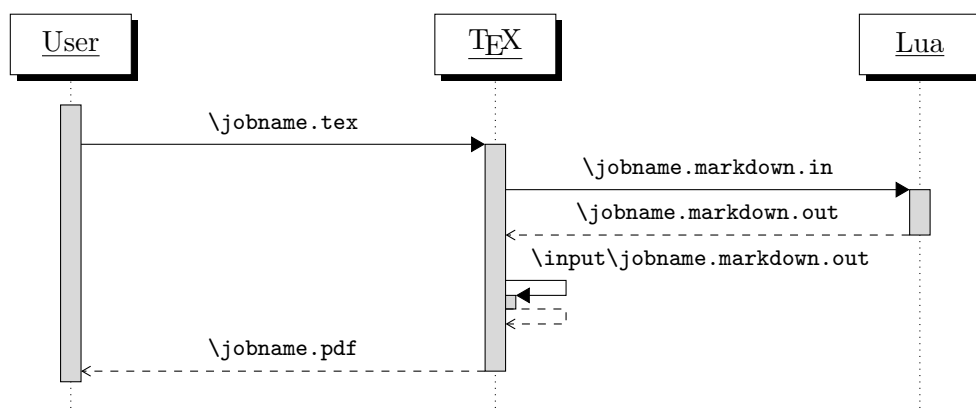


Figure 2: A sequence diagram of the Markdown package typesetting a markdown document using the $\text{T}_{\text{E}}\text{X}$ interface

```
59
60 HELP_STRING = [[
61 Usage: texlua ]] .. arg[0] .. [[ [OPTIONS] -- [INPUT_FILE] [OUTPUT_FILE]
62 where OPTIONS are documented in the Lua interface section of the
63 technical Markdown package documentation.
64
65 When OUTPUT_FILE is unspecified, the result of the conversion will be
66 written to the standard output. When INPUT_FILE is also unspecified, the
67 result of the conversion will be read from the standard input.
68
69 Report bugs to: witiko@mail.muni.cz
70 Markdown package home page: <https://github.com/witiko/markdown>]]
71
```

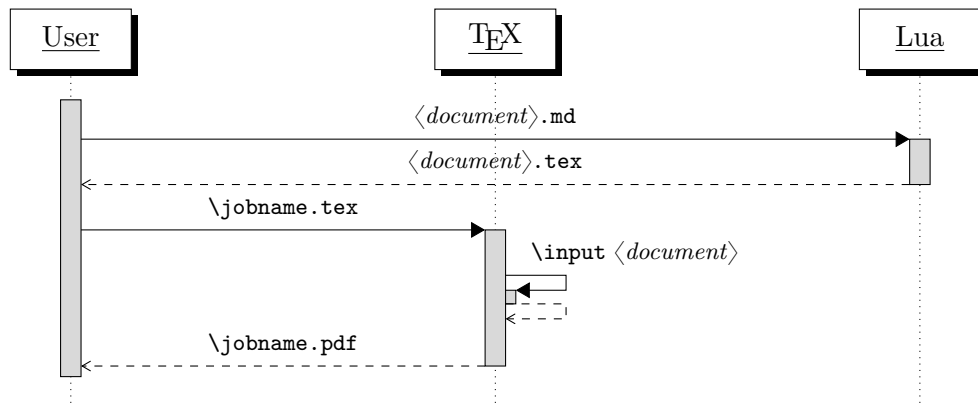


Figure 3: A sequence diagram of the Markdown package typesetting a markdown document using the Lua command-line interface

```

72 VERSION_STRING = [[
73 markdown-cli.lua (Markdown) ]] .. metadata.version .. [[
74
75 Copyright (C) ]] .. table.concat(metadata.copyright,
76                                     "\nCopyright (C) ") .. [[
77
78 License: ]] .. metadata.license
79
80 local function warn(s)
81   io.stderr:write("Warning: " .. s .. "\n") end
82
83 local function error(s)
84   io.stderr:write("Error: " .. s .. "\n")
85   os.exit(1) end
86
87 local process_options = true
88 local options = {}
89 local input_filename
90 local output_filename
91 for i = 1, #arg do
92   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.

```

93   if arg[i] == "--" then
94     process_options = false
95     goto continue

```

Unless the `--` argument has been specified before, an argument containing the equals sign (=) is assumed to be an option specification in a $\langle key \rangle = \langle value \rangle$ format. The available options are listed in Section 2.1.2 on page 7.

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

The `defaultOptions` table is consulted to identify whether $\langle value \rangle$ should be parsed as a string or as a boolean.

```
98         default_type = type(defaultOptions[key])
99         if default_type == "boolean" then
100             options[key] = (value == "true")
101         elseif default_type == "number" then
102             options[key] = tonumber(value)
103         else
104             if default_type ~= "string" then
105                 if default_type == "nil" then
106                     warn('Option "' .. key .. '" not recognized.')
107                 else
108                     warn('Option "' .. key .. '" type not recognized, please file ' ..
109                         'a report to the package maintainer.')
110                 end
111                 warn('Parsing the ' .. 'value "' .. value .. '" of option "' ..
112                     key .. '" as a string.')
113             end
114             options[key] = value
115         end
116         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.

```
117     elseif arg[i] == "--help" or arg[i] == "-h" then
118         print(HELP_STRING)
119         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.

```
120     elseif arg[i] == "--version" or arg[i] == "-v" then
121         print(VERSION_STRING)
122         os.exit()
123     end
124 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 \TeX document.

```

125  if input_filename == nil then
126    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 \TeX document that will result from the conversion.

```

127  elseif output_filename == nil then
128    output_filename = arg[i]
129  else
130    error('Unexpected argument: "' .. arg[i] .. "'')
131  end
132  ::continue::
133 end

```

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

```

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

```

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

```



```

2.2 Plain \TeX Interface

The plain \TeX interface provides macros for the typesetting of markdown input from within plain \TeX , for setting the Lua interface options (see Section 2.1.2 on page 7) used during the conversion from markdown to plain \TeX and for changing the way markdown the tokens are rendered.

```

134 \def\markdownLastModified{$(LAST_MODIFIED)}%
135 \def\markdownVersion{$(VERSION)}%

```

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

```



```

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

```
136 \let\markdownBegin\relax
137 \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_EX [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_EX 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_EX code showcases the usage of the `\markdownBegin` and `\markdownEnd` macros:

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


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

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

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

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

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

2.2.2 Options

The plain \TeX options are represented by \TeX commands. Some of them map directly to the options recognized by the Lua interface (see Section 2.1.2 on page 7), while some of them are specific to the plain \TeX interface.

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

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

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

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

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

2.2.2.2 File and Directory Names The `\markdownOptionHelperScriptFileName` macro sets the filename of the helper Lua script file that is created during the conversion from markdown to plain \TeX in \TeX engines without the `\directlua`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.2.2.3 Lua Interface Options The following macros map directly to the options recognized by the Lua interface (see Section 2.1.2 on page 7) and are not processed by the plain T_EX implementation, only passed along to Lua. They are undefined, which makes them fall back to the default values provided by the Lua interface.

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

```
147 \let\markdownOptionBlankBeforeBlockquote\undefined
148 \let\markdownOptionBlankBeforeCodeFence\undefined
149 \let\markdownOptionBlankBeforeHeading\undefined
150 \let\markdownOptionBreakableBlockquotes\undefined
151 \let\markdownOptionCitations\undefined
152 \let\markdownOptionCitationNbsps\undefined
153 \let\markdownOptionContentBlocks\undefined
154 \let\markdownOptionContentBlocksLanguageMap\undefined
155 \let\markdownOptionDefinitionLists\undefined
156 \let\markdownOptionEagerCache\undefined
157 \let\markdownOptionFootnotes\undefined
158 \let\markdownOptionFencedCode\undefined
159 \let\markdownOptionHardLineBreaks\undefined
160 \let\markdownOptionHashEnumerators\undefined
161 \let\markdownOptionHeaderAttributes\undefined
162 \let\markdownOptionHtml\undefined
163 \let\markdownOptionHybrid\undefined
164 \let\markdownOptionInlineFootnotes\undefined
165 \let\markdownOptionJekyllData\undefined
166 \let\markdownOptionPipeTables\undefined
167 \let\markdownOptionPreserveTabs\undefined
168 \let\markdownOptionShiftHeadings\undefined
169 \let\markdownOptionSlice\undefined
170 \let\markdownOptionSmartEllipses\undefined
171 \let\markdownOptionStartNumber\undefined
172 \let\markdownOptionStripIndent\undefined
173 \let\markdownOptionTableCaptions\undefined
174 \let\markdownOptionTaskLists\undefined
175 \let\markdownOptionTeXComments\undefined
176 \let\markdownOptionTightLists\undefined
```

2.2.2.4 Miscellaneous Options The `\markdownOptionStripPercentSigns` macro controls whether a percent sign (Markdown input (see Section [\vref{sec:buffering}](#)) or not

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

2.2.3 Token Renderers

The following TeX macros may occur inside the output of the converter functions exposed by the Lua interface (see Section [2.1.1 on page 6](#)) 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 on page 38](#)).

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.

```
178 \def\markdownRendererTickedBox{%
179   \markdownRendererTickedBoxPrototype}%
180 \def\markdownRendererHalfTickedBox{%
181   \markdownRendererHalfTickedBoxPrototype}%
182 \def\markdownRendererUntickedBox{%
183   \markdownRendererUntickedBoxPrototype}%
```

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

```
184 \def\markdownRendererInterblockSeparator{%
185   \markdownRendererInterblockSeparatorPrototype}%
```

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

```
186 \def\markdownRendererLineBreak{%
187   \markdownRendererLineBreakPrototype}%
```

2.2.3.4 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.

```
188 \def\markdownRendererEllipsis{%
189   \markdownRendererEllipsisPrototype}%
```

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

```
190 \def\markdownRendererNbsp{%
191   \markdownRendererNbspPrototype}%
```

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

```
192 \def\markdownRendererLeftBrace{%
193   \markdownRendererLeftBracePrototype}%
194 \def\markdownRendererRightBrace{%
195   \markdownRendererRightBracePrototype}%
196 \def\markdownRendererDollarSign{%
197   \markdownRendererDollarSignPrototype}%
198 \def\markdownRendererPercentSign{%
199   \markdownRendererPercentSignPrototype}%
200 \def\markdownRendererAmpersand{%
201   \markdownRendererAmpersandPrototype}%
202 \def\markdownRendererUnderscore{%
203   \markdownRendererUnderscorePrototype}%
204 \def\markdownRendererHash{%
205   \markdownRendererHashPrototype}%
206 \def\markdownRendererCircumflex{%
207   \markdownRendererCircumflexPrototype}%
208 \def\markdownRendererBackslash{%
209   \markdownRendererBackslashPrototype}%
210 \def\markdownRendererTilde{%
211   \markdownRendererTildePrototype}%
212 \def\markdownRendererPipe{%
213   \markdownRendererPipePrototype}%
```

2.2.3.7 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.

```
214 \def\markdownRendererCodeSpan{%
215   \markdownRendererCodeSpanPrototype}%
```

2.2.3.8 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.

```
216 \def\markdownRendererLink{%
217   \markdownRendererLinkPrototype}%
```

2.2.3.9 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.

```
218 \def\markdownRendererImage{%  
219 \markdownRendererImagePrototype}%
```

2.2.3.10 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.

```
220 \def\markdownRendererContentBlock{%  
221 \markdownRendererContentBlockPrototype}%
```

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

```
222 \def\markdownRendererContentBlockOnlineImage{%  
223 \markdownRendererContentBlockOnlineImagePrototype}%
```

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`⁶ 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_EX 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.

```
224 \def\markdownRendererContentBlockCode{%  
225 \markdownRendererContentBlockCodePrototype}%
```

2.2.3.11 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.

⁶Local files take precedence. Filenames other than `markdown-languages.json` may be specified using the `contentBlocksLanguageMap` Lua option.

```
226 \def\markdownRendererU1Begin{%
227   \markdownRendererU1BeginPrototype}%
```

The `\markdownRendererU1BeginTight` 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.

```
228 \def\markdownRendererU1BeginTight{%
229   \markdownRendererU1BeginTightPrototype}%
```

The `\markdownRendererU1Item` macro represents an item in a bulleted list. The macro receives no arguments.

```
230 \def\markdownRendererU1Item{%
231   \markdownRendererU1ItemPrototype}%
```

The `\markdownRendererU1ItemEnd` macro represents the end of an item in a bulleted list. The macro receives no arguments.

```
232 \def\markdownRendererU1ItemEnd{%
233   \markdownRendererU1ItemEndPrototype}%
```

The `\markdownRendererU1End` 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.

```
234 \def\markdownRendererU1End{%
235   \markdownRendererU1EndPrototype}%
```

The `\markdownRendererU1EndTight` 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.

```
236 \def\markdownRendererU1EndTight{%
237   \markdownRendererU1EndTightPrototype}%
```

2.2.3.12 Ordered List Renderers The `\markdownRendererO1Begin` 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.

```
238 \def\markdownRendererO1Begin{%
239   \markdownRendererO1BeginPrototype}%
```

The `\markdownRendererO1BeginTight` 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.

```
240 \def\markdownRendererO1BeginTight{%
241   \markdownRendererO1BeginTightPrototype}%
```

The `\markdownRendererO1Item` 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.

```
242 \def\markdownRendererO1Item{%  
243   \markdownRendererO1ItemPrototype}%
```

The `\markdownRendererO1ItemEnd` macro represents the end of an item in an ordered list. The macro receives no arguments.

```
244 \def\markdownRendererO1ItemEnd{%  
245   \markdownRendererO1ItemEndPrototype}%
```

The `\markdownRendererO1ItemWithNumber` 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.

```
246 \def\markdownRendererO1ItemWithNumber{%  
247   \markdownRendererO1ItemWithNumberPrototype}%
```

The `\markdownRendererO1End` 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.

```
248 \def\markdownRendererO1End{%  
249   \markdownRendererO1EndPrototype}%
```

The `\markdownRendererO1EndTight` 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.

```
250 \def\markdownRendererO1EndTight{%  
251   \markdownRendererO1EndTightPrototype}%
```

2.2.3.13 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.

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

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.


```
254 \def\markdownRendererDlBeginTight{%
255   \markdownRendererDlBeginTightPrototype}%
```

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

```
256 \def\markdownRendererDlItem{%
257   \markdownRendererDlItemPrototype}%
```

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

```
258 \def\markdownRendererDlItemEnd{%
259   \markdownRendererDlItemEndPrototype}%
```

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

```
260 \def\markdownRendererDlDefinitionBegin{%
261   \markdownRendererDlDefinitionBeginPrototype}%
```

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

```
262 \def\markdownRendererDlDefinitionEnd{%
263   \markdownRendererDlDefinitionEndPrototype}%
```

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.

```
264 \def\markdownRendererDlEnd{%
265   \markdownRendererDlEndPrototype}%
```

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.

```
266 \def\markdownRendererDlEndTight{%
267   \markdownRendererDlEndTightPrototype}%
```

2.2.3.14 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.

```
268 \def\markdownRendererEmphasis{%
269   \markdownRendererEmphasisPrototype}%
```

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

```
270 \def\markdownRendererStrongEmphasis{%  
271   \markdownRendererStrongEmphasisPrototype}%
```

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

```
272 \def\markdownRendererBlockQuoteBegin{%  
273   \markdownRendererBlockQuoteBeginPrototype}%
```

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

```
274 \def\markdownRendererBlockQuoteEnd{%  
275   \markdownRendererBlockQuoteEndPrototype}%
```

2.2.3.16 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.

```
276 \def\markdownRendererInputVerbatim{%  
277   \markdownRendererInputVerbatimPrototype}%
```

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.

```
278 \def\markdownRendererInputFencedCode{%  
279   \markdownRendererInputFencedCodePrototype}%
```

2.2.3.17 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.

```
280 \def\markdownRendererJekyllDataBegin{%  
281   \markdownRendererJekyllDataBeginPrototype}%
```

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.

```
282 \def\markdownRendererJekyllDataEnd{%  
283   \markdownRendererJekyllDataEndPrototype}%
```

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.

```
284 \def\markdownRendererJekyllDataMappingBegin{%  
285   \markdownRendererJekyllDataMappingBeginPrototype}%
```

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.

```
286 \def\markdownRendererJekyllDataMappingEnd{%  
287   \markdownRendererJekyllDataMappingEndPrototype}%
```

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.

```
288 \def\markdownRendererJekyllDataSequenceBegin{%  
289   \markdownRendererJekyllDataSequenceBeginPrototype}%
```

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.

```
290 \def\markdownRendererJekyllDataSequenceEnd{%  
291   \markdownRendererJekyllDataSequenceEndPrototype}%
```

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.

```
292 \def\markdownRendererJekyllDataBoolean{%  
293   \markdownRendererJekyllDataBooleanPrototype}%
```

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.

```
294 \def\markdownRendererJekyllDataNumber{%  
295   \markdownRendererJekyllDataNumberPrototype}%
```

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.

```
296 \def\markdownRendererJekyllDataString{%  
297   \markdownRendererJekyllDataStringPrototype}%
```

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.

```
298 \def\markdownRendererJekyllDataEmpty{%  
299   \markdownRendererJekyllDataEmptyPrototype}%
```

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

```
300 \def\markdownRendererHeadingOne{%  
301   \markdownRendererHeadingOnePrototype}%
```

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

```
302 \def\markdownRendererHeadingTwo{%  
303   \markdownRendererHeadingTwoPrototype}%
```

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

```
304 \def\markdownRendererHeadingThree{%  
305   \markdownRendererHeadingThreePrototype}%
```

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

```
306 \def\markdownRendererHeadingFour{%  
307   \markdownRendererHeadingFourPrototype}%
```

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

```
308 \def\markdownRendererHeadingFive{%  
309   \markdownRendererHeadingFivePrototype}%
```

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

```
310 \def\markdownRendererHeadingSix{%  
311   \markdownRendererHeadingSixPrototype}%
```

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

```
312 \def\markdownRendererHorizontalRule{%  
313   \markdownRendererHorizontalRulePrototype}%
```

2.2.3.20 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.

```
314 \def\markdownRendererFootnote{%  
315   \markdownRendererFootnotePrototype}%
```

2.2.3.21 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.

```
316 \def\markdownRendererCite{%  
317   \markdownRendererCitePrototype}%
```

2.2.3.22 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.

```
318 \def\markdownRendererTextCite{%  
319   \markdownRendererTextCitePrototype}%
```

2.2.3.23 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.

```
320 \def\markdownRendererTable{%  
321   \markdownRendererTablePrototype}%
```

2.2.3.24 Inline HTML Comment Renderer 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.

```
322 \def\markdownRendererInlineHtmlComment{%
323   \markdownRendererInlineHtmlCommentPrototype}%
```

2.2.4 Token Renderer Prototypes

The following $\text{T}_{\text{E}}\text{X}$ macros provide definitions for the token renderers (see Section 2.2.3 on page 28) 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 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ and $\text{C}_{\text{O}}\text{N}\text{T}_{\text{E}}\text{X}\text{T}$ implementations (see sections 3.3 on page 173 and 3.4 on page 193).

```
324 \def\markdownRendererInterblockSeparatorPrototype{}%
325 \def\markdownRendererLineBreakPrototype{}%
326 \def\markdownRendererEllipsisPrototype{}%
327 \def\markdownRendererNbspPrototype{}%
328 \def\markdownRendererLeftBracePrototype{}%
329 \def\markdownRendererRightBracePrototype{}%
330 \def\markdownRendererDollarSignPrototype{}%
331 \def\markdownRendererPercentSignPrototype{}%
332 \def\markdownRendererAmpersandPrototype{}%
333 \def\markdownRendererUnderscorePrototype{}%
334 \def\markdownRendererHashPrototype{}%
335 \def\markdownRendererCircumflexPrototype{}%
336 \def\markdownRendererBackslashPrototype{}%
337 \def\markdownRendererTildePrototype{}%
338 \def\markdownRendererPipePrototype{}%
339 \def\markdownRendererCodeSpanPrototype#1{}%
340 \def\markdownRendererLinkPrototype#1#2#3#4{}%
341 \def\markdownRendererImagePrototype#1#2#3#4{}%
342 \def\markdownRendererContentBlockPrototype#1#2#3#4{}%
343 \def\markdownRendererContentBlockOnlineImagePrototype#1#2#3#4{}%
344 \def\markdownRendererContentBlockCodePrototype#1#2#3#4#5{}%
345 \def\markdownRendererUlBeginPrototype{}%
346 \def\markdownRendererUlBeginTightPrototype{}%
347 \def\markdownRendererUlItemPrototype{}%
348 \def\markdownRendererUlItemEndPrototype{}%
349 \def\markdownRendererUlEndPrototype{}%
350 \def\markdownRendererUlEndTightPrototype{}%
351 \def\markdownRendererOlBeginPrototype{}%
352 \def\markdownRendererOlBeginTightPrototype{}%
353 \def\markdownRendererOlItemPrototype{}%
354 \def\markdownRendererOlItemWithNumberPrototype#1{}%
355 \def\markdownRendererOlItemEndPrototype{}%
```

```

356 \def\markdownRendererO1EndPrototype{}%
357 \def\markdownRendererO1EndTightPrototype{}%
358 \def\markdownRendererDlBeginPrototype{}%
359 \def\markdownRendererDlBeginTightPrototype{}%
360 \def\markdownRendererDlItemPrototype#1{}%
361 \def\markdownRendererDlItemEndPrototype{}%
362 \def\markdownRendererDlDefinitionBeginPrototype{}%
363 \def\markdownRendererDlDefinitionEndPrototype{}%
364 \def\markdownRendererDlEndPrototype{}%
365 \def\markdownRendererDlEndTightPrototype{}%
366 \def\markdownRendererEmphasisPrototype#1{}%
367 \def\markdownRendererStrongEmphasisPrototype#1{}%
368 \def\markdownRendererBlockQuoteBeginPrototype{}%
369 \def\markdownRendererBlockQuoteEndPrototype{}%
370 \def\markdownRendererInputVerbatimPrototype#1{}%
371 \def\markdownRendererInputFencedCodePrototype#1#2{}%
372 \def\markdownRendererJekyllDataBooleanPrototype#1#2{}%
373 \def\markdownRendererJekyllDataEmptyPrototype#1{}%
374 \def\markdownRendererJekyllDataNumberPrototype#1#2{}%
375 \def\markdownRendererJekyllDataStringPrototype#1#2{}%
376 \def\markdownRendererJekyllDataBeginPrototype{}%
377 \def\markdownRendererJekyllDataEndPrototype{}%
378 \def\markdownRendererJekyllDataSequenceBeginPrototype#1#2{}%
379 \def\markdownRendererJekyllDataSequenceEndPrototype{}%
380 \def\markdownRendererJekyllDataMappingBeginPrototype#1#2{}%
381 \def\markdownRendererJekyllDataMappingEndPrototype{}%
382 \def\markdownRendererHeadingOnePrototype#1{}%
383 \def\markdownRendererHeadingTwoPrototype#1{}%
384 \def\markdownRendererHeadingThreePrototype#1{}%
385 \def\markdownRendererHeadingFourPrototype#1{}%
386 \def\markdownRendererHeadingFivePrototype#1{}%
387 \def\markdownRendererHeadingSixPrototype#1{}%
388 \def\markdownRendererHorizontalRulePrototype{}%
389 \def\markdownRendererFootnotePrototype#1{}%
390 \def\markdownRendererCitePrototype#1{}%
391 \def\markdownRendererTextCitePrototype#1{}%
392 \def\markdownRendererTablePrototype#1#2#3{}%
393 \def\markdownRendererInlineHtmlCommentPrototype#1{}%
394 \def\markdownRendererTickedBoxPrototype{}%
395 \def\markdownRendererHalfTickedBoxPrototype{}%
396 \def\markdownRendererUntickedBoxPrototype{}%

```

2.2.5 Logging Facilities

The `\markdownInfo`, `\markdownWarning`, and `\markdownError` macros perform logging for the Markdown package. Their first argument specifies the text of the info, warning, or error message. The `\markdownError` macro receives a second argument

that provides a help text. You may redefine these macros to redirect and process the info, warning, and error messages.

2.2.6 Miscellanea

The `\markdownMakeOther` macro is used by the package, when a T_EX engine that does not support direct Lua access is starting to buffer a text. The plain T_EX implementation changes the category code of plain T_EX 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*.

```
397 \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 T_EX 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.

```
398 \let\markdownReadAndConvert\relax
```

```
399 \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*.

```
400 \catcode`\|=0\catcode`\=12%
401 |gdef|markdownBegin{%
402   |markdownReadAndConvert{\markdownEnd}%
403   {|\markdownEnd}}%
404 |endgroup
```

The macro is exposed in the interface, so that the user can create their own markdown environments. Due to the way the arguments are passed to Lua (see Section 3.2.7 on page 171), the first argument may not contain the string `]]` (regardless of the category code of the bracket symbol (`]`)).

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

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

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

```
405 \ifx\markdownMode\undefined
```

```
406 \ifx\directlua\undefined
```



```

407     \def\markdownMode{0}%
408     \else
409     \def\markdownMode{2}%
410     \fi
411 \fi

```

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

```

412 \def\markdownLuaRegisterIBCallback#1{\relax}%
413 \def\markdownLuaUnregisterIBCallback#1{\relax}%

```

2.3 \LaTeX Interface

The \LaTeX interface provides \LaTeX environments for the typesetting of markdown input from within \LaTeX , facilities for setting Lua interface options (see Section 2.1.2 on page 7) used during the conversion from markdown to plain \TeX , and facilities for changing the way markdown tokens are rendered. The rest of the interface is inherited from the plain \TeX interface (see Section 2.2 on page 23).

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

```

\usepackage[<options>]{markdown}

```

where *<options>* are the \LaTeX interface options (see Section 2.3.2 on the following page). Note that *<options>* inside the `\usepackage` macro may not set the `markdownRenderers` (see Section 2.3.2.5 on page 51) and `markdownRendererPrototypes` (see Section 2.3.2.6 on page 55) keys. This limitation is due to the way $\text{\LaTeX} 2_{\epsilon}$ parses package options.

2.3.1 Typesetting Markdown

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

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

```

414 \newenvironment{markdown}\relax\relax
415 \newenvironment{markdown*}[1]\relax\relax

```

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

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

The following example L^AT_EX code showcases the usage of the `markdown` and `markdown*` environments:

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

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

The following example L^AT_EX 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 L^AT_EX 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 on the next page, and the L^AT_EX themes described in Section 2.3.2.2 on page 44, and the L^AT_EX setup snippets described in Section 2.3.2.3 on page 48, L^AT_EX options map directly to the options recognized by the plain T_EX interface (see Section 2.2.2 on page 25) and to the markdown token renderers and their prototypes recognized by the plain T_EX interface (see Sections 2.2.3 on page 28 and 2.2.4 on page 38).

The L^AT_EX options may be specified when loading the L^AT_EX package, when using the `markdown*` L^AT_EX environment or the `\markdownInput` macro (see Section 2.3 on

page 41), or via the `\markdownSetup` macro. The `\markdownSetup` macro receives the options to set up as its only argument:

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

We may also store L^AT_EX 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:

```
418 \newcommand\markdownSetupSnippet[2]{%
419   \@ifundefined
420     {markdownLaTeXSetupSnippet\markdownLaTeXThemeName#1}{%
421     \newtoks\next
422     \next={#2}%
423     \expandafter\let\csname markdownLaTeXSetupSnippet%
424     \markdownLaTeXThemeName#1\endcsname=\next
425   }{%
426     \markdownWarning
427     {Redefined setup snippet \markdownLaTeXThemeName#1}%
428     \csname markdownLaTeXSetupSnippet%
429     \markdownLaTeXThemeName#1\endcsname={#2}%
430   }}%
```

See Section 2.3.2.2 on the following page for information on interactions between setup snippets and L^AT_EX themes. See Section 2.3.2.3 on page 48 for information about invoking the stored setup snippets.

2.3.2.1 No default token renderer prototypes Default token renderer prototypes require L^AT_EX 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_EX implementation (see Section 3.2.3 on page 162) and prevent the soft L^AT_EX prerequisites in Section 1.1.3 on page 3 from being loaded:

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

```
431 \newif\ifmarkdownLaTeXPlain
432 \markdownLaTeXPlainfalse
433 \define@key{markdownOptions}{plain}[true]{%
434   \ifmarkdownLaTeXLoaded
435     \markdownWarning
436     {The plain option must be specified when loading the package}%
437   \else
438     \markdownLaTeXPlaintrue
439   \fi}
```

2.3.2.2 L^AT_EX themes

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

The L^AT_EX option with key `theme` loads a L^AT_EX 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^AT_EX package, which provides similar functionality with its `\usetheme` macro [5, Section 15.1].

Theme names must be qualified to minimize naming conflicts between different themes intended for a single L^AT_EX document class or for a single L^AT_EX 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^AT_EX packages are identified only by their filenames, not by their pathnames. [6] 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^AT_EX package named `markdownthemewitiko_beamer_MU.sty`.

If the L^AT_EX 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^AT_EX 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^AT_EX package, and finally the `markdownthemewitiko_dot.sty` L^AT_EX package:

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

```
440 \newif\ifmarkdownLaTeXLoaded
441 \markdownLaTeXLoadedfalse
442 \AtEndOfPackage{\markdownLaTeXLoadedtrue}%
443 \define@key{markdownOptions}{theme}{%
```

```

444 \IfSubStr{#1}{/}{/}{/}{%
445   \markdownError
446   {Won't load theme with unqualified name #1}%
447   {Theme names must contain at least one forward slash}}%
448 \StrSubstitute{#1}{/}{_}{\markdownLaTeXThemePackageName}%
449 \edef\markdownLaTeXThemePackageName{%
450   markdowntheme\markdownLaTeXThemePackageName}%
451 \expandafter\markdownLaTeXThemeLoad\expandafter{%
452   \markdownLaTeXThemePackageName}{#1/}}%
453 \newcommand\markdownLaTeXThemeName{}%
454 \newcommand\markdownLaTeXThemeLoad[2]{%
455   \ifmarkdownLaTeXLoaded
456     \def\markdownLaTeXThemeName{#2}%
457     \RequirePackage{#1}%
458     \def\markdownLaTeXThemeName{}%
459   \else
460     \AtEndOfPackage{%
461       \def\markdownLaTeXThemeName{#2}%
462       \RequirePackage{#1}%
463       \def\markdownLaTeXThemeName{}}%
464   \fi}%

```

The \LaTeX themes have a useful synergy with the setup snippets (see Section 2.3.2 on page 42): To make it less likely that different themes will define setup snippets with the same name, we will prepend \langle 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 \LaTeX , themes may not be loaded after the beginning of a \LaTeX document.

```
465 \@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 on the next page.

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

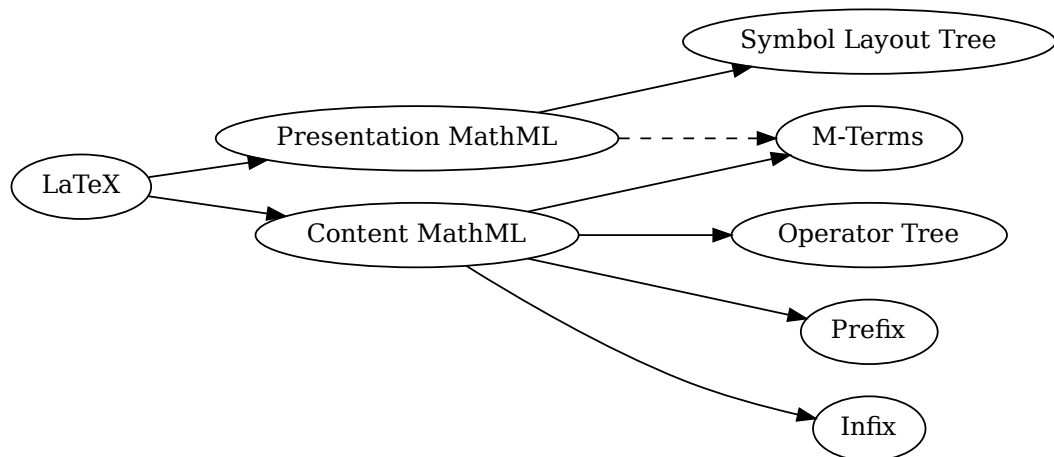
466 \ProvidesPackage{markdownthemewitiko\_dot}[2021/03/09]%

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

```

\documentclass{article}
\usepackage[theme=witiko/graphicx/http]{markdown}
\begin{document}
\begin{markdown}
![img](https://github.com/witiko/markdown/raw/main/markdown.png)

```



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

```

 "The banner of the Markdown package")
\end{markdown}
\end{document}

```

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

```
467 \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”.

```

\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

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

Please, see Section 3.3.3.1 on page 175 for implementation details of the example themes.

**2.3.2.3 L<sup>A</sup>T<sub>E</sub>X setup snippets** The L<sup>A</sup>T<sub>E</sub>X option with key `snippet` invokes a snippet named `<value>`:

```
469 \define@key{markdownOptions}{snippet}{%
470 \expandafter\markdownSetup\expandafter{%
471 \the\csname markdownLaTeXSetupSnippet#1\endcsname}}%
```

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

```

\markdownSetupSnippet{romanNumerals}{
 renderers = {
 oItemWithNumber = {\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<sub>E</sub>X Interface Options** The following options map directly to the option macros exposed by the plain T<sub>E</sub>X interface (see Section 2.2.2 on page 25).

```

472 \define@key{markdownOptions}{helperScriptFileName}{%
473 \def\markdownOptionHelperScriptFileName{#1}}%
474 \define@key{markdownOptions}{inputTempFileName}{%
475 \def\markdownOptionInputTempFileName{#1}}%
476 \define@key{markdownOptions}{outputTempFileName}{%
477 \def\markdownOptionOutputTempFileName{#1}}%
478 \define@key{markdownOptions}{errorTempFileName}{%
479 \def\markdownOptionErrorTempFileName{#1}}%
480 \define@key{markdownOptions}{cacheDir}{%
481 \def\markdownOptionCacheDir{#1}}%
482 \define@key{markdownOptions}{outputDir}{%
483 \def\markdownOptionOutputDir{#1}}%
484 \define@key{markdownOptions}{blankBeforeBlockquote}[true]{%
485 \def\markdownOptionBlankBeforeBlockquote{#1}}%
486 \define@key{markdownOptions}{blankBeforeCodeFence}[true]{%
487 \def\markdownOptionBlankBeforeCodeFence{#1}}%
488 \define@key{markdownOptions}{blankBeforeHeading}[true]{%
489 \def\markdownOptionBlankBeforeHeading{#1}}%
490 \define@key{markdownOptions}{breakableBlockquotes}[true]{%
491 \def\markdownOptionBreakableBlockquotes{#1}}%
492 \define@key{markdownOptions}{citations}[true]{%
493 \def\markdownOptionCitations{#1}}%
494 \define@key{markdownOptions}{citationNbsps}[true]{%
495 \def\markdownOptionCitationNbsps{#1}}%
496 \define@key{markdownOptions}{contentBlocks}[true]{%
497 \def\markdownOptionContentBlocks{#1}}%
498 \define@key{markdownOptions}{codeSpans}[true]{%
499 \def\markdownOptionCodeSpans{#1}}%
500 \define@key{markdownOptions}{contentBlocksLanguageMap}{%

```

```

501 \def\markdownOptionContentBlocksLanguageMap{#1}}%
502 \define@key{markdownOptions}{definitionLists}[true]{%
503 \def\markdownOptionDefinitionLists{#1}}%
504 \define@key{markdownOptions}{eagerCache}[true]{%
505 \def\markdownOptionEagerCache{#1}}%
506 \define@key{markdownOptions}{footnotes}[true]{%
507 \def\markdownOptionFootnotes{#1}}%
508 \define@key{markdownOptions}{fencedCode}[true]{%
509 \def\markdownOptionFencedCode{#1}}%
510 \define@key{markdownOptions}{jekyllData}[true]{%
511 \def\markdownOptionJekyllData{#1}}%
512 \define@key{markdownOptions}{hardLineBreaks}[true]{%
513 \def\markdownOptionHardLineBreaks{#1}}%
514 \define@key{markdownOptions}{hashEnumerators}[true]{%
515 \def\markdownOptionHashEnumerators{#1}}%
516 \define@key{markdownOptions}{headerAttributes}[true]{%
517 \def\markdownOptionHeaderAttributes{#1}}%
518 \define@key{markdownOptions}{html}[true]{%
519 \def\markdownOptionHtml{#1}}%
520 \define@key{markdownOptions}{hybrid}[true]{%
521 \def\markdownOptionHybrid{#1}}%
522 \define@key{markdownOptions}{inlineFootnotes}[true]{%
523 \def\markdownOptionInlineFootnotes{#1}}%
524 \define@key{markdownOptions}{pipeTables}[true]{%
525 \def\markdownOptionPipeTables{#1}}%
526 \define@key{markdownOptions}{preserveTabs}[true]{%
527 \def\markdownOptionPreserveTabs{#1}}%
528 \define@key{markdownOptions}{smartEllipses}[true]{%
529 \def\markdownOptionSmartEllipses{#1}}%
530 \define@key{markdownOptions}{shiftHeadings}{%
531 \def\markdownOptionShiftHeadings{#1}}%
532 \define@key{markdownOptions}{slice}{%
533 \def\markdownOptionSlice{#1}}%
534 \define@key{markdownOptions}{startNumber}[true]{%
535 \def\markdownOptionStartNumber{#1}}%
536 \define@key{markdownOptions}{stripIndent}[true]{%
537 \def\markdownOptionStripIndent{#1}}%
538 \define@key{markdownOptions}{tableCaptions}[true]{%
539 \def\markdownOptionTableCaptions{#1}}%
540 \define@key{markdownOptions}{taskLists}[true]{%
541 \def\markdownOptionTaskLists{#1}}%
542 \define@key{markdownOptions}{texComments}[true]{%
543 \def\markdownOptionTeXComments{#1}}%
544 \define@key{markdownOptions}{tightLists}[true]{%
545 \def\markdownOptionTightLists{#1}}%
546 \define@key{markdownOptions}{underscores}[true]{%
547 \def\markdownOptionUnderscores{#1}}%

```

```

548 \define@key{markdownOptions}{stripPercentSigns}[true]{%
549 \def\markdownOptionStripPercentSigns{#1}}%

```

The `\markdownOptionFinalizeCache` and `\markdownOptionFrozenCache` plain TeX options are exposed through L<sup>A</sup>T<sub>E</sub>X options with keys `finalizeCache` and `frozenCache`.

To ensure compatibility with the `minted` package [7, Section 5.1], which supports the `finalizcache` and `frozencache` package options with similar semantics, the Markdown package also recognizes these as aliases and recognizes them as document class options. By passing `finalizcache` and `frozencache` as document class options, you may conveniently control the behavior of both packages at once:

```

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

```

We hope that other packages will support the `finalizcache` and `frozencache` package options in the future, so that they can become a standard interface for preparing L<sup>A</sup>T<sub>E</sub>X document sources for distribution.

```

550 \define@key{markdownOptions}{finalizeCache}[true]{%
551 \def\markdownOptionFinalizeCache{#1}}%
552 \DeclareOption{finalizcache}{\markdownSetup{finalizeCache}}
553 \define@key{markdownOptions}{frozenCache}[true]{%
554 \def\markdownOptionFrozenCache{#1}}%
555 \DeclareOption{frozencache}{\markdownSetup{frozenCache}}
556 \define@key{markdownOptions}{frozenCacheFileName}{%
557 \def\markdownOptionFrozenCacheFileName{#1}}%

```

The following example L<sup>A</sup>T<sub>E</sub>X code showcases a possible configuration of plain TeX interface options `\markdownOptionHybrid`, `\markdownOptionSmartEllipses`, and `\markdownOptionCacheDir`.

```

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

```

### 2.3.2.5 Plain TeX Markdown Token Renderers

The L<sup>A</sup>T<sub>E</sub>X interface recognizes an option with the `renderers` key, whose value must be a list of options that map directly to the markdown token renderer macros exposed by the plain TeX interface (see Section 2.2.3 on page 28).

```

558 \define@key{markdownRenderers}{interblockSeparator}{%
559 \renewcommand\markdownRendererInterblockSeparator{#1}}%
560 \define@key{markdownRenderers}{lineBreak}{%
561 \renewcommand\markdownRendererLineBreak{#1}}%
562 \define@key{markdownRenderers}{ellipsis}{%
563 \renewcommand\markdownRendererEllipsis{#1}}%
564 \define@key{markdownRenderers}{nbsp}{%
565 \renewcommand\markdownRendererNbsp{#1}}%
566 \define@key{markdownRenderers}{leftBrace}{%
567 \renewcommand\markdownRendererLeftBrace{#1}}%
568 \define@key{markdownRenderers}{rightBrace}{%
569 \renewcommand\markdownRendererRightBrace{#1}}%
570 \define@key{markdownRenderers}{dollarSign}{%
571 \renewcommand\markdownRendererDollarSign{#1}}%
572 \define@key{markdownRenderers}{percentSign}{%
573 \renewcommand\markdownRendererPercentSign{#1}}%
574 \define@key{markdownRenderers}{ampersand}{%
575 \renewcommand\markdownRendererAmpersand{#1}}%
576 \define@key{markdownRenderers}{underscore}{%
577 \renewcommand\markdownRendererUnderscore{#1}}%
578 \define@key{markdownRenderers}{hash}{%
579 \renewcommand\markdownRendererHash{#1}}%
580 \define@key{markdownRenderers}{circumflex}{%
581 \renewcommand\markdownRendererCircumflex{#1}}%
582 \define@key{markdownRenderers}{backslash}{%
583 \renewcommand\markdownRendererBackslash{#1}}%
584 \define@key{markdownRenderers}{tilde}{%
585 \renewcommand\markdownRendererTilde{#1}}%
586 \define@key{markdownRenderers}{pipe}{%
587 \renewcommand\markdownRendererPipe{#1}}%
588 \define@key{markdownRenderers}{codeSpan}{%
589 \renewcommand\markdownRendererCodeSpan[1]{#1}}%
590 \define@key{markdownRenderers}{link}{%
591 \renewcommand\markdownRendererLink[4]{#1}}%
592 \define@key{markdownRenderers}{contentBlock}{%
593 \renewcommand\markdownRendererContentBlock[4]{#1}}%
594 \define@key{markdownRenderers}{contentBlockOnlineImage}{%
595 \renewcommand\markdownRendererContentBlockOnlineImage[4]{#1}}%
596 \define@key{markdownRenderers}{contentBlockCode}{%
597 \renewcommand\markdownRendererContentBlockCode[5]{#1}}%
598 \define@key{markdownRenderers}{image}{%
599 \renewcommand\markdownRendererImage[4]{#1}}%
600 \define@key{markdownRenderers}{ulBegin}{%
601 \renewcommand\markdownRendererUlBegin{#1}}%
602 \define@key{markdownRenderers}{ulBeginTight}{%
603 \renewcommand\markdownRendererUlBeginTight{#1}}%
604 \define@key{markdownRenderers}{ulItem}{%

```

```

605 \renewcommand\markdownRendererUlItem{#1}}%
606 \define@key{markdownRenderers}{ulItemEnd}{%
607 \renewcommand\markdownRendererUlItemEnd{#1}}%
608 \define@key{markdownRenderers}{ulEnd}{%
609 \renewcommand\markdownRendererUlEnd{#1}}%
610 \define@key{markdownRenderers}{ulEndTight}{%
611 \renewcommand\markdownRendererUlEndTight{#1}}%
612 \define@key{markdownRenderers}{olBegin}{%
613 \renewcommand\markdownRendererOlBegin{#1}}%
614 \define@key{markdownRenderers}{olBeginTight}{%
615 \renewcommand\markdownRendererOlBeginTight{#1}}%
616 \define@key{markdownRenderers}{olItem}{%
617 \renewcommand\markdownRendererOlItem{#1}}%
618 \define@key{markdownRenderers}{olItemWithNumber}{%
619 \renewcommand\markdownRendererOlItemWithNumber [1]{#1}}%
620 \define@key{markdownRenderers}{olItemEnd}{%
621 \renewcommand\markdownRendererOlItemEnd{#1}}%
622 \define@key{markdownRenderers}{olEnd}{%
623 \renewcommand\markdownRendererOlEnd{#1}}%
624 \define@key{markdownRenderers}{olEndTight}{%
625 \renewcommand\markdownRendererOlEndTight{#1}}%
626 \define@key{markdownRenderers}{dlBegin}{%
627 \renewcommand\markdownRendererDlBegin{#1}}%
628 \define@key{markdownRenderers}{dlBeginTight}{%
629 \renewcommand\markdownRendererDlBeginTight{#1}}%
630 \define@key{markdownRenderers}{dlItem}{%
631 \renewcommand\markdownRendererDlItem [1]{#1}}%
632 \define@key{markdownRenderers}{dlItemEnd}{%
633 \renewcommand\markdownRendererDlItemEnd{#1}}%
634 \define@key{markdownRenderers}{dlDefinitionBegin}{%
635 \renewcommand\markdownRendererDlDefinitionBegin{#1}}%
636 \define@key{markdownRenderers}{dlDefinitionEnd}{%
637 \renewcommand\markdownRendererDlDefinitionEnd{#1}}%
638 \define@key{markdownRenderers}{dlEnd}{%
639 \renewcommand\markdownRendererDlEnd{#1}}%
640 \define@key{markdownRenderers}{dlEndTight}{%
641 \renewcommand\markdownRendererDlEndTight{#1}}%
642 \define@key{markdownRenderers}{emphasis}{%
643 \renewcommand\markdownRendererEmphasis [1]{#1}}%
644 \define@key{markdownRenderers}{strongEmphasis}{%
645 \renewcommand\markdownRendererStrongEmphasis [1]{#1}}%
646 \define@key{markdownRenderers}{blockquoteBegin}{%
647 \renewcommand\markdownRendererBlockQuoteBegin{#1}}%
648 \define@key{markdownRenderers}{blockquoteEnd}{%
649 \renewcommand\markdownRendererBlockQuoteEnd{#1}}%
650 \define@key{markdownRenderers}{inputVerbatim}{%
651 \renewcommand\markdownRendererInputVerbatim [1]{#1}}%

```

```

652 \define@key{markdownRenderers}{inputFencedCode}{%
653 \renewcommand\markdownRendererInputFencedCode[2]{#1}}%
654 \define@key{markdownRenderers}{jekyllDataBoolean}{%
655 \renewcommand\markdownRendererJekyllDataBoolean[2]{#1}}%
656 \define@key{markdownRenderers}{jekyllDataEmpty}{%
657 \renewcommand\markdownRendererJekyllDataEmpty[1]{#1}}%
658 \define@key{markdownRenderers}{jekyllDataNumber}{%
659 \renewcommand\markdownRendererJekyllDataNumber[2]{#1}}%
660 \define@key{markdownRenderers}{jekyllDataString}{%
661 \renewcommand\markdownRendererJekyllDataString[2]{#1}}%
662 \define@key{markdownRenderers}{jekyllDataBegin}{%
663 \renewcommand\markdownRendererJekyllDataBegin{#1}}%
664 \define@key{markdownRenderers}{jekyllDataEnd}{%
665 \renewcommand\markdownRendererJekyllDataEnd{#1}}%
666 \define@key{markdownRenderers}{jekyllDataSequenceBegin}{%
667 \renewcommand\markdownRendererJekyllDataSequenceBegin[2]{#1}}%
668 \define@key{markdownRenderers}{jekyllDataSequenceEnd}{%
669 \renewcommand\markdownRendererJekyllDataSequenceEnd{#1}}%
670 \define@key{markdownRenderers}{jekyllDataMappingBegin}{%
671 \renewcommand\markdownRendererJekyllDataMappingBegin[2]{#1}}%
672 \define@key{markdownRenderers}{jekyllDataMappingEnd}{%
673 \renewcommand\markdownRendererJekyllDataMappingEnd{#1}}%
674 \define@key{markdownRenderers}{headingOne}{%
675 \renewcommand\markdownRendererHeadingOne[1]{#1}}%
676 \define@key{markdownRenderers}{headingTwo}{%
677 \renewcommand\markdownRendererHeadingTwo[1]{#1}}%
678 \define@key{markdownRenderers}{headingThree}{%
679 \renewcommand\markdownRendererHeadingThree[1]{#1}}%
680 \define@key{markdownRenderers}{headingFour}{%
681 \renewcommand\markdownRendererHeadingFour[1]{#1}}%
682 \define@key{markdownRenderers}{headingFive}{%
683 \renewcommand\markdownRendererHeadingFive[1]{#1}}%
684 \define@key{markdownRenderers}{headingSix}{%
685 \renewcommand\markdownRendererHeadingSix[1]{#1}}%
686 \define@key{markdownRenderers}{horizontalRule}{%
687 \renewcommand\markdownRendererHorizontalRule{#1}}%
688 \define@key{markdownRenderers}{footnote}{%
689 \renewcommand\markdownRendererFootnote[1]{#1}}%
690 \define@key{markdownRenderers}{cite}{%
691 \renewcommand\markdownRendererCite[1]{#1}}%
692 \define@key{markdownRenderers}{textCite}{%
693 \renewcommand\markdownRendererTextCite[1]{#1}}%
694 \define@key{markdownRenderers}{table}{%
695 \renewcommand\markdownRendererTable[3]{#1}}%
696 \define@key{markdownRenderers}{inlineHtmlComment}{%
697 \renewcommand\markdownRendererInlineHtmlComment[1]{#1}}%
698 \define@key{markdownRenderers}{tickedBox}{%

```

```

699 \renewcommand\markdownRendererTickedBox{#1}}%
700 \define@key{markdownRenderers}{halfTickedBox}{%
701 \renewcommand\markdownRendererHalfTickedBox{#1}}%
702 \define@key{markdownRenderers}{untickedBox}{%
703 \renewcommand\markdownRendererUntickedBox{#1}}%

```

The following example L<sup>A</sup>T<sub>E</sub>X 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<sub>E</sub>X Markdown Token Renderer Prototypes** The L<sup>A</sup>T<sub>E</sub>X 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<sub>E</sub>X interface (see Section 2.2.4 on page 38).

```

704 \define@key{markdownRendererPrototypes}{interblockSeparator}{%
705 \renewcommand\markdownRendererInterblockSeparatorPrototype{#1}}%
706 \define@key{markdownRendererPrototypes}{lineBreak}{%
707 \renewcommand\markdownRendererLineBreakPrototype{#1}}%
708 \define@key{markdownRendererPrototypes}{ellipsis}{%
709 \renewcommand\markdownRendererEllipsisPrototype{#1}}%
710 \define@key{markdownRendererPrototypes}{nbsp}{%
711 \renewcommand\markdownRendererNbspPrototype{#1}}%
712 \define@key{markdownRendererPrototypes}{leftBrace}{%
713 \renewcommand\markdownRendererLeftBracePrototype{#1}}%
714 \define@key{markdownRendererPrototypes}{rightBrace}{%
715 \renewcommand\markdownRendererRightBracePrototype{#1}}%
716 \define@key{markdownRendererPrototypes}{dollarSign}{%
717 \renewcommand\markdownRendererDollarSignPrototype{#1}}%
718 \define@key{markdownRendererPrototypes}{percentSign}{%
719 \renewcommand\markdownRendererPercentSignPrototype{#1}}%
720 \define@key{markdownRendererPrototypes}{ampersand}{%
721 \renewcommand\markdownRendererAmpersandPrototype{#1}}%
722 \define@key{markdownRendererPrototypes}{underscore}{%
723 \renewcommand\markdownRendererUnderscorePrototype{#1}}%
724 \define@key{markdownRendererPrototypes}{hash}{%
725 \renewcommand\markdownRendererHashPrototype{#1}}%
726 \define@key{markdownRendererPrototypes}{circumflex}{%
727 \renewcommand\markdownRendererCircumflexPrototype{#1}}%

```

```

728 \define@key{markdownRendererPrototypes}{backslash}{%
729 \renewcommand\markdownRendererBackslashPrototype{#1}}%
730 \define@key{markdownRendererPrototypes}{tilde}{%
731 \renewcommand\markdownRendererTildePrototype{#1}}%
732 \define@key{markdownRendererPrototypes}{pipe}{%
733 \renewcommand\markdownRendererPipePrototype{#1}}%
734 \define@key{markdownRendererPrototypes}{codeSpan}{%
735 \renewcommand\markdownRendererCodeSpanPrototype[1]{#1}}%
736 \define@key{markdownRendererPrototypes}{link}{%
737 \renewcommand\markdownRendererLinkPrototype[4]{#1}}%
738 \define@key{markdownRendererPrototypes}{contentBlock}{%
739 \renewcommand\markdownRendererContentBlockPrototype[4]{#1}}%
740 \define@key{markdownRendererPrototypes}{contentBlockOnlineImage}{%
741 \renewcommand\markdownRendererContentBlockOnlineImagePrototype[4]{#1}}%
742 \define@key{markdownRendererPrototypes}{contentBlockCode}{%
743 \renewcommand\markdownRendererContentBlockCodePrototype[5]{#1}}%
744 \define@key{markdownRendererPrototypes}{image}{%
745 \renewcommand\markdownRendererImagePrototype[4]{#1}}%
746 \define@key{markdownRendererPrototypes}{ulBegin}{%
747 \renewcommand\markdownRendererUlBeginPrototype{#1}}%
748 \define@key{markdownRendererPrototypes}{ulBeginTight}{%
749 \renewcommand\markdownRendererUlBeginTightPrototype{#1}}%
750 \define@key{markdownRendererPrototypes}{ulItem}{%
751 \renewcommand\markdownRendererUlItemPrototype{#1}}%
752 \define@key{markdownRendererPrototypes}{ulItemEnd}{%
753 \renewcommand\markdownRendererUlItemEndPrototype{#1}}%
754 \define@key{markdownRendererPrototypes}{ulEnd}{%
755 \renewcommand\markdownRendererUlEndPrototype{#1}}%
756 \define@key{markdownRendererPrototypes}{ulEndTight}{%
757 \renewcommand\markdownRendererUlEndTightPrototype{#1}}%
758 \define@key{markdownRendererPrototypes}{olBegin}{%
759 \renewcommand\markdownRendererOlBeginPrototype{#1}}%
760 \define@key{markdownRendererPrototypes}{olBeginTight}{%
761 \renewcommand\markdownRendererOlBeginTightPrototype{#1}}%
762 \define@key{markdownRendererPrototypes}{olItem}{%
763 \renewcommand\markdownRendererOlItemPrototype{#1}}%
764 \define@key{markdownRendererPrototypes}{olItemWithNumber}{%
765 \renewcommand\markdownRendererOlItemWithNumberPrototype[1]{#1}}%
766 \define@key{markdownRendererPrototypes}{olItemEnd}{%
767 \renewcommand\markdownRendererOlItemEndPrototype{#1}}%
768 \define@key{markdownRendererPrototypes}{olEnd}{%
769 \renewcommand\markdownRendererOlEndPrototype{#1}}%
770 \define@key{markdownRendererPrototypes}{olEndTight}{%
771 \renewcommand\markdownRendererOlEndTightPrototype{#1}}%
772 \define@key{markdownRendererPrototypes}{dlBegin}{%
773 \renewcommand\markdownRendererDlBeginPrototype{#1}}%
774 \define@key{markdownRendererPrototypes}{dlBeginTight}{%

```



```

775 \renewcommand\markdownRendererDlBeginTightPrototype{#1}}%
776 \define@key{markdownRendererPrototypes}{dlItem}{%
777 \renewcommand\markdownRendererDlItemPrototype[1]{#1}}%
778 \define@key{markdownRendererPrototypes}{dlItemEnd}{%
779 \renewcommand\markdownRendererDlItemEndPrototype{#1}}%
780 \define@key{markdownRendererPrototypes}{dlDefinitionBegin}{%
781 \renewcommand\markdownRendererDlDefinitionBeginPrototype{#1}}%
782 \define@key{markdownRendererPrototypes}{dlDefinitionEnd}{%
783 \renewcommand\markdownRendererDlDefinitionEndPrototype{#1}}%
784 \define@key{markdownRendererPrototypes}{dlEnd}{%
785 \renewcommand\markdownRendererDlEndPrototype{#1}}%
786 \define@key{markdownRendererPrototypes}{dlEndTight}{%
787 \renewcommand\markdownRendererDlEndTightPrototype{#1}}%
788 \define@key{markdownRendererPrototypes}{emphasis}{%
789 \renewcommand\markdownRendererEmphasisPrototype[1]{#1}}%
790 \define@key{markdownRendererPrototypes}{strongEmphasis}{%
791 \renewcommand\markdownRendererStrongEmphasisPrototype[1]{#1}}%
792 \define@key{markdownRendererPrototypes}{blockquoteBegin}{%
793 \renewcommand\markdownRendererBlockQuoteBeginPrototype{#1}}%
794 \define@key{markdownRendererPrototypes}{blockquoteEnd}{%
795 \renewcommand\markdownRendererBlockQuoteEndPrototype{#1}}%
796 \define@key{markdownRendererPrototypes}{inputVerbatim}{%
797 \renewcommand\markdownRendererInputVerbatimPrototype[1]{#1}}%
798 \define@key{markdownRendererPrototypes}{inputFencedCode}{%
799 \renewcommand\markdownRendererInputFencedCodePrototype[2]{#1}}%
800 \define@key{markdownRendererPrototypes}{jekyllDataBoolean}{%
801 \renewcommand\markdownRendererJekyllDataBooleanPrototype[2]{#1}}%
802 \define@key{markdownRendererPrototypes}{jekyllDataEmpty}{%
803 \renewcommand\markdownRendererJekyllDataEmptyPrototype[1]{#1}}%
804 \define@key{markdownRendererPrototypes}{jekyllDataNumber}{%
805 \renewcommand\markdownRendererJekyllDataNumberPrototype[2]{#1}}%
806 \define@key{markdownRendererPrototypes}{jekyllDataString}{%
807 \renewcommand\markdownRendererJekyllDataStringPrototype[2]{#1}}%
808 \define@key{markdownRendererPrototypes}{jekyllDataBegin}{%
809 \renewcommand\markdownRendererJekyllDataBeginPrototype{#1}}%
810 \define@key{markdownRendererPrototypes}{jekyllDataEnd}{%
811 \renewcommand\markdownRendererJekyllDataEndPrototype{#1}}%
812 \define@key{markdownRendererPrototypes}{jekyllDataSequenceBegin}{%
813 \renewcommand\markdownRendererJekyllDataSequenceBeginPrototype[2]{#1}}%
814 \define@key{markdownRendererPrototypes}{jekyllDataSequenceEnd}{%
815 \renewcommand\markdownRendererJekyllDataSequenceEndPrototype{#1}}%
816 \define@key{markdownRendererPrototypes}{jekyllDataMappingBegin}{%
817 \renewcommand\markdownRendererJekyllDataMappingBeginPrototype[2]{#1}}%
818 \define@key{markdownRendererPrototypes}{jekyllDataMappingEnd}{%
819 \renewcommand\markdownRendererJekyllDataMappingEndPrototype{#1}}%
820 \define@key{markdownRendererPrototypes}{headingOne}{%
821 \renewcommand\markdownRendererHeadingOnePrototype[1]{#1}}%

```

```

822 \define@key{markdownRendererPrototypes}{headingTwo}{%
823 \renewcommand\markdownRendererHeadingTwoPrototype[1]{#1}}%
824 \define@key{markdownRendererPrototypes}{headingThree}{%
825 \renewcommand\markdownRendererHeadingThreePrototype[1]{#1}}%
826 \define@key{markdownRendererPrototypes}{headingFour}{%
827 \renewcommand\markdownRendererHeadingFourPrototype[1]{#1}}%
828 \define@key{markdownRendererPrototypes}{headingFive}{%
829 \renewcommand\markdownRendererHeadingFivePrototype[1]{#1}}%
830 \define@key{markdownRendererPrototypes}{headingSix}{%
831 \renewcommand\markdownRendererHeadingSixPrototype[1]{#1}}%
832 \define@key{markdownRendererPrototypes}{horizontalRule}{%
833 \renewcommand\markdownRendererHorizontalRulePrototype{#1}}%
834 \define@key{markdownRendererPrototypes}{footnote}{%
835 \renewcommand\markdownRendererFootnotePrototype[1]{#1}}%
836 \define@key{markdownRendererPrototypes}{cite}{%
837 \renewcommand\markdownRendererCitePrototype[1]{#1}}%
838 \define@key{markdownRendererPrototypes}{textCite}{%
839 \renewcommand\markdownRendererTextCitePrototype[1]{#1}}%
840 \define@key{markdownRendererPrototypes}{table}{%
841 \renewcommand\markdownRendererTablePrototype[3]{#1}}%
842 \define@key{markdownRendererPrototypes}{inlineHtmlComment}{%
843 \renewcommand\markdownRendererInlineHtmlCommentPrototype[1]{#1}}%
844 \define@key{markdownRendererPrototypes}{tickedBox}{%
845 \renewcommand\markdownRendererTickedBoxPrototype{#1}}%
846 \define@key{markdownRendererPrototypes}{halfTickedBox}{%
847 \renewcommand\markdownRendererHalfTickedBoxPrototype{#1}}%
848 \define@key{markdownRendererPrototypes}{untickedBox}{%
849 \renewcommand\markdownRendererUntickedBoxPrototype{#1}}%

```

The following example L<sup>A</sup>T<sub>E</sub>X 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<sub>E</sub>Xt Interface

The ConT<sub>E</sub>Xt interface provides a start-stop macro pair for the typesetting of markdown input from within ConT<sub>E</sub>Xt. The rest of the interface is inherited from the plain T<sub>E</sub>X interface (see Section 2.2 on page 23).

```

850 \writestatus{loading}{ConTeXt User Module / markdown}%
851 \startmodule[markdown]
852 \unprotect

```

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

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

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

```

853 \let\startmarkdown\relax
854 \let\stopmarkdown\relax

```

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

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

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

```

\usemodule[t][markdown]
\starttext
\startmarkdown
Hello world ...
\stopmarkdown
\stoptext

```

## 3 Implementation

This part of the documentation describes the implementation of the interfaces exposed by the package (see Section 2 on page 6) and is aimed at the developers of the package, as well as the curious users.

Figure 1 on page 7 shows the high-level structure of the Markdown package: The translation from markdown to TeX *token renderers* is performed by the Lua layer. The plain TeX layer provides default definitions for the token renderers. The L<sup>A</sup>T<sub>E</sub>X

and ConT<sub>E</sub>Xt layers correct idiosyncrasies of the respective T<sub>E</sub>X formats, and provide format-specific default definitions for the token renderers.

### 3.1 Lua Implementation

The Lua implementation implements `writer` and `reader` objects that provide the conversion from markdown to plain T<sub>E</sub>X.

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<sub>E</sub>X writer were hidden behind the converter functions exposed by the Lua interface (see Section 2.1 on page 6).

```
855 local upper, gsub, format, length =
856 string.upper, string.gsub, string.format, string.len
857 local concat = table.concat
858 local P, R, S, V, C, Cg, Cb, Cmt, Cc, Ct, B, Cs, any =
859 lpeg.P, lpeg.R, lpeg.S, lpeg.V, lpeg.C, lpeg.Cg, lpeg.Cb,
860 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<sub>E</sub>X 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.

```
861 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.

```
862 function util.err(msg, exit_code)
863 io.stderr:write("markdown.lua: " .. msg .. "\n")
864 os.exit(exit_code or 1)
865 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.

```
866 function util.cache(dir, string, salt, transform, suffix)
867 local digest = md5.sumhexa(string .. (salt or ""))
868 local name = util.pathname(dir, digest .. suffix)
869 local file = io.open(name, "r")
870 if file == nil then -- If no cache entry exists, then create a new one.
871 local file = assert(io.open(name, "w"),
872 [[could not open file]] .. name .. [[for writing]])
873 local result = string
```

```

874 if transform ~= nil then
875 result = transform(result)
876 end
877 assert(file:write(result))
878 assert(file:close())
879 end
880 return name
881 end

```

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

```

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

```

The `util.expand_tabs_in_line` expands tabs in string `s`. If `tabstop` is specified, it is used as the tab stop width. Otherwise, the tab stop width of 4 characters is used. The method is a copy of the tab expansion algorithm from Ierusalimschy [8, Chapter 21].

```

887 function util.expand_tabs_in_line(s, tabstop)
888 local tab = tabstop or 4
889 local corr = 0
890 return (s:gsub("\t", function(p)
891 local sp = tab - (p - 1 + corr) % tab
892 corr = corr - 1 + sp
893 return string.rep(" ", sp)
894 end))
895 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.

```

896 function util.walk(t, f)
897 local typ = type(t)
898 if typ == "string" then
899 f(t)
900 elseif typ == "table" then
901 local i = 1
902 local n
903 n = t[i]
904 while n do
905 util.walk(n, f)
906 i = i + 1
907 n = t[i]
908 end
909 elseif typ == "function" then

```

```

910 local ok, val = pcall(t)
911 if ok then
912 util.walk(val,f)
913 end
914 else
915 f(tostring(t))
916 end
917 end

```

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

```

918 function util.flatten(ary)
919 local new = {}
920 for _,v in ipairs(ary) do
921 if type(v) == "table" then
922 for _,w in ipairs(util.flatten(v)) do
923 new[#new + 1] = w
924 end
925 else
926 new[#new + 1] = v
927 end
928 end
929 return new
930 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.

```

931 function util.rope_to_string(rope)
932 local buffer = {}
933 util.walk(rope, function(x) buffer[#buffer + 1] = x end)
934 return table.concat(buffer)
935 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.

```

936 function util.rope_last(rope)
937 if #rope == 0 then
938 return nil
939 else
940 local l = rope[#rope]
941 if type(l) == "table" then
942 return util.rope_last(l)
943 else
944 return l
945 end
946 end
947 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  $1 \leq i \leq \#ary$ .

```

948 function util.intersperse(ary, x)
949 local new = {}
950 local l = #ary
951 for i,v in ipairs(ary) do
952 local n = #new
953 new[n + 1] = v
954 if i ~= l then
955 new[n + 2] = x
956 end
957 end
958 return new
959 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  $1 \leq i \leq \#ary$ .

```

960 function util.map(ary, f)
961 local new = {}
962 for i,v in ipairs(ary) do
963 new[i] = f(v)
964 end
965 return new
966 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.

```

967 function util.escaper(char_escapes, string_escapes)

```

Build a string of escapable characters.

```

968 local char_escapes_list = ""
969 for i,_ in pairs(char_escapes) do
970 char_escapes_list = char_escapes_list .. i
971 end

```

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

```

972 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.

```
973 if string_escapes then
974 for k,v in pairs(string_escapes) do
975 escapable = P(k) / v + escapable
976 end
977 end
```

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

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

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

```
979 return function(s)
980 return lpeg.match(escape_string, s)
981 end
982 end
```

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

```
983 function util.pathname(dir, file)
984 if #dir == 0 then
985 return file
986 else
987 return dir .. "/" .. file
988 end
989 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.

```
990 local entities = {}
991
992 local character_entities = {
993 ["Tab"] = 9,
994 ["NewLine"] = 10,
995 ["excl"] = 33,
996 ["quot"] = 34,
997 ["QUOT"] = 34,
998 ["num"] = 35,
999 ["dollar"] = 36,
1000 ["percent"] = 37,
```



1001 ["amp"] = 38,  
1002 ["AMP"] = 38,  
1003 ["apos"] = 39,  
1004 ["lpar"] = 40,  
1005 ["rpar"] = 41,  
1006 ["ast"] = 42,  
1007 ["midast"] = 42,  
1008 ["plus"] = 43,  
1009 ["comma"] = 44,  
1010 ["period"] = 46,  
1011 ["sol"] = 47,  
1012 ["colon"] = 58,  
1013 ["semi"] = 59,  
1014 ["lt"] = 60,  
1015 ["LT"] = 60,  
1016 ["equals"] = 61,  
1017 ["gt"] = 62,  
1018 ["GT"] = 62,  
1019 ["quest"] = 63,  
1020 ["commat"] = 64,  
1021 ["lsqb"] = 91,  
1022 ["lbrack"] = 91,  
1023 ["bsol"] = 92,  
1024 ["rsqb"] = 93,  
1025 ["rbrack"] = 93,  
1026 ["Hat"] = 94,  
1027 ["lowbar"] = 95,  
1028 ["grave"] = 96,  
1029 ["DiacriticalGrave"] = 96,  
1030 ["lcub"] = 123,  
1031 ["lbrace"] = 123,  
1032 ["verbar"] = 124,  
1033 ["vert"] = 124,  
1034 ["VerticalLine"] = 124,  
1035 ["rcub"] = 125,  
1036 ["rbrace"] = 125,  
1037 ["nbsp"] = 160,  
1038 ["NonBreakingSpace"] = 160,  
1039 ["iexcl"] = 161,  
1040 ["cent"] = 162,  
1041 ["pound"] = 163,  
1042 ["curren"] = 164,  
1043 ["yen"] = 165,  
1044 ["brvbar"] = 166,  
1045 ["sect"] = 167,  
1046 ["Dot"] = 168,  
1047 ["die"] = 168,

1048 ["DoubleDot"] = 168,  
1049 ["uml"] = 168,  
1050 ["copy"] = 169,  
1051 ["COPY"] = 169,  
1052 ["ordf"] = 170,  
1053 ["laquo"] = 171,  
1054 ["not"] = 172,  
1055 ["shy"] = 173,  
1056 ["reg"] = 174,  
1057 ["circledR"] = 174,  
1058 ["REG"] = 174,  
1059 ["macr"] = 175,  
1060 ["OverBar"] = 175,  
1061 ["strns"] = 175,  
1062 ["deg"] = 176,  
1063 ["plusmn"] = 177,  
1064 ["pm"] = 177,  
1065 ["PlusMinus"] = 177,  
1066 ["sup2"] = 178,  
1067 ["sup3"] = 179,  
1068 ["acute"] = 180,  
1069 ["DiacriticalAcute"] = 180,  
1070 ["micro"] = 181,  
1071 ["para"] = 182,  
1072 ["middot"] = 183,  
1073 ["centerdot"] = 183,  
1074 ["CenterDot"] = 183,  
1075 ["cedil"] = 184,  
1076 ["Cedilla"] = 184,  
1077 ["sup1"] = 185,  
1078 ["ordm"] = 186,  
1079 ["raquo"] = 187,  
1080 ["frac14"] = 188,  
1081 ["frac12"] = 189,  
1082 ["half"] = 189,  
1083 ["frac34"] = 190,  
1084 ["iquest"] = 191,  
1085 ["Agrave"] = 192,  
1086 ["Aacute"] = 193,  
1087 ["Acirc"] = 194,  
1088 ["Atilde"] = 195,  
1089 ["Auml"] = 196,  
1090 ["Aring"] = 197,  
1091 ["AElig"] = 198,  
1092 ["Ccedil"] = 199,  
1093 ["Egrave"] = 200,  
1094 ["Eacute"] = 201,

1095 ["Ecirc"] = 202,  
1096 ["Euml"] = 203,  
1097 ["Igrave"] = 204,  
1098 ["Iacute"] = 205,  
1099 ["Icirc"] = 206,  
1100 ["Iuml"] = 207,  
1101 ["ETH"] = 208,  
1102 ["Ntilde"] = 209,  
1103 ["Ograve"] = 210,  
1104 ["Oacute"] = 211,  
1105 ["Ocirc"] = 212,  
1106 ["Otilde"] = 213,  
1107 ["Ouml"] = 214,  
1108 ["times"] = 215,  
1109 ["Oslash"] = 216,  
1110 ["Ugrave"] = 217,  
1111 ["Uacute"] = 218,  
1112 ["Ucirc"] = 219,  
1113 ["Uuml"] = 220,  
1114 ["Yacute"] = 221,  
1115 ["THORN"] = 222,  
1116 ["szlig"] = 223,  
1117 ["agrave"] = 224,  
1118 ["aacute"] = 225,  
1119 ["acirc"] = 226,  
1120 ["atilde"] = 227,  
1121 ["auml"] = 228,  
1122 ["aring"] = 229,  
1123 ["aelig"] = 230,  
1124 ["ccedil"] = 231,  
1125 ["egrave"] = 232,  
1126 ["eacute"] = 233,  
1127 ["ecirc"] = 234,  
1128 ["euml"] = 235,  
1129 ["igrave"] = 236,  
1130 ["iacute"] = 237,  
1131 ["icirc"] = 238,  
1132 ["iuml"] = 239,  
1133 ["eth"] = 240,  
1134 ["ntilde"] = 241,  
1135 ["ograve"] = 242,  
1136 ["oacute"] = 243,  
1137 ["ocirc"] = 244,  
1138 ["otilde"] = 245,  
1139 ["ouml"] = 246,  
1140 ["divide"] = 247,  
1141 ["div"] = 247,

1142 ["oslash"] = 248,  
1143 ["ugrave"] = 249,  
1144 ["uacute"] = 250,  
1145 ["ucirc"] = 251,  
1146 ["uuml"] = 252,  
1147 ["yacute"] = 253,  
1148 ["thorn"] = 254,  
1149 ["yuml"] = 255,  
1150 ["Amacr"] = 256,  
1151 ["amacr"] = 257,  
1152 ["Abreve"] = 258,  
1153 ["abreve"] = 259,  
1154 ["Aogon"] = 260,  
1155 ["aogon"] = 261,  
1156 ["Cacute"] = 262,  
1157 ["cacute"] = 263,  
1158 ["Ccirc"] = 264,  
1159 ["ccirc"] = 265,  
1160 ["Cdot"] = 266,  
1161 ["cdot"] = 267,  
1162 ["Ccaron"] = 268,  
1163 ["ccaron"] = 269,  
1164 ["Dcaron"] = 270,  
1165 ["dcaron"] = 271,  
1166 ["Dstrok"] = 272,  
1167 ["dstrok"] = 273,  
1168 ["Emacr"] = 274,  
1169 ["emacr"] = 275,  
1170 ["Edot"] = 278,  
1171 ["edot"] = 279,  
1172 ["Eogon"] = 280,  
1173 ["eogon"] = 281,  
1174 ["Ecaron"] = 282,  
1175 ["ecaron"] = 283,  
1176 ["Gcirc"] = 284,  
1177 ["gcirc"] = 285,  
1178 ["Gbreve"] = 286,  
1179 ["gbreve"] = 287,  
1180 ["Gdot"] = 288,  
1181 ["gdot"] = 289,  
1182 ["Gcedil"] = 290,  
1183 ["Hcirc"] = 292,  
1184 ["hcirc"] = 293,  
1185 ["Hstrok"] = 294,  
1186 ["hstrok"] = 295,  
1187 ["Itilde"] = 296,  
1188 ["itilde"] = 297,

1189 ["Imacr"] = 298,  
1190 ["imacr"] = 299,  
1191 ["Iogon"] = 302,  
1192 ["iogon"] = 303,  
1193 ["Idot"] = 304,  
1194 ["imath"] = 305,  
1195 ["inodot"] = 305,  
1196 ["IJlig"] = 306,  
1197 ["ijlig"] = 307,  
1198 ["Jcirc"] = 308,  
1199 ["jcirc"] = 309,  
1200 ["Kcedil"] = 310,  
1201 ["kcedil"] = 311,  
1202 ["kgreen"] = 312,  
1203 ["Lacute"] = 313,  
1204 ["lacute"] = 314,  
1205 ["Lcedil"] = 315,  
1206 ["lcedil"] = 316,  
1207 ["Lcaron"] = 317,  
1208 ["lcaron"] = 318,  
1209 ["Lmidot"] = 319,  
1210 ["lmidot"] = 320,  
1211 ["Lstrok"] = 321,  
1212 ["lstrok"] = 322,  
1213 ["Nacute"] = 323,  
1214 ["nacute"] = 324,  
1215 ["Ncedil"] = 325,  
1216 ["ncedil"] = 326,  
1217 ["Ncaron"] = 327,  
1218 ["ncaron"] = 328,  
1219 ["napos"] = 329,  
1220 ["ENG"] = 330,  
1221 ["eng"] = 331,  
1222 ["Omacr"] = 332,  
1223 ["omacr"] = 333,  
1224 ["Odblac"] = 336,  
1225 ["odblac"] = 337,  
1226 ["OElig"] = 338,  
1227 ["oelig"] = 339,  
1228 ["Racute"] = 340,  
1229 ["racute"] = 341,  
1230 ["Rcedil"] = 342,  
1231 ["rcedil"] = 343,  
1232 ["Rcaron"] = 344,  
1233 ["rcaron"] = 345,  
1234 ["Sacute"] = 346,  
1235 ["sacute"] = 347,

1236 ["Scirc"] = 348,  
1237 ["scirc"] = 349,  
1238 ["Scedil"] = 350,  
1239 ["scedil"] = 351,  
1240 ["Scaron"] = 352,  
1241 ["scaron"] = 353,  
1242 ["Tcedil"] = 354,  
1243 ["tcedil"] = 355,  
1244 ["Tcaron"] = 356,  
1245 ["tcaron"] = 357,  
1246 ["Tstrok"] = 358,  
1247 ["tstrok"] = 359,  
1248 ["Utilde"] = 360,  
1249 ["utilde"] = 361,  
1250 ["Umacr"] = 362,  
1251 ["umacr"] = 363,  
1252 ["Ubreve"] = 364,  
1253 ["ubreve"] = 365,  
1254 ["Uring"] = 366,  
1255 ["uring"] = 367,  
1256 ["Udblac"] = 368,  
1257 ["udblac"] = 369,  
1258 ["Uogon"] = 370,  
1259 ["uogon"] = 371,  
1260 ["Wcirc"] = 372,  
1261 ["wcirc"] = 373,  
1262 ["Ycirc"] = 374,  
1263 ["ycirc"] = 375,  
1264 ["Yuml"] = 376,  
1265 ["Zacute"] = 377,  
1266 ["zacute"] = 378,  
1267 ["Zdot"] = 379,  
1268 ["zdot"] = 380,  
1269 ["Zcaron"] = 381,  
1270 ["zcaron"] = 382,  
1271 ["fnof"] = 402,  
1272 ["imped"] = 437,  
1273 ["gacute"] = 501,  
1274 ["jmath"] = 567,  
1275 ["circ"] = 710,  
1276 ["caron"] = 711,  
1277 ["Hacek"] = 711,  
1278 ["breve"] = 728,  
1279 ["Breve"] = 728,  
1280 ["dot"] = 729,  
1281 ["DiacriticalDot"] = 729,  
1282 ["ring"] = 730,

1283 ["ogon"] = 731,  
1284 ["tilde"] = 732,  
1285 ["DiacriticalTilde"] = 732,  
1286 ["dblac"] = 733,  
1287 ["DiacriticalDoubleAcute"] = 733,  
1288 ["DownBreve"] = 785,  
1289 ["UnderBar"] = 818,  
1290 ["Alpha"] = 913,  
1291 ["Beta"] = 914,  
1292 ["Gamma"] = 915,  
1293 ["Delta"] = 916,  
1294 ["Epsilon"] = 917,  
1295 ["Zeta"] = 918,  
1296 ["Eta"] = 919,  
1297 ["Theta"] = 920,  
1298 ["Iota"] = 921,  
1299 ["Kappa"] = 922,  
1300 ["Lambda"] = 923,  
1301 ["Mu"] = 924,  
1302 ["Nu"] = 925,  
1303 ["Xi"] = 926,  
1304 ["Omicron"] = 927,  
1305 ["Pi"] = 928,  
1306 ["Rho"] = 929,  
1307 ["Sigma"] = 931,  
1308 ["Tau"] = 932,  
1309 ["Upsilon"] = 933,  
1310 ["Phi"] = 934,  
1311 ["Chi"] = 935,  
1312 ["Psi"] = 936,  
1313 ["Omega"] = 937,  
1314 ["alpha"] = 945,  
1315 ["beta"] = 946,  
1316 ["gamma"] = 947,  
1317 ["delta"] = 948,  
1318 ["epsiv"] = 949,  
1319 ["varepsilon"] = 949,  
1320 ["epsilon"] = 949,  
1321 ["zeta"] = 950,  
1322 ["eta"] = 951,  
1323 ["theta"] = 952,  
1324 ["iota"] = 953,  
1325 ["kappa"] = 954,  
1326 ["lambda"] = 955,  
1327 ["mu"] = 956,  
1328 ["nu"] = 957,  
1329 ["xi"] = 958,

1330 ["omicron"] = 959,  
 1331 ["pi"] = 960,  
 1332 ["rho"] = 961,  
 1333 ["sigmav"] = 962,  
 1334 ["varsigma"] = 962,  
 1335 ["sigmaf"] = 962,  
 1336 ["sigma"] = 963,  
 1337 ["tau"] = 964,  
 1338 ["upsilon"] = 965,  
 1339 ["upsilon"] = 965,  
 1340 ["phi"] = 966,  
 1341 ["phiv"] = 966,  
 1342 ["varphi"] = 966,  
 1343 ["chi"] = 967,  
 1344 ["psi"] = 968,  
 1345 ["omega"] = 969,  
 1346 ["thetav"] = 977,  
 1347 ["vartheta"] = 977,  
 1348 ["thetasym"] = 977,  
 1349 ["Upsi"] = 978,  
 1350 ["upsih"] = 978,  
 1351 ["straightphi"] = 981,  
 1352 ["piv"] = 982,  
 1353 ["varpi"] = 982,  
 1354 ["Gammad"] = 988,  
 1355 ["gammad"] = 989,  
 1356 ["digamma"] = 989,  
 1357 ["kappav"] = 1008,  
 1358 ["varkappa"] = 1008,  
 1359 ["rhov"] = 1009,  
 1360 ["varrho"] = 1009,  
 1361 ["epsi"] = 1013,  
 1362 ["straightepsilon"] = 1013,  
 1363 ["bepsi"] = 1014,  
 1364 ["backepsilon"] = 1014,  
 1365 ["IOcy"] = 1025,  
 1366 ["DJcy"] = 1026,  
 1367 ["GJcy"] = 1027,  
 1368 ["Jukcy"] = 1028,  
 1369 ["DScy"] = 1029,  
 1370 ["Iukcy"] = 1030,  
 1371 ["YIcy"] = 1031,  
 1372 ["Jsercy"] = 1032,  
 1373 ["LJcy"] = 1033,  
 1374 ["NJcy"] = 1034,  
 1375 ["TSHcy"] = 1035,  
 1376 ["KJcy"] = 1036,



1377 ["Ubrcy"] = 1038,  
1378 ["DZcy"] = 1039,  
1379 ["Acy"] = 1040,  
1380 ["Bcy"] = 1041,  
1381 ["Vcy"] = 1042,  
1382 ["Gcy"] = 1043,  
1383 ["Dcy"] = 1044,  
1384 ["IEcy"] = 1045,  
1385 ["ZHcy"] = 1046,  
1386 ["Zcy"] = 1047,  
1387 ["Icy"] = 1048,  
1388 ["Jcy"] = 1049,  
1389 ["Kcy"] = 1050,  
1390 ["Lcy"] = 1051,  
1391 ["Mcy"] = 1052,  
1392 ["Ncy"] = 1053,  
1393 ["Ocy"] = 1054,  
1394 ["Pcy"] = 1055,  
1395 ["Rcy"] = 1056,  
1396 ["Scy"] = 1057,  
1397 ["Tcy"] = 1058,  
1398 ["Ucy"] = 1059,  
1399 ["Fcy"] = 1060,  
1400 ["KHcy"] = 1061,  
1401 ["TScy"] = 1062,  
1402 ["CHcy"] = 1063,  
1403 ["SHcy"] = 1064,  
1404 ["SHCHcy"] = 1065,  
1405 ["HARDcy"] = 1066,  
1406 ["Ycy"] = 1067,  
1407 ["SOFTcy"] = 1068,  
1408 ["Ecy"] = 1069,  
1409 ["YUcy"] = 1070,  
1410 ["YAcy"] = 1071,  
1411 ["acy"] = 1072,  
1412 ["bcy"] = 1073,  
1413 ["vcy"] = 1074,  
1414 ["gcy"] = 1075,  
1415 ["dcy"] = 1076,  
1416 ["iecy"] = 1077,  
1417 ["zhcy"] = 1078,  
1418 ["zcy"] = 1079,  
1419 ["icy"] = 1080,  
1420 ["jcy"] = 1081,  
1421 ["kcy"] = 1082,  
1422 ["lcy"] = 1083,  
1423 ["mcy"] = 1084,

1424 ["ncy"] = 1085,  
1425 ["ocy"] = 1086,  
1426 ["pcy"] = 1087,  
1427 ["rcy"] = 1088,  
1428 ["scy"] = 1089,  
1429 ["tcy"] = 1090,  
1430 ["ucy"] = 1091,  
1431 ["fcy"] = 1092,  
1432 ["khcy"] = 1093,  
1433 ["tscy"] = 1094,  
1434 ["chcy"] = 1095,  
1435 ["shcy"] = 1096,  
1436 ["shchcy"] = 1097,  
1437 ["hardcy"] = 1098,  
1438 ["ycy"] = 1099,  
1439 ["softcy"] = 1100,  
1440 ["ecy"] = 1101,  
1441 ["yucy"] = 1102,  
1442 ["yacy"] = 1103,  
1443 ["iocy"] = 1105,  
1444 ["djcy"] = 1106,  
1445 ["gjcy"] = 1107,  
1446 ["jukcy"] = 1108,  
1447 ["dscy"] = 1109,  
1448 ["iukcy"] = 1110,  
1449 ["yicy"] = 1111,  
1450 ["jsercy"] = 1112,  
1451 ["ljcy"] = 1113,  
1452 ["njcy"] = 1114,  
1453 ["tshcy"] = 1115,  
1454 ["kjcy"] = 1116,  
1455 ["ubrscy"] = 1118,  
1456 ["dzcy"] = 1119,  
1457 ["ensp"] = 8194,  
1458 ["emsp"] = 8195,  
1459 ["emsp13"] = 8196,  
1460 ["emsp14"] = 8197,  
1461 ["numsp"] = 8199,  
1462 ["puncsp"] = 8200,  
1463 ["thinsp"] = 8201,  
1464 ["ThinSpace"] = 8201,  
1465 ["hairsp"] = 8202,  
1466 ["VeryThinSpace"] = 8202,  
1467 ["ZeroWidthSpace"] = 8203,  
1468 ["NegativeVeryThinSpace"] = 8203,  
1469 ["NegativeThinSpace"] = 8203,  
1470 ["NegativeMediumSpace"] = 8203,

1471 ["NegativeThickSpace"] = 8203,  
1472 ["zwnj"] = 8204,  
1473 ["zwj"] = 8205,  
1474 ["lrm"] = 8206,  
1475 ["rlm"] = 8207,  
1476 ["hyphen"] = 8208,  
1477 ["dash"] = 8208,  
1478 ["ndash"] = 8211,  
1479 ["mdash"] = 8212,  
1480 ["horbar"] = 8213,  
1481 ["Verbar"] = 8214,  
1482 ["Vert"] = 8214,  
1483 ["lsquo"] = 8216,  
1484 ["OpenCurlyQuote"] = 8216,  
1485 ["rsquo"] = 8217,  
1486 ["rsquor"] = 8217,  
1487 ["CloseCurlyQuote"] = 8217,  
1488 ["lsquor"] = 8218,  
1489 ["sbquo"] = 8218,  
1490 ["ldquo"] = 8220,  
1491 ["OpenCurlyDoubleQuote"] = 8220,  
1492 ["rdquo"] = 8221,  
1493 ["rdquor"] = 8221,  
1494 ["CloseCurlyDoubleQuote"] = 8221,  
1495 ["ldquor"] = 8222,  
1496 ["bdquo"] = 8222,  
1497 ["dagger"] = 8224,  
1498 ["Dagger"] = 8225,  
1499 ["ddagger"] = 8225,  
1500 ["bull"] = 8226,  
1501 ["bullet"] = 8226,  
1502 ["nldr"] = 8229,  
1503 ["hellip"] = 8230,  
1504 ["mldr"] = 8230,  
1505 ["permil"] = 8240,  
1506 ["pertenk"] = 8241,  
1507 ["prime"] = 8242,  
1508 ["Prime"] = 8243,  
1509 ["tprime"] = 8244,  
1510 ["bprime"] = 8245,  
1511 ["backprime"] = 8245,  
1512 ["lsaquo"] = 8249,  
1513 ["rsaquo"] = 8250,  
1514 ["oline"] = 8254,  
1515 ["caret"] = 8257,  
1516 ["hybull"] = 8259,  
1517 ["frasl"] = 8260,

1518 ["bsemi"] = 8271,  
1519 ["qprime"] = 8279,  
1520 ["MediumSpace"] = 8287,  
1521 ["NoBreak"] = 8288,  
1522 ["ApplyFunction"] = 8289,  
1523 ["af"] = 8289,  
1524 ["InvisibleTimes"] = 8290,  
1525 ["it"] = 8290,  
1526 ["InvisibleComma"] = 8291,  
1527 ["ic"] = 8291,  
1528 ["euro"] = 8364,  
1529 ["tdot"] = 8411,  
1530 ["TripleDot"] = 8411,  
1531 ["DotDot"] = 8412,  
1532 ["Copf"] = 8450,  
1533 ["complexes"] = 8450,  
1534 ["incare"] = 8453,  
1535 ["gscr"] = 8458,  
1536 ["hamilt"] = 8459,  
1537 ["HilbertSpace"] = 8459,  
1538 ["Hscr"] = 8459,  
1539 ["Hfr"] = 8460,  
1540 ["Poincareplane"] = 8460,  
1541 ["quaternions"] = 8461,  
1542 ["Hopf"] = 8461,  
1543 ["planckh"] = 8462,  
1544 ["planck"] = 8463,  
1545 ["hbar"] = 8463,  
1546 ["plankv"] = 8463,  
1547 ["hslash"] = 8463,  
1548 ["Iscr"] = 8464,  
1549 ["imagline"] = 8464,  
1550 ["image"] = 8465,  
1551 ["Im"] = 8465,  
1552 ["imagpart"] = 8465,  
1553 ["Ifr"] = 8465,  
1554 ["Lscr"] = 8466,  
1555 ["lagran"] = 8466,  
1556 ["Laplacetrif"] = 8466,  
1557 ["ell"] = 8467,  
1558 ["Nopf"] = 8469,  
1559 ["naturals"] = 8469,  
1560 ["numero"] = 8470,  
1561 ["copysr"] = 8471,  
1562 ["weierp"] = 8472,  
1563 ["wp"] = 8472,  
1564 ["Popf"] = 8473,

1565 ["primes"] = 8473,  
1566 ["rationals"] = 8474,  
1567 ["Qopf"] = 8474,  
1568 ["Rscr"] = 8475,  
1569 ["realine"] = 8475,  
1570 ["real"] = 8476,  
1571 ["Re"] = 8476,  
1572 ["realpart"] = 8476,  
1573 ["Rfr"] = 8476,  
1574 ["reals"] = 8477,  
1575 ["Ropf"] = 8477,  
1576 ["rx"] = 8478,  
1577 ["trade"] = 8482,  
1578 ["TRADE"] = 8482,  
1579 ["integers"] = 8484,  
1580 ["Zopf"] = 8484,  
1581 ["ohm"] = 8486,  
1582 ["mho"] = 8487,  
1583 ["Zfr"] = 8488,  
1584 ["zeetrf"] = 8488,  
1585 ["iiota"] = 8489,  
1586 ["angst"] = 8491,  
1587 ["bernou"] = 8492,  
1588 ["Bernoullis"] = 8492,  
1589 ["Bscr"] = 8492,  
1590 ["Cfr"] = 8493,  
1591 ["Cayleys"] = 8493,  
1592 ["escr"] = 8495,  
1593 ["Escr"] = 8496,  
1594 ["expectation"] = 8496,  
1595 ["Fscr"] = 8497,  
1596 ["Fouriertrf"] = 8497,  
1597 ["phmmat"] = 8499,  
1598 ["Mellintrf"] = 8499,  
1599 ["Mscr"] = 8499,  
1600 ["order"] = 8500,  
1601 ["orderof"] = 8500,  
1602 ["oscr"] = 8500,  
1603 ["alefsym"] = 8501,  
1604 ["aleph"] = 8501,  
1605 ["beth"] = 8502,  
1606 ["gimel"] = 8503,  
1607 ["daleth"] = 8504,  
1608 ["CapitalDifferentialD"] = 8517,  
1609 ["DD"] = 8517,  
1610 ["DifferentialD"] = 8518,  
1611 ["dd"] = 8518,

1612 ["ExponentialE"] = 8519,  
1613 ["exponentiale"] = 8519,  
1614 ["ee"] = 8519,  
1615 ["ImaginaryI"] = 8520,  
1616 ["ii"] = 8520,  
1617 ["frac13"] = 8531,  
1618 ["frac23"] = 8532,  
1619 ["frac15"] = 8533,  
1620 ["frac25"] = 8534,  
1621 ["frac35"] = 8535,  
1622 ["frac45"] = 8536,  
1623 ["frac16"] = 8537,  
1624 ["frac56"] = 8538,  
1625 ["frac18"] = 8539,  
1626 ["frac38"] = 8540,  
1627 ["frac58"] = 8541,  
1628 ["frac78"] = 8542,  
1629 ["larr"] = 8592,  
1630 ["leftarrow"] = 8592,  
1631 ["LeftArrow"] = 8592,  
1632 ["slarr"] = 8592,  
1633 ["ShortLeftArrow"] = 8592,  
1634 ["uarr"] = 8593,  
1635 ["uparrow"] = 8593,  
1636 ["UpArrow"] = 8593,  
1637 ["ShortUpArrow"] = 8593,  
1638 ["rarr"] = 8594,  
1639 ["rightarrow"] = 8594,  
1640 ["RightArrow"] = 8594,  
1641 ["srarr"] = 8594,  
1642 ["ShortRightArrow"] = 8594,  
1643 ["darr"] = 8595,  
1644 ["downarrow"] = 8595,  
1645 ["DownArrow"] = 8595,  
1646 ["ShortDownArrow"] = 8595,  
1647 ["harr"] = 8596,  
1648 ["leftrightarrow"] = 8596,  
1649 ["LeftRightArrow"] = 8596,  
1650 ["varr"] = 8597,  
1651 ["updownarrow"] = 8597,  
1652 ["UpDownArrow"] = 8597,  
1653 ["nwarr"] = 8598,  
1654 ["UpperLeftArrow"] = 8598,  
1655 ["nwarrow"] = 8598,  
1656 ["nearr"] = 8599,  
1657 ["UpperRightArrow"] = 8599,  
1658 ["nearrow"] = 8599,

```

1659 ["searr"] = 8600,
1660 ["searrow"] = 8600,
1661 ["LowerRightArrow"] = 8600,
1662 ["swarr"] = 8601,
1663 ["swarrow"] = 8601,
1664 ["LowerLeftArrow"] = 8601,
1665 ["nlarr"] = 8602,
1666 ["nleftarrow"] = 8602,
1667 ["nrarr"] = 8603,
1668 ["nrightarrow"] = 8603,
1669 ["rarrw"] = 8605,
1670 ["rightsquigarrow"] = 8605,
1671 ["Larr"] = 8606,
1672 ["twoheadleftarrow"] = 8606,
1673 ["Uarr"] = 8607,
1674 ["Rarr"] = 8608,
1675 ["twoheadrightarrow"] = 8608,
1676 ["Darr"] = 8609,
1677 ["larrtl"] = 8610,
1678 ["leftarrowtail"] = 8610,
1679 ["rarrtl"] = 8611,
1680 ["rightarrowtail"] = 8611,
1681 ["LeftTeeArrow"] = 8612,
1682 ["mapstoleft"] = 8612,
1683 ["UpTeeArrow"] = 8613,
1684 ["mapstoup"] = 8613,
1685 ["map"] = 8614,
1686 ["RightTeeArrow"] = 8614,
1687 ["mapsto"] = 8614,
1688 ["DownTeeArrow"] = 8615,
1689 ["mapstodown"] = 8615,
1690 ["larrhk"] = 8617,
1691 ["hookleftarrow"] = 8617,
1692 ["rarrhk"] = 8618,
1693 ["hookrightarrow"] = 8618,
1694 ["larrlp"] = 8619,
1695 ["looparrowleft"] = 8619,
1696 ["rarrlp"] = 8620,
1697 ["looparrowright"] = 8620,
1698 ["harrw"] = 8621,
1699 ["leftrightsquigarrow"] = 8621,
1700 ["nharr"] = 8622,
1701 ["nleftrightarrow"] = 8622,
1702 ["lsh"] = 8624,
1703 ["Lsh"] = 8624,
1704 ["rsh"] = 8625,
1705 ["Rsh"] = 8625,

```

```

1706 ["ldsh"] = 8626,
1707 ["rdsh"] = 8627,
1708 ["crarr"] = 8629,
1709 ["cularr"] = 8630,
1710 ["curvearrowleft"] = 8630,
1711 ["curarr"] = 8631,
1712 ["curvearrowright"] = 8631,
1713 ["olarr"] = 8634,
1714 ["circlearrowleft"] = 8634,
1715 ["orarr"] = 8635,
1716 ["circlearrowright"] = 8635,
1717 ["lharu"] = 8636,
1718 ["LeftVector"] = 8636,
1719 ["leftharpoonup"] = 8636,
1720 ["lhard"] = 8637,
1721 ["leftharpoondown"] = 8637,
1722 ["DownLeftVector"] = 8637,
1723 ["uharr"] = 8638,
1724 ["upharpoonright"] = 8638,
1725 ["RightUpVector"] = 8638,
1726 ["uharl"] = 8639,
1727 ["upharpoonleft"] = 8639,
1728 ["LeftUpVector"] = 8639,
1729 ["rharu"] = 8640,
1730 ["RightVector"] = 8640,
1731 ["rightharpoonup"] = 8640,
1732 ["rhard"] = 8641,
1733 ["rightharpoondown"] = 8641,
1734 ["DownRightVector"] = 8641,
1735 ["dharr"] = 8642,
1736 ["RightDownVector"] = 8642,
1737 ["downharpoonright"] = 8642,
1738 ["dharl"] = 8643,
1739 ["LeftDownVector"] = 8643,
1740 ["downharpoonleft"] = 8643,
1741 ["rlarr"] = 8644,
1742 ["rightleftarrows"] = 8644,
1743 ["RightArrowLeftArrow"] = 8644,
1744 ["udarr"] = 8645,
1745 ["UpArrowDownArrow"] = 8645,
1746 ["lrarr"] = 8646,
1747 ["leftrightarrows"] = 8646,
1748 ["LeftArrowRightArrow"] = 8646,
1749 ["llarr"] = 8647,
1750 ["leftleftarrows"] = 8647,
1751 ["uuarr"] = 8648,
1752 ["upuparrows"] = 8648,

```



1753 ["rrarr"] = 8649,  
1754 ["rightrightarrows"] = 8649,  
1755 ["ddarr"] = 8650,  
1756 ["downdownarrows"] = 8650,  
1757 ["lrhar"] = 8651,  
1758 ["ReverseEquilibrium"] = 8651,  
1759 ["leftrightharpoons"] = 8651,  
1760 ["rlhar"] = 8652,  
1761 ["rightleftharpoons"] = 8652,  
1762 ["Equilibrium"] = 8652,  
1763 ["nLArr"] = 8653,  
1764 ["nLeftarrow"] = 8653,  
1765 ["nhArr"] = 8654,  
1766 ["nLeftrightarrow"] = 8654,  
1767 ["nrArr"] = 8655,  
1768 ["nrightarrow"] = 8655,  
1769 ["lArr"] = 8656,  
1770 ["Leftarrow"] = 8656,  
1771 ["DoubleLeftArrow"] = 8656,  
1772 ["uArr"] = 8657,  
1773 ["Uparrow"] = 8657,  
1774 ["DoubleUpArrow"] = 8657,  
1775 ["rArr"] = 8658,  
1776 ["Rightarrow"] = 8658,  
1777 ["Implies"] = 8658,  
1778 ["DoubleRightArrow"] = 8658,  
1779 ["dArr"] = 8659,  
1780 ["Downarrow"] = 8659,  
1781 ["DoubleDownArrow"] = 8659,  
1782 ["hArr"] = 8660,  
1783 ["Leftrightarrow"] = 8660,  
1784 ["DoubleLeftRightArrow"] = 8660,  
1785 ["iff"] = 8660,  
1786 ["vArr"] = 8661,  
1787 ["Updownarrow"] = 8661,  
1788 ["DoubleUpDownArrow"] = 8661,  
1789 ["nwArr"] = 8662,  
1790 ["neArr"] = 8663,  
1791 ["seArr"] = 8664,  
1792 ["swArr"] = 8665,  
1793 ["lAarr"] = 8666,  
1794 ["Lleftarrow"] = 8666,  
1795 ["rAarr"] = 8667,  
1796 ["Rrightarrow"] = 8667,  
1797 ["zigrarr"] = 8669,  
1798 ["larrb"] = 8676,  
1799 ["LeftArrowBar"] = 8676,

1800 ["rarrb"] = 8677,  
1801 ["RightArrowBar"] = 8677,  
1802 ["duarr"] = 8693,  
1803 ["DownArrowUpArrow"] = 8693,  
1804 ["loarr"] = 8701,  
1805 ["roarr"] = 8702,  
1806 ["hoarr"] = 8703,  
1807 ["forall"] = 8704,  
1808 ["ForAll"] = 8704,  
1809 ["comp"] = 8705,  
1810 ["complement"] = 8705,  
1811 ["part"] = 8706,  
1812 ["PartialD"] = 8706,  
1813 ["exist"] = 8707,  
1814 ["Exists"] = 8707,  
1815 ["nexist"] = 8708,  
1816 ["NotExists"] = 8708,  
1817 ["nexists"] = 8708,  
1818 ["empty"] = 8709,  
1819 ["emptyset"] = 8709,  
1820 ["emptyv"] = 8709,  
1821 ["varnothing"] = 8709,  
1822 ["nabla"] = 8711,  
1823 ["Del"] = 8711,  
1824 ["isin"] = 8712,  
1825 ["isinv"] = 8712,  
1826 ["Element"] = 8712,  
1827 ["in"] = 8712,  
1828 ["notin"] = 8713,  
1829 ["NotElement"] = 8713,  
1830 ["notinva"] = 8713,  
1831 ["niv"] = 8715,  
1832 ["ReverseElement"] = 8715,  
1833 ["ni"] = 8715,  
1834 ["SuchThat"] = 8715,  
1835 ["notni"] = 8716,  
1836 ["notniva"] = 8716,  
1837 ["NotReverseElement"] = 8716,  
1838 ["prod"] = 8719,  
1839 ["Product"] = 8719,  
1840 ["coprod"] = 8720,  
1841 ["Coproduct"] = 8720,  
1842 ["sum"] = 8721,  
1843 ["Sum"] = 8721,  
1844 ["minus"] = 8722,  
1845 ["mnplus"] = 8723,  
1846 ["mp"] = 8723,

1847 ["MinusPlus"] = 8723,  
1848 ["plusdo"] = 8724,  
1849 ["dotplus"] = 8724,  
1850 ["setmn"] = 8726,  
1851 ["setminus"] = 8726,  
1852 ["Backslash"] = 8726,  
1853 ["ssetmn"] = 8726,  
1854 ["smallsetminus"] = 8726,  
1855 ["lowast"] = 8727,  
1856 ["compfn"] = 8728,  
1857 ["SmallCircle"] = 8728,  
1858 ["radic"] = 8730,  
1859 ["Sqrt"] = 8730,  
1860 ["prop"] = 8733,  
1861 ["propto"] = 8733,  
1862 ["Proportional"] = 8733,  
1863 ["vprop"] = 8733,  
1864 ["varpropto"] = 8733,  
1865 ["infin"] = 8734,  
1866 ["angrt"] = 8735,  
1867 ["ang"] = 8736,  
1868 ["angle"] = 8736,  
1869 ["angmsd"] = 8737,  
1870 ["measuredangle"] = 8737,  
1871 ["angsph"] = 8738,  
1872 ["mid"] = 8739,  
1873 ["VerticalBar"] = 8739,  
1874 ["smid"] = 8739,  
1875 ["shortmid"] = 8739,  
1876 ["nmid"] = 8740,  
1877 ["NotVerticalBar"] = 8740,  
1878 ["nsmid"] = 8740,  
1879 ["nshortmid"] = 8740,  
1880 ["par"] = 8741,  
1881 ["parallel"] = 8741,  
1882 ["DoubleVerticalBar"] = 8741,  
1883 ["spar"] = 8741,  
1884 ["shortparallel"] = 8741,  
1885 ["npar"] = 8742,  
1886 ["nparallel"] = 8742,  
1887 ["NotDoubleVerticalBar"] = 8742,  
1888 ["nspar"] = 8742,  
1889 ["nshortparallel"] = 8742,  
1890 ["and"] = 8743,  
1891 ["wedge"] = 8743,  
1892 ["or"] = 8744,  
1893 ["vee"] = 8744,

1894 ["cap"] = 8745,  
1895 ["cup"] = 8746,  
1896 ["int"] = 8747,  
1897 ["Integral"] = 8747,  
1898 ["Int"] = 8748,  
1899 ["tint"] = 8749,  
1900 ["iiint"] = 8749,  
1901 ["conint"] = 8750,  
1902 ["oint"] = 8750,  
1903 ["ContourIntegral"] = 8750,  
1904 ["Conint"] = 8751,  
1905 ["DoubleContourIntegral"] = 8751,  
1906 ["Cconint"] = 8752,  
1907 ["cwint"] = 8753,  
1908 ["cwconint"] = 8754,  
1909 ["ClockwiseContourIntegral"] = 8754,  
1910 ["awconint"] = 8755,  
1911 ["CounterClockwiseContourIntegral"] = 8755,  
1912 ["there4"] = 8756,  
1913 ["therefore"] = 8756,  
1914 ["Therefore"] = 8756,  
1915 ["becaus"] = 8757,  
1916 ["because"] = 8757,  
1917 ["Because"] = 8757,  
1918 ["ratio"] = 8758,  
1919 ["Colon"] = 8759,  
1920 ["Proportion"] = 8759,  
1921 ["minusd"] = 8760,  
1922 ["dotminus"] = 8760,  
1923 ["mDDot"] = 8762,  
1924 ["homtht"] = 8763,  
1925 ["sim"] = 8764,  
1926 ["Tilde"] = 8764,  
1927 ["thksim"] = 8764,  
1928 ["thicksim"] = 8764,  
1929 ["bsim"] = 8765,  
1930 ["backsim"] = 8765,  
1931 ["ac"] = 8766,  
1932 ["mstpos"] = 8766,  
1933 ["acd"] = 8767,  
1934 ["wreath"] = 8768,  
1935 ["VerticalTilde"] = 8768,  
1936 ["wr"] = 8768,  
1937 ["nsim"] = 8769,  
1938 ["NotTilde"] = 8769,  
1939 ["esim"] = 8770,  
1940 ["EqualTilde"] = 8770,

1941 ["eqsim"] = 8770,  
 1942 ["sime"] = 8771,  
 1943 ["TildeEqual"] = 8771,  
 1944 ["simeq"] = 8771,  
 1945 ["nsime"] = 8772,  
 1946 ["nsimeq"] = 8772,  
 1947 ["NotTildeEqual"] = 8772,  
 1948 ["cong"] = 8773,  
 1949 ["TildeFullEqual"] = 8773,  
 1950 ["simne"] = 8774,  
 1951 ["ncong"] = 8775,  
 1952 ["NotTildeFullEqual"] = 8775,  
 1953 ["asymp"] = 8776,  
 1954 ["ap"] = 8776,  
 1955 ["TildeTilde"] = 8776,  
 1956 ["approx"] = 8776,  
 1957 ["thkap"] = 8776,  
 1958 ["thickapprox"] = 8776,  
 1959 ["nap"] = 8777,  
 1960 ["NotTildeTilde"] = 8777,  
 1961 ["naprox"] = 8777,  
 1962 ["ape"] = 8778,  
 1963 ["approxpeq"] = 8778,  
 1964 ["apid"] = 8779,  
 1965 ["bcong"] = 8780,  
 1966 ["backcong"] = 8780,  
 1967 ["asympeq"] = 8781,  
 1968 ["CupCap"] = 8781,  
 1969 ["bump"] = 8782,  
 1970 ["HumpDownHump"] = 8782,  
 1971 ["Bumpeq"] = 8782,  
 1972 ["bumpe"] = 8783,  
 1973 ["HumpEqual"] = 8783,  
 1974 ["bumpeq"] = 8783,  
 1975 ["esdot"] = 8784,  
 1976 ["DotEqual"] = 8784,  
 1977 ["doteq"] = 8784,  
 1978 ["eDot"] = 8785,  
 1979 ["doteqdot"] = 8785,  
 1980 ["efDot"] = 8786,  
 1981 ["fallingdotseq"] = 8786,  
 1982 ["erDot"] = 8787,  
 1983 ["risingdotseq"] = 8787,  
 1984 ["colone"] = 8788,  
 1985 ["coloneq"] = 8788,  
 1986 ["Assign"] = 8788,  
 1987 ["ecolon"] = 8789,

1988 ["eqcolon"] = 8789,  
 1989 ["ecir"] = 8790,  
 1990 ["eqcirc"] = 8790,  
 1991 ["cire"] = 8791,  
 1992 ["circeq"] = 8791,  
 1993 ["wedgeq"] = 8793,  
 1994 ["veeeq"] = 8794,  
 1995 ["trie"] = 8796,  
 1996 ["triangleq"] = 8796,  
 1997 ["equest"] = 8799,  
 1998 ["questeq"] = 8799,  
 1999 ["ne"] = 8800,  
 2000 ["NotEqual"] = 8800,  
 2001 ["equiv"] = 8801,  
 2002 ["Congruent"] = 8801,  
 2003 ["nequiv"] = 8802,  
 2004 ["NotCongruent"] = 8802,  
 2005 ["le"] = 8804,  
 2006 ["leq"] = 8804,  
 2007 ["ge"] = 8805,  
 2008 ["GreaterEqual"] = 8805,  
 2009 ["geq"] = 8805,  
 2010 ["lE"] = 8806,  
 2011 ["LessFullEqual"] = 8806,  
 2012 ["leqq"] = 8806,  
 2013 ["gE"] = 8807,  
 2014 ["GreaterFullEqual"] = 8807,  
 2015 ["geqq"] = 8807,  
 2016 ["lnE"] = 8808,  
 2017 ["lneqq"] = 8808,  
 2018 ["gnE"] = 8809,  
 2019 ["gneqq"] = 8809,  
 2020 ["Lt"] = 8810,  
 2021 ["NestedLessLess"] = 8810,  
 2022 ["ll"] = 8810,  
 2023 ["Gt"] = 8811,  
 2024 ["NestedGreaterGreater"] = 8811,  
 2025 ["gg"] = 8811,  
 2026 ["twixt"] = 8812,  
 2027 ["between"] = 8812,  
 2028 ["NotCupCap"] = 8813,  
 2029 ["nlt"] = 8814,  
 2030 ["NotLess"] = 8814,  
 2031 ["nless"] = 8814,  
 2032 ["ngt"] = 8815,  
 2033 ["NotGreater"] = 8815,  
 2034 ["ngtr"] = 8815,

2035 ["nle"] = 8816,  
 2036 ["NotLessEqual"] = 8816,  
 2037 ["nleq"] = 8816,  
 2038 ["nge"] = 8817,  
 2039 ["NotGreaterEqual"] = 8817,  
 2040 ["ngeq"] = 8817,  
 2041 ["lsim"] = 8818,  
 2042 ["LessTilde"] = 8818,  
 2043 ["lesssim"] = 8818,  
 2044 ["gsim"] = 8819,  
 2045 ["gtrsim"] = 8819,  
 2046 ["GreaterTilde"] = 8819,  
 2047 ["nlsim"] = 8820,  
 2048 ["NotLessTilde"] = 8820,  
 2049 ["ngsim"] = 8821,  
 2050 ["NotGreaterTilde"] = 8821,  
 2051 ["lg"] = 8822,  
 2052 ["lessgtr"] = 8822,  
 2053 ["LessGreater"] = 8822,  
 2054 ["gl"] = 8823,  
 2055 ["gtrless"] = 8823,  
 2056 ["GreaterLess"] = 8823,  
 2057 ["ntlg"] = 8824,  
 2058 ["NotLessGreater"] = 8824,  
 2059 ["ntgl"] = 8825,  
 2060 ["NotGreaterLess"] = 8825,  
 2061 ["pr"] = 8826,  
 2062 ["Precedes"] = 8826,  
 2063 ["prec"] = 8826,  
 2064 ["sc"] = 8827,  
 2065 ["Succeeds"] = 8827,  
 2066 ["succ"] = 8827,  
 2067 ["prcue"] = 8828,  
 2068 ["PrecedesSlantEqual"] = 8828,  
 2069 ["preccurlyeq"] = 8828,  
 2070 ["sccue"] = 8829,  
 2071 ["SucceedsSlantEqual"] = 8829,  
 2072 ["succcurlyeq"] = 8829,  
 2073 ["prsim"] = 8830,  
 2074 ["precsim"] = 8830,  
 2075 ["PrecedesTilde"] = 8830,  
 2076 ["scsim"] = 8831,  
 2077 ["succsim"] = 8831,  
 2078 ["SucceedsTilde"] = 8831,  
 2079 ["npr"] = 8832,  
 2080 ["nprec"] = 8832,  
 2081 ["NotPrecedes"] = 8832,

2082 ["nsc"] = 8833,  
 2083 ["nsucc"] = 8833,  
 2084 ["NotSucceeds"] = 8833,  
 2085 ["sub"] = 8834,  
 2086 ["subset"] = 8834,  
 2087 ["sup"] = 8835,  
 2088 ["supset"] = 8835,  
 2089 ["Superset"] = 8835,  
 2090 ["nsub"] = 8836,  
 2091 ["nsup"] = 8837,  
 2092 ["sube"] = 8838,  
 2093 ["SubsetEqual"] = 8838,  
 2094 ["subseteq"] = 8838,  
 2095 ["supe"] = 8839,  
 2096 ["supseteq"] = 8839,  
 2097 ["SupersetEqual"] = 8839,  
 2098 ["nsube"] = 8840,  
 2099 ["nsubseteq"] = 8840,  
 2100 ["NotSubsetEqual"] = 8840,  
 2101 ["nsupe"] = 8841,  
 2102 ["nsupseteq"] = 8841,  
 2103 ["NotSupersetEqual"] = 8841,  
 2104 ["subne"] = 8842,  
 2105 ["subsetneq"] = 8842,  
 2106 ["supne"] = 8843,  
 2107 ["supsetneq"] = 8843,  
 2108 ["cupdot"] = 8845,  
 2109 ["uplus"] = 8846,  
 2110 ["UnionPlus"] = 8846,  
 2111 ["sqsub"] = 8847,  
 2112 ["SquareSubset"] = 8847,  
 2113 ["sqsubset"] = 8847,  
 2114 ["sqsup"] = 8848,  
 2115 ["SquareSuperset"] = 8848,  
 2116 ["sqsupset"] = 8848,  
 2117 ["sqsube"] = 8849,  
 2118 ["SquareSubsetEqual"] = 8849,  
 2119 ["sqsubseteq"] = 8849,  
 2120 ["sqsupe"] = 8850,  
 2121 ["SquareSupersetEqual"] = 8850,  
 2122 ["sqsupseteq"] = 8850,  
 2123 ["sqcap"] = 8851,  
 2124 ["SquareIntersection"] = 8851,  
 2125 ["sqcup"] = 8852,  
 2126 ["SquareUnion"] = 8852,  
 2127 ["oplus"] = 8853,  
 2128 ["CirclePlus"] = 8853,



2129 ["ominus"] = 8854,  
 2130 ["CircleMinus"] = 8854,  
 2131 ["otimes"] = 8855,  
 2132 ["CircleTimes"] = 8855,  
 2133 ["osol"] = 8856,  
 2134 ["odot"] = 8857,  
 2135 ["CircleDot"] = 8857,  
 2136 ["ocir"] = 8858,  
 2137 ["circledcirc"] = 8858,  
 2138 ["oast"] = 8859,  
 2139 ["circledast"] = 8859,  
 2140 ["odash"] = 8861,  
 2141 ["circleddash"] = 8861,  
 2142 ["plusb"] = 8862,  
 2143 ["boxplus"] = 8862,  
 2144 ["minusb"] = 8863,  
 2145 ["boxminus"] = 8863,  
 2146 ["timesb"] = 8864,  
 2147 ["boxtimes"] = 8864,  
 2148 ["sdotb"] = 8865,  
 2149 ["dotsquare"] = 8865,  
 2150 ["vdash"] = 8866,  
 2151 ["RightTee"] = 8866,  
 2152 ["dashv"] = 8867,  
 2153 ["LeftTee"] = 8867,  
 2154 ["top"] = 8868,  
 2155 ["DownTee"] = 8868,  
 2156 ["bottom"] = 8869,  
 2157 ["bot"] = 8869,  
 2158 ["perp"] = 8869,  
 2159 ["UpTee"] = 8869,  
 2160 ["models"] = 8871,  
 2161 ["vDash"] = 8872,  
 2162 ["DoubleRightTee"] = 8872,  
 2163 ["Vdash"] = 8873,  
 2164 ["Vvdash"] = 8874,  
 2165 ["VDash"] = 8875,  
 2166 ["nvdash"] = 8876,  
 2167 ["nvDash"] = 8877,  
 2168 ["nVdash"] = 8878,  
 2169 ["nVDash"] = 8879,  
 2170 ["prurel"] = 8880,  
 2171 ["vltri"] = 8882,  
 2172 ["vartriangleleft"] = 8882,  
 2173 ["LeftTriangle"] = 8882,  
 2174 ["vrtri"] = 8883,  
 2175 ["vartriangleright"] = 8883,

2176 ["RightTriangle"] = 8883,  
 2177 ["ltrie"] = 8884,  
 2178 ["trianglelefteq"] = 8884,  
 2179 ["LeftTriangleEqual"] = 8884,  
 2180 ["rtrie"] = 8885,  
 2181 ["trianglerighteq"] = 8885,  
 2182 ["RightTriangleEqual"] = 8885,  
 2183 ["origof"] = 8886,  
 2184 ["imof"] = 8887,  
 2185 ["mumap"] = 8888,  
 2186 ["multimap"] = 8888,  
 2187 ["hercon"] = 8889,  
 2188 ["intcal"] = 8890,  
 2189 ["intercal"] = 8890,  
 2190 ["veebar"] = 8891,  
 2191 ["barvee"] = 8893,  
 2192 ["angrtvb"] = 8894,  
 2193 ["lrtri"] = 8895,  
 2194 ["xwedge"] = 8896,  
 2195 ["Wedge"] = 8896,  
 2196 ["bigwedge"] = 8896,  
 2197 ["xvee"] = 8897,  
 2198 ["Vee"] = 8897,  
 2199 ["bigvee"] = 8897,  
 2200 ["xcap"] = 8898,  
 2201 ["Intersection"] = 8898,  
 2202 ["bigcap"] = 8898,  
 2203 ["xcup"] = 8899,  
 2204 ["Union"] = 8899,  
 2205 ["bigcup"] = 8899,  
 2206 ["diam"] = 8900,  
 2207 ["diamond"] = 8900,  
 2208 ["Diamond"] = 8900,  
 2209 ["sdot"] = 8901,  
 2210 ["sstarf"] = 8902,  
 2211 ["Star"] = 8902,  
 2212 ["divonx"] = 8903,  
 2213 ["divideontimes"] = 8903,  
 2214 ["bowtie"] = 8904,  
 2215 ["ltimes"] = 8905,  
 2216 ["rtimes"] = 8906,  
 2217 ["lthree"] = 8907,  
 2218 ["leftthreetimes"] = 8907,  
 2219 ["rthree"] = 8908,  
 2220 ["rightthreetimes"] = 8908,  
 2221 ["bsime"] = 8909,  
 2222 ["backsimeq"] = 8909,

2223 ["cuvee"] = 8910,  
2224 ["curlyvee"] = 8910,  
2225 ["cuwed"] = 8911,  
2226 ["curlywedge"] = 8911,  
2227 ["Sub"] = 8912,  
2228 ["Subset"] = 8912,  
2229 ["Sup"] = 8913,  
2230 ["Supset"] = 8913,  
2231 ["Cap"] = 8914,  
2232 ["Cup"] = 8915,  
2233 ["fork"] = 8916,  
2234 ["pitchfork"] = 8916,  
2235 ["epar"] = 8917,  
2236 ["ltdot"] = 8918,  
2237 ["lessdot"] = 8918,  
2238 ["gtdot"] = 8919,  
2239 ["gtrdot"] = 8919,  
2240 ["Ll"] = 8920,  
2241 ["Gg"] = 8921,  
2242 ["ggg"] = 8921,  
2243 ["leg"] = 8922,  
2244 ["LessEqualGreater"] = 8922,  
2245 ["lesseqgtr"] = 8922,  
2246 ["gel"] = 8923,  
2247 ["gtreqless"] = 8923,  
2248 ["GreaterEqualLess"] = 8923,  
2249 ["cuepr"] = 8926,  
2250 ["curlyeqprec"] = 8926,  
2251 ["cuesc"] = 8927,  
2252 ["curlyeqsucc"] = 8927,  
2253 ["nprcue"] = 8928,  
2254 ["NotPrecedesSlantEqual"] = 8928,  
2255 ["nsccue"] = 8929,  
2256 ["NotSucceedsSlantEqual"] = 8929,  
2257 ["nsqsube"] = 8930,  
2258 ["NotSquareSubsetEqual"] = 8930,  
2259 ["nsqsupe"] = 8931,  
2260 ["NotSquareSupersetEqual"] = 8931,  
2261 ["lnsim"] = 8934,  
2262 ["gnsim"] = 8935,  
2263 ["prnsim"] = 8936,  
2264 ["precnsim"] = 8936,  
2265 ["scnsim"] = 8937,  
2266 ["succnsim"] = 8937,  
2267 ["nltri"] = 8938,  
2268 ["ntriangleleft"] = 8938,  
2269 ["NotLeftTriangle"] = 8938,

2270 ["nrtri"] = 8939,  
 2271 ["ntriangleright"] = 8939,  
 2272 ["NotRightTriangle"] = 8939,  
 2273 ["nltrie"] = 8940,  
 2274 ["ntrianglelefteq"] = 8940,  
 2275 ["NotLeftTriangleEqual"] = 8940,  
 2276 ["nrtrie"] = 8941,  
 2277 ["ntrianglerighteq"] = 8941,  
 2278 ["NotRightTriangleEqual"] = 8941,  
 2279 ["vellip"] = 8942,  
 2280 ["ctdot"] = 8943,  
 2281 ["utdot"] = 8944,  
 2282 ["dtdot"] = 8945,  
 2283 ["disin"] = 8946,  
 2284 ["isinsv"] = 8947,  
 2285 ["isins"] = 8948,  
 2286 ["isindot"] = 8949,  
 2287 ["notinvc"] = 8950,  
 2288 ["notinvb"] = 8951,  
 2289 ["isinE"] = 8953,  
 2290 ["nisd"] = 8954,  
 2291 ["xnis"] = 8955,  
 2292 ["nis"] = 8956,  
 2293 ["notnivc"] = 8957,  
 2294 ["notnivb"] = 8958,  
 2295 ["barwed"] = 8965,  
 2296 ["barwedge"] = 8965,  
 2297 ["Barwed"] = 8966,  
 2298 ["doublebarwedge"] = 8966,  
 2299 ["lceil"] = 8968,  
 2300 ["LeftCeiling"] = 8968,  
 2301 ["rceil"] = 8969,  
 2302 ["RightCeiling"] = 8969,  
 2303 ["lfloor"] = 8970,  
 2304 ["LeftFloor"] = 8970,  
 2305 ["rfloor"] = 8971,  
 2306 ["RightFloor"] = 8971,  
 2307 ["drcrop"] = 8972,  
 2308 ["dlcrop"] = 8973,  
 2309 ["urcrop"] = 8974,  
 2310 ["ulcrop"] = 8975,  
 2311 ["bnot"] = 8976,  
 2312 ["proflin"] = 8978,  
 2313 ["profsurf"] = 8979,  
 2314 ["telrec"] = 8981,  
 2315 ["target"] = 8982,  
 2316 ["ulcorn"] = 8988,

2317 ["ulcorner"] = 8988,  
2318 ["urcorn"] = 8989,  
2319 ["urcorner"] = 8989,  
2320 ["dlcorn"] = 8990,  
2321 ["llcorner"] = 8990,  
2322 ["drcorn"] = 8991,  
2323 ["lrcorner"] = 8991,  
2324 ["frown"] = 8994,  
2325 ["sfrown"] = 8994,  
2326 ["smile"] = 8995,  
2327 ["ssmile"] = 8995,  
2328 ["cylcty"] = 9005,  
2329 ["profalar"] = 9006,  
2330 ["topbot"] = 9014,  
2331 ["ovbar"] = 9021,  
2332 ["solbar"] = 9023,  
2333 ["angzarr"] = 9084,  
2334 ["lmoust"] = 9136,  
2335 ["lmoustache"] = 9136,  
2336 ["rmoust"] = 9137,  
2337 ["rmoustache"] = 9137,  
2338 ["tbrk"] = 9140,  
2339 ["OverBracket"] = 9140,  
2340 ["bbrk"] = 9141,  
2341 ["UnderBracket"] = 9141,  
2342 ["bbrktbrk"] = 9142,  
2343 ["OverParenthesis"] = 9180,  
2344 ["UnderParenthesis"] = 9181,  
2345 ["OverBrace"] = 9182,  
2346 ["UnderBrace"] = 9183,  
2347 ["trpezium"] = 9186,  
2348 ["elinters"] = 9191,  
2349 ["blank"] = 9251,  
2350 ["oS"] = 9416,  
2351 ["circledS"] = 9416,  
2352 ["boxh"] = 9472,  
2353 ["HorizontalLine"] = 9472,  
2354 ["boxv"] = 9474,  
2355 ["boxdr"] = 9484,  
2356 ["boxdl"] = 9488,  
2357 ["boxur"] = 9492,  
2358 ["boxul"] = 9496,  
2359 ["boxvr"] = 9500,  
2360 ["boxvl"] = 9508,  
2361 ["boxhd"] = 9516,  
2362 ["boxhu"] = 9524,  
2363 ["boxvh"] = 9532,

2364 ["boxH"] = 9552,  
2365 ["boxV"] = 9553,  
2366 ["boxdR"] = 9554,  
2367 ["boxDr"] = 9555,  
2368 ["boxDR"] = 9556,  
2369 ["boxdL"] = 9557,  
2370 ["boxDl"] = 9558,  
2371 ["boxDL"] = 9559,  
2372 ["boxuR"] = 9560,  
2373 ["boxUr"] = 9561,  
2374 ["boxUR"] = 9562,  
2375 ["boxuL"] = 9563,  
2376 ["boxUl"] = 9564,  
2377 ["boxUL"] = 9565,  
2378 ["boxvR"] = 9566,  
2379 ["boxVr"] = 9567,  
2380 ["boxVR"] = 9568,  
2381 ["boxvL"] = 9569,  
2382 ["boxVl"] = 9570,  
2383 ["boxVL"] = 9571,  
2384 ["boxHd"] = 9572,  
2385 ["boxhD"] = 9573,  
2386 ["boxHD"] = 9574,  
2387 ["boxHu"] = 9575,  
2388 ["boxhU"] = 9576,  
2389 ["boxHU"] = 9577,  
2390 ["boxvH"] = 9578,  
2391 ["boxVh"] = 9579,  
2392 ["boxVH"] = 9580,  
2393 ["uhblk"] = 9600,  
2394 ["lhblk"] = 9604,  
2395 ["block"] = 9608,  
2396 ["blk14"] = 9617,  
2397 ["blk12"] = 9618,  
2398 ["blk34"] = 9619,  
2399 ["squ"] = 9633,  
2400 ["square"] = 9633,  
2401 ["Square"] = 9633,  
2402 ["squf"] = 9642,  
2403 ["squarf"] = 9642,  
2404 ["blacksquare"] = 9642,  
2405 ["FilledVerySmallSquare"] = 9642,  
2406 ["EmptyVerySmallSquare"] = 9643,  
2407 ["rect"] = 9645,  
2408 ["marker"] = 9646,  
2409 ["fltns"] = 9649,  
2410 ["xutri"] = 9651,

2411 ["bigtriangleup"] = 9651,  
2412 ["utrif"] = 9652,  
2413 ["blacktriangle"] = 9652,  
2414 ["utri"] = 9653,  
2415 ["triangle"] = 9653,  
2416 ["rtrif"] = 9656,  
2417 ["blacktriangleright"] = 9656,  
2418 ["rtri"] = 9657,  
2419 ["triangleright"] = 9657,  
2420 ["xdtri"] = 9661,  
2421 ["bigtriangledown"] = 9661,  
2422 ["dtrif"] = 9662,  
2423 ["blacktriangledown"] = 9662,  
2424 ["dtri"] = 9663,  
2425 ["triangledown"] = 9663,  
2426 ["ltrif"] = 9666,  
2427 ["blacktriangleleft"] = 9666,  
2428 ["ltri"] = 9667,  
2429 ["triangleleft"] = 9667,  
2430 ["loz"] = 9674,  
2431 ["lozenge"] = 9674,  
2432 ["cir"] = 9675,  
2433 ["tridot"] = 9708,  
2434 ["xcirc"] = 9711,  
2435 ["bigcirc"] = 9711,  
2436 ["ultri"] = 9720,  
2437 ["urtri"] = 9721,  
2438 ["lltri"] = 9722,  
2439 ["EmptySmallSquare"] = 9723,  
2440 ["FilledSmallSquare"] = 9724,  
2441 ["starf"] = 9733,  
2442 ["bigstar"] = 9733,  
2443 ["star"] = 9734,  
2444 ["phone"] = 9742,  
2445 ["female"] = 9792,  
2446 ["male"] = 9794,  
2447 ["spades"] = 9824,  
2448 ["spadesuit"] = 9824,  
2449 ["clubs"] = 9827,  
2450 ["clubsuit"] = 9827,  
2451 ["hearts"] = 9829,  
2452 ["heartsuit"] = 9829,  
2453 ["diams"] = 9830,  
2454 ["diamondsuit"] = 9830,  
2455 ["sung"] = 9834,  
2456 ["flat"] = 9837,  
2457 ["natur"] = 9838,

2458 ["natural"] = 9838,  
 2459 ["sharp"] = 9839,  
 2460 ["check"] = 10003,  
 2461 ["checkmark"] = 10003,  
 2462 ["cross"] = 10007,  
 2463 ["malt"] = 10016,  
 2464 ["maltese"] = 10016,  
 2465 ["sext"] = 10038,  
 2466 ["VerticalSeparator"] = 10072,  
 2467 ["lbrk"] = 10098,  
 2468 ["rbrk"] = 10099,  
 2469 ["lobrk"] = 10214,  
 2470 ["LeftDoubleBracket"] = 10214,  
 2471 ["robrk"] = 10215,  
 2472 ["RightDoubleBracket"] = 10215,  
 2473 ["lang"] = 10216,  
 2474 ["LeftAngleBracket"] = 10216,  
 2475 ["langle"] = 10216,  
 2476 ["rang"] = 10217,  
 2477 ["RightAngleBracket"] = 10217,  
 2478 ["rangle"] = 10217,  
 2479 ["Lang"] = 10218,  
 2480 ["Rang"] = 10219,  
 2481 ["loang"] = 10220,  
 2482 ["roang"] = 10221,  
 2483 ["xlarr"] = 10229,  
 2484 ["longleftarrow"] = 10229,  
 2485 ["LongLeftArrow"] = 10229,  
 2486 ["xrarr"] = 10230,  
 2487 ["longrightarrow"] = 10230,  
 2488 ["LongRightArrow"] = 10230,  
 2489 ["xharr"] = 10231,  
 2490 ["longleftrightarrow"] = 10231,  
 2491 ["LongLeftRightArrow"] = 10231,  
 2492 ["xlArr"] = 10232,  
 2493 ["Longleftarrow"] = 10232,  
 2494 ["DoubleLongLeftArrow"] = 10232,  
 2495 ["xrArr"] = 10233,  
 2496 ["Longrightarrow"] = 10233,  
 2497 ["DoubleLongRightArrow"] = 10233,  
 2498 ["xhArr"] = 10234,  
 2499 ["Longleftrightarrow"] = 10234,  
 2500 ["DoubleLongLeftRightArrow"] = 10234,  
 2501 ["xmap"] = 10236,  
 2502 ["longmapsto"] = 10236,  
 2503 ["dzigrarr"] = 10239,  
 2504 ["nvlArr"] = 10498,



2505 ["nvrArr"] = 10499,  
2506 ["nvHarr"] = 10500,  
2507 ["Map"] = 10501,  
2508 ["lbarr"] = 10508,  
2509 ["rbarr"] = 10509,  
2510 ["bkarow"] = 10509,  
2511 ["lBarr"] = 10510,  
2512 ["rBarr"] = 10511,  
2513 ["dbkarow"] = 10511,  
2514 ["RBarr"] = 10512,  
2515 ["drbkarow"] = 10512,  
2516 ["DDotrahd"] = 10513,  
2517 ["UpArrowBar"] = 10514,  
2518 ["DownArrowBar"] = 10515,  
2519 ["Rarrtl"] = 10518,  
2520 ["latail"] = 10521,  
2521 ["ratail"] = 10522,  
2522 ["lAtail"] = 10523,  
2523 ["rAtail"] = 10524,  
2524 ["larrfs"] = 10525,  
2525 ["rarrfs"] = 10526,  
2526 ["larrbfs"] = 10527,  
2527 ["rarrbfs"] = 10528,  
2528 ["nwarhk"] = 10531,  
2529 ["nearhk"] = 10532,  
2530 ["searhk"] = 10533,  
2531 ["hksearow"] = 10533,  
2532 ["swarhk"] = 10534,  
2533 ["hkswarow"] = 10534,  
2534 ["nwnear"] = 10535,  
2535 ["nesear"] = 10536,  
2536 ["toea"] = 10536,  
2537 ["seswar"] = 10537,  
2538 ["tosa"] = 10537,  
2539 ["swnwar"] = 10538,  
2540 ["rarrc"] = 10547,  
2541 ["cudarr"] = 10549,  
2542 ["ldca"] = 10550,  
2543 ["rdca"] = 10551,  
2544 ["cudarrl"] = 10552,  
2545 ["larrpl"] = 10553,  
2546 ["curarrm"] = 10556,  
2547 ["cularrp"] = 10557,  
2548 ["rarrpl"] = 10565,  
2549 ["harrcir"] = 10568,  
2550 ["Uarroccir"] = 10569,  
2551 ["lurdshar"] = 10570,

2552 ["ldrushar"] = 10571,  
2553 ["LeftRightVector"] = 10574,  
2554 ["RightUpDownVector"] = 10575,  
2555 ["DownLeftRightVector"] = 10576,  
2556 ["LeftUpDownVector"] = 10577,  
2557 ["LeftVectorBar"] = 10578,  
2558 ["RightVectorBar"] = 10579,  
2559 ["RightUpVectorBar"] = 10580,  
2560 ["RightDownVectorBar"] = 10581,  
2561 ["DownLeftVectorBar"] = 10582,  
2562 ["DownRightVectorBar"] = 10583,  
2563 ["LeftUpVectorBar"] = 10584,  
2564 ["LeftDownVectorBar"] = 10585,  
2565 ["LeftTeeVector"] = 10586,  
2566 ["RightTeeVector"] = 10587,  
2567 ["RightUpTeeVector"] = 10588,  
2568 ["RightDownTeeVector"] = 10589,  
2569 ["DownLeftTeeVector"] = 10590,  
2570 ["DownRightTeeVector"] = 10591,  
2571 ["LeftUpTeeVector"] = 10592,  
2572 ["LeftDownTeeVector"] = 10593,  
2573 ["lHar"] = 10594,  
2574 ["uHar"] = 10595,  
2575 ["rHar"] = 10596,  
2576 ["dHar"] = 10597,  
2577 ["luruhar"] = 10598,  
2578 ["ldrdhar"] = 10599,  
2579 ["ruluhar"] = 10600,  
2580 ["rdldhar"] = 10601,  
2581 ["lharul"] = 10602,  
2582 ["llhard"] = 10603,  
2583 ["rharul"] = 10604,  
2584 ["lrhard"] = 10605,  
2585 ["udhar"] = 10606,  
2586 ["UpEquilibrium"] = 10606,  
2587 ["duhar"] = 10607,  
2588 ["ReverseUpEquilibrium"] = 10607,  
2589 ["RoundImplies"] = 10608,  
2590 ["erarr"] = 10609,  
2591 ["simrarr"] = 10610,  
2592 ["larrsim"] = 10611,  
2593 ["rarrsim"] = 10612,  
2594 ["rarrap"] = 10613,  
2595 ["ltlarr"] = 10614,  
2596 ["gtrarr"] = 10616,  
2597 ["subrarr"] = 10617,  
2598 ["suplarr"] = 10619,

2599 ["lfisht"] = 10620,  
2600 ["rfisht"] = 10621,  
2601 ["ufisht"] = 10622,  
2602 ["dfisht"] = 10623,  
2603 ["lopar"] = 10629,  
2604 ["ropar"] = 10630,  
2605 ["lbrke"] = 10635,  
2606 ["rbrke"] = 10636,  
2607 ["lbrkslu"] = 10637,  
2608 ["rbrksld"] = 10638,  
2609 ["lbrksld"] = 10639,  
2610 ["rbrkslu"] = 10640,  
2611 ["langd"] = 10641,  
2612 ["rangd"] = 10642,  
2613 ["lparlt"] = 10643,  
2614 ["rpargt"] = 10644,  
2615 ["gtlPar"] = 10645,  
2616 ["ltrPar"] = 10646,  
2617 ["vzigzag"] = 10650,  
2618 ["vangrt"] = 10652,  
2619 ["angrtvbd"] = 10653,  
2620 ["ange"] = 10660,  
2621 ["range"] = 10661,  
2622 ["dwangle"] = 10662,  
2623 ["uwangle"] = 10663,  
2624 ["angmsdaa"] = 10664,  
2625 ["angmsdab"] = 10665,  
2626 ["angmsdac"] = 10666,  
2627 ["angmsdad"] = 10667,  
2628 ["angmsdae"] = 10668,  
2629 ["angmsdaf"] = 10669,  
2630 ["angmsdag"] = 10670,  
2631 ["angmsdah"] = 10671,  
2632 ["bemptyv"] = 10672,  
2633 ["demptyv"] = 10673,  
2634 ["cemptyv"] = 10674,  
2635 ["raemptyv"] = 10675,  
2636 ["laemptyv"] = 10676,  
2637 ["ohbar"] = 10677,  
2638 ["omid"] = 10678,  
2639 ["opar"] = 10679,  
2640 ["operp"] = 10681,  
2641 ["olcross"] = 10683,  
2642 ["odsold"] = 10684,  
2643 ["olcir"] = 10686,  
2644 ["ofcir"] = 10687,  
2645 ["olt"] = 10688,

2646 ["ogt"] = 10689,  
 2647 ["cirscir"] = 10690,  
 2648 ["cirE"] = 10691,  
 2649 ["solb"] = 10692,  
 2650 ["bsolb"] = 10693,  
 2651 ["boxbox"] = 10697,  
 2652 ["trisb"] = 10701,  
 2653 ["rtriltri"] = 10702,  
 2654 ["LeftTriangleBar"] = 10703,  
 2655 ["RightTriangleBar"] = 10704,  
 2656 ["race"] = 10714,  
 2657 ["iinfin"] = 10716,  
 2658 ["infintie"] = 10717,  
 2659 ["nvinfin"] = 10718,  
 2660 ["eparsl"] = 10723,  
 2661 ["smeparsl"] = 10724,  
 2662 ["eqvparsl"] = 10725,  
 2663 ["lozf"] = 10731,  
 2664 ["blacklozenge"] = 10731,  
 2665 ["RuleDelayed"] = 10740,  
 2666 ["dsol"] = 10742,  
 2667 ["xodot"] = 10752,  
 2668 ["bigodot"] = 10752,  
 2669 ["xoplus"] = 10753,  
 2670 ["bigoplus"] = 10753,  
 2671 ["xotime"] = 10754,  
 2672 ["bigotimes"] = 10754,  
 2673 ["xuplus"] = 10756,  
 2674 ["biguplus"] = 10756,  
 2675 ["xsqcup"] = 10758,  
 2676 ["bigsqcup"] = 10758,  
 2677 ["qint"] = 10764,  
 2678 ["iiiint"] = 10764,  
 2679 ["fpartint"] = 10765,  
 2680 ["cirfnint"] = 10768,  
 2681 ["awint"] = 10769,  
 2682 ["rppolint"] = 10770,  
 2683 ["scpolint"] = 10771,  
 2684 ["npolint"] = 10772,  
 2685 ["pointint"] = 10773,  
 2686 ["quatint"] = 10774,  
 2687 ["intlarhk"] = 10775,  
 2688 ["pluscir"] = 10786,  
 2689 ["plusacir"] = 10787,  
 2690 ["simplus"] = 10788,  
 2691 ["plusdu"] = 10789,  
 2692 ["plussim"] = 10790,

2693 ["plustwo"] = 10791,  
 2694 ["mcomma"] = 10793,  
 2695 ["minusdu"] = 10794,  
 2696 ["loplus"] = 10797,  
 2697 ["roplus"] = 10798,  
 2698 ["Cross"] = 10799,  
 2699 ["timesd"] = 10800,  
 2700 ["timesbar"] = 10801,  
 2701 ["smashp"] = 10803,  
 2702 ["lotimes"] = 10804,  
 2703 ["rotimes"] = 10805,  
 2704 ["otimesas"] = 10806,  
 2705 ["Otimes"] = 10807,  
 2706 ["odiv"] = 10808,  
 2707 ["triplus"] = 10809,  
 2708 ["triminus"] = 10810,  
 2709 ["tritime"] = 10811,  
 2710 ["iproduct"] = 10812,  
 2711 ["intprod"] = 10812,  
 2712 ["amalg"] = 10815,  
 2713 ["capdot"] = 10816,  
 2714 ["ncup"] = 10818,  
 2715 ["ncap"] = 10819,  
 2716 ["capand"] = 10820,  
 2717 ["cupor"] = 10821,  
 2718 ["cupcap"] = 10822,  
 2719 ["capcup"] = 10823,  
 2720 ["cupbrcap"] = 10824,  
 2721 ["capbrcup"] = 10825,  
 2722 ["cupcup"] = 10826,  
 2723 ["capcap"] = 10827,  
 2724 ["ccups"] = 10828,  
 2725 ["ccaps"] = 10829,  
 2726 ["ccupssm"] = 10832,  
 2727 ["And"] = 10835,  
 2728 ["Or"] = 10836,  
 2729 ["andand"] = 10837,  
 2730 ["oror"] = 10838,  
 2731 ["orslope"] = 10839,  
 2732 ["andslope"] = 10840,  
 2733 ["andv"] = 10842,  
 2734 ["orv"] = 10843,  
 2735 ["andd"] = 10844,  
 2736 ["ord"] = 10845,  
 2737 ["wedbar"] = 10847,  
 2738 ["sdote"] = 10854,  
 2739 ["simdot"] = 10858,

2740 ["congdot"] = 10861,  
 2741 ["easter"] = 10862,  
 2742 ["apacir"] = 10863,  
 2743 ["apE"] = 10864,  
 2744 ["eplus"] = 10865,  
 2745 ["pluse"] = 10866,  
 2746 ["Esim"] = 10867,  
 2747 ["Colone"] = 10868,  
 2748 ["Equal"] = 10869,  
 2749 ["eDDot"] = 10871,  
 2750 ["ddotseq"] = 10871,  
 2751 ["equivDD"] = 10872,  
 2752 ["ltcir"] = 10873,  
 2753 ["gtcir"] = 10874,  
 2754 ["ltquest"] = 10875,  
 2755 ["gtquest"] = 10876,  
 2756 ["les"] = 10877,  
 2757 ["LessSlantEqual"] = 10877,  
 2758 ["leqslant"] = 10877,  
 2759 ["ges"] = 10878,  
 2760 ["GreaterSlantEqual"] = 10878,  
 2761 ["geqslant"] = 10878,  
 2762 ["lesdot"] = 10879,  
 2763 ["gesdot"] = 10880,  
 2764 ["lesdoto"] = 10881,  
 2765 ["gesdoto"] = 10882,  
 2766 ["lesdotor"] = 10883,  
 2767 ["gesdotor"] = 10884,  
 2768 ["lap"] = 10885,  
 2769 ["lessapprox"] = 10885,  
 2770 ["gap"] = 10886,  
 2771 ["gtrapprox"] = 10886,  
 2772 ["lne"] = 10887,  
 2773 ["lneq"] = 10887,  
 2774 ["gne"] = 10888,  
 2775 ["gneq"] = 10888,  
 2776 ["lnap"] = 10889,  
 2777 ["lnapprox"] = 10889,  
 2778 ["gnap"] = 10890,  
 2779 ["gnapprox"] = 10890,  
 2780 ["lEg"] = 10891,  
 2781 ["lesseqqgtr"] = 10891,  
 2782 ["gEl"] = 10892,  
 2783 ["gtreqqless"] = 10892,  
 2784 ["lsime"] = 10893,  
 2785 ["gsime"] = 10894,  
 2786 ["lsimg"] = 10895,

2787 ["gsiml"] = 10896,  
2788 ["lgE"] = 10897,  
2789 ["glE"] = 10898,  
2790 ["lesges"] = 10899,  
2791 ["gesles"] = 10900,  
2792 ["els"] = 10901,  
2793 ["eqslantless"] = 10901,  
2794 ["egs"] = 10902,  
2795 ["eqslantgtr"] = 10902,  
2796 ["elsdot"] = 10903,  
2797 ["egsdot"] = 10904,  
2798 ["el"] = 10905,  
2799 ["eg"] = 10906,  
2800 ["siml"] = 10909,  
2801 ["simg"] = 10910,  
2802 ["simLE"] = 10911,  
2803 ["simGE"] = 10912,  
2804 ["LessLess"] = 10913,  
2805 ["GreaterGreater"] = 10914,  
2806 ["glj"] = 10916,  
2807 ["gla"] = 10917,  
2808 ["ltcc"] = 10918,  
2809 ["gtcc"] = 10919,  
2810 ["lescc"] = 10920,  
2811 ["gescc"] = 10921,  
2812 ["smt"] = 10922,  
2813 ["lat"] = 10923,  
2814 ["smtE"] = 10924,  
2815 ["late"] = 10925,  
2816 ["bumpE"] = 10926,  
2817 ["pre"] = 10927,  
2818 ["preceq"] = 10927,  
2819 ["PrecedesEqual"] = 10927,  
2820 ["sce"] = 10928,  
2821 ["succeq"] = 10928,  
2822 ["SucceedsEqual"] = 10928,  
2823 ["prE"] = 10931,  
2824 ["scE"] = 10932,  
2825 ["prnE"] = 10933,  
2826 ["precneqq"] = 10933,  
2827 ["scnE"] = 10934,  
2828 ["succneqq"] = 10934,  
2829 ["prap"] = 10935,  
2830 ["precapprox"] = 10935,  
2831 ["scap"] = 10936,  
2832 ["succapprox"] = 10936,  
2833 ["prnap"] = 10937,

2834 ["precnapprox"] = 10937,  
 2835 ["scnap"] = 10938,  
 2836 ["succnapprox"] = 10938,  
 2837 ["Pr"] = 10939,  
 2838 ["Sc"] = 10940,  
 2839 ["subdot"] = 10941,  
 2840 ["supdot"] = 10942,  
 2841 ["subplus"] = 10943,  
 2842 ["supplus"] = 10944,  
 2843 ["submult"] = 10945,  
 2844 ["supmult"] = 10946,  
 2845 ["subedot"] = 10947,  
 2846 ["supedot"] = 10948,  
 2847 ["subE"] = 10949,  
 2848 ["subseteqq"] = 10949,  
 2849 ["supE"] = 10950,  
 2850 ["supseteqq"] = 10950,  
 2851 ["subsim"] = 10951,  
 2852 ["supsim"] = 10952,  
 2853 ["subnE"] = 10955,  
 2854 ["subsetneqq"] = 10955,  
 2855 ["supnE"] = 10956,  
 2856 ["supsetneqq"] = 10956,  
 2857 ["csub"] = 10959,  
 2858 ["csup"] = 10960,  
 2859 ["csube"] = 10961,  
 2860 ["csupe"] = 10962,  
 2861 ["subsup"] = 10963,  
 2862 ["supsub"] = 10964,  
 2863 ["subsub"] = 10965,  
 2864 ["supsup"] = 10966,  
 2865 ["suphsub"] = 10967,  
 2866 ["supdsub"] = 10968,  
 2867 ["forkv"] = 10969,  
 2868 ["topfork"] = 10970,  
 2869 ["mlcp"] = 10971,  
 2870 ["Dashv"] = 10980,  
 2871 ["DoubleLeftTee"] = 10980,  
 2872 ["Vdashl"] = 10982,  
 2873 ["Barv"] = 10983,  
 2874 ["vBar"] = 10984,  
 2875 ["vBarv"] = 10985,  
 2876 ["Vbar"] = 10987,  
 2877 ["Not"] = 10988,  
 2878 ["bNot"] = 10989,  
 2879 ["rnmid"] = 10990,  
 2880 ["cirmid"] = 10991,



2881 ["midcir"] = 10992,  
2882 ["topcir"] = 10993,  
2883 ["nhpar"] = 10994,  
2884 ["parsim"] = 10995,  
2885 ["parsl"] = 11005,  
2886 ["fflig"] = 64256,  
2887 ["filig"] = 64257,  
2888 ["fllig"] = 64258,  
2889 ["ffilig"] = 64259,  
2890 ["ffllig"] = 64260,  
2891 ["Ascr"] = 119964,  
2892 ["Cscr"] = 119966,  
2893 ["Dscr"] = 119967,  
2894 ["Gscr"] = 119970,  
2895 ["Jscr"] = 119973,  
2896 ["Kscr"] = 119974,  
2897 ["Nscr"] = 119977,  
2898 ["Oscr"] = 119978,  
2899 ["Pscr"] = 119979,  
2900 ["Qscr"] = 119980,  
2901 ["Sscr"] = 119982,  
2902 ["Tscr"] = 119983,  
2903 ["Uscr"] = 119984,  
2904 ["Vscr"] = 119985,  
2905 ["Wscr"] = 119986,  
2906 ["Xscr"] = 119987,  
2907 ["Yscr"] = 119988,  
2908 ["Zscr"] = 119989,  
2909 ["ascr"] = 119990,  
2910 ["bscr"] = 119991,  
2911 ["cscr"] = 119992,  
2912 ["dscr"] = 119993,  
2913 ["fscr"] = 119995,  
2914 ["hscr"] = 119997,  
2915 ["iscr"] = 119998,  
2916 ["jscr"] = 119999,  
2917 ["kscr"] = 120000,  
2918 ["lscr"] = 120001,  
2919 ["mscr"] = 120002,  
2920 ["nscr"] = 120003,  
2921 ["pscr"] = 120005,  
2922 ["qscr"] = 120006,  
2923 ["rscr"] = 120007,  
2924 ["sscr"] = 120008,  
2925 ["tscr"] = 120009,  
2926 ["uscr"] = 120010,  
2927 ["vscr"] = 120011,

2928 ["wscr"] = 120012,  
2929 ["xscr"] = 120013,  
2930 ["yscr"] = 120014,  
2931 ["zscr"] = 120015,  
2932 ["Afr"] = 120068,  
2933 ["Bfr"] = 120069,  
2934 ["Dfr"] = 120071,  
2935 ["Efr"] = 120072,  
2936 ["Ffr"] = 120073,  
2937 ["Gfr"] = 120074,  
2938 ["Jfr"] = 120077,  
2939 ["Kfr"] = 120078,  
2940 ["Lfr"] = 120079,  
2941 ["Mfr"] = 120080,  
2942 ["Nfr"] = 120081,  
2943 ["Ofr"] = 120082,  
2944 ["Pfr"] = 120083,  
2945 ["Qfr"] = 120084,  
2946 ["Sfr"] = 120086,  
2947 ["Tfr"] = 120087,  
2948 ["Ufr"] = 120088,  
2949 ["Vfr"] = 120089,  
2950 ["Wfr"] = 120090,  
2951 ["Xfr"] = 120091,  
2952 ["Yfr"] = 120092,  
2953 ["afr"] = 120094,  
2954 ["bfr"] = 120095,  
2955 ["cfr"] = 120096,  
2956 ["dfr"] = 120097,  
2957 ["efr"] = 120098,  
2958 ["ffr"] = 120099,  
2959 ["gfr"] = 120100,  
2960 ["hfr"] = 120101,  
2961 ["ifr"] = 120102,  
2962 ["jfr"] = 120103,  
2963 ["kfr"] = 120104,  
2964 ["lfr"] = 120105,  
2965 ["mfr"] = 120106,  
2966 ["nfr"] = 120107,  
2967 ["ofr"] = 120108,  
2968 ["pfr"] = 120109,  
2969 ["qfr"] = 120110,  
2970 ["rfr"] = 120111,  
2971 ["sfr"] = 120112,  
2972 ["tfr"] = 120113,  
2973 ["ufr"] = 120114,  
2974 ["vfr"] = 120115,

2975 ["wfr"] = 120116,  
2976 ["xfr"] = 120117,  
2977 ["yfr"] = 120118,  
2978 ["zfr"] = 120119,  
2979 ["Aopf"] = 120120,  
2980 ["Bopf"] = 120121,  
2981 ["Dopf"] = 120123,  
2982 ["Eopf"] = 120124,  
2983 ["Fopf"] = 120125,  
2984 ["Gopf"] = 120126,  
2985 ["Iopf"] = 120128,  
2986 ["Jopf"] = 120129,  
2987 ["Kopf"] = 120130,  
2988 ["Lopf"] = 120131,  
2989 ["Mopf"] = 120132,  
2990 ["Oopf"] = 120134,  
2991 ["Sopf"] = 120138,  
2992 ["Topf"] = 120139,  
2993 ["Uopf"] = 120140,  
2994 ["Vopf"] = 120141,  
2995 ["Wopf"] = 120142,  
2996 ["Xopf"] = 120143,  
2997 ["Yopf"] = 120144,  
2998 ["aopf"] = 120146,  
2999 ["bopf"] = 120147,  
3000 ["copf"] = 120148,  
3001 ["dopf"] = 120149,  
3002 ["eopf"] = 120150,  
3003 ["fopf"] = 120151,  
3004 ["gopf"] = 120152,  
3005 ["hopf"] = 120153,  
3006 ["iopf"] = 120154,  
3007 ["jopf"] = 120155,  
3008 ["kopf"] = 120156,  
3009 ["lopf"] = 120157,  
3010 ["mopf"] = 120158,  
3011 ["nopf"] = 120159,  
3012 ["oopf"] = 120160,  
3013 ["popf"] = 120161,  
3014 ["qopf"] = 120162,  
3015 ["ropf"] = 120163,  
3016 ["sopf"] = 120164,  
3017 ["topf"] = 120165,  
3018 ["uopf"] = 120166,  
3019 ["vopf"] = 120167,  
3020 ["wopf"] = 120168,  
3021 ["xopf"] = 120169,

```

3022 ["yopf"] = 120170,
3023 ["zopf"] = 120171,
3024 }

```

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

```

3025 function entities.dec_entity(s)
3026 return unicode.utf8.char(tonumber(s))
3027 end

```

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

```

3028 function entities.hex_entity(s)
3029 return unicode.utf8.char(tonumber("0x"..s))
3030 end

```

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

```

3031 function entities.char_entity(s)
3032 local n = character_entities[s]
3033 if n == nil then
3034 return "&" .. s .. ";"
3035 end
3036 return unicode.utf8.char(n)
3037 end

```

### 3.1.3 Plain T<sub>E</sub>X Writer

This section documents the `writer` object, which implements the routines for producing the T<sub>E</sub>X output. The object is an amalgamate of the generic, T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X 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 on page 6), 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.

```

3038 M.writer = {}

```

The `writer.new` method creates and returns a new T<sub>E</sub>X writer object associated with the Lua interface options (see Section 2.1.2 on page 7) `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.

```

3039 function M.writer.new(options)
3040 local self = {}
3041 options = options or {}

```

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

```

3042 setmetatable(options, { __index = function (_, key)
3043 return defaultOptions[key] end })

```

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

```

3044 local slice_specifiers = {}
3045 for specifier in options.slice:gmatch("[^%s]+") do
3046 table.insert(slice_specifiers, specifier)
3047 end
3048
3049 if #slice_specifiers == 2 then
3050 self.slice_begin, self.slice_end = table.unpack(slice_specifiers)
3051 local slice_begin_type = self.slice_begin:sub(1, 1)
3052 if slice_begin_type ~= "^" and slice_begin_type ~= "$" then
3053 self.slice_begin = "^" .. self.slice_begin
3054 end
3055 local slice_end_type = self.slice_end:sub(1, 1)
3056 if slice_end_type ~= "^" and slice_end_type ~= "$" then
3057 self.slice_end = "$" .. self.slice_end
3058 end
3059 elseif #slice_specifiers == 1 then
3060 self.slice_begin = "^" .. slice_specifiers[1]
3061 self.slice_end = "$" .. slice_specifiers[1]
3062 end
3063
3064 if self.slice_begin == "^" and self.slice_end ~= "^" then
3065 self.is_writing = true
3066 else
3067 self.is_writing = false
3068 end

```

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

```

3069 self.suffix = ".tex"

```

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

```

3070 self.space = " "

```

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

```

3071 self.nbsp = "\\markdownRendererNbsp{}"

```

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

```

3072 function self.plain(s)
3073 return s
3074 end

```

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

```
3075 function self.paragraph(s)
3076 if not self.is_writing then return "" end
3077 return s
3078 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.

```
3079 function self.pack(name)
3080 return [[\input]] .. name .. [[\relax]]
3081 end
```

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

```
3082 function self.interblocksep()
3083 if not self.is_writing then return "" end
3084 return "\\markdownRendererInterblockSeparator\n{}"
3085 end
```

Define `writer->eof` as the end of file marker in the output format.

```
3086 self.eof = [[\relax]]
```

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

```
3087 self.linebreak = "\\markdownRendererLineBreak\n{}"
```

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

```
3088 self.ellipsis = "\\markdownRendererEllipsis{}"
```

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

```
3089 function self.hrule()
3090 if not self.is_writing then return "" end
3091 return "\\markdownRendererHorizontalRule{}"
3092 end
```

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

```
3093 local escaped_uri_chars = {
3094 [{"{"} = "\\markdownRendererLeftBrace{}",
3095 [{"}"} = "\\markdownRendererRightBrace{}",
3096 [{"%"} = "\\markdownRendererPercentSign{}",
3097 [{"\\"}] = "\\markdownRendererBackslash{}",
3098 }
3099 local escaped_citation_chars = {
3100 [{"{"} = "\\markdownRendererLeftBrace{}",
3101 [{"}"} = "\\markdownRendererRightBrace{}",
3102 [{"%"} = "\\markdownRendererPercentSign{}",
3103 [{"\\"}] = "\\markdownRendererBackslash{}",
3104 [{"#"} = "\\markdownRendererHash{}",
```

```

3105 }
3106 local escaped_minimal_strings = {
3107 ["^"] = "\\markdownRendererCircumflex\\markdownRendererCircumflex ",
3108 ["☒"] = "\\markdownRendererTickedBox{}",
3109 ["☐"] = "\\markdownRendererHalfTickedBox{}",
3110 ["□"] = "\\markdownRendererUntickedBox{}",
3111 }

```

Define a table `escaped_chars` containing the mapping from special plain T<sub>E</sub>X characters (including the active pipe character (|) of ConT<sub>E</sub>Xt) that need to be escaped for typeset content.

```

3112 local escaped_chars = {
3113 "{" = "\\markdownRendererLeftBrace{}",
3114 "}" = "\\markdownRendererRightBrace{}",
3115 "%" = "\\markdownRendererPercentSign{}",
3116 "\\" = "\\markdownRendererBackslash{}",
3117 "#" = "\\markdownRendererHash{}",
3118 "$" = "\\markdownRendererDollarSign{}",
3119 "&" = "\\markdownRendererAmpersand{}",
3120 "_" = "\\markdownRendererUnderscore{}",
3121 "^" = "\\markdownRendererCircumflex{}",
3122 "~" = "\\markdownRendererTilde{}",
3123 "|" = "\\markdownRendererPipe{}",
3124 }

```

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

```

3125 local escape = util.escaper(escaped_chars, escaped_minimal_strings)
3126 local escape_citation = util.escaper(escaped_citation_chars,
3127 escaped_minimal_strings)
3128 local escape_uri = util.escaper(escaped_uri_chars, escaped_minimal_strings)
3129 local escape_minimal = util.escaper({}, escaped_minimal_strings)

```

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

```

3130 if options.hybrid then
3131 self.string = escape_minimal
3132 self.citation = escape_minimal
3133 self.uri = escape_minimal
3134 else
3135 self.string = escape
3136 self.citation = escape_citation

```

```

3137 self.uri = escape_uri
3138 end

```

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

```

3139 self.escape = escape

```

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

```

3140 function self.code(s)
3141 return {"\\markdownRendererCodeSpan{",self.escape(s),"}"}
3142 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.

```

3143 function self.link(lab,src,tit)
3144 return {"\\markdownRendererLink{",lab,"}",
3145 "{",self.escape(src),"}",
3146 "{",self.uri(src),"}",
3147 "{",self.string(tit or ""),"}"}
3148 end

```

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

```

3149 function self.table(rows, caption)
3150 if not self.is_writing then return "" end
3151 local buffer = {"\\markdownRendererTable{",
3152 caption or "", "}{" , #rows - 1, "}{" , #rows[1], "}"}
3153 local temp = rows[2] -- put alignments on the first row
3154 rows[2] = rows[1]
3155 rows[1] = temp
3156 for i, row in ipairs(rows) do
3157 table.insert(buffer, "{")
3158 for _, column in ipairs(row) do
3159 if i > 1 then -- do not use braces for alignments
3160 table.insert(buffer, "{")
3161 end
3162 table.insert(buffer, column)
3163 if i > 1 then
3164 table.insert(buffer, "}")
3165 end
3166 end
3167 table.insert(buffer, "}")
3168 end
3169 return buffer

```



```
3170 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.

```
3171 function self.image(lab,src,tit)
3172 return {"\\markdownRendererImage{" ,lab,"} ",
3173 "{" ,self.string(src),"} ",
3174 "{" ,self.uri(src),"} ",
3175 "{" ,self.string(tit or ""),"}"}
3176 end
```

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

```
3177 local languages_json = (function()
3178 local ran_ok, kpse = pcall(require, "kpse")
3179 if ran_ok then
3180 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:

```
3181 else
3182 kpse = {lookup=function(filename, options) return filename end}
3183 end
3184 local base, prev, curr
3185 for _, filename in ipairs{kpse.lookup(options.contentBlocksLanguageMap,
3186 { all=true })} do
3187 local file = io.open(filename, "r")
3188 if not file then goto continue end
3189 json = file:read("*all"):gsub('("[^\n]-"):','[%1]=')
3190 curr = (function()
3191 local _ENV={ json=json, load=load } -- run in sandbox
3192 return load("return "..json)()
3193 end)()
3194 if type(curr) == "table" then
3195 if base == nil then
3196 base = curr
3197 else
3198 setmetatable(prev, { __index = curr })
3199 end
3200 prev = curr
3201 end
3202 ::continue::
3203 end
```

```

3204 return base or {}
3205 end()

```

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

```

3206 function self.contentblock(src,suf,type,tit)
3207 if not self.is_writing then return "" end
3208 src = src..".."..suf
3209 suf = suf:lower()
3210 if type == "onlineimage" then
3211 return {"\\markdownRendererContentBlockOnlineImage{" ,suf,"} ",
3212 "{" ,self.string(src),"} ",
3213 "{" ,self.uri(src),"} ",
3214 "{" ,self.string(tit or ""),"} "}
3215 elseif languages_json[suf] then
3216 return {"\\markdownRendererContentBlockCode{" ,suf,"} ",
3217 "{" ,self.string(languages_json[suf]),"} ",
3218 "{" ,self.string(src),"} ",
3219 "{" ,self.uri(src),"} ",
3220 "{" ,self.string(tit or ""),"} "}
3221 else
3222 return {"\\markdownRendererContentBlock{" ,suf,"} ",
3223 "{" ,self.string(src),"} ",
3224 "{" ,self.uri(src),"} ",
3225 "{" ,self.string(tit or ""),"} "}
3226 end
3227 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.

```

3228 local function ulitem(s)
3229 return {"\\markdownRendererUlItem " ,s,
3230 "\\markdownRendererUlItemEnd "}
3231 end
3232
3233 function self.bulletlist(items,tight)
3234 if not self.is_writing then return "" end
3235 local buffer = {}
3236 for _,item in ipairs(items) do
3237 buffer[#buffer + 1] = ulitem(item)
3238 end
3239 local contents = util.intersperse(buffer,"\\n")
3240 if tight and options.tightLists then
3241 return {"\\markdownRendererUlBeginTight\\n" ,contents,

```

```

3242 "\n\\markdownRendererUlEndTight "}
3243 else
3244 return {"\\markdownRendererUlBegin\n",contents,
3245 "\n\\markdownRendererUlEnd "}
3246 end
3247 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.

```

3248 local function olitem(s,num)
3249 if num ~= nil then
3250 return {"\\markdownRendererOlItemWithNumber{" ,num,"} ",s,
3251 "\\markdownRendererOlItemEnd "}
3252 else
3253 return {"\\markdownRendererOlItem ",s,
3254 "\\markdownRendererOlItemEnd "}
3255 end
3256 end
3257
3258 function self.orderedlist(items,tight,startnum)
3259 if not self.is_writing then return "" end
3260 local buffer = {}
3261 local num = startnum
3262 for _,item in ipairs(items) do
3263 buffer[#buffer + 1] = olitem(item,num)
3264 if num ~= nil then
3265 num = num + 1
3266 end
3267 end
3268 local contents = util.intersperse(buffer,"\n")
3269 if tight and options.tightLists then
3270 return {"\\markdownRendererOlBeginTight\n",contents,
3271 "\n\\markdownRendererOlEndTight "}
3272 else
3273 return {"\\markdownRendererOlBegin\n",contents,
3274 "\n\\markdownRendererOlEnd "}
3275 end
3276 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.

```

3277 function self.inline_html_comment(contents)
3278 return {"\\markdownRendererInlineHtmlComment{" ,contents,"}"}
3279 end

```

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

```

3280 local function dlitem(term, defs)
3281 local retVal = {"\\markdownRendererDlItem{" ,term,"}"}
3282 for _, def in ipairs(defs) do
3283 retVal[#retVal+1] = {"\\markdownRendererDlDefinitionBegin " ,def,
3284 "\\markdownRendererDlDefinitionEnd "}
3285 end
3286 retVal[#retVal+1] = "\\markdownRendererDlItemEnd "
3287 return retVal
3288 end
3289
3290 function self.definitionlist(items,tight)
3291 if not self.is_writing then return "" end
3292 local buffer = {}
3293 for _,item in ipairs(items) do
3294 buffer[#buffer + 1] = dlitem(item.term, item.definitions)
3295 end
3296 if tight and options.tightLists then
3297 return {"\\markdownRendererDlBeginTight\\n", buffer,
3298 "\\n\\markdownRendererDlEndTight"}
3299 else
3300 return {"\\markdownRendererDlBegin\\n", buffer,
3301 "\\n\\markdownRendererDlEnd"}
3302 end
3303 end

```

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

```

3304 function self.emphasis(s)
3305 return {"\\markdownRendererEmphasis{" ,s,"}"}
3306 end

```

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

```

3307 function self.tickbox(f)
3308 if f == 1.0 then
3309 return "☒ "
3310 elseif f == 0.0 then
3311 return "☐ "
3312 else
3313 return "◻ "
3314 end
3315 end

```

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

```
3316 function self.strong(s)
3317 return {"\\markdownRendererStrongEmphasis{" ,s,"}"}
3318 end
```

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

```
3319 function self.blockquote(s)
3320 if #util.ropetostring(s) == 0 then return "" end
3321 return {"\\markdownRendererBlockQuoteBegin\\n",s,
3322 "\\n\\markdownRendererBlockQuoteEnd "}
3323 end
```

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

```
3324 function self.verbatim(s)
3325 if not self.is_writing then return "" end
3326 s = string.gsub(s, '\\r\\n%s]*$', '')
3327 local name = util.cache(options.cacheDir, s, nil, nil, ".verbatim")
3328 return {"\\markdownRendererInputVerbatim{" ,name,"}"}
3329 end
```

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

```
3330 function self.fencedCode(i, s)
3331 if not self.is_writing then return "" end
3332 s = string.gsub(s, '\\r\\n%s]*$', '')
3333 local name = util.cache(options.cacheDir, s, nil, nil, ".verbatim")
3334 return {"\\markdownRendererInputFencedCode{" ,name,"}{",i,"}"}
3335 end
```

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

```
3336 function self.jekyllData(d, t, p)
3337 if not self.is_writing then return "" end
3338
3339 local buf = {}
3340
3341 local keys = {}
3342 for k, _ in pairs(d) do
3343 table.insert(keys, k)
3344 end
3345 table.sort(keys)
```

```

3346
3347 if not p then
3348 table.insert(buf, "\\markdownRendererJekyllDataBegin")
3349 end
3350
3351 if #d > 0 then
3352 table.insert(buf, "\\markdownRendererJekyllDataSequenceBegin{")
3353 table.insert(buf, self.uri(p or "null"))
3354 table.insert(buf, "}{")
3355 table.insert(buf, #keys)
3356 table.insert(buf, "}")
3357 else
3358 table.insert(buf, "\\markdownRendererJekyllDataMappingBegin{")
3359 table.insert(buf, self.uri(p or "null"))
3360 table.insert(buf, "}{")
3361 table.insert(buf, #keys)
3362 table.insert(buf, "}")
3363 end
3364
3365 for _, k in ipairs(keys) do
3366 local v = d[k]
3367 local typ = type(v)
3368 k = tostring(k or "null")
3369 if typ == "table" and next(v) ~= nil then
3370 table.insert(
3371 buf,
3372 self.jekyllData(v, t, k)
3373)
3374 else
3375 k = self.uri(k)
3376 v = tostring(v)
3377 if typ == "boolean" then
3378 table.insert(buf, "\\markdownRendererJekyllDataBoolean{")
3379 table.insert(buf, k)
3380 table.insert(buf, "}{")
3381 table.insert(buf, v)
3382 table.insert(buf, "}")
3383 elseif typ == "number" then
3384 table.insert(buf, "\\markdownRendererJekyllDataNumber{")
3385 table.insert(buf, k)
3386 table.insert(buf, "}{")
3387 table.insert(buf, v)
3388 table.insert(buf, "}")
3389 elseif typ == "string" then
3390 table.insert(buf, "\\markdownRendererJekyllDataString{")
3391 table.insert(buf, k)
3392 table.insert(buf, "}{")

```

```

3393 table.insert(buf, t(v))
3394 table.insert(buf, "}")
3395 elseif typ == "table" then
3396 table.insert(buf, "\\markdownRendererJekyllDataEmpty{")
3397 table.insert(buf, k)
3398 table.insert(buf, "}")
3399 else
3400 error(format("Unexpected type %s for value of " ..
3401 "YAML key %s", typ, k))
3402 end
3403 end
3404 end
3405
3406 if #d > 0 then
3407 table.insert(buf, "\\markdownRendererJekyllDataSequenceEnd")
3408 else
3409 table.insert(buf, "\\markdownRendererJekyllDataMappingEnd")
3410 end
3411
3412 if not p then
3413 table.insert(buf, "\\markdownRendererJekyllDataEnd")
3414 end
3415
3416 return buf
3417 end

```

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

```

3418 self.active_headings = {}

```

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

```

3419 function self.heading(s,level,attributes)
3420 local active_headings = self.active_headings
3421 local slice_begin_type = self.slice_begin:sub(1, 1)
3422 local slice_begin_identifier = self.slice_begin:sub(2) or ""
3423 local slice_end_type = self.slice_end:sub(1, 1)
3424 local slice_end_identifier = self.slice_end:sub(2) or ""
3425
3426 while #active_headings < level do
3427 -- push empty identifiers for implied sections
3428 table.insert(active_headings, {})
3429 end
3430
3431 while #active_headings >= level do
3432 -- pop identifiers for sections that have ended
3433 local active_identifiers = active_headings[#active_headings]
3434 if active_identifiers[slice_begin_identifier] ~= nil

```

```

3435 and slice_begin_type == "$" then
3436 self.is_writing = true
3437 end
3438 if active_identifiers[slice_end_identifier] ~= nil
3439 and slice_end_type == "$" then
3440 self.is_writing = false
3441 end
3442 table.remove(active_headings, #active_headings)
3443 end
3444
3445 -- push identifiers for the new section
3446 attributes = attributes or {}
3447 local identifiers = {}
3448 for index = 1, #attributes do
3449 attribute = attributes[index]
3450 identifiers[attribute:sub(2)] = true
3451 end
3452 if identifiers[slice_begin_identifier] ~= nil
3453 and slice_begin_type == "^" then
3454 self.is_writing = true
3455 end
3456 if identifiers[slice_end_identifier] ~= nil
3457 and slice_end_type == "^" then
3458 self.is_writing = false
3459 end
3460 table.insert(active_headings, identifiers)
3461
3462 if not self.is_writing then return "" end
3463
3464 local cmd
3465 level = level + options.shiftHeadings
3466 if level <= 1 then
3467 cmd = "\\markdownRendererHeadingOne"
3468 elseif level == 2 then
3469 cmd = "\\markdownRendererHeadingTwo"
3470 elseif level == 3 then
3471 cmd = "\\markdownRendererHeadingThree"
3472 elseif level == 4 then
3473 cmd = "\\markdownRendererHeadingFour"
3474 elseif level == 5 then
3475 cmd = "\\markdownRendererHeadingFive"
3476 elseif level >= 6 then
3477 cmd = "\\markdownRendererHeadingSix"
3478 else
3479 cmd = ""
3480 end
3481 return {cmd, "{" , s, " "}

```



```
3482 end
```

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

```
3483 function self.note(s)
3484 return {"\\markdownRendererFootnote{",s,""}
3485 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.

```
3486 function self.citations(text_cites, cites)
3487 local buffer = {"\\markdownRenderer", text_cites and "TextCite" or "Cite",
3488 "{", #cites, "}"}
3489 for _,cite in ipairs(cites) do
3490 buffer[#buffer+1] = {cite.suppress_author and "-" or "+", "{",
3491 cite.prenote or "", "}{" , cite.postnote or "", "}{" , cite.name, "}"}
3492 end
3493 return buffer
3494 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.

```
3495 function self.get_state()
3496 return {
3497 is_writing=self.is_writing,
3498 active_headings={table.unpack(self.active_headings)},
3499 }
3500 end
```

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

```
3501 function self.set_state(s)
3502 local previous_state = self.get_state()
3503 for key, value in pairs(s) do
```

```

3504 self[key] = value
3505 end
3506 return previous_state
3507 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`.

```

3508 function self.defer_call(f)
3509 local previous_state = self.get_state()
3510 return function(...)
3511 local state = self.set_state(previous_state)
3512 local return_value = f(...)
3513 self.set_state(state)
3514 return return_value
3515 end
3516 end
3517
3518 return self
3519 end

```

### 3.1.4 Parsers

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

```

3520 local parsers = {}

```

#### 3.1.4.1 Basic Parsers

```

3521 parsers.percent = P("%")
3522 parsers.at = P("@")
3523 parsers.comma = P(",")
3524 parsers.asterisk = P("*")
3525 parsers.dash = P("-")
3526 parsers.plus = P("+")
3527 parsers.underscore = P("_")
3528 parsers.period = P(".")
3529 parsers.hash = P("#")
3530 parsers.ampersand = P("&")
3531 parsers.backtick = P("`")
3532 parsers.less = P("<")
3533 parsers.more = P(">")
3534 parsers.space = P(" ")
3535 parsers.squote = P("'")
3536 parsers.dquote = P('"')
3537 parsers.lparent = P("(")
3538 parsers.rparent = P(")")

```

```

3539 parsers.lbracket = P("[")
3540 parsers.rbracket = P("]")
3541 parsers.lbrace = P("{")
3542 parsers.rbrace = P("}")
3543 parsers.circumflex = P("^")
3544 parsers.slash = P("/")
3545 parsers.equal = P("=")
3546 parsers.colon = P(":")
3547 parsers.semicolon = P(";")
3548 parsers.exclamation = P("!")
3549 parsers.pipe = P("|")
3550 parsers.tilde = P("~")
3551 parsers.backslash = P("\\")
3552 parsers.tab = P("\t")
3553 parsers.newline = P("\n")
3554 parsers.tightblocksep = P("\001")
3555
3556 parsers.digit = R("09")
3557 parsers.hexdigit = R("09", "af", "AF")
3558 parsers.letter = R("AZ", "az")
3559 parsers.alphanumeric = R("AZ", "az", "09")
3560 parsers.keyword = parsers.letter
3561 * parsers.alphanumeric^0
3562 parsers.citation_chars = parsers.alphanumeric
3563 + S("#$%&-+<>~/_")
3564 parsers.internal_punctuation = S(";, .?")
3565
3566 parsers.doubleasterisks = P("**")
3567 parsers.doubleunderscores = P("__")
3568 parsers.fourspace = P(" ")
3569
3570 parsers.any = P(1)
3571 parsers.fail = parsers.any - 1
3572
3573 parsers.escapable = S("*_{}[]()+_!<>#-~:~@;")
3574 parsers.anyescaped = parsers.backslash / " " * parsers.escapable
3575 + parsers.any
3576
3577 parsers.spacechar = S("\t ")
3578 parsers.spacing = S(" \n\r\t")
3579 parsers.nonspacechar = parsers.any - parsers.spacing
3580 parsers.optionalspace = parsers.spacechar^0
3581
3582 parsers.specialchar = S("*_`&[]<!\\\.@-~")
3583
3584 parsers.normalchar = parsers.any - (parsers.specialchar
3585 + parsers.spacing)

```

```

3586 + parsers.tightblocksep)
3587 parsers.eof = -parsers.any
3588 parsers.nonindentspace = parsers.space-3 * - parsers.spacechar
3589 parsers.indent = parsers.space-3 * parsers.tab
3590 + parsers.fourspaces / ""
3591 parsers.linechar = P(1 - parsers.newline)
3592
3593 parsers.blankline = parsers.optionalspace
3594 * parsers.newline / "\n"
3595 parsers.blanklines = parsers.blankline0
3596 parsers.skipblanklines = (parsers.optionalspace * parsers.newline)0
3597 parsers.indentedline = parsers.indent / ""
3598 * C(parsers.linechar1 * parsers.newline-
1)
3599 parsers.optionallyindentedline = parsers.indent-1 / ""
3600 * C(parsers.linechar1 * parsers.newline-
1)
3601 parsers.sp = parsers.spacing0
3602 parsers.spnl = parsers.optionalspace
3603 * (parsers.newline * parsers.optionalspace)-
1
3604 parsers.line = parsers.linechar0 * parsers.newline
3605 parsers.nonemptyline = parsers.line - parsers.blankline

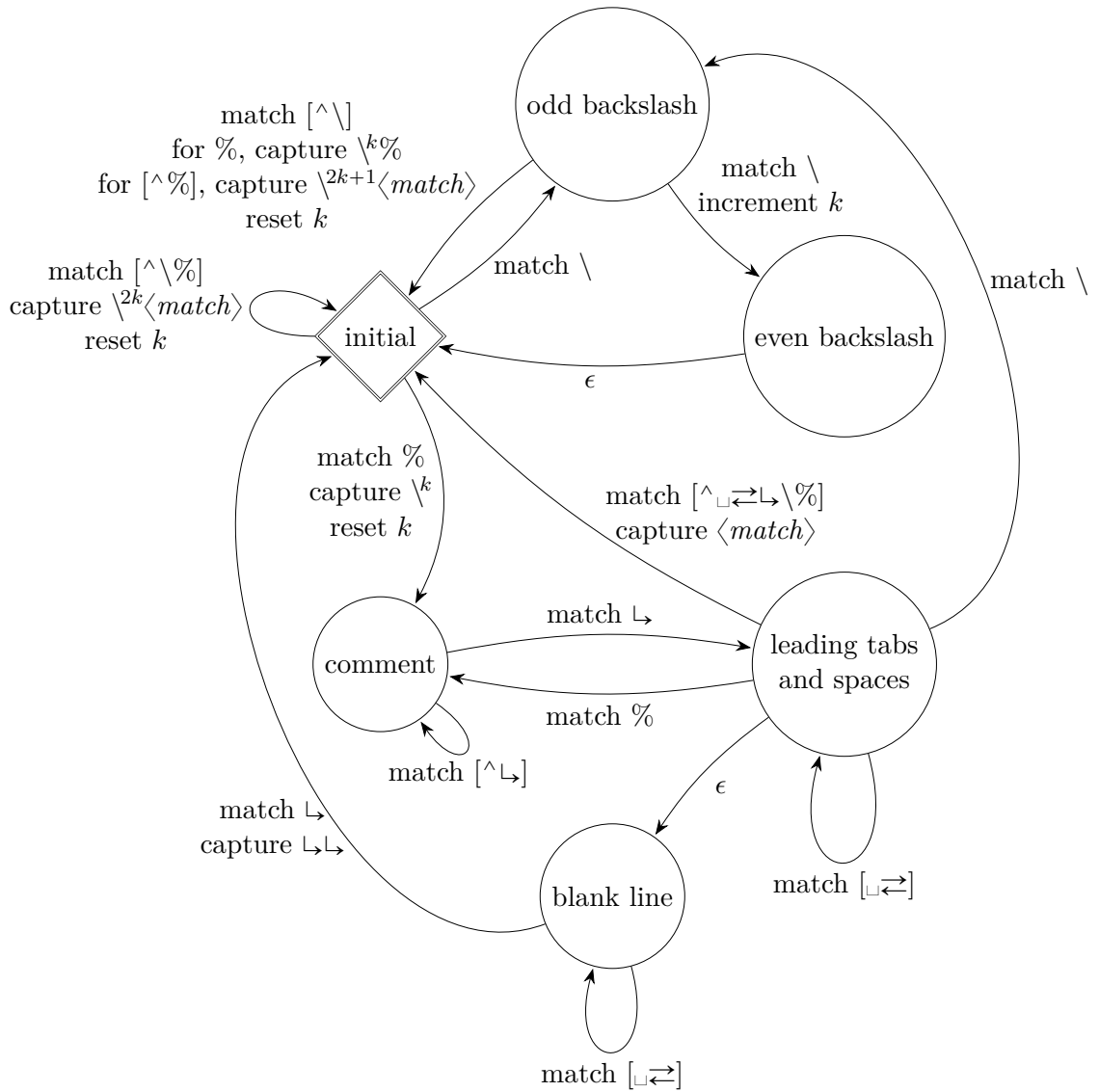
```

The `parsers.commented_line1` parser recognizes the regular language of T<sub>E</sub>X comments, see an equivalent finite automaton in Figure 6 on the next page.

```

3606 parsers.commented_line_letter = parsers.linechar
3607 + parsers.newline
3608 - parsers.backslash
3609 - parsers.percent
3610 parsers.commented_line = Cg(Cc(""), "backslashes")
3611 * ((#(parsers.commented_line_letter
3612 - parsers.newline)
3613 * Cb("backslashes")
3614 * Cs(parsers.commented_line_letter
3615 - parsers.newline)1 -- initial
3616 * Cg(Cc(""), "backslashes"))
3617 + #(parsers.backslash * parsers.backslash)
3618 * Cg((parsers.backslash -- even backslash
3619 * parsers.backslash)1, "backslashes")
3620 + (parsers.backslash
3621 * (#parsers.percent
3622 * Cb("backslashes")
3623 / function(backslashes)
3624 return string.rep("\\", #backslashes / 2)
3625 end
3626 * C(parsers.percent)

```



**Figure 6: A pushdown automaton that recognizes TeX comments**

```

3627 + #parsers.commented_line_letter
3628 * Cb("backslashes")
3629 * Cc("\\")
3630 * C(parsers.commented_line_letter))
3631 * Cg(Cc(""), "backslashes"))^0
3632 * (#parsers.percent
3633 * Cb("backslashes")
3634 / function(backslashes)
3635 return string.rep("\\", #backslashes / 2)
3636 end
3637 * ((parsers.percent -- comment
3638 * parsers.line
3639 * #parsers.blankline) -- blank line
3640 / "\n"
3641 + parsers.percent -- comment
3642 * parsers.line
3643 * parsers.optionalspace) -- leading tabs and spaces
3644 + #(parsers.newline)
3645 * Cb("backslashes")
3646 * C(parsers.newline))
3647
3648 parsers.chunk = parsers.line * (parsers.optionallyindentedline
3649 - parsers.blankline)^0
3650
3651 parsers.css_identifier_char = R("AZ", "az", "09") + S("-_")
3652 parsers.css_identifier = (parsers.hash + parsers.period)
3653 * (((parsers.css_identifier_char
3654 - parsers.dash - parsers.digit)
3655 * parsers.css_identifier_char^1)
3656 + (parsers.dash
3657 * (parsers.css_identifier_char
3658 - parsers.digit)
3659 * parsers.css_identifier_char^0))
3660 parsers.attribute_name_char = parsers.any - parsers.space
3661 - parsers.squote - parsers.dquote
3662 - parsers.more - parsers.slash
3663 - parsers.equal
3664 parsers.attribute_value_char = parsers.any - parsers.dquote
3665 - parsers.more
3666
3667 -- block followed by 0 or more optionally
3668 -- indented blocks with first line indented.
3669 parsers.indented_blocks = function(bl)
3670 return Cs(bl
3671 * (parsers.blankline^1 * parsers.indent * -parsers.blankline * bl)^0
3672 * (parsers.blankline^1 + parsers.eof))
3673 end

```

### 3.1.4.2 Parsers Used for Markdown Lists

```
3674 parsers.bulletchar = C(parsers.plus + parsers.asterisk + parsers.dash)
3675
3676 parsers.bullet = (parsers.bulletchar * #parsers.spacing
3677 * (parsers.tab + parsers.space^-3)
3678 + parsers.space * parsers.bulletchar * #parsers.spacing
3679 * (parsers.tab + parsers.space^-2)
3680 + parsers.space * parsers.space * parsers.bulletchar
3681 * #parsers.spacing
3682 * (parsers.tab + parsers.space^-1)
3683 + parsers.space * parsers.space * parsers.space
3684 * parsers.bulletchar * #parsers.spacing
3685)
3686
3687 local function tickbox(interior)
3688 return parsers.optionalspace * parsers.lbracket
3689 * interior * parsers.rbracket * parsers.spacechar^1
3690 end
3691
3692 parsers.ticked_box = tickbox(S("xX")) * Cc(1.0)
3693 parsers.halfticked_box = tickbox(S("./")) * Cc(0.5)
3694 parsers.unticked_box = tickbox(parsers.spacechar^1) * Cc(0.0)
3695
```

### 3.1.4.3 Parsers Used for Markdown Code Spans

```
3696 parsers.openticks = Cg(parsers.backtick^1, "ticks")
3697
3698 local function captures_equal_length(s,i,a,b)
3699 return #a == #b and i
3700 end
3701
3702 parsers.closeticks = parsers.space^-1
3703 * Cmt(C(parsers.backtick^1)
3704 * Cb("ticks"), captures_equal_length)
3705
3706 parsers.intickschar = (parsers.any - S("\n\r`"))
3707 + (parsers.newline * -parsers.blankline)
3708 + (parsers.space - parsers.closeticks)
3709 + (parsers.backtick^1 - parsers.closeticks)
3710
3711 parsers.inticks = parsers.openticks * parsers.space^-1
3712 * C(parsers.intickschar^0) * parsers.closeticks
```

### 3.1.4.4 Parsers Used for Fenced Code Blocks

```
3713 local function captures_geq_length(s,i,a,b)
3714 return #a >= #b and i
```

```

3715 end
3716
3717 parsers.infostring = (parsers.linechar - (parsers.backtick
3718 + parsers.space^1 * (parsers.newline + parsers.eof)))^0
3719
3720 local fenceindent
3721 parsers.fencehead = function(char)
3722 return C(parsers.nonindentspace) / function(s) fenceindent = #s end
3723 * Cg(char^3, "fencelength")
3724 * parsers.optionalspace * C(parsers.infostring)
3725 * parsers.optionalspace * (parsers.newline + parsers.eof)
3726 end
3727
3728 parsers.fencetail = function(char)
3729 return parsers.nonindentspace
3730 * Cmt(C(char^3) * Cb("fencelength"), captures_geq_length)
3731 * parsers.optionalspace * (parsers.newline + parsers.eof)
3732 + parsers.eof
3733 end
3734
3735 parsers.fencedline = function(char)
3736 return C(parsers.line - parsers.fencetail(char))
3737 / function(s)
3738 i = 1
3739 remaining = fenceindent
3740 while true do
3741 c = s:sub(i, i)
3742 if c == " " and remaining > 0 then
3743 remaining = remaining - 1
3744 i = i + 1
3745 elseif c == "\t" and remaining > 3 then
3746 remaining = remaining - 4
3747 i = i + 1
3748 else
3749 break
3750 end
3751 end
3752 return s:sub(i)
3753 end
3754 end

```

### 3.1.4.5 Parsers Used for Markdown Tags and Links

```

3755 parsers.leader = parsers.space^-3
3756
3757 -- content in balanced brackets, parentheses, or quotes:
3758 parsers.bracketed = P{ parsers.lbracket

```



```

3759 * ((parsers.anyescaped - (parsers.lbracket
3760 + parsers.rbracket
3761 + parsers.blankline^2)
3762) + V(1))^0
3763 * parsers.rbracket }
3764
3765 parsers.inparens = P{ parsers.lparent
3766 * ((parsers.anyescaped - (parsers.lparent
3767 + parsers.rparent
3768 + parsers.blankline^2)
3769) + V(1))^0
3770 * parsers.rparent }
3771
3772 parsers.squoted = P{ parsers.squote * parsers.alphanumeric
3773 * ((parsers.anyescaped - (parsers.squote
3774 + parsers.blankline^2)
3775) + V(1))^0
3776 * parsers.squote }
3777
3778 parsers.dquoted = P{ parsers.dquote * parsers.alphanumeric
3779 * ((parsers.anyescaped - (parsers.dquote
3780 + parsers.blankline^2)
3781) + V(1))^0
3782 * parsers.dquote }
3783
3784 -- bracketed tag for markdown links, allowing nested brackets:
3785 parsers.tag = parsers.lbracket
3786 * Cs((parsers.alphanumeric^1
3787 + parsers.bracketed
3788 + parsers.inticks
3789 + (parsers.anyescaped
3790 - (parsers.rbracket + parsers.blankline^2)))^0)
3791 * parsers.rbracket
3792
3793 -- url for markdown links, allowing nested brackets:
3794 parsers.url = parsers.less * Cs((parsers.anyescaped
3795 - parsers.more)^0)
3796 * parsers.more
3797 + Cs((parsers.inparens + (parsers.anyescaped
3798 - parsers.spacing
3799 - parsers.rparent))^1)
3800
3801 -- quoted text, possibly with nested quotes:
3802 parsers.title_s = parsers.squote * Cs(((parsers.anyescaped-parsers.squote)
3803 + parsers.squoted)^0)
3804 * parsers.squote
3805

```

```

3806 parsers.title_d = parsers.dquote * Cs(((parsers.anyescaped-parsers.dquote)
3807 + parsers.dquoted)^0)
3808 * parsers.dquote
3809
3810 parsers.title_p = parsers.lparent
3811 * Cs((parsers.inparens + (parsers.anyescaped-parsers.rparent))^0)
3812 * parsers.rparent
3813
3814 parsers.title = parsers.title_d + parsers.title_s + parsers.title_p
3815
3816 parsers.optionaltitle
3817 = parsers.spnl * parsers.title * parsers.spacechar^0
3818 + Cc("")

```

### 3.1.4.6 Parsers Used for iA Writer Content Blocks

```

3819 parsers.contentblock_tail
3820 = parsers.optionaltitle
3821 * (parsers.newline + parsers.eof)
3822
3823 -- case insensitive online image suffix:
3824 parsers.onlineimagesuffix
3825 = (function(...)
3826 local parser = nil
3827 for _,suffix in ipairs({...}) do
3828 local pattern=nil
3829 for i=1,#suffix do
3830 local char=suffix:sub(i,i)
3831 char = S(char:lower()..char:upper())
3832 if pattern == nil then
3833 pattern = char
3834 else
3835 pattern = pattern * char
3836 end
3837 end
3838 if parser == nil then
3839 parser = pattern
3840 else
3841 parser = parser + pattern
3842 end
3843 end
3844 return parser
3845 end)("png", "jpg", "jpeg", "gif", "tif", "tiff")
3846
3847 -- online image url for iA Writer content blocks with mandatory suffix,
3848 -- allowing nested brackets:
3849 parsers.onlineimageurl

```

```

3850 = (parsers.less
3851 * Cs((parsers.anyescaped
3852 - parsers.more
3853 - #(parsers.period
3854 * parsers.onlineimagesuffix
3855 * parsers.more
3856 * parsers.contentblock_tail))^0)
3857 * parsers.period
3858 * Cs(parsers.onlineimagesuffix)
3859 * parsers.more
3860 + (Cs((parsers.inparens
3861 + (parsers.anyescaped
3862 - parsers.spacing
3863 - parsers.rparent
3864 - #(parsers.period
3865 * parsers.onlineimagesuffix
3866 * parsers.contentblock_tail)))^0)
3867 * parsers.period
3868 * Cs(parsers.onlineimagesuffix))
3869) * Cc("onlineimage")
3870
3871 -- filename for iA Writer content blocks with mandatory suffix:
3872 parsers.localfilepath
3873 = parsers.slash
3874 * Cs((parsers.anyescaped
3875 - parsers.tab
3876 - parsers.newline
3877 - #(parsers.period
3878 * parsers.alphanumeric^1
3879 * parsers.contentblock_tail))^1)
3880 * parsers.period
3881 * Cs(parsers.alphanumeric^1)
3882 * Cc("localfile")

```

### 3.1.4.7 Parsers Used for Citations

```

3883 parsers.citation_name = Cs(parsers.dash^-1) * parsers.at
3884 * Cs(parsers.citation_chars
3885 * (((parsers.citation_chars + parsers.internal_punctuation
3886 - parsers.comma - parsers.semicolon)
3887 * -#((parsers.internal_punctuation - parsers.comma
3888 - parsers.semicolon)^0
3889 * -(parsers.citation_chars + parsers.internal_punctuat
3890 - parsers.comma - parsers.semicolon)))^0
3891 * parsers.citation_chars)^-1)
3892
3893 parsers.citation_body_prenote

```

```

3894 = Cs((parsers.alphanumeric^1
3895 + parsers.bracketed
3896 + parsers.inticks
3897 + (parsers.anyescaped
3898 - (parsers.rbracket + parsers.blankline^2))
3899 - (parsers.spnl * parsers.dash^-1 * parsers.at))^0)
3900
3901 parsers.citation_body_postnote
3902 = Cs((parsers.alphanumeric^1
3903 + parsers.bracketed
3904 + parsers.inticks
3905 + (parsers.anyescaped
3906 - (parsers.rbracket + parsers.semicolon
3907 + parsers.blankline^2))
3908 - (parsers.spnl * parsers.rbracket))^0)
3909
3910 parsers.citation_body_chunk
3911 = parsers.citation_body_prenote
3912 * parsers.spnl * parsers.citation_name
3913 * (parsers.internal_punctuation - parsers.semicolon)^-
3914 1
3915 * parsers.spnl * parsers.citation_body_postnote
3916
3917 parsers.citation_body
3918 = parsers.citation_body_chunk
3919 * (parsers.semicolon * parsers.spnl
3920 * parsers.citation_body_chunk)^0
3921
3922 parsers.citation_headless_body_postnote
3923 = Cs((parsers.alphanumeric^1
3924 + parsers.bracketed
3925 + parsers.inticks
3926 + (parsers.anyescaped
3927 - (parsers.rbracket + parsers.at
3928 + parsers.semicolon + parsers.blankline^2))
3929 - (parsers.spnl * parsers.rbracket))^0)
3930
3931 parsers.citation_headless_body
3932 = parsers.citation_headless_body_postnote
3933 * (parsers.sp * parsers.semicolon * parsers.spnl
3934 * parsers.citation_body_chunk)^0

```

### 3.1.4.8 Parsers Used for Footnotes

```

3934 local function strip_first_char(s)
3935 return s:sub(2)
3936 end

```

```

3937
3938 parsers.RawNoteRef = #(parsers.lbracket * parsers.circumflex)
3939 * parsers.tag / strip_first_char

```

### 3.1.4.9 Parsers Used for Tables

```

3940 local function make_pipe_table_rectangular(rows)
3941 local num_columns = #rows[2]
3942 local rectangular_rows = {}
3943 for i = 1, #rows do
3944 local row = rows[i]
3945 local rectangular_row = {}
3946 for j = 1, num_columns do
3947 rectangular_row[j] = row[j] or ""
3948 end
3949 table.insert(rectangular_rows, rectangular_row)
3950 end
3951 return rectangular_rows
3952 end
3953
3954 local function pipe_table_row(allow_empty_first_column
3955 , nonempty_column
3956 , column_separator
3957 , column)
3958 local row_beginning
3959 if allow_empty_first_column then
3960 row_beginning = -- empty first column
3961 #(parsers.spacechar^4
3962 * column_separator)
3963 * parsers.optionalspace
3964 * column
3965 * parsers.optionalspace
3966 -- non-empty first column
3967 + parsers.nonindentspace
3968 * nonempty_column^-1
3969 * parsers.optionalspace
3970 else
3971 row_beginning = parsers.nonindentspace
3972 * nonempty_column^-1
3973 * parsers.optionalspace
3974 end
3975
3976 return Ct(row_beginning
3977 * (-- single column with no leading pipes
3978 #(column_separator
3979 * parsers.optionalspace
3980 * parsers.newline)

```

```

3981 * column_separator
3982 * parsers.optionalspace
3983 -- single column with leading pipes or
3984 -- more than a single column
3985 + (column_separator
3986 * parsers.optionalspace
3987 * column
3988 * parsers.optionalspace)^1
3989 * (column_separator
3990 * parsers.optionalspace)^-1))
3991 end
3992
3993 parsers.table_hline_separator = parsers.pipe + parsers.plus
3994 parsers.table_hline_column = (parsers.dash
3995 - #(parsers.dash
3996 * (parsers.spacechar
3997 + parsers.table_hline_separator
3998 + parsers.newline)))^1
3999 * (parsers.colon * Cc("r")
4000 + parsers.dash * Cc("d"))
4001 + parsers.colon
4002 * (parsers.dash
4003 - #(parsers.dash
4004 * (parsers.spacechar
4005 + parsers.table_hline_separator
4006 + parsers.newline)))^1
4007 * (parsers.colon * Cc("c")
4008 + parsers.dash * Cc("l"))
4009 parsers.table_hline = pipe_table_row(false
4010 , parsers.table_hline_column
4011 , parsers.table_hline_separator
4012 , parsers.table_hline_column)
4013 parsers.table_caption_beginning = parsers.skipblanklines
4014 * parsers.nonindentospace
4015 * (P("Table")^-1 * parsers.colon)
4016 * parsers.optionalspace

```

### 3.1.4.10 Parsers Used for HTML

```

4017 -- case-insensitive match (we assume s is lowercase). must be single byte encoding
4018 parsers.keyword_exact = function(s)
4019 local parser = P(0)
4020 for i=1,#s do
4021 local c = s:sub(i,i)
4022 local m = c .. upper(c)
4023 parser = parser * S(m)
4024 end

```

```

4025 return parser
4026 end
4027
4028 parsers.block_keyword =
4029 parsers.keyword_exact("address") + parsers.keyword_exact("blockquote") +
4030 parsers.keyword_exact("center") + parsers.keyword_exact("del") +
4031 parsers.keyword_exact("dir") + parsers.keyword_exact("div") +
4032 parsers.keyword_exact("p") + parsers.keyword_exact("pre") +
4033 parsers.keyword_exact("li") + parsers.keyword_exact("ol") +
4034 parsers.keyword_exact("ul") + parsers.keyword_exact("dl") +
4035 parsers.keyword_exact("dd") + parsers.keyword_exact("form") +
4036 parsers.keyword_exact("fieldset") + parsers.keyword_exact("isindex") +
4037 parsers.keyword_exact("ins") + parsers.keyword_exact("menu") +
4038 parsers.keyword_exact("noframes") + parsers.keyword_exact("frameset") +
4039 parsers.keyword_exact("h1") + parsers.keyword_exact("h2") +
4040 parsers.keyword_exact("h3") + parsers.keyword_exact("h4") +
4041 parsers.keyword_exact("h5") + parsers.keyword_exact("h6") +
4042 parsers.keyword_exact("hr") + parsers.keyword_exact("script") +
4043 parsers.keyword_exact("noscript") + parsers.keyword_exact("table") +
4044 parsers.keyword_exact("tbody") + parsers.keyword_exact("tfoot") +
4045 parsers.keyword_exact("thead") + parsers.keyword_exact("th") +
4046 parsers.keyword_exact("td") + parsers.keyword_exact("tr")
4047
4048 -- There is no reason to support bad html, so we expect quoted attributes
4049 parsers.htmlattributevalue
4050 = parsers.quote * (parsers.any - (parsers.blankline
4051 + parsers.quote))^0
4052 * parsers.quote
4053 + parsers.dquote * (parsers.any - (parsers.blankline
4054 + parsers.dquote))^0
4055 * parsers.dquote
4056
4057 parsers.htmlattribute = parsers.spacing^1
4058 * (parsers.alphanumeric + S("_-"))^1
4059 * parsers.sp * parsers.equal * parsers.sp
4060 * parsers.htmlattributevalue
4061
4062 parsers.htmlcomment = P("<!--")
4063 * parsers.optionalspace
4064 * Cs((parsers.any - parsers.optionalspace * P("-->"))^0)
4065 * parsers.optionalspace
4066 * P("-->")
4067
4068 parsers.htmlinstruction = P("<?") * (parsers.any - P(">"))^0 * P(">")
4069
4070 parsers.openelt_any = parsers.less * parsers.keyword * parsers.htmlattribute^0
4071 * parsers.sp * parsers.more

```

```

4072
4073 parsers.openelt_exact = function(s)
4074 return parsers.less * parsers.sp * parsers.keyword_exact(s)
4075 * parsers.htmlattribute^0 * parsers.sp * parsers.more
4076 end
4077
4078 parsers.openelt_block = parsers.sp * parsers.block_keyword
4079 * parsers.htmlattribute^0 * parsers.sp * parsers.more
4080
4081 parsers.closeelt_any = parsers.less * parsers.sp * parsers.slash
4082 * parsers.keyword * parsers.sp * parsers.more
4083
4084 parsers.closeelt_exact = function(s)
4085 return parsers.less * parsers.sp * parsers.slash * parsers.keyword_exact(s)
4086 * parsers.sp * parsers.more
4087 end
4088
4089 parsers.emptyelt_any = parsers.less * parsers.sp * parsers.keyword
4090 * parsers.htmlattribute^0 * parsers.sp * parsers.slash
4091 * parsers.more
4092
4093 parsers.emptyelt_block = parsers.less * parsers.sp * parsers.block_keyword
4094 * parsers.htmlattribute^0 * parsers.sp * parsers.slash
4095 * parsers.more
4096
4097 parsers.displaytext = (parsers.any - parsers.less)^1
4098
4099 -- return content between two matched HTML tags
4100 parsers.in_matched = function(s)
4101 return { parsers.openelt_exact(s)
4102 * (V(1) + parsers.displaytext
4103 + (parsers.less - parsers.closeelt_exact(s)))^0
4104 * parsers.closeelt_exact(s) }
4105 end
4106
4107 local function parse_matched_tags(s,pos)
4108 local t = string.lower(lpeg.match(C(parsers.keyword),s,pos))
4109 return lpeg.match(parsers.in_matched(t),s,pos-1)
4110 end
4111
4112 parsers.in_matched_block_tags = parsers.less
4113 * Cmt(#parsers.openelt_block, parse_matched_tags)
4114
4115 parsers.displayhtml = parsers.htmlcomment / ""
4116 + parsers.emptyelt_block
4117 + parsers.openelt_exact("hr")
4118 + parsers.in_matched_block_tags

```



```
4119 + parsers.htmlinstruction
```

### 3.1.4.11 Parsers Used for HTML Entities

```
4120 parsers.hexentity = parsers.ampersand * parsers.hash * S("Xx")
4121 * C(parsers.hexdigit^1) * parsers.semicolon
4122 parsers.decentity = parsers.ampersand * parsers.hash
4123 * C(parsers.digit^1) * parsers.semicolon
4124 parsers.tagentity = parsers.ampersand * C(parsers.alphanumeric^1)
4125 * parsers.semicolon
```

### 3.1.4.12 Helpers for References

```
4126 -- parse a reference definition: [foo]: /bar "title"
4127 parsers.define_reference_parser = parsers.leader * parsers.tag * parsers.colon
4128 * parsers.spacechar^0 * parsers.url
4129 * parsers.optionaltitle * parsers.blankline^1
```

### 3.1.4.13 Inline Elements

```
4130 parsers.Inline = V("Inline")
4131 parsers.IndentedInline = V("IndentedInline")
4132
4133 -- parse many p between starter and ender
4134 parsers.between = function(p, starter, ender)
4135 local ender2 = B(parsers.nonspacechar) * ender
4136 return (starter * #parsers.nonspacechar * Ct(p * (p - ender2)^0) * ender2)
4137 end
4138
4139 parsers.urlchar = parsers.anyescaped - parsers.newline - parsers.more
```

### 3.1.4.14 Block Elements

```
4140 parsers.Block = V("Block")
4141
4142 parsers.OfflineImageURL
4143 = parsers.leader
4144 * parsers.onlineimageurl
4145 * parsers.optionaltitle
4146
4147 parsers.LocalFilePath
4148 = parsers.leader
4149 * parsers.localfilepath
4150 * parsers.optionaltitle
4151
4152 parsers.TildeFencedCode
4153 = parsers.fencehead(parsers.tilde)
4154 * Cs(parsers.fencedline(parsers.tilde)^0)
4155 * parsers.fencetail(parsers.tilde)
```

```

4156
4157 parsers.BacktickFencedCode
4158 = parsers.fencehead(parsers.backtick)
4159 * Cs(parsers.fencedline(parsers.backtick)^0)
4160 * parsers.fencetail(parsers.backtick)
4161
4162 parsers.JekyllFencedCode
4163 = parsers.fencehead(parsers.dash)
4164 * Cs(parsers.fencedline(parsers.dash)^0)
4165 * parsers.fencetail(parsers.dash)
4166
4167 parsers.lineof = function(c)
4168 return (parsers.lead * (P(c) * parsers.optionalspace)^3
4169 * (parsers.newline * parsers.blankline^1
4170 + parsers.newline^-1 * parsers.eof))
4171 end

```

### 3.1.4.15 Lists

```

4172 parsers.defstartchar = S("~:")
4173 parsers.defstart = (parsers.defstartchar * #parsers.spacing
4174 * (parsers.tab + parsers.space^-
4175 3)
4176 + parsers.space * parsers.defstartchar * #parsers.spacing
4177 * (parsers.tab + parsers.space^-2)
4178 + parsers.space * parsers.space * parsers.defstartchar
4179 * #parsers.spacing
4180 * (parsers.tab + parsers.space^-1)
4181 + parsers.space * parsers.space * parsers.space
4182 * parsers.defstartchar * #parsers.spacing
4183)
4184 parsers.dlchunk = Cs(parsers.line * (parsers.indentedline - parsers.blankline)^0)

```

### 3.1.4.16 Headings

```

4185 parsers.heading_attribute = C(parsers.css_identifier)
4186 + C((parsers.attribute_name_char
4187 - parsers.rbrace)^1
4188 * parsers.equal
4189 * (parsers.attribute_value_char
4190 - parsers.rbrace)^1)
4191 parsers.HeadingAttributes = parsers.lbrace
4192 * parsers.heading_attribute
4193 * (parsers.spacechar^1
4194 * parsers.heading_attribute)^0
4195 * parsers.rbrace
4196

```

```

4197 -- parse Atx heading start and return level
4198 parsers.HeadingStart = #parsers.hash * C(parsers.hash^-6)
4199 * -parsers.hash / length
4200
4201 -- parse setext header ending and return level
4202 parsers.HeadingLevel = parsers.equal^1 * Cc(1) + parsers.dash^1 * Cc(2)
4203
4204 local function strip_atx_end(s)
4205 return s:gsub("#%s*\n$", "")
4206 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 on page 6), 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 on page 7) `options` and with a writer object `writer`. When `options` are unspecified, it is assumed that an empty table was passed to the method.

The objects produced by the `reader.new` method expose instance methods and variables of their own. As a convention, I will refer to these *member*s as `reader->member`.

```

4207 M.reader = {}
4208 function M.reader.new(writer, options)
4209 local self = {}
4210 options = options or {}

```

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

```

4211 setmetatable(options, { __index = function (_, key)
4212 return defaultOptions[key] end })

```

**3.1.5.1 Top-Level Helper Functions** Define `normalize_tag` as a function that normalizes a markdown reference tag by lowercasing it, and by collapsing any adjacent whitespace characters.

```

4213 local function normalize_tag(tag)
4214 return string.lower(
4215 gsub(util.ropetostring(tag), "[\n\r\t]+", " "))
4216 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.

```
4217 local function iterlines(s, f)
4218 rope = lpeg.match(Ct((parsers.line / f)^1), s)
4219 return util.rope_to_string(rope)
4220 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.

```
4221 local expandtabs
4222 if options.preserveTabs then
4223 expandtabs = function(s) return s end
4224 else
4225 expandtabs = function(s)
4226 if s:find("\t") then
4227 return iterlines(s, util.expand_tabs_in_line)
4228 else
4229 return s
4230 end
4231 end
4232 end
```

The `larsers` (as in ‘`local \luamref{parsers}''`) hash table stores `\acro{peg} patterns` tions’, which impedes their reuse between different `reader` objects.

```
4233 local larsers = {}
```

### 3.1.5.2 Top-Level Parser Functions

```
4234 local function create_parser(name, grammar, toplevel)
4235 return function(str)
```

If the parser is top-level and the `stripIndent` Lua option is enabled, we will first expand tabs in the input string `str` into spaces and then we will count the minimum indent across all lines, skipping blank lines. Next, we will remove the minimum indent from all lines.

```
4236 if toplevel and options.stripIndent then
4237 local min_prefix_length, min_prefix = nil, ''
4238 str = iterlines(str, function(line)
4239 if lpeg.match(parsers.nonemptyline, line) == nil then
4240 return line
4241 end
4242 line = util.expand_tabs_in_line(line)
4243 prefix = lpeg.match(C(parsers.optionalspace), line)
4244 local prefix_length = #prefix
4245 local is_shorter = min_prefix_length == nil
4246 is_shorter = is_shorter or prefix_length < min_prefix_length
4247 if is_shorter then
```

```

4248 min_prefix_length, min_prefix = prefix_length, prefix
4249 end
4250 return line
4251 end)
4252 str = str:gsub('^' .. min_prefix, '')
4253 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.

```

4254 if toplevel and (options.texComments or options.hybrid) then
4255 str = lpeg.match(Ct(parsers.commented_line^1), str)
4256 str = util.rope_to_string(str)
4257 end
4258 local res = lpeg.match(grammar(), str)
4259 if res == nil then
4260 error(format("%s failed on:\n%s", name, str:sub(1,20)))
4261 else
4262 return res
4263 end
4264 end
4265 end
4266
4267 local parse_blocks
4268 = create_parser("parse_blocks",
4269 function()
4270 return larsers.blocks
4271 end, false)
4272
4273 local parse_blocks_toplevel
4274 = create_parser("parse_blocks_toplevel",
4275 function()
4276 return larsers.blocks_toplevel
4277 end, true)
4278
4279 local parse_inlines
4280 = create_parser("parse_inlines",
4281 function()
4282 return larsers.inlines
4283 end, false)
4284
4285 local parse_inlines_no_link
4286 = create_parser("parse_inlines_no_link",
4287 function()
4288 return larsers.inlines_no_link
4289 end, false)
4290

```

```

4291 local parse_inlines_no_inline_note
4292 = create_parser("parse_inlines_no_inline_note",
4293 function()
4294 return larsers.inlines_no_inline_note
4295 end, false)
4296
4297 local parse_inlines_no_html
4298 = create_parser("parse_inlines_no_html",
4299 function()
4300 return larsers.inlines_no_html
4301 end, false)
4302
4303 local parse_inlines_nbsp
4304 = create_parser("parse_inlines_nbsp",
4305 function()
4306 return larsers.inlines_nbsp
4307 end, false)

```

### 3.1.5.3 Parsers Used for Markdown Lists (local)

```

4308 if options.hashEnumerators then
4309 larsers.dig = parsers.digit + parsers.hash
4310 else
4311 larsers.dig = parsers.digit
4312 end
4313
4314 larsers.enumerator = C(larsers.dig^3 * parsers.period) * #parsers.spacing
4315 + C(larsers.dig^2 * parsers.period) * #parsers.spacing
4316 * (parsers.tab + parsers.space^1)
4317 + C(larsers.dig * parsers.period) * #parsers.spacing
4318 * (parsers.tab + parsers.space^-2)
4319 + parsers.space * C(larsers.dig^2 * parsers.period)
4320 * #parsers.spacing
4321 + parsers.space * C(larsers.dig * parsers.period)
4322 * #parsers.spacing
4323 * (parsers.tab + parsers.space^-1)
4324 + parsers.space * parsers.space * C(larsers.dig^1
4325 * parsers.period) * #parsers.spacing

```

### 3.1.5.4 Parsers Used for Blockquotes (local)

```

4326 -- strip off leading > and indents, and run through blocks
4327 larsers.blockquote_body = ((parsers.leader * parsers.more * parsers.space^-
4328 1)/""
4329 * parsers.linechar^0 * parsers.newline)^1
4330 * (-(parsers.leader * parsers.more
4331 + parsers.blankline) * parsers.linechar^1
4332 * parsers.newline)^0

```

```

4332
4333 if not options.breakableBlockquotes then
4334 larsers.blockquote_body = larsers.blockquote_body
4335 * (parsers.blankline^0 / "")
4336 end

```

### 3.1.5.5 Parsers Used for Citations (local)

```

4337 larsers.citations = function(text_cites, raw_cites)
4338 local function normalize(str)
4339 if str == "" then
4340 str = nil
4341 else
4342 str = (options.citationNbsps and parse_inlines_nbsp or
4343 parse_inlines)(str)
4344 end
4345 return str
4346 end
4347
4348 local cites = {}
4349 for i = 1,#raw_cites,4 do
4350 cites[#cites+1] = {
4351 prenote = normalize(raw_cites[i]),
4352 suppress_author = raw_cites[i+1] == "-",
4353 name = writer.citation(raw_cites[i+2]),
4354 postnote = normalize(raw_cites[i+3]),
4355 }
4356 end
4357 return writer.citations(text_cites, cites)
4358 end

```

### 3.1.5.6 Parsers Used for Footnotes (local)

```

4359 local rawnotes = {}
4360
4361 -- like indirect_link
4362 local function lookup_note(ref)
4363 return writer.defer_call(function()
4364 local found = rawnotes[normalize_tag(ref)]
4365 if found then
4366 return writer.note(parse_blocks_toplevel(found))
4367 else
4368 return {"[", parse_inlines("^" .. ref), "]" }
4369 end
4370 end)
4371 end
4372
4373 local function register_note(ref,rawnote)

```

```

4374 rawnotes[normalize_tag(ref)] = rawnote
4375 return ""
4376 end
4377
4378 larsers.NoteRef = parsers.RawNoteRef / lookup_note
4379
4380
4381 larsers.NoteBlock = parsers.leader * parsers.RawNoteRef * parsers.colon
4382 * parsers.spnl * parsers.indented_blocks(parsers.chunk)
4383 / register_note
4384
4385 larsers.InlineNote = parsers.circumflex
4386 * (parsers.tag / parse_inlines_no_inline_note) -- no notes inside
4387 / writer.note

```

### 3.1.5.7 Parsers Used for Tables (local)

```

4388 larsers.table_row = pipe_table_row(true
4389 , (C((parsers.linechar - parsers.pipe)^1)
4390 / parse_inlines)
4391 , parsers.pipe
4392 , (C((parsers.linechar - parsers.pipe)^0)
4393 / parse_inlines))
4394
4395 if options.tableCaptions then
4396 larsers.table_caption = #parsers.table_caption_beginning
4397 * parsers.table_caption_beginning
4398 * Ct(parsers.IndentedInline^1)
4399 * parsers.newline
4400 else
4401 larsers.table_caption = parsers.fail
4402 end
4403
4404 larsers.PipeTable = Ct(larsers.table_row * parsers.newline
4405 * parsers.table_hline
4406 * (parsers.newline * larsers.table_row)^0)
4407 / make_pipe_table_rectangular
4408 * larsers.table_caption^-1
4409 / writer.table

```

### 3.1.5.8 Helpers for Links and References (local)

```

4410 -- List of references defined in the document
4411 local references
4412
4413 -- add a reference to the list
4414 local function register_link(tag,url,title)
4415 references[normalize_tag(tag)] = { url = url, title = title }

```



```

4416 return ""
4417 end
4418
4419 -- lookup link reference and return either
4420 -- the link or nil and fallback text.
4421 local function lookup_reference(label,sps,tag)
4422 local tagpart
4423 if not tag then
4424 tag = label
4425 tagpart = ""
4426 elseif tag == "" then
4427 tag = label
4428 tagpart = "[]"
4429 else
4430 tagpart = {"[", parse_inlines(tag), "]" }
4431 end
4432 if sps then
4433 tagpart = {sps, tagpart}
4434 end
4435 local r = references[normalize_tag(tag)]
4436 if r then
4437 return r
4438 else
4439 return nil, {"[", parse_inlines(label), "]", tagpart}
4440 end
4441 end
4442
4443 -- lookup link reference and return a link, if the reference is found,
4444 -- or a bracketed label otherwise.
4445 local function indirect_link(label,sps,tag)
4446 return writer.defer_call(function()
4447 local r,fallback = lookup_reference(label,sps,tag)
4448 if r then
4449 return writer.link(parse_inlines_no_link(label), r.url, r.title)
4450 else
4451 return fallback
4452 end
4453 end)
4454 end
4455
4456 -- lookup image reference and return an image, if the reference is found,
4457 -- or a bracketed label otherwise.
4458 local function indirect_image(label,sps,tag)
4459 return writer.defer_call(function()
4460 local r,fallback = lookup_reference(label,sps,tag)
4461 if r then
4462 return writer.image(writer.string(label), r.url, r.title)

```

```

4463 else
4464 return {"!", fallback}
4465 end
4466 end)
4467 end

```

### 3.1.5.9 Inline Elements (local)

```

4468 larsers.Str = (parsers.normalchar * (parsers.normalchar + parsers.at)^0)
4469 / writer.string
4470
4471 larsers.Symbol = (parsers.specialchar - parsers.tightblocksep)
4472 / writer.string
4473
4474 larsers.Ellipsis = P("...") / writer.ellipsis
4475
4476 larsers.Smart = larsers.Ellipsis
4477
4478 larsers.Code = parsers.inticks / writer.code
4479
4480 if options.blankBeforeBlockquote then
4481 larsers.bqstart = parsers.fail
4482 else
4483 larsers.bqstart = parsers.more
4484 end
4485
4486 if options.blankBeforeHeading then
4487 larsers.headerstart = parsers.fail
4488 else
4489 larsers.headerstart = parsers.hash
4490 + (parsers.line * (parsers.equal^1 + parsers.dash^1)
4491 * parsers.optionalspace * parsers.newline)
4492 end
4493
4494 if not options.fencedCode or options.blankBeforeCodeFence then
4495 larsers.fencestart = parsers.fail
4496 else
4497 larsers.fencestart = parsers.fencehead(parsers.backtick)
4498 + parsers.fencehead(parsers.tilde)
4499 end
4500
4501 larsers.Endline = parsers.newline * -(-- newline, but not before...
4502 parsers.blankline -- paragraph break
4503 + parsers.tightblocksep -- nested list
4504 + parsers.eof -- end of document
4505 + larsers.bqstart
4506 + larsers.headerstart

```

```

4507 + larsers.fencestart
4508) * parsers.spacechar^0
4509 / (options.hardLineBreaks and writer.linebreak
4510 or writer.space)
4511
4512 larsers.OptionalIndent
4513 = parsers.spacechar^1 / writer.space
4514
4515 larsers.Space
4516 = parsers.spacechar^2 * larsers.Endline / writer.linebreak
4517 + parsers.spacechar^1 * larsers.Endline^-1 * parsers.eof / ""
4518 + parsers.spacechar^1 * larsers.Endline
4519 * parsers.optionalspace
4520 / (options.hardLineBreaks
4521 and writer.linebreak
4522 or writer.space)
4523 + parsers.spacechar^1 * parsers.optionalspace
4524 / writer.space
4525
4526 larsers.NonbreakingEndline
4527 = parsers.newline * -(-- newline, but not before...
4528 parsers.blankline -- paragraph break
4529 + parsers.tightblocksep -- nested list
4530 + parsers.eof -- end of document
4531 + larsers.bqstart
4532 + larsers.headerstart
4533 + larsers.fencestart
4534) * parsers.spacechar^0
4535 / (options.hardLineBreaks and writer.linebreak
4536 or writer.nbsp)
4537
4538 larsers.NonbreakingSpace
4539 = parsers.spacechar^2 * larsers.Endline / writer.linebreak
4540 + parsers.spacechar^1 * larsers.Endline^-1 * parsers.eof / ""
4541 + parsers.spacechar^1 * larsers.Endline
4542 * parsers.optionalspace
4543 / (options.hardLineBreaks
4544 and writer.linebreak
4545 or writer.nbsp)
4546 + parsers.spacechar^1 * parsers.optionalspace
4547 / writer.nbsp
4548
4549 if options.underscores then
4550 larsers.Strong = (parsers.between(parsers.Inline, parsers.doubleasterisks,
4551 parsers.doubleasterisks)
4552 + parsers.between(parsers.Inline, parsers.doubleunderscores,
4553 parsers.doubleunderscores)
4554) / writer.strong

```

```

4554
4555 larsers.Emph = (parsers.between(parsers.Inline, parsers.asterisk,
4556 parsers.asterisk)
4557 + parsers.between(parsers.Inline, parsers.underscore,
4558 parsers.underscore)
4559) / writer.emphasis
4560 else
4561 larsers.Strong = (parsers.between(parsers.Inline, parsers.doubleasterisks,
4562 parsers.doubleasterisks)
4563) / writer.strong
4564
4565 larsers.Emph = (parsers.between(parsers.Inline, parsers.asterisk,
4566 parsers.asterisk)
4567) / writer.emphasis
4568 end
4569
4570 larsers.AutoLinkUrl = parsers.less
4571 * C(parsers.alphanumeric^1 * P("://") * parsers.urlchar^1)
4572 * parsers.more
4573 / function(url)
4574 return writer.link(writer.escape(url), url)
4575 end
4576
4577 larsers.AutoLinkEmail = parsers.less
4578 * C((parsers.alphanumeric + S("-._+"))^1
4579 * P("@") * parsers.urlchar^1)
4580 * parsers.more
4581 / function(email)
4582 return writer.link(writer.escape(email),
4583 "mailto:".email)
4584 end
4585
4586 larsers.DirectLink = (parsers.tag / parse_inlines_no_link) -- no links inside link
4587 * parsers.spnl
4588 * parsers.lparent
4589 * (parsers.url + Cc("")) -- link can be empty [foo]()
4590 * parsers.optionaltitle
4591 * parsers.rparent
4592 / writer.link
4593
4594 larsers.IndirectLink = parsers.tag * (C(parsers.spnl) * parsers.tag)^-
4595
4596
4597 -- parse a link or image (direct or indirect)
4598 larsers.Link = larsers.DirectLink + larsers.IndirectLink
4599

```

```

4600 larsers.DirectImage = parsers.exclamation
4601 * (parsers.tag / parse_inlines)
4602 * parsers.spnl
4603 * parsers.lparent
4604 * (parsers.url + Cc("")) -- link can be empty [foo]()
4605 * parsers.optionaltitle
4606 * parsers.rparent
4607 / writer.image
4608
4609 larsers.IndirectImage = parsers.exclamation * parsers.tag
4610 * (C(parsers.spnl) * parsers.tag)^-1 / indirect_image
4611
4612 larsers.Image = larsers.DirectImage + larsers.IndirectImage
4613
4614 larsers.TextCitations = Ct((parsers.spnl
4615 * Cc("")
4616 * parsers.citation_name
4617 * ((parsers.spnl
4618 * parsers.lbracket
4619 * parsers.citation_headless_body
4620 * parsers.rbracket) + Cc("")))^1)
4621 / function(raw_cites)
4622 return larsers.citations(true, raw_cites)
4623 end
4624
4625 larsers.ParenthesizedCitations
4626 = Ct((parsers.spnl
4627 * parsers.lbracket
4628 * parsers.citation_body
4629 * parsers.rbracket)^1)
4630 / function(raw_cites)
4631 return larsers.citations(false, raw_cites)
4632 end
4633
4634 larsers.Citations = larsers.TextCitations + larsers.ParenthesizedCitations
4635
4636 -- avoid parsing long strings of * or _ as emph/strong
4637 larsers.UlOrStarLine = parsers.asterisk^4 + parsers.underscore^4
4638 / writer.string
4639
4640 larsers.EscapedChar = parsers.backslash * C(parsers.escapable) / writer.string
4641
4642 larsers.InlineHtml = parsers.emptyelt_any
4643 + (parsers.htmlcomment / parse_inlines_no_html)
4644 / writer.inline_html_comment
4645 + parsers.htmlinstruction
4646 + parsers.openelt_any

```

```

4647 + parsers.closeelt_any
4648
4649 larsers.HtmlEntity = parsers.hexentity / entities.hex_entity / writer.string
4650 + parsers.decententity / entities.dec_entity / writer.string
4651 + parsers.tagentity / entities.char_entity / writer.string

```

### 3.1.5.10 Block Elements (local)

```

4652 larsers.ContentBlock = parsers.leader
4653 * (parsers.localfilepath + parsers.onlineimageurl)
4654 * parsers.contentblock_tail
4655 / writer.contentblock
4656
4657 larsers.DisplayHtml = parsers.displayhtml
4658
4659 larsers.Verbatim = Cs((parsers.blanklines
4660 * ((parsers.indentedline - parsers.blankline))^1)^1
4661) / expandtabs / writer.verbatim
4662
4663 larsers.FencedCode = (parsers.TildeFencedCode
4664 + parsers.BacktickFencedCode)
4665 / function(infostring, code)
4666 return writer.fencedCode(writer.string(infostring),
4667 expandtabs(code))
4668 end
4669
4670 larsers.JekyllData = P("----")
4671 * parsers.blankline / 0
4672 * #(-parsers.blankline) -- if followed by blank, it's an hrule
4673 * C((parsers.line - P("----") - P("..."))^0)
4674 * (P("----") + P("..."))
4675 / function(text)
4676 local tinyyaml = require("markdown-tinyyaml")
4677 data = tinyyaml.parse(text)
4678 return writer.jekyllData(data, function(s)
4679 return parse_blocks(s)
4680 end, nil)
4681 end
4682
4683 larsers.Blockquote = Cs(larsers.blockquote_body^1)
4684 / parse_blocks_toplevel / writer.blockquote
4685
4686 larsers.HorizontalRule = (parsers.lineof(parsers.asterisk)
4687 + parsers.lineof(parsers.dash)
4688 + parsers.lineof(parsers.underscore)
4689) / writer.hrule
4690

```

```

4691 larsers.Reference = parsers.define_reference_parser / register_link
4692
4693 larsers.Paragraph = parsers.nonindentspace * Ct(parsers.Inline^1)
4694 * parsers.newline
4695 * (parsers.blankline^1
4696 + #parsers.hash
4697 + #(parsers.leader * parsers.more * parsers.space^-
1)
4698)
4699 / writer.paragraph
4700
4701 larsers.ToplevelParagraph
4702 = parsers.nonindentspace * Ct(parsers.Inline^1)
4703 * (parsers.newline
4704 * (parsers.blankline^1
4705 + #parsers.hash
4706 + #(parsers.leader * parsers.more * parsers.space^-
1)
4707 + parsers.eof
4708)
4709 + parsers.eof)
4710 / writer.paragraph
4711
4712 larsers.Plain = parsers.nonindentspace * Ct(parsers.Inline^1)
4713 / writer.plain

```

### 3.1.5.11 Lists (local)

```

4714 larsers.starter = parsers.bullet + larsers.enumerator
4715
4716 if options.taskLists then
4717 larsers.tickbox = (parsers.ticked_box
4718 + parsers.halfticked_box
4719 + parsers.unticked_box
4720) / writer.tickbox
4721 else
4722 larsers.tickbox = parsers.fail
4723 end
4724
4725 -- we use \001 as a separator between a tight list item and a
4726 -- nested list under it.
4727 larsers.NestedList = Cs((parsers.optionallyindentedline
4728 - larsers.starter)^1)
4729 / function(a) return "\001"..a end
4730
4731 larsers.ListBlockLine = parsers.optionallyindentedline
4732 - parsers.blankline - (parsers.indent^-1

```

```

4733 * larsers.starter)
4734
4735 larsers.ListBlock = parsers.line * larsers.ListBlockLine^0
4736
4737 larsers.ListContinuationBlock = parsers.blanklines * (parsers.indent / "")
4738 * larsers.ListBlock
4739
4740 larsers.TightListItem = function(starter)
4741 return -larsers.HorizontalRule
4742 * (Cs(starter / "" * larsers.tickbox^-1 * larsers.ListBlock * larsers.Nested
1)
4743 / parse_blocks)
4744 * -(parsers.blanklines * parsers.indent)
4745 end
4746
4747 larsers.LooseListItem = function(starter)
4748 return -larsers.HorizontalRule
4749 * Cs(starter / "" * larsers.tickbox^-1 * larsers.ListBlock * Cc("\n")
4750 * (larsers.NestedList + larsers.ListContinuationBlock^0)
4751 * (parsers.blanklines / "\n\n")
4752) / parse_blocks
4753 end
4754
4755 larsers.BulletList = (Ct(larsers.TightListItem(parsers.bullet)^1) * Cc(true)
4756 * parsers.skipblanklines * -parsers.bullet
4757 + Ct(larsers.LooseListItem(parsers.bullet)^1) * Cc(false)
4758 * parsers.skipblanklines)
4759 / writer.bulletlist
4760
4761 local function ordered_list(items,tight,startNumber)
4762 if options.startNumber then
4763 startNumber = tonumber(startNumber) or 1 -- fallback for '#'
4764 if startNumber ~= nil then
4765 startNumber = math.floor(startNumber)
4766 end
4767 else
4768 startNumber = nil
4769 end
4770 return writer.orderedlist(items,tight,startNumber)
4771 end
4772
4773 larsers.OrderedList = Cg(larsers.enumerator, "listtype") *
4774 (Ct(larsers.TightListItem(Cb("listtype")))
4775 * larsers.TightListItem(larsers.enumerator)^0)
4776 * Cc(true) * parsers.skipblanklines * -larsers.enumerator
4777 + Ct(larsers.LooseListItem(Cb("listtype")))
4778 * larsers.LooseListItem(larsers.enumerator)^0)

```



```

4779 * Cc(false) * parsers.skipblanklines
4780) * Cb("listtype") / ordered_list
4781
4782 local function definition_list_item(term, defs, tight)
4783 return { term = parse_inlines(term), definitions = defs }
4784 end
4785
4786 larsers.DefinitionListItemLoose = C(parsers.line) * parsers.skipblanklines
4787 * Ct((parsers.defstart
4788 * parsers.indented_blocks(parsers.dlchunk)
4789 / parse_blocks_toplevel)^1)
4790 * Cc(false) / definition_list_item
4791
4792 larsers.DefinitionListItemTight = C(parsers.line)
4793 * Ct((parsers.defstart * parsers.dlchunk
4794 / parse_blocks)^1)
4795 * Cc(true) / definition_list_item
4796
4797 larsers.DefinitionList = (Ct(larsers.DefinitionListItemLoose^1) * Cc(false)
4798 + Ct(larsers.DefinitionListItemTight^1)
4799 * (parsers.skipblanklines
4800 * -larsers.DefinitionListItemLoose * Cc(true))
4801) / writer.definitionlist

```

### 3.1.5.12 Blank (local)

```

4802 larsers.Blank = parsers.blankline / ""
4803 + larsers.NoteBlock
4804 + larsers.Reference
4805 + (parsers.tightblocksep / "\n")

```

### 3.1.5.13 Headings (local)

```

4806 -- parse atx header
4807 if options.headerAttributes then
4808 larsers.AtxHeading = Cg(parsers.HeadingStart,"level")
4809 * parsers.optionalspace
4810 * (C(((parsers.linechar
4811 - ((parsers.hash^1
4812 * parsers.optionalspace
4813 * parsers.HeadingAttributes^-1
4814 + parsers.HeadingAttributes)
4815 * parsers.optionalspace
4816 * parsers.newline))
4817 * (parsers.linechar
4818 - parsers.hash
4819 - parsers.lbrace)^0)^1)
4820 / parse_inlines)

```

```

4821 * Cg(Ct(parsers.newline
4822 + (parsers.hash^1
4823 * parsers.optionalspace
4824 * parsers.HeadingAttributes~-1
4825 + parsers.HeadingAttributes)
4826 * parsers.optionalspace
4827 * parsers.newline), "attributes")
4828 * Cb("level")
4829 * Cb("attributes")
4830 / writer.heading
4831
4832 larsers.SettextHeading = #(parsers.line * S("--"))
4833 * (C((parsers.linechar
4834 - (parsers.HeadingAttributes
4835 * parsers.optionalspace
4836 * parsers.newline))
4837 * (parsers.linechar
4838 - parsers.lbrace)^0)^1)
4839 / parse_inlines)
4840 * Cg(Ct(parsers.newline
4841 + (parsers.HeadingAttributes
4842 * parsers.optionalspace
4843 * parsers.newline)), "attributes")
4844 * parsers.HeadingLevel
4845 * Cb("attributes")
4846 * parsers.optionalspace
4847 * parsers.newline
4848 / writer.heading
4849 else
4850 larsers.AtxHeading = Cg(parsers.HeadingStart, "level")
4851 * parsers.optionalspace
4852 * (C(parsers.line) / strip_atx_end / parse_inlines)
4853 * Cb("level")
4854 / writer.heading
4855
4856 larsers.SettextHeading = #(parsers.line * S("--"))
4857 * Ct(parsers.linechar^1 / parse_inlines)
4858 * parsers.newline
4859 * parsers.HeadingLevel
4860 * parsers.optionalspace
4861 * parsers.newline
4862 / writer.heading
4863 end
4864
4865 larsers.Heading = larsers.AtxHeading + larsers.SettextHeading

```

### 3.1.5.14 Syntax Specification

```
4866 local syntax =
4867 { "Blocks",
4868
4869 Blocks = larsers.Blank^0 * parsers.Block^-1
4870 * (larsers.Blank^0 / writer.interblocksep
4871 * parsers.Block)^0
4872 * larsers.Blank^0 * parsers.eof,
4873
4874 Blank = larsers.Blank,
4875
4876 JekyllData = larsers.JekyllData,
4877
4878 Block = V("ContentBlock")
4879 + V("JekyllData")
4880 + V("Blockquote")
4881 + V("PipeTable")
4882 + V("Verbatim")
4883 + V("FencedCode")
4884 + V("HorizontalRule")
4885 + V("BulletList")
4886 + V("OrderedList")
4887 + V("Heading")
4888 + V("DefinitionList")
4889 + V("DisplayHtml")
4890 + V("Paragraph")
4891 + V("Plain"),
4892
4893 ContentBlock = larsers.ContentBlock,
4894 Blockquote = larsers.Blockquote,
4895 Verbatim = larsers.Verbatim,
4896 FencedCode = larsers.FencedCode,
4897 HorizontalRule = larsers.HorizontalRule,
4898 BulletList = larsers.BulletList,
4899 OrderedList = larsers.OrderedList,
4900 Heading = larsers.Heading,
4901 DefinitionList = larsers.DefinitionList,
4902 DisplayHtml = larsers.DisplayHtml,
4903 Paragraph = larsers.Paragraph,
4904 PipeTable = larsers.PipeTable,
4905 Plain = larsers.Plain,
4906
4907 Inline = V("Str")
4908 + V("Space")
4909 + V("Endline")
4910 + V("U1OrStarLine")
4911 + V("Strong")
```

```

4912 + V("Emph")
4913 + V("InlineNote")
4914 + V("NoteRef")
4915 + V("Citations")
4916 + V("Link")
4917 + V("Image")
4918 + V("Code")
4919 + V("AutoLinkUrl")
4920 + V("AutoLinkEmail")
4921 + V("InlineHtml")
4922 + V("HtmlEntity")
4923 + V("EscapedChar")
4924 + V("Smart")
4925 + V("Symbol"),
4926
4927 IndentedInline = V("Str")
4928 + V("OptionalIndent")
4929 + V("Endline")
4930 + V("U1OrStarLine")
4931 + V("Strong")
4932 + V("Emph")
4933 + V("InlineNote")
4934 + V("NoteRef")
4935 + V("Citations")
4936 + V("Link")
4937 + V("Image")
4938 + V("Code")
4939 + V("AutoLinkUrl")
4940 + V("AutoLinkEmail")
4941 + V("InlineHtml")
4942 + V("HtmlEntity")
4943 + V("EscapedChar")
4944 + V("Smart")
4945 + V("Symbol"),
4946
4947 Str = larsers.Str,
4948 Space = larsers.Space,
4949 OptionalIndent = larsers.OptionalIndent,
4950 Endline = larsers.Endline,
4951 U1OrStarLine = larsers.U1OrStarLine,
4952 Strong = larsers.Strong,
4953 Emph = larsers.Emph,
4954 InlineNote = larsers.InlineNote,
4955 NoteRef = larsers.NoteRef,
4956 Citations = larsers.Citations,
4957 Link = larsers.Link,
4958 Image = larsers.Image,

```

```

4959 Code = larsers.Code,
4960 AutoLinkUrl = larsers.AutoLinkUrl,
4961 AutoLinkEmail = larsers.AutoLinkEmail,
4962 InlineHtml = larsers.InlineHtml,
4963 HtmlEntity = larsers.HtmlEntity,
4964 EscapedChar = larsers.EscapedChar,
4965 Smart = larsers.Smart,
4966 Symbol = larsers.Symbol,
4967 }
4968
4969 if not options.citations then
4970 syntax.Citations = parsers.fail
4971 end
4972
4973 if not options.contentBlocks then
4974 syntax.ContentBlock = parsers.fail
4975 end
4976
4977 if not options.codeSpans then
4978 syntax.Code = parsers.fail
4979 end
4980
4981 if not options.definitionLists then
4982 syntax.DefinitionList = parsers.fail
4983 end
4984
4985 if not options.fencedCode then
4986 syntax.FencedCode = parsers.fail
4987 end
4988
4989 if not options.footnotes then
4990 syntax.NoteRef = parsers.fail
4991 end
4992
4993 if not options.html then
4994 syntax.DisplayHtml = parsers.fail
4995 syntax.InlineHtml = parsers.fail
4996 syntax.HtmlEntity = parsers.fail
4997 end
4998
4999 if not options.inlineFootnotes then
5000 syntax.InlineNote = parsers.fail
5001 end
5002
5003 if not options.jekyllData then
5004 syntax.JekyllData = parsers.fail
5005 end

```

```

5006
5007 if options.preserveTabs then
5008 options.stripIndent = false
5009 end
5010
5011 if not options.pipeTables then
5012 syntax.PipeTable = parsers.fail
5013 end
5014
5015 if not options.smartEllipses then
5016 syntax.Smart = parsers.fail
5017 end
5018
5019 local blocks_toplevel_t = util.table_copy(syntax)
5020 blocks_toplevel_t.Paragraph = larsers.ToplevelParagraph
5021 larsers.blocks_toplevel = Ct(blocks_toplevel_t)
5022
5023 larsers.blocks = Ct(syntax)
5024
5025 local inlines_t = util.table_copy(syntax)
5026 inlines_t[1] = "Inlines"
5027 inlines_t.Inlines = parsers.Inline~0 * (parsers.spacing~0 * parsers.eof / "")
5028 larsers.inlines = Ct(inlines_t)
5029
5030 local inlines_no_link_t = util.table_copy(inlines_t)
5031 inlines_no_link_t.Link = parsers.fail
5032 larsers.inlines_no_link = Ct(inlines_no_link_t)
5033
5034 local inlines_no_inline_note_t = util.table_copy(inlines_t)
5035 inlines_no_inline_note_t.InlineNote = parsers.fail
5036 larsers.inlines_no_inline_note = Ct(inlines_no_inline_note_t)
5037
5038 local inlines_no_html_t = util.table_copy(inlines_t)
5039 inlines_no_html_t.DisplayHtml = parsers.fail
5040 inlines_no_html_t.InlineHtml = parsers.fail
5041 inlines_no_html_t.HtmlEntity = parsers.fail
5042 larsers.inlines_no_html = Ct(inlines_no_html_t)
5043
5044 local inlines_nbsp_t = util.table_copy(inlines_t)
5045 inlines_nbsp_t.Endline = larsers.NonbreakingEndline
5046 inlines_nbsp_t.Space = larsers.NonbreakingSpace
5047 larsers.inlines_nbsp = Ct(inlines_nbsp_t)

```

**3.1.5.15 Exported Conversion Function** Define `reader->convert` as a function that converts markdown string `input` into a plain T<sub>E</sub>X output and returns it. Note that the converter assumes that the input has UNIX line endings.

```

5048 function self.convert(input)
5049 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 on page 7). The `cacheDir` option is disregarded.

```

5050 local opt_string = {}
5051 for k,_ in pairs(defaultOptions) do
5052 local v = options[k]
5053 if k ~= "cacheDir" then
5054 opt_string[#opt_string+1] = k .. "=" .. tostring(v)
5055 end
5056 end
5057 table.sort(opt_string)
5058 local salt = table.concat(opt_string, ",") .. "," .. metadata.version
5059 local output

```

If we cache markdown documents, produce the cache file and transform its filename to plain  $\text{\TeX}$  output via the `writer->pack` method.

```

5060 local function convert(input)
5061 return util.rope_to_string(parse_blocks_toplevel(input)) .. writer.eof
5062 end
5063 if options.eagerCache or options.finalizeCache then
5064 local name = util.cache(options.cacheDir, input, salt, convert, ".md" .. writer.s
5065 output = writer.pack(name)

```

Otherwise, return the result of the conversion directly.

```

5066 else
5067 output = convert(input)
5068 end

```

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

```

5069 if options.finalizeCache then
5070 local file, mode
5071 if options.frozenCacheCounter > 0 then
5072 mode = "a"
5073 else
5074 mode = "w"
5075 end
5076 file = assert(io.open(options.frozenCacheFileName, mode),
5077 [[could not open file]] .. options.frozenCacheFileName
5078 .. [{" for writing"}])
5079 assert(file:write([[\\expandafter\\global\\expandafter\\def\\csname]]
5080 .. [[markdownFrozenCache]] .. options.frozenCacheCounter
5081 .. [[\\endcsname{]] .. output .. [{}]] .. "\\n"))
5082 assert(file:close())

```

```

5083 end
5084 return output
5085 end
5086 return self
5087 end

```

### 3.1.6 Conversion from Markdown to Plain T<sub>E</sub>X

The `new` method returns the `reader->convert` function of a reader object associated with the Lua interface options (see Section 2.1.2 on page 7) `options` and with a writer object associated with `options`.

```

5088 function M.new(options)
5089 local writer = M.writer.new(options)
5090 local reader = M.reader.new(writer, options)
5091 return reader.convert
5092 end
5093
5094 return M

```

### 3.1.7 Command-Line Implementation

The command-line implementation provides the actual conversion routine for the command-line interface described in Section 2.1.5 on page 20.

```

5095
5096 local input
5097 if input_filename then
5098 local input_file = assert(io.open(input_filename, "r"),
5099 [[could not open file]] .. input_filename .. [[for reading]])
5100 input = assert(input_file:read("*a"))
5101 assert(input_file:close())
5102 else
5103 input = assert(io.read("*a"))
5104 end
5105

```

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

```

5106 local lfs = require("lfs")
5107 if options.cacheDir and not lfs.isdir(options.cacheDir) then
5108 assert(lfs.mkdir(options["cacheDir"]))
5109 end
5110
5111 local ran_ok, kpse = pcall(require, "kpse")
5112 if ran_ok then kpse.set_program_name("luatex") end
5113 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.



```

5114 if metadata.version ~= md.metadata.version then
5115 warn("markdown-cli.lua " .. metadata.version .. " used with " ..
5116 "markdown.lua " .. md.metadata.version .. ".")
5117 end
5118 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.
5119 local output = convert(input:gsub("\r\n?", "\n") .. "\n")
5120
5121 if output_filename then
5122 local output_file = assert(io.open(output_filename, "w"),
5123 [[could not open file]] .. output_filename .. [[for writing]])
5124 assert(output_file:write(output))
5125 assert(output_file:close())
5126 else
5127 assert(io.write(output))
5128 end

```

## 3.2 Plain T<sub>E</sub>X Implementation

The plain T<sub>E</sub>X implementation provides macros for the interfacing between T<sub>E</sub>X 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<sub>E</sub>X exposed by the plain T<sub>E</sub>X interface (see Section 2.2 on page 23).

### 3.2.1 Logging Facilities

```

5129 \ifx\markdownInfo\undefined
5130 \def\markdownInfo#1{%
5131 \immediate\write-1{(1.\the\inputlineno) markdown.tex info: #1.}}%
5132 \fi
5133 \ifx\markdownWarning\undefined
5134 \def\markdownWarning#1{%
5135 \immediate\write16{(1.\the\inputlineno) markdown.tex warning: #1}}%
5136 \fi
5137 \ifx\markdownError\undefined
5138 \def\markdownError#1#2{%
5139 \errhelp{#2.}%
5140 \errmessage{(1.\the\inputlineno) markdown.tex error: #1}}%
5141 \fi

```

### 3.2.2 Finalizing and Freezing the Cache

When the `\markdownOptionFinalizeCache` option is enabled, then the `\markdownFrozenCacheCounter` counter is used to enumerate the markdown documents using the Lua interface `frozenCacheCounter` option.

When the `\markdownOptionFrozenCache` option is enabled, then the `\markdownFrozenCacheCounter` counter is used to render markdown documents from the frozen cache without invoking Lua.

```
5142 \newcount\markdownFrozenCacheCounter
```

### 3.2.3 Token Renderer Prototypes

The following definitions should be considered placeholder.

```
5143 \def\markdownRendererInterblockSeparatorPrototype{\par}%
5144 \def\markdownRendererLineBreakPrototype{\hfil\break}%
5145 \let\markdownRendererEllipsisPrototype\dots
5146 \def\markdownRendererNbspPrototype{~}%
5147 \def\markdownRendererLeftBracePrototype{\char`}%
5148 \def\markdownRendererRightBracePrototype{\char`}%
5149 \def\markdownRendererDollarSignPrototype{\char`$}%
5150 \def\markdownRendererPercentSignPrototype{\char`}%
5151 \def\markdownRendererAmpersandPrototype{&}%
5152 \def\markdownRendererUnderscorePrototype{\char`_%}
5153 \def\markdownRendererHashPrototype{\char`#}%
5154 \def\markdownRendererCircumflexPrototype{\char`^}%
5155 \def\markdownRendererBackslashPrototype{\char`\}%
5156 \def\markdownRendererTildePrototype{\char`~}%
5157 \def\markdownRendererPipePrototype{|}%
5158 \def\markdownRendererCodeSpanPrototype#1{\tt#1}%
5159 \def\markdownRendererLinkPrototype#1#2#3#4#5#6#7#8#9#10#11#12#13#14#15#16#17#18#19#20#21#22#23#24#25#26#27#28#29#30#31#32#33#34#35#36#37#38#39#40#41#42#43#44#45#46#47#48#49#50#51#52#53#54#55#56#57#58#59#60#61#62#63#64#65#66#67#68#69#70#71#72#73#74#75#76#77#78#79#80#81#82#83#84#85#86#87#88#89#90#91#92#93#94#95#96#97#98#99#100#101#102#103#104#105#106#107#108#109#110#111#112#113#114#115#116#117#118#119#120#121#122#123#124#125#126#127#128#129#130#131#132#133#134#135#136#137#138#139#140#141#142#143#144#145#146#147#148#149#150#151#152#153#154#155#156#157#158#159#160#161#162#163#164#165#166#167#168#169#170#171#172#173#174#175#176#177#178#179#180#181#182#183#184#185#186#187#188#189#190#191#192#193#194#195#196#197#198#199#200#201#202#203#204#205#206#207#208#209#210#211#212#213#214#215#216#217#218#219#220#221#222#223#224#225#226#227#228#229#230#231#232#233#234#235#236#237#238#239#240#241#242#243#244#245#246#247#248#249#250#251#252#253#254#255#256#257#258#259#260#261#262#263#264#265#266#267#268#269#270#271#272#273#274#275#276#277#278#279#280#281#282#283#284#285#286#287#288#289#290#291#292#293#294#295#296#297#298#299#300#301#302#303#304#305#306#307#308#309#310#311#312#313#314#315#316#317#318#319#320#321#322#323#324#325#326#327#328#329#330#331#332#333#334#335#336#337#338#339#340#341#342#343#344#345#346#347#348#349#350#351#352#353#354#355#356#357#358#359#360#361#362#363#364#365#366#367#368#369#370#371#372#373#374#375#376#377#378#379#380#381#382#383#384#385#386#387#388#389#390#391#392#393#394#395#396#397#398#399#400#401#402#403#404#405#406#407#408#409#410#411#412#413#414#415#416#417#418#419#420#421#422#423#424#425#426#427#428#429#430#431#432#433#434#435#436#437#438#439#440#441#442#443#444#445#446#447#448#449#450#451#452#453#454#455#456#457#458#459#460#461#462#463#464#465#466#467#468#469#470#471#472#473#474#475#476#477#478#479#480#481#482#483#484#485#486#487#488#489#490#491#492#493#494#495#496#497#498#499#500#501#502#503#504#505#506#507#508#509#510#511#512#513#514#515#516#517#518#519#520#521#522#523#524#525#526#527#528#529#530#531#532#533#534#535#536#537#538#539#540#541#542#543#544#545#546#547#548#549#550#551#552#553#554#555#556#557#558#559#560#561#562#563#564#565#566#567#568#569#570#571#572#573#574#575#576#577#578#579#580#581#582#583#584#585#586#587#588#589#590#591#592#593#594#595#596#597#598#599#600#601#602#603#604#605#606#607#608#609#610#611#612#613#614#615#616#617#618#619#620#621#622#623#624#625#626#627#628#629#630#631#632#633#634#635#636#637#638#639#640#641#642#643#644#645#646#647#648#649#650#651#652#653#654#655#656#657#658#659#660#661#662#663#664#665#666#667#668#669#670#671#672#673#674#675#676#677#678#679#680#681#682#683#684#685#686#687#688#689#690#691#692#693#694#695#696#697#698#699#700#701#702#703#704#705#706#707#708#709#710#711#712#713#714#715#716#717#718#719#720#721#722#723#724#725#726#727#728#729#730#731#732#733#734#735#736#737#738#739#740#741#742#743#744#745#746#747#748#749#750#751#752#753#754#755#756#757#758#759#760#761#762#763#764#765#766#767#768#769#770#771#772#773#774#775#776#777#778#779#780#781#782#783#784#785#786#787#788#789#790#791#792#793#794#795#796#797#798#799#800#801#802#803#804#805#806#807#808#809#810#811#812#813#814#815#816#817#818#819#820#821#822#823#824#825#826#827#828#829#830#831#832#833#834#835#836#837#838#839#840#841#842#843#844#845#846#847#848#849#850#851#852#853#854#855#856#857#858#859#860#861#862#863#864#865#866#867#868#869#870#871#872#873#874#875#876#877#878#879#880#881#882#883#884#885#886#887#888#889#890#891#892#893#894#895#896#897#898#899#900#901#902#903#904#905#906#907#908#909#910#911#912#913#914#915#916#917#918#919#920#921#922#923#924#925#926#927#928#929#930#931#932#933#934#935#936#937#938#939#940#941#942#943#944#945#946#947#948#949#950#951#952#953#954#955#956#957#958#959#960#961#962#963#964#965#966#967#968#969#970#971#972#973#974#975#976#977#978#979#980#981#982#983#984#985#986#987#988#989#990#991#992#993#994#995#996#997#998#999#1000#1001#1002#1003#1004#1005#1006#1007#1008#1009#1010#1011#1012#1013#1014#1015#1016#1017#1018#1019#1020#1021#1022#1023#1024#1025#1026#1027#1028#1029#1030#1031#1032#1033#1034#1035#1036#1037#1038#1039#1040#1041#1042#1043#1044#1045#1046#1047#1048#1049#1050#1051#1052#1053#1054#1055#1056#1057#1058#1059#1060#1061#1062#1063#1064#1065#1066#1067#1068#1069#1070#1071#1072#1073#1074#1075#1076#1077#1078#1079#1080#1081#1082#1083#1084#1085#1086#1087#1088#1089#1090#1091#1092#1093#1094#1095#1096#1097#1098#1099#1100#1101#1102#1103#1104#1105#1106#1107#1108#1109#1110#1111#1112#1113#1114#1115#1116#1117#1118#1119#1120#1121#1122#1123#1124#1125#1126#1127#1128#1129#1130#1131#1132#1133#1134#1135#1136#1137#1138#1139#1140#1141#1142#1143#1144#1145#1146#1147#1148#1149#1150#1151#1152#1153#1154#1155#1156#1157#1158#1159#1160#1161#1162#1163#1164#1165#1166#1167#1168#1169#1170#1171#1172#1173#1174#1175#1176#1177#1178#1179#1180#1181#1182#1183#1184#1185#1186#1187#1188#1189#1190#1191#1192#1193#1194#1195#1196#1197#1198#1199#1200#1201#1202#1203#1204#1205#1206#1207#1208#1209#1210#1211#1212#1213#1214#1215#1216#1217#1218#1219#1220#1221#1222#1223#1224#1225#1226#1227#1228#1229#1230#1231#1232#1233#1234#1235#1236#1237#1238#1239#1240#1241#1242#1243#1244#1245#1246#1247#1248#1249#1250#1251#1252#1253#1254#1255#1256#1257#1258#1259#1260#1261#1262#1263#1264#1265#1266#1267#1268#1269#1270#1271#1272#1273#1274#1275#1276#1277#1278#1279#1280#1281#1282#1283#1284#1285#1286#1287#1288#1289#1290#1291#1292#1293#1294#1295#1296#1297#1298#1299#1300#1301#1302#1303#1304#1305#1306#1307#1308#1309#1310#1311#1312#1313#1314#1315#1316#1317#1318#1319#1320#1321#1322#1323#1324#1325#1326#1327#1328#1329#1330#1331#1332#1333#1334#1335#1336#1337#1338#1339#1340#1341#1342#1343#1344#1345#1346#1347#1348#1349#1350#1351#1352#1353#1354#1355#1356#1357#1358#1359#1360#1361#1362#1363#1364#1365#1366#1367#1368#1369#1370#1371#1372#1373#1374#1375#1376#1377#1378#1379#1380#1381#1382#1383#1384#1385#1386#1387#1388#1389#1390#1391#1392#1393#1394#1395#1396#1397#1398#1399#1400#1401#1402#1403#1404#1405#1406#1407#1408#1409#1410#1411#1412#1413#1414#1415#1416#1417#1418#1419#1420#1421#1422#1423#1424#1425#1426#1427#1428#1429#1430#1431#1432#1433#1434#1435#1436#1437#1438#1439#1440#1441#1442#1443#1444#1445#1446#1447#1448#1449#1450#1451#1452#1453#1454#1455#1456#1457#1458#1459#1460#1461#1462#1463#1464#1465#1466#1467#1468#1469#1470#1471#1472#1473#1474#1475#1476#1477#1478#1479#1480#1481#1482#1483#1484#1485#1486#1487#1488#1489#1490#1491#1492#1493#1494#1495#1496#1497#1498#1499#1500#1501#1502#1503#1504#1505#1506#1507#1508#1509#1510#1511#1512#1513#1514#1515#1516#1517#1518#1519#1520#1521#1522#1523#1524#1525#1526#1527#1528#1529#1530#1531#1532#1533#1534#1535#1536#1537#1538#1539#1540#1541#1542#1543#1544#1545#1546#1547#1548#1549#1550#1551#1552#1553#1554#1555#1556#1557#1558#1559#1560#1561#1562#1563#1564#1565#1566#1567#1568#1569#1570#1571#1572#1573#1574#1575#1576#1577#1578#1579#1580#1581#1582#1583#1584#1585#1586#1587#1588#1589#1590#1591#1592#1593#1594#1595#1596#1597#1598#1599#1600#1601#1602#1603#1604#1605#1606#1607#1608#1609#1610#1611#1612#1613#1614#1615#1616#1617#1618#1619#1620#1621#1622#1623#1624#1625#1626#1627#1628#1629#1630#1631#1632#1633#1634#1635#1636#1637#1638#1639#1640#1641#1642#1643#1644#1645#1646#1647#1648#1649#1650#1651#1652#1653#1654#1655#1656#1657#1658#1659#1660#1661#1662#1663#1664#1665#1666#1667#1668#1669#1670#1671#1672#1673#1674#1675#1676#1677#1678#1679#1680#1681#1682#1683#1684#1685#1686#1687#1688#1689#1690#1691#1692#1693#1694#1695#1696#1697#1698#1699#1700#1701#1702#1703#1704#1705#1706#1707#1708#1709#1710#1711#1712#1713#1714#1715#1716#1717#1718#1719#1720#1721#1722#1723#1724#1725#1726#1727#1728#1729#1730#1731#1732#1733#1734#1735#1736#1737#1738#1739#1740#1741#1742#1743#1744#1745#1746#1747#1748#1749#1750#1751#1752#1753#1754#1755#1756#1757#1758#1759#1760#1761#1762#1763#1764#1765#1766#1767#1768#1769#1770#1771#1772#1773#1774#1775#1776#1777#1778#1779#1780#1781#1782#1783#1784#1785#1786#1787#1788#1789#1790#1791#1792#1793#1794#1795#1796#1797#1798#1799#1800#1801#1802#1803#1804#1805#1806#1807#1808#1809#1810#1811#1812#1813#1814#1815#1816#1817#1818#1819#1820#1821#1822#1823#1824#1825#1826#1827#1828#1829#1830#1831#1832#1833#1834#1835#1836#1837#1838#1839#1840#1841#1842#1843#1844#1845#1846#1847#1848#1849#1850#1851#1852#1853#1854#1855#1856#1857#1858#1859#1860#1861#1862#1863#1864#1865#1866#1867#1868#1869#1870#1871#1872#1873#1874#1875#1876#1877#1878#1879#1880#1881#1882#1883#1884#1885#1886#1887#1888#1889#1890#1891#1892#1893#1894#1895#1896#1897#1898#1899#1900#1901#1902#1903#1904#1905#1906#1907#1908#1909#1910#1911#1912#1913#1914#1915#1916#1917#1918#1919#1920#1921#1922#1923#1924#1925#1926#1927#1928#1929#1930#1931#1932#1933#1934#1935#1936#1937#1938#1939#1940#1941#1942#1943#1944#1945#1946#1947#1948#1949#1950#1951#1952#1953#1954#1955#1956#1957#1958#1959#1960#1961#1962#1963#1964#1965#1966#1967#1968#1969#1970#1971#1972#1973#1974#1975#1976#1977#1978#1979#1980#1981#1982#1983#1984#1985#1986#1987#1988#1989#1990#1991#1992#1993#1994#1995#1996#1997#1998#1999#2000#2001#2002#2003#2004#2005#2006#2007#2008#2009#2010#2011#2012#2013#2014#2015#2016#2017#2018#2019#2020#2021#2022#2023#2024#2025#2026#2027#2028#2029#2030#2031#2032#2033#2034#2035#2036#2037#2038#2039#2040#2041#2042#2043#2044#2045#2046#2047#2048#2049#2050#2051#2052#2053#2054#2055#2056#2057#2058#2059#2060#2061#2062#2063#2064#2065#2066#2067#2068#2069#2070#2071#2072#2073#2074#2075#2076#2077#2078#2079#2080#2081#2082#2083#2084#2085#2086#2087#2088#2089#2090#2091#2092#2093#2094#2095#2096#2097#2098#2099#2100#2101#2102#2103#2104#2105#2106#2107#2108#2109#2110#2111#2112#2113#2114#2115#2116#2117#2118#2119#2120#2121#2122#2123#2124#2125#2126#2127#2128#2129#2130#2131#2132#2133#2134#2135#2136#2137#2138#2139#2140#2141#2142#2143#2144#2145#2146#2147#2148#2149#2150#2151#2152#2153#2154#2155#2156#2157#2158#2159#2160#2161#2162#2163#2164#2165#2166#2167#2168#2169#2170#2171#2172#2173#2174#2175#2176#2177#2178#2179#2180#2181#2182#2183#2184#2185#2186#2187#2188#2189#2190#2191#2192#2193#2194#2195#2196#2197#2198#2199#2200#2201#2202#2203#2204#2205#2206#2207#2208#2209#2210#2211#2212#2213#2214#2215#2216#2217#2218#2219#2220#2221#2222#2223#2224#2225#2226#2227#2228#2229#2230#2231#2232#2233#2234#2235#2236#2237#2238#2239#2240#2241#2242#2243#2244#2245#2246#2247#2248#2249#2250#2251#2252#2253#2254#2255#2256#2257#2258#2259#2260#2261#2262#2263#2264#2265#2266#2267#2268#2269#2270#2271#2272#2273#2274#2275#2276#2277#2278#2279#2280#2281#2282#2283#2284#2285#2286#2287#2288#2289#2290#2291#2292#2293#2294#2295#2296#2297#2298#2299#2300#2301#2302#2303#2304#2305#2306#2307#2308#2309#2310#2311#2312#2313#2314#2315#2316#2317#2318#2319#2320#2321#2322#2323#2324#2325#2326#2327#2328#2329#2330#2331#2332#2333#2334#2335#2336#2337#2338#2339#2340#2341#2342#2343#2344#2345#2346#2347#2348#2349#2350#2351#2352#2353#2354#2355#2356#2357#2358#2359#2360#2361#2362#2363#2364#2365#2366#2367#2368#2369#2370#2371#2372#2373#2374#2375#2376#2377#2378#2379#2380#2381#2382#2383#2384#2385#2386#2387#2388#2389#2390#2391#2392#2393#2394#2395#2396#2397#2398#2399#2400#2401#2402#2403#2404#2405#2406#2407#2408#2409#2410#2411#2412#2413#2414#2415#2416#2417#2418#2419#2420#2421#2422#2423#2424#2425#2426#2427#2428#2429#2430#2431#2432#2433#2434#2435#2436#2437#2438#2439#2440#2441#2442#2443#2444#2445#2446#2447#2448#2449#2450#2451#2452#2453#2454#2455#2456#2457#2458#2459#2460#2461#2462#2463#2464#2465#2466#2467#2468#2469#2470#2471#2472#2473#2474#2475#2476#2477#2478#2479#2480#2481#2482#2483#2484#2485#2486#2487#2488#2489#2490#2491#2492#2493#2494#2495#2496#2497#2498#2499#2500#2501#2502#2503#2504#2505#2506#2507#2508#2509#2510#2511#2512#2513#2514#2515#2516#2517#2518#2519#2520#2521#2522#2523#2524#2525#2526#2527#2528#2529#2530#2531#2532#2533#2534#2535#2536#2537#2538#2539#2540#2541#2542#2543#2544#2545#2546#2547#2548#2549#2550#2551#2552#2553#2554#2555#2556#2557#2558#2559#2560#2561#2562#2563#2564#2565#2566#2567#2568#2569#2570#2571#2572#2573#2574#2575#2576#2577#2578#2579#2580#2581#2582#2583#2584#2585#2586#2587#2588#2589#2590#2591#2592
```

```

5181 \def\markdownRendererDlBeginTightPrototype{}%
5182 \def\markdownRendererDlItemPrototype#1{#1}%
5183 \def\markdownRendererDlItemEndPrototype{}%
5184 \def\markdownRendererDlDefinitionBeginPrototype{}%
5185 \def\markdownRendererDlDefinitionEndPrototype{\par}%
5186 \def\markdownRendererDlEndPrototype{}%
5187 \def\markdownRendererDlEndTightPrototype{}%
5188 \def\markdownRendererEmphasisPrototype#1{\it#1}%
5189 \def\markdownRendererStrongEmphasisPrototype#1{\bf#1}%
5190 \def\markdownRendererBlockQuoteBeginPrototype{\par\begingroup\it}%
5191 \def\markdownRendererBlockQuoteEndPrototype{\endgroup\par}%
5192 \def\markdownRendererInputVerbatimPrototype#1{%
5193 \par{\tt\input#1\relax{}}\par}%
5194 \def\markdownRendererInputFencedCodePrototype#1#2{%
5195 \markdownRendererInputVerbatimPrototype{#1}}%
5196 \def\markdownRendererHeadingOnePrototype#1{#1}%
5197 \def\markdownRendererHeadingTwoPrototype#1{#1}%
5198 \def\markdownRendererHeadingThreePrototype#1{#1}%
5199 \def\markdownRendererHeadingFourPrototype#1{#1}%
5200 \def\markdownRendererHeadingFivePrototype#1{#1}%
5201 \def\markdownRendererHeadingSixPrototype#1{#1}%
5202 \def\markdownRendererHorizontalRulePrototype{}%
5203 \def\markdownRendererFootnotePrototype#1{#1}%
5204 \def\markdownRendererCitePrototype#1{}%
5205 \def\markdownRendererTextCitePrototype#1{}%
5206 \def\markdownRendererTickedBoxPrototype{[X]}%
5207 \def\markdownRendererHalfTickedBoxPrototype{[/]}%
5208 \def\markdownRendererUntickedBoxPrototype{[]}%

```

### 3.2.4 Lua Snippets

The `\markdownLuaOptions` macro expands to a Lua table that contains the plain  $\TeX$  options (see Section 2.2.2 on page 25) in a format recognized by Lua (see Section 2.1.2 on page 7).

```

5209 \def\markdownLuaOptions{%
5210 \ifx\markdownOptionBlankBeforeBlockquote\undefined\else
5211 blankBeforeBlockquote = \markdownOptionBlankBeforeBlockquote,
5212 \fi
5213 \ifx\markdownOptionBlankBeforeCodeFence\undefined\else
5214 blankBeforeCodeFence = \markdownOptionBlankBeforeCodeFence,
5215 \fi
5216 \ifx\markdownOptionBlankBeforeHeading\undefined\else
5217 blankBeforeHeading = \markdownOptionBlankBeforeHeading,
5218 \fi
5219 \ifx\markdownOptionBreakableBlockquotes\undefined\else
5220 breakableBlockquotes = \markdownOptionBreakableBlockquotes,

```

```

5221 \fi
5222 cacheDir = "\markdownOptionCacheDir",
5223 \ifx\markdownOptionCitations\undefined\else
5224 citations = \markdownOptionCitations,
5225 \fi
5226 \ifx\markdownOptionCitationNbsps\undefined\else
5227 citationNbsps = \markdownOptionCitationNbsps,
5228 \fi
5229 \ifx\markdownOptionCodeSpans\undefined\else
5230 codeSpans = \markdownOptionCodeSpans,
5231 \fi
5232 \ifx\markdownOptionContentBlocks\undefined\else
5233 contentBlocks = \markdownOptionContentBlocks,
5234 \fi
5235 \ifx\markdownOptionContentBlocksLanguageMap\undefined\else
5236 contentBlocksLanguageMap =
5237 "\markdownOptionContentBlocksLanguageMap",
5238 \fi
5239 \ifx\markdownOptionDefinitionLists\undefined\else
5240 definitionLists = \markdownOptionDefinitionLists,
5241 \fi
5242 \ifx\markdownOptionEagerCache\undefined\else
5243 eagerCache = \markdownOptionEagerCache,
5244 \fi
5245 \ifx\markdownOptionFinalizeCache\undefined\else
5246 finalizeCache = \markdownOptionFinalizeCache,
5247 \fi
5248 frozenCacheFileName = "\markdownOptionFrozenCacheFileName",
5249 frozenCacheCounter = \the\markdownFrozenCacheCounter,
5250 \ifx\markdownOptionFootnotes\undefined\else
5251 footnotes = \markdownOptionFootnotes,
5252 \fi
5253 \ifx\markdownOptionFencedCode\undefined\else
5254 fencedCode = \markdownOptionFencedCode,
5255 \fi
5256 \ifx\markdownOptionHardLineBreaks\undefined\else
5257 hardLineBreaks = \markdownOptionHardLineBreaks,
5258 \fi
5259 \ifx\markdownOptionHashEnumerators\undefined\else
5260 hashEnumerators = \markdownOptionHashEnumerators,
5261 \fi
5262 \ifx\markdownOptionHeaderAttributes\undefined\else
5263 headerAttributes = \markdownOptionHeaderAttributes,
5264 \fi
5265 \ifx\markdownOptionHtml\undefined\else
5266 html = \markdownOptionHtml,
5267 \fi

```

```

5268 \ifx\markdownOptionHybrid\undefined\else
5269 hybrid = \markdownOptionHybrid,
5270 \fi
5271 \ifx\markdownOptionInlineFootnotes\undefined\else
5272 inlineFootnotes = \markdownOptionInlineFootnotes,
5273 \fi
5274 \ifx\markdownOptionJekyllData\undefined\else
5275 jekyllData = \markdownOptionJekyllData,
5276 \fi
5277 \ifx\markdownOptionPipeTables\undefined\else
5278 pipeTables = \markdownOptionPipeTables,
5279 \fi
5280 \ifx\markdownOptionPreserveTabs\undefined\else
5281 preserveTabs = \markdownOptionPreserveTabs,
5282 \fi
5283 \ifx\markdownOptionShiftHeadings\undefined\else
5284 shiftHeadings = "\markdownOptionShiftHeadings",
5285 \fi
5286 \ifx\markdownOptionSlice\undefined\else
5287 slice = "\markdownOptionSlice",
5288 \fi
5289 \ifx\markdownOptionSmartEllipses\undefined\else
5290 smartEllipses = \markdownOptionSmartEllipses,
5291 \fi
5292 \ifx\markdownOptionStartNumber\undefined\else
5293 startNumber = \markdownOptionStartNumber,
5294 \fi
5295 \ifx\markdownOptionStripIndent\undefined\else
5296 stripIndent = \markdownOptionStripIndent,
5297 \fi
5298 \ifx\markdownOptionTableCaptions\undefined\else
5299 tableCaptions = \markdownOptionTableCaptions,
5300 \fi
5301 \ifx\markdownOptionTaskLists\undefined\else
5302 taskLists = \markdownOptionTaskLists,
5303 \fi
5304 \ifx\markdownOptionTeXComments\undefined\else
5305 texComments = \markdownOptionTeXComments,
5306 \fi
5307 \ifx\markdownOptionTightLists\undefined\else
5308 tightLists = \markdownOptionTightLists,
5309 \fi
5310 \ifx\markdownOptionUnderscores\undefined\else
5311 underscores = \markdownOptionUnderscores,
5312 \fi}
5313 }%

```

The `\markdownPrepare` macro contains the Lua code that is executed prior to

any conversion from markdown to plain T<sub>E</sub>X. It exposes the `convert` function for the use by any further Lua code.

```
5314 \def\markdownPrepare{%
```

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

```
5315 local lfs = require("lfs")
5316 local cacheDir = "\markdownOptionCacheDir"
5317 if not lfs.isdir(cacheDir) then
5318 assert(lfs.mkdir(cacheDir))
5319 end
```

Next, load the `markdown` module and create a converter function using the plain T<sub>E</sub>X options, which were serialized to a Lua table via the `\markdownLuaOptions` macro.

```
5320 local md = require("markdown")
5321 local convert = md.new(\markdownLuaOptions)
5322 }%
```

### 3.2.5 Buffering Markdown Input

The `\markdownIfOption{<name>}{<iftrue>}{<iffalse>}` macro is provided for testing, whether the value of `\markdownOption<name>` is `true`. If the value is `true`, then `<iftrue>` is expanded, otherwise `<iffalse>` is expanded.

```
5323 \def\markdownIfOption#1#2#3{%
5324 \begingroup
5325 \def\next{true}%
5326 \expandafter\ifx\csname markdownOption#1\endcsname\next
5327 \endgroup#2\else\endgroup#3\fi}%
```

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.

```
5328 \csname newread\endcsname\markdownInputFileStream
5329 \csname newwrite\endcsname\markdownOutputFileStream
```

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

```
5330 \begingroup
5331 \catcode\^^I=12%
5332 \gdef\markdownReadAndConvertTab{^^I}%
5333 \endgroup
```

The `\markdownReadAndConvert` macro is largely a rewrite of the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> `\filecontents` macro to plain T<sub>E</sub>X.

```
5334 \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 (‘so that we can remove percent signs from the beginning of lines when `\markdownOptionStripPercentSigns` is enabled.

```
5335 \catcode\^^M=13%
5336 \catcode\^^I=13%
5337 \catcode|=0%
5338 \catcode\|=12%
5339 |catcode`=14%
5340 |catcode`|=12@
5341 |gdef|markdownReadAndConvert#1#2{@
5342 |begingroup@
```

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

```
5343 |markdownIfOption{FrozenCache}{@
5344 |immediate|openout|markdownOutputFileStream@
5345 |markdownOptionInputTempFileName|relax@
5346 |markdownInfo{Buffering markdown input into the temporary @
5347 input file "|markdownOptionInputTempFileName" and scanning @
5348 for the closing token sequence "#1"}@
5349 }@
```

Locally change the category of the special plain TeX characters to *other* in order to prevent unwanted interpretation of the input. Change also the category of the space character, so that we can retrieve it unaltered.

```
5350 |def|do##1{|catcode`##1=12}|dospecials@
5351 |catcode`=12@
5352 |markdownMakeOther@
```

The `\markdownReadAndConvertStripPercentSigns` macro will process the individual lines of output, stripping away leading percent signs (`\mref{markdownOptionStripPercentSigns`

```
5353 |def|markdownReadAndConvertStripPercentSign##1{@
5354 |markdownIfOption{StripPercentSigns}{@
5355 |if##1%@
5356 |expandafter|expandafter|expandafter@
5357 |markdownReadAndConvertProcessLine@
5358 |else@
5359 |expandafter|expandafter|expandafter@
5360 |markdownReadAndConvertProcessLine@
5361 |expandafter|expandafter|expandafter##1@
5362 |fi@
5363 }{@
5364 |expandafter@
5365 |markdownReadAndConvertProcessLine@
5366 |expandafter##1@
5367 }@
5368 }@
```

The `\markdownReadAndConvertProcessLine` macro will process the individual lines of output. Notice the use of the comments (`@`) to ensure that the entire macro is at a single line and therefore no (active) newline symbols (`^^M`) are produced.

```
5369 |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.

```
5370 |ifx|relax##3|relax@
5371 |markdownIfOption{FrozenCache}{-}{@
5372 |immediate|write|markdownOutputFileStream{##1}@
5373 }@
5374 |else@
```

When the ending token sequence appears in the line, make the next newline character close the `\markdownOptionInputTempFileName` file, return the character categories back to the former state, convert the `\markdownOptionInputTempFileName` file from markdown to plain TEX, `\input` the result of the conversion, and expand the ending control sequence.

```
5375 |def^^M{@
5376 |markdownInfo{The ending token sequence was found}@
5377 |markdownIfOption{FrozenCache}{-}{@
5378 |immediate|closeout|markdownOutputFileStream@
5379 }@
5380 |endgroup@
5381 |markdownInput{@
5382 |markdownOptionOutputDir@
5383 /|markdownOptionInputTempFileName@
5384 }@
5385 #2}@
5386 |fi@
```

Repeat with the next line.

```
5387 ^^M}@
```

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

```
5388 |catcode`|^I=13@
5389 |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.

```
5390 |catcode`|^M=13@
5391 |def^^M##1^^M{@
```



```

5392 |def^^M####1^^M{@
5393 |markdownReadAndConvertStripPercentSign####1#1#1|relax}@
5394 ^^M}@
5395 ^^M}@

```

Reset the character categories back to the former state.

```
5396 |endgroup
```

### 3.2.6 Lua Shell Escape Bridge

The following  $\TeX$  code is intended for  $\TeX$  engines that do not provide direct access to Lua, but expose the shell of the operating system. This corresponds to the `\markdownMode` values of 0 and 1.

The `\markdownLuaExecute` macro defined here and in Section 3.2.7 on page 171 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].

```

5397 \ifnum\markdownMode<2\relax
5398 \ifnum\markdownMode=0\relax
5399 \markdownInfo{Using mode 0: Shell escape via write18}%
5400 \else
5401 \markdownInfo{Using mode 1: Shell escape via os.execute}%
5402 \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.

```

5403 \ifx\pdfshellescape\undefined
5404 \ifx\shellescape\undefined
5405 \ifnum\markdownMode=0\relax
5406 \def\markdownExecuteShellEscape{1}%
5407 \else
5408 \def\markdownExecuteShellEscape{%
5409 \directlua{tex.sprint(status.shell_escape or "1")}}%
5410 \fi
5411 \else
5412 \let\markdownExecuteShellEscape\shellescape
5413 \fi
5414 \else
5415 \let\markdownExecuteShellEscape\pdfshellescape
5416 \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.

```
5417 \ifnum\markdownMode=0\relax
5418 \def\markdownExecuteDirect#1{\immediate\write18{#1}}%
5419 \else
5420 \def\markdownExecuteDirect#1{%
5421 \directlua{os.execute("\luaescapestring{#1}")}%
5422 \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.

```
5423 \def\markdownExecute#1{%
5424 \ifnum\markdownExecuteShellEscape=1\relax
5425 \markdownExecuteDirect{#1}%
5426 \else
5427 \markdownError{I can not access the shell}{Either run the TeX
5428 compiler with the --shell-escape or the --enable-write18 flag,
5429 or set shell_escape=t in the texmf.cnf file}%
5430 \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.

```
5431 \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.

```
5432 \catcode`\|=0%
5433 \catcode`\|=12%
5434 \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.

```
5435 |immediate|openout|markdownOutputFileStream=%
5436 |markdownOptionHelperScriptFileName
5437 |markdownInfo{Writing a helper Lua script to the file
5438 "|markdownOptionHelperScriptFileName"}%
5439 |immediate|write|markdownOutputFileStream{%
5440 local ran_ok, error = pcall(function()
5441 local ran_ok, kpse = pcall(require, "kpse")
5442 if ran_ok then kpse.set_program_name("luatex") end
5443 #1
5444 end)
```

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

```
5445 if not ran_ok then
5446 local file = io.open("%
5447 |markdownOptionOutputDir
5448 /|markdownOptionErrorTempFileName", "w")
5449 if file then
5450 file:write(error .. "\n")
5451 file:close()
5452 end
5453 print('\markdownError{An error was encountered while executing
5454 Lua code}{For further clues, examine the file
5455 "|markdownOptionOutputDir
5456 /|markdownOptionErrorTempFileName"}')
5457 end}%
5458 |immediate|closeout|markdownOutputFileStream
```

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

```
5459 |markdownInfo{Executing a helper Lua script from the file
5460 "|markdownOptionHelperScriptFileName" and storing the result in the
5461 file "|markdownOptionOutputTempFileName"}%
5462 |markdownExecute{texlua "|markdownOptionOutputDir
5463 /|markdownOptionHelperScriptFileName" > %
5464 "|markdownOptionOutputDir
5465 /|markdownOptionOutputTempFileName"}%
```

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

```
5466 |input|markdownOptionOutputTempFileName|relax}%
5467 |endgroup
```

### 3.2.7 Direct Lua Access

The following `TeX` code is intended for `TeX` engines that provide direct access to Lua (`LuaTeX`). The macro `\markdownLuaExecute` defined here and in Section 3.2.6 on page 169 are meant to be indistinguishable to the remaining code. This corresponds to the `\markdownMode` value of 2.

```
5468 \else
5469 \markdownInfo{Using mode 2: Direct Lua access}%
```

The direct Lua access version of the `\markdownLuaExecute` macro is defined in terms of the `\directlua` primitive. The `print` function is set as an alias to the `\tex.print` method in order to mimic the behaviour of the `\markdownLuaExecute` definition from Section 3.2.6 on page 169,

```
5470 \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.

```

5471 \catcode`|=0%
5472 \catcode`\|=12%
5473 |gdef|markdownLuaExecute#1{%
5474 |directlua{%
5475 local function print(input)
5476 local output = {}
5477 for line in input:gmatch("[^\r\n]+") do
5478 table.insert(output, line)
5479 end
5480 tex.print(output)
5481 end
5482 #1
5483 }%
5484 }%
5485 |endgroup
5486 \fi

```

### 3.2.8 Typesetting Markdown

The `\markdownInput` macro uses an implementation of the `\markdownLuaExecute` macro to convert the contents of the file whose filename it has received as its single argument from markdown to plain TeX.

```
5487 \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.

```

5488 \catcode`|=0%
5489 \catcode`\|=12%
5490 |gdef|markdownInput#1{%

```

Change the category code of the percent sign (‘of the `hybrid` Lua option or a malevolent actor can’t produce TeX comments in the plain TeX output of the Markdown package.

```

5491 |begingroup
5492 |catcode`|=12

```

If we are reading from the frozen cache, input it, expand the corresponding `\markdownFrozenCache<number>` macro, and increment `\markdownFrozenCacheCounter`.

```

5493 |markdownIfOption{FrozenCache}{%
5494 |ifnum|markdownFrozenCacheCounter=0|relax
5495 |markdownInfo{Reading frozen cache from
5496 "|markdownOptionFrozenCacheFileName"%
5497 |input|markdownOptionFrozenCacheFileName|relax
5498 |fi
5499 |markdownInfo{Including markdown document number

```

```

5500 " |the|markdownFrozenCacheCounter" from frozen cache}%
5501 |csname markdownFrozenCache|the|markdownFrozenCacheCounter|endcsname
5502 |global|advance|markdownFrozenCacheCounter by 1|relax
5503 }-%
5504 |markdownInfo{Including markdown document "#1"}%

```

Attempt to open the markdown document to record it in the `.log` and `.fls` files. This allows external programs such as `LATEXMk` to track changes to the markdown document.

```

5505 |openin|markdownInputFileStream#1
5506 |closein|markdownInputFileStream
5507 |markdownLuaExecute{%
5508 |markdownPrepare
5509 local file = assert(io.open("#1", "r"),
5510 [[could not open file "#1" for reading]])
5511 local input = assert(file:read("*a"))
5512 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.

```

5513 print(convert(input:gsub("\r\n?", "\n") .. "\n"))}%

```

If we are finalizing the frozen cache, increment `\markdownFrozenCacheCounter`.

```

5514 |markdownIfOption{FinalizeCache}{%
5515 |global|advance|markdownFrozenCacheCounter by 1|relax
5516 }%
5517 }%
5518 |endgroup
5519 }%
5520 |endgroup

```

### 3.3 `LATEX` Implementation

The `LATEX` implementation makes use of the fact that, apart from some subtle differences, `LATEX` implements the majority of the plain `TEX` format [9, Section 9]. As a consequence, we can directly reuse the existing plain `TEX` implementation.

The `LATEX` implementation redefines the plain `TEX` logging macros (see Section 3.2.1 on page 161) to use the `LATEX` `\PackageInfo`, `\PackageWarning`, and `\PackageError` macros.

```

5521 \newcommand\markdownInfo[1]{\PackageInfo{markdown}{#1}}%
5522 \newcommand\markdownWarning[1]{\PackageWarning{markdown}{#1}}%
5523 \newcommand\markdownError[2]{\PackageError{markdown}{#1}{#2.}}%
5524 \input markdown/markdown
5525 \def\markdownVersionSpace{ }%
5526 \ProvidesPackage{markdown}[\markdownLastModified\markdownVersionSpace v%
5527 \markdownVersion\markdownVersionSpace markdown renderer]%

```

### 3.3.1 Logging Facilities

The  $\LaTeX$  implementation redefines the plain  $\TeX$  logging macros (see Section 3.2.1 on page 161) 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 on page 42).

```
5528 \let\markdownInputPlainTeX\markdownInput
5529 \renewcommand\markdownInput[2] [] {%
5530 \begingroup
5531 \markdownSetup{#1}%
5532 \markdownInputPlainTeX{#2}%
5533 \endgroup}%
```

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

```
5534 \renewenvironment{markdown}{%
5535 \markdownReadAndConvert@markdown{}}{%
5536 \markdownEnd}%
5537 \renewenvironment{markdown*}[1] {%
5538 \markdownSetup{#1}%
5539 \markdownReadAndConvert@markdown*}{%
5540 \markdownEnd}%
5541 \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*.

```
5542 \catcode`\|=0\catcode`\<=1\catcode`\>=2%
5543 \catcode`\|=12\catcode`\{=12\catcode`\}=12%
5544 |gdef|markdownReadAndConvert@markdown#1<%
5545 |markdownReadAndConvert<\end{markdown#1}>%
5546 <|end<markdown#1>>>%
5547 |endgroup
```

### 3.3.3 Options

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

```
5548 \DeclareOption*{%
5549 \expandafter\markdownSetup\expandafter{\CurrentOption}}%
```

```
5550 \ProcessOptions\relax
```

After processing the options, activate the `renderers` and `rendererPrototypes` keys.

```
5551 \define@key{markdownOptions}{renderers}{%
5552 \setkeys{markdownRenderers}{#1}%
5553 \def\KV@prefix{KV@markdownOptions@}}%
5554 \define@key{markdownOptions}{rendererPrototypes}{%
5555 \setkeys{markdownRendererPrototypes}{#1}%
5556 \def\KV@prefix{KV@markdownOptions@}}%
```

**3.3.3.1 L<sup>A</sup>T<sub>E</sub>X Themes** This section implements example themes provided with the Markdown package.

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

```
5557 \markdownSetup{fencedCode}%
```

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

```
5558 \RequirePackage{ifthen,grffile}
```

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

```
5559 \let\markdown@witiko@dot@oldRendererInputFencedCodePrototype
5560 \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 T<sub>E</sub>X option is disabled and the code block has not been previously typeset:

```
5561 \renewcommand\markdownRendererInputFencedCode[2]{%
5562 \def\next##1 ##2\relax{%
5563 \ifthenelse{\equal{##1}{dot}}{%
5564 \markdownIfOption{FrozenCache}{}{%
5565 \immediate\write18{%
5566 if ! test -e #1.pdf.source || ! diff #1 #1.pdf.source;
5567 then
5568 dot -Tpdf -o #1.pdf #1;
5569 cp #1 #1.pdf.source;
5570 fi}}%
5571 \next##1 ##2\relax}%
5572 }
```

We include the typeset image using the image token renderer:

```
5571 \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:

```
5572 }{%
5573 \markdown@witiko@dot@oldRendererInputFencedCodePrototype{#1}{#2}%
5574 }%
5575 }%
5576 \next#2 \relax}%
```

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

```
5577 \let\markdown@witiko@graphicx@http@oldRendererImagePrototype
5578 \markdownRendererImagePrototype
```

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

```
5579 \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.

```
5580 \newcount\markdown@witiko@graphicx@http@counter
5581 \markdown@witiko@graphicx@http@counter=0
5582 \newcommand\markdown@witiko@graphicx@http@filename{%
5583 \markdownOptionCacheDir/witiko_graphicx_http%
5584 .\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.

```
5585 \newcommand\markdown@witiko@graphicx@http@download[2]{%
5586 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:

```
5587 \begingroup
5588 \catcode`\%=12
5589 \catcode`\^^A=14
```

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

```
5590 \global\def\markdownRendererImagePrototype#1#2#3#4{^^A
5591 \begingroup
5592 \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:

```
5593 \markdownIfOption{FrozenCache}{}{^^A
5594 \immediate\write18{^^A
5595 if printf '%s' "#3" | grep -q -E '^https?:';
5596 then
```

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

```
5597 OUTPUT_PREFIX="\markdownOptionCacheDir";
5598 OUTPUT_BODY="$(printf '%s' '#3' | md5sum | cut -d' ' -f1)";
```



```

5599 OUTPUT_SUFFIX="$(printf '%s' '#3' | sed 's/.*[.]/')";
5600 OUTPUT="$OUTPUT_PREFIX/$OUTPUT_BODY.$OUTPUT_SUFFIX";
 The image will be downloaded only if it has not already been downloaded:
5601 if ! [-e "$OUTPUT"];
5602 then
5603 \markdown@witiko@graphicx@http@download{'#3'}{"$OUTPUT"};
5604 printf '%s' "$OUTPUT" > "\filename";
5605 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:

```

5606 else
5607 printf '%s' '#3' > "\filename";
5608 fi}}^^A

```

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

```

5609 \CatchFileDef{\filename}{\filename}{\newlinechar=-1}^^A
5610 \markdown@witiko@graphicx@http@oldRendererImagePrototype^^A
5611 {#1}{#2}{\filename}{#4}^^A
5612 \endgroup
5613 \global\advance\markdown@witiko@graphicx@http@counter by 1\relax}^^A
5614 \endgroup

```

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

```

5615 \renewcommand\markdownRendererTildePrototype{~}%

```

### 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 on page 43), none of it will take effect.

```

5616 \ifmarkdownLaTeXplain\else

```

If the `\markdownOptionTightLists` macro expands to `false`, do not load the `paralist` package. This is necessary for  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$  document classes that do not play nice with `paralist`, such as `beamer`. If the `\markdownOptionTightLists` is undefined and the `beamer` document class is in use, then do not load the `paralist` package either.

```

5617 \RequirePackage{ifthen}
5618
5619 \ifx\markdownOptionTightLists\undefined
5620 \@ifclassloaded{beamer}{}{%
5621 \RequirePackage{paralist}}%
5622 \else
5623 \ifthenelse{\equal{\markdownOptionTightLists}{false}}{}{%
5624 \RequirePackage{paralist}}%

```

5625 \fi

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.

```
5626 \@ifpackageloaded{paralist}{
5627 \markdownSetup{rendererPrototypes={
5628 ulBeginTight = {\begin{compactitem}},
5629 ulEndTight = {\end{compactitem}},
5630 olBeginTight = {\begin{compactenum}},
5631 olEndTight = {\end{compactenum}},
5632 dlBeginTight = {\begin{compactdesc}},
5633 dlEndTight = {\end{compactdesc}}}}
5634 }{
5635 \markdownSetup{rendererPrototypes={
5636 ulBeginTight = {\markdownRendererUlBegin},
5637 ulEndTight = {\markdownRendererUlEnd},
5638 olBeginTight = {\markdownRendererOlBegin},
5639 olEndTight = {\markdownRendererOlEnd},
5640 dlBeginTight = {\markdownRendererDlBegin},
5641 dlEndTight = {\markdownRendererDlEnd}}}}
5642 \RequirePackage{amsmath}
5643 \RequirePackage{amssymb}
5644 \RequirePackage{csvsimple}
5645 \RequirePackage{fancyvrb}
5646 \RequirePackage{graphicx}
5647 \markdownSetup{rendererPrototypes={
5648 lineBreak = {\},
5649 leftBrace = {\textbraceleft},
5650 rightBrace = {\textbraceright},
5651 dollarSign = {\textdollar},
5652 underscore = {\textunderscore},
5653 circumflex = {\textasciicircum},
5654 backslash = {\textbackslash},
5655 tilde = {\textasciitilde},
5656 pipe = {\textbar},
5657 codeSpan = {\texttt{#1}},
5658 contentBlock = {%
5659 \ifthenelse{\equal{#1}{csv}}{%
5660 \begin{table}%
5661 \begin{center}%
5662 \csvautotabular{#3}%
5663 \end{center}
5664 \ifx\empty#4\empty\else
5665 \caption{#4}%
5666 \fi
5667 \end{table}%
}
```

```

5668 }{%
5669 \ifthenelse{\equal{#1}{tex}}{%
5670 \catcode`\%=14\relax
5671 \input #3\relax
5672 \catcode`\%=12\relax
5673 }{%
5674 \markdownInput{#3}%
5675 }%
5676 }%
5677 },
5678 image = {%
5679 \begin{figure}%
5680 \begin{center}%
5681 \includegraphics{#3}%
5682 \end{center}%
5683 \ifx\empty#4\empty\else
5684 \caption{#4}%
5685 \fi
5686 \label{fig:#1}%
5687 \end{figure}},
5688 ulBegin = {\begin{itemize}},
5689 ulEnd = {\end{itemize}},
5690 olBegin = {\begin{enumerate}},
5691 olItem = {\item{}},
5692 olItemWithNumber = {\item[#1.]},
5693 olEnd = {\end{enumerate}},
5694 dlBegin = {\begin{description}},
5695 dlItem = {\item[#1]},
5696 dlEnd = {\end{description}},
5697 emphasis = {\emph{#1}},
5698 tickedBox = {\\boxtimes},
5699 halfTickedBox = {\\boxdot},
5700 untickedBox = {\\square},
5701 blockQuoteBegin = {\begin{quotation}},
5702 blockQuoteEnd = {\end{quotation}},
5703 inputVerbatim = {\VerbatimInput{#1}},
5704 inputFencedCode = {%
5705 \ifx\relax#2\relax
5706 \VerbatimInput{#1}%
5707 \else
5708 \@ifundefined{minted@code}{%
5709 \@ifundefined{lst@version}{%
5710 \markdownRendererInputFencedCode{#1}{}%

```

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

```

5711 }{%
5712 \lstinputlisting[language=#2]{#1}%
5713 }%

```

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

```
5714 }{%
5715 \inputminted{#2}{#1}%
5716 }%
5717 \fi},
5718 horizontalRule = {\noindent\rule[0.5ex]{\linewidth}{1pt}},
5719 footnote = {\footnote{#1}}}
```

Support the nesting of strong emphasis.

```
5720 \def\markdownLATEXStrongEmphasis#1{%
5721 \IfSubStr\f@series{b}{\textnormal{#1}}{\textbf{#1}}}
5722 \markdownSetup{rendererPrototypes={strongEmphasis={%
5723 \protect\markdownLATEXStrongEmphasis{#1}}}}
```

Support  $\LaTeX$  document classes that do not provide chapters.

```
5724 \@ifundefined{chapter}{%
5725 \markdownSetup{rendererPrototypes = {
5726 headingOne = {\section{#1}},
5727 headingTwo = {\subsection{#1}},
5728 headingThree = {\subsubsection{#1}},
5729 headingFour = {\paragraph{#1}\leavevmode},
5730 headingFive = {\subparagraph{#1}\leavevmode}}}
5731 }{%
5732 \markdownSetup{rendererPrototypes = {
5733 headingOne = {\chapter{#1}},
5734 headingTwo = {\section{#1}},
5735 headingThree = {\subsection{#1}},
5736 headingFour = {\subsubsection{#1}},
5737 headingFive = {\paragraph{#1}\leavevmode},
5738 headingSix = {\subparagraph{#1}\leavevmode}}}
5739 }%
```

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

```
5740 \markdownSetup{
5741 rendererPrototypes = {
5742 ulItem = {%
5743 \futurelet\markdownLaTeXCheckbox\markdownLaTeXUListItem
5744 },
5745 },
5746 }
5747 \def\markdownLaTeXUListItem{%
5748 \ifx\markdownLaTeXCheckbox\markdownRendererTickedBox
5749 \item[\markdownLaTeXCheckbox]%
5750 \expandafter\@gobble
5751 \else
```

```

5752 \ifx\markdownLaTeXCheckbox\markdownRendererHalfTickedBox
5753 \item[\markdownLaTeXCheckbox]%
5754 \expandafter\expandafter\expandafter\@gobble
5755 \else
5756 \ifx\markdownLaTeXCheckbox\markdownRendererUntickedBox
5757 \item[\markdownLaTeXCheckbox]%
5758 \expandafter\expandafter\expandafter\expandafter
5759 \expandafter\expandafter\expandafter\@gobble
5760 \else
5761 \item{}%
5762 \fi
5763 \fi
5764 \fi
5765 }

```

**3.3.4.2 Citations** Here is a basic implementation for citations that uses the L<sup>A</sup>T<sub>E</sub>X `\cite` macro. There are also implementations that use the natbib `\citep`, and `\citet` macros, and the BibL<sup>A</sup>T<sub>E</sub>X `\autocites` and `\textcites` macros. These implementations will be used, when the respective packages are loaded.

```

5766 \newcount\markdownLaTeXCitationsCounter
5767
5768 % Basic implementation
5769 \RequirePackage{gobble}
5770 \def\markdownLaTeXBasicCitations#1#2#3#4#5#6{%
5771 \advance\markdownLaTeXCitationsCounter by 1\relax
5772 \ifx\relax#4\relax
5773 \ifx\relax#5\relax
5774 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5775 \cite{#1#2#6}% Without prenotes and postnotes, just accumulate cites
5776 \expandafter\expandafter\expandafter
5777 \expandafter\expandafter\expandafter\expandafter
5778 \@gobblethree
5779 \fi
5780 \else% Before a postnote (#5), dump the accumulator
5781 \ifx\relax#1\relax\else
5782 \cite{#1}%
5783 \fi
5784 \cite[#5]{#6}%
5785 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5786 \else
5787 \expandafter\expandafter\expandafter
5788 \expandafter\expandafter\expandafter\expandafter
5789 \expandafter\expandafter\expandafter
5790 \expandafter\expandafter\expandafter\expandafter
5791 \markdownLaTeXBasicCitations
5792 \fi

```

```

5793 \expandafter\expandafter\expandafter
5794 \expandafter\expandafter\expandafter\expandafter{%
5795 \expandafter\expandafter\expandafter
5796 \expandafter\expandafter\expandafter\expandafter}%
5797 \expandafter\expandafter\expandafter
5798 \expandafter\expandafter\expandafter\expandafter{%
5799 \expandafter\expandafter\expandafter
5800 \expandafter\expandafter\expandafter\expandafter}%
5801 \expandafter\expandafter\expandafter
5802 \@gobblethree
5803 \fi
5804 \else% Before a prenote (#4), dump the accumulator
5805 \ifx\relax#1\relax\else
5806 \cite{#1}%
5807 \fi
5808 \ifnum\markdownLaTeXCitationsCounter>1\relax
5809 \space % Insert a space before the prenote in later citations
5810 \fi
5811 #4-\expandafter\cite\ifx\relax#5\relax{#6}\else[#5]{#6}\fi
5812 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5813 \else
5814 \expandafter\expandafter\expandafter
5815 \expandafter\expandafter\expandafter\expandafter
5816 \markdownLaTeXBasicCitations
5817 \fi
5818 \expandafter\expandafter\expandafter{%
5819 \expandafter\expandafter\expandafter}%
5820 \expandafter\expandafter\expandafter{%
5821 \expandafter\expandafter\expandafter}%
5822 \expandafter
5823 \@gobblethree
5824 \fi\markdownLaTeXBasicCitations{#1#2#6},}
5825 \let\markdownLaTeXBasicTextCitations\markdownLaTeXBasicCitations
5826
5827 % Natbib implementation
5828 \def\markdownLaTeXNatbibCitations#1#2#3#4#5{%
5829 \advance\markdownLaTeXCitationsCounter by 1\relax
5830 \ifx\relax#3\relax
5831 \ifx\relax#4\relax
5832 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5833 \citep{#1,#5}% Without prenotes and postnotes, just accumulate cites
5834 \expandafter\expandafter\expandafter
5835 \expandafter\expandafter\expandafter\expandafter
5836 \@gobbletwo
5837 \fi
5838 \else% Before a postnote (#4), dump the accumulator
5839 \ifx\relax#1\relax\else

```

```

5840 \citep{#1}%
5841 \fi
5842 \citep[] [#4]{#5}%
5843 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5844 \else
5845 \expandafter\expandafter\expandafter
5846 \expandafter\expandafter\expandafter\expandafter
5847 \expandafter\expandafter\expandafter
5848 \expandafter\expandafter\expandafter\expandafter
5849 \markdownLaTeXNatbibCitations
5850 \fi
5851 \expandafter\expandafter\expandafter
5852 \expandafter\expandafter\expandafter\expandafter{%
5853 \expandafter\expandafter\expandafter
5854 \expandafter\expandafter\expandafter\expandafter}%
5855 \expandafter\expandafter\expandafter
5856 \@gobbletwo
5857 \fi
5858 \else% Before a prenote (#3), dump the accumulator
5859 \ifx\relax#1\relax\relax\else
5860 \citep{#1}%
5861 \fi
5862 \citep[#3] [#4]{#5}%
5863 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5864 \else
5865 \expandafter\expandafter\expandafter
5866 \expandafter\expandafter\expandafter\expandafter
5867 \markdownLaTeXNatbibCitations
5868 \fi
5869 \expandafter\expandafter\expandafter{%
5870 \expandafter\expandafter\expandafter}%
5871 \expandafter
5872 \@gobbletwo
5873 \fi\markdownLaTeXNatbibCitations{#1,#5}}
5874 \def\markdownLaTeXNatbibTextCitations#1#2#3#4#5{%
5875 \advance\markdownLaTeXCitationsCounter by 1\relax
5876 \ifx\relax#3\relax
5877 \ifx\relax#4\relax
5878 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5879 \citet{#1,#5}% Without prenotes and postnotes, just accumulate cites
5880 \expandafter\expandafter\expandafter
5881 \expandafter\expandafter\expandafter\expandafter
5882 \@gobbletwo
5883 \fi
5884 \else% After a prenote or a postnote, dump the accumulator
5885 \ifx\relax#1\relax\else
5886 \citet{#1}%

```

```

5887 \fi
5888 , \citet[#3][#4]{#5}%
5889 \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
5890 ,
5891 \else
5892 \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
5893 ,
5894 \fi
5895 \fi
5896 \expandafter\expandafter\expandafter
5897 \expandafter\expandafter\expandafter\expandafter
5898 \markdownLaTeXNatbibTextCitations
5899 \expandafter\expandafter\expandafter
5900 \expandafter\expandafter\expandafter\expandafter{%
5901 \expandafter\expandafter\expandafter
5902 \expandafter\expandafter\expandafter\expandafter}%
5903 \expandafter\expandafter\expandafter
5904 \@gobbletwo
5905 \fi
5906 \else% After a prenote or a postnote, dump the accumulator
5907 \ifx\relax#1\relax\relax\else
5908 \citet{#1}%
5909 \fi
5910 , \citet[#3][#4]{#5}%
5911 \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
5912 ,
5913 \else
5914 \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
5915 ,
5916 \fi
5917 \fi
5918 \expandafter\expandafter\expandafter
5919 \markdownLaTeXNatbibTextCitations
5920 \expandafter\expandafter\expandafter{%
5921 \expandafter\expandafter\expandafter}%
5922 \expandafter
5923 \@gobbletwo
5924 \fi\markdownLaTeXNatbibTextCitations{#1,#5}}
5925
5926 % BibLaTeX implementation
5927 \def\markdownLaTeXBibLaTeXCitations#1#2#3#4#5{%
5928 \advance\markdownLaTeXCitationsCounter by 1\relax
5929 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5930 \autocites#1[#3][#4]{#5}%
5931 \expandafter\@gobbletwo
5932 \fi\markdownLaTeXBibLaTeXCitations{#1[#3][#4]{#5}}
5933 \def\markdownLaTeXBibLaTeXTextCitations#1#2#3#4#5{%

```



```

5934 \advance\markdownLaTeXCitationsCounter by 1\relax
5935 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5936 \textcites#1[#3][#4]{#5}%
5937 \expandafter\@gobbletwo
5938 \fi\markdownLaTeXBibLaTeXTextCitations{#1[#3][#4]{#5}}
5939
5940 \markdownSetup{rendererPrototypes = {
5941 cite = {%
5942 \markdownLaTeXCitationsCounter=1%
5943 \def\markdownLaTeXCitationsTotal{#1}%
5944 \@ifundefined{autocites}{%
5945 \@ifundefined{citep}{%
5946 \expandafter\expandafter\expandafter
5947 \markdownLaTeXBasicCitations
5948 \expandafter\expandafter\expandafter{%
5949 \expandafter\expandafter\expandafter}%
5950 \expandafter\expandafter\expandafter{%
5951 \expandafter\expandafter\expandafter}%
5952 }{%
5953 \expandafter\expandafter\expandafter
5954 \markdownLaTeXNatbibCitations
5955 \expandafter\expandafter\expandafter{%
5956 \expandafter\expandafter\expandafter}%
5957 }%
5958 }{%
5959 \expandafter\expandafter\expandafter
5960 \markdownLaTeXBibLaTeXCitations
5961 \expandafter{\expandafter}%
5962 }},
5963 textCite = {%
5964 \markdownLaTeXCitationsCounter=1%
5965 \def\markdownLaTeXCitationsTotal{#1}%
5966 \@ifundefined{autocites}{%
5967 \@ifundefined{citep}{%
5968 \expandafter\expandafter\expandafter
5969 \markdownLaTeXBasicTextCitations
5970 \expandafter\expandafter\expandafter{%
5971 \expandafter\expandafter\expandafter}%
5972 \expandafter\expandafter\expandafter{%
5973 \expandafter\expandafter\expandafter}%
5974 }{%
5975 \expandafter\expandafter\expandafter
5976 \markdownLaTeXNatbibTextCitations
5977 \expandafter\expandafter\expandafter{%
5978 \expandafter\expandafter\expandafter}%
5979 }%
5980 }{%

```

```

5981 \expandafter\expandafter\expandafter
5982 \markdownLaTeXBibLaTeXTextCitations
5983 \expandafter{\expandafter}%
5984 }}}

```

**3.3.4.3 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. After the hyperlink has been typeset, we restore the original catcode.

```

5985 \RequirePackage{url}
5986 \def\markdownRendererLinkPrototype{%
5987 \begingroup
5988 \catcode`\#=12
5989 \def\next##1##2##3##4{%
5990 ##1\footnote{%
5991 \ifx\empty##4\empty\else##4: \fi\texttt<\url{##3}\texttt>}%
5992 \endgroup}%
5993 \next}

```

**3.3.4.4 Tables** Here is a basic implementation of tables. If the booktabs package is loaded, then it is used to produce horizontal lines.

```

5994 \newcount\markdownLaTeXRowCount
5995 \newcount\markdownLaTeXRowTotal
5996 \newcount\markdownLaTeXColumnCounter
5997 \newcount\markdownLaTeXColumnTotal
5998 \newtoks\markdownLaTeXTable
5999 \newtoks\markdownLaTeXTableAlignment
6000 \newtoks\markdownLaTeXTableEnd
6001 \@ifpackageloaded{booktabs}{
6002 \let\markdownLaTeXTopRule\toprule
6003 \let\markdownLaTeXMidRule\midrule
6004 \let\markdownLaTeXBottomRule\bottomrule
6005 }{
6006 \let\markdownLaTeXTopRule\hline
6007 \let\markdownLaTeXMidRule\hline
6008 \let\markdownLaTeXBottomRule\hline
6009 }
6010 \markdownSetup{rendererPrototypes={
6011 table = {%
6012 \markdownLaTeXTable={}%
6013 \markdownLaTeXTableAlignment={}%
6014 \markdownLaTeXTableEnd={%
6015 \markdownLaTeXBottomRule
6016 \end{tabular}}}%
6017 \ifx\empty#1\empty\else

```

```

6018 \addto@hook\markdownLaTeXTable{%
6019 \begin{table}
6020 \centering}%
6021 \addto@hook\markdownLaTeXTableEnd{%
6022 \caption{#1}
6023 \end{table}}%
6024 \fi
6025 \addto@hook\markdownLaTeXTable{\begin{tabular}}%
6026 \markdownLaTeXRowCounter=0%
6027 \markdownLaTeXRowTotal=#2%
6028 \markdownLaTeXColumnTotal=#3%
6029 \markdownLaTeXRenderTableRow
6030 }
6031 }}
6032 \def\markdownLaTeXRenderTableRow#1{%
6033 \markdownLaTeXColumnCounter=0%
6034 \ifnum\markdownLaTeXRowCounter=0\relax
6035 \markdownLaTeXReadAlignments#1%
6036 \markdownLaTeXTable=\expandafter\expandafter\expandafter{%
6037 \expandafter\the\expandafter\markdownLaTeXTable\expandafter{%
6038 \the\markdownLaTeXTableAlignment}}%
6039 \addto@hook\markdownLaTeXTable{\markdownLaTeXTopRule}%
6040 \else
6041 \markdownLaTeXRenderTableCell#1%
6042 \fi
6043 \ifnum\markdownLaTeXRowCounter=1\relax
6044 \addto@hook\markdownLaTeXTable\markdownLaTeXMidRule
6045 \fi
6046 \advance\markdownLaTeXRowCounter by 1\relax
6047 \ifnum\markdownLaTeXRowCounter>\markdownLaTeXRowTotal\relax
6048 \the\markdownLaTeXTable
6049 \the\markdownLaTeXTableEnd
6050 \expandafter\@gobble
6051 \fi\markdownLaTeXRenderTableRow}
6052 \def\markdownLaTeXReadAlignments#1{%
6053 \advance\markdownLaTeXColumnCounter by 1\relax
6054 \if#1d%
6055 \addto@hook\markdownLaTeXTableAlignment{1}%
6056 \else
6057 \addto@hook\markdownLaTeXTableAlignment{#1}%
6058 \fi
6059 \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax\else
6060 \expandafter\@gobble
6061 \fi\markdownLaTeXReadAlignments}
6062 \def\markdownLaTeXRenderTableCell#1{%
6063 \advance\markdownLaTeXColumnCounter by 1\relax
6064 \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax

```

```

6065 \addto@hook\markdownLaTeXTable{#1&}%
6066 \else
6067 \addto@hook\markdownLaTeXTable{#1\\}%
6068 \expandafter\@gobble
6069 \fi\markdownLaTeXRenderTableCell}
6070 \fi

```

**3.3.4.5 YAML Metadata** To parse the YAML metadata we will use the `expl3` language from the `LATEX3` kernel.

```

6071 \RequirePackage{expl3}
6072 \ExplSyntaxOn

```

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  $p$ .

```

6073 \seq_new:N \g_@@_jekyll_data_datatypes_seq
6074 \tl_const:Nn \c_@@_jekyll_data_sequence_tl { sequence }
6075 \tl_const:Nn \c_@@_jekyll_data_mapping_tl { mapping }
6076 \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.

```

6077 \seq_new:N \g_@@_jekyll_data_wildcard_absolute_address_seq
6078 \cs_new:Nn \markdown_jekyll_data_push_address_segment:n
6079 {
6080 \seq_if_empty:NF
6081 \g_@@_jekyll_data_datatypes_seq
6082 {
6083 \seq_get_right:NN
6084 \g_@@_jekyll_data_datatypes_seq
6085 \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.

```
6086 \tl_if_eq:NNTF
6087 \l_tmpa_tl
6088 \c_@@_jekyll_data_sequence_tl
6089 {
6090 \seq_put_right:Nn
6091 \g_@@_jekyll_data_wildcard_absolute_address_seq
6092 { * }
6093 }
6094 {
6095 \seq_put_right:Nn
6096 \g_@@_jekyll_data_wildcard_absolute_address_seq
6097 { #1 }
6098 }
6099 }
6100 }
```

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.

```
6101 \tl_new:N \g_@@_jekyll_data_wildcard_absolute_address_tl
6102 \tl_new:N \g_@@_jekyll_data_wildcard_relative_address_tl
```

```

6103 \cs_new:Nn \markdown_jekyll_data_concatenate_address:NN
6104 {
6105 \seq_pop_left:NN #1 \l_tmpa_tl
6106 \tl_set:Nx #2 { / \seq_use:Nn #1 { / } }
6107 \seq_put_left:NV #1 \l_tmpa_tl
6108 }
6109 \cs_new:Nn \markdown_jekyll_data_update_address_tls:
6110 {
6111 \markdown_jekyll_data_concatenate_address:NN
6112 \g_@@_jekyll_data_wildcard_absolute_address_seq
6113 \g_@@_jekyll_data_wildcard_absolute_address_tl
6114 \seq_get_right:NN
6115 \g_@@_jekyll_data_wildcard_absolute_address_seq
6116 \g_@@_jekyll_data_wildcard_relative_address_tl
6117 }

```

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.

```

6118 \cs_new:Nn \markdown_jekyll_data_push:nN
6119 {
6120 \markdown_jekyll_data_push_address_segment:n
6121 { #1 }
6122 \seq_put_right:NV
6123 \g_@@_jekyll_data_datatypes_seq
6124 #2
6125 \markdown_jekyll_data_update_address_tls:
6126 }
6127 \cs_new:Nn \markdown_jekyll_data_pop:
6128 {
6129 \seq_pop_right:NN
6130 \g_@@_jekyll_data_wildcard_absolute_address_seq
6131 \l_tmpa_tl
6132 \seq_pop_right:NN
6133 \g_@@_jekyll_data_datatypes_seq
6134 \l_tmpa_tl
6135 \markdown_jekyll_data_update_address_tls:
6136 }

```

To interface with the user, we use `markdown/jekyllData` key-values from the `l3keys` module of the `LATEX3` kernel. The default setup 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.

```

6137 \keys_define:nn
6138 { markdown/jekyllData }
6139 {
6140 author .code:n = { \author{#1} },
6141 date .code:n = { \date{#1} },

```

```

6142 title .code:n = { \title{#1} },
6143 }

```

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.

```

6144 \cs_new:Nn \markdown_jekyll_data_set_keyval:nn
6145 {
6146 \keys_set_known:nn
6147 { markdown/jekyllData }
6148 { { #1 } = { #2 } }
6149 }
6150 \cs_generate_variant:Nn
6151 \markdown_jekyll_data_set_keyval:nn
6152 { Vn }
6153 \cs_new:Nn \markdown_jekyll_data_set_keyvals:nn
6154 {
6155 \markdown_jekyll_data_push:nN
6156 { #1 }
6157 \c_@@_jekyll_data_scalar_tl
6158 \markdown_jekyll_data_set_keyval:Vn
6159 \g_@@_jekyll_data_wildcard_absolute_address_tl
6160 { #2 }
6161 \markdown_jekyll_data_set_keyval:Vn
6162 \g_@@_jekyll_data_wildcard_relative_address_tl
6163 { #2 }
6164 \markdown_jekyll_data_pop:
6165 }

```

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

```

6166 \markdownSetup{
6167 rendererPrototypes = {
6168 jekyllDataSequenceBegin = {
6169 \markdown_jekyll_data_push:nN
6170 { #1 }
6171 \c_@@_jekyll_data_sequence_tl
6172 },
6173 jekyllDataMappingBegin = {
6174 \markdown_jekyll_data_push:nN
6175 { #1 }
6176 \c_@@_jekyll_data_mapping_tl
6177 },
6178 jekyllDataSequenceEnd = {
6179 \markdown_jekyll_data_pop:
6180 },
6181 jekyllDataMappingEnd = {
6182 \markdown_jekyll_data_pop:

```

```

6183 },
6184 jekyllDataBoolean = {
6185 \markdown_jekyll_data_set_keyvals:nn
6186 { #1 }
6187 { #2 }
6188 },
6189 jekyllDataEmpty = { },
6190 jekyllDataNumber = {
6191 \markdown_jekyll_data_set_keyvals:nn
6192 { #1 }
6193 { #2 }
6194 },
6195 jekyllDataString = {
6196 \markdown_jekyll_data_set_keyvals:nn
6197 { #1 }
6198 { #2 }
6199 },

```

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.

```

6200 },
6201 }
6202 \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
6203 \markdownSetup{
6204 rendererPrototypes = {
6205 jekyllDataEnd = {
6206 \IfFormatAtLeastTF
6207 { 2020-10-01 }
6208 { \AddToHook{begindocument/end}{\maketitle} }
6209 {
6210 \ifx\@onlypreamble\@notprerr
6211 % We are in the document
6212 \maketitle
6213 \else
6214 % We are in the preamble
6215 \RequirePackage{etoolbox}
6216 \AfterEndPreamble{\maketitle}
6217 \fi
6218 }
6219 },
6220 },
6221 }
6222
6223 \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.

```
6224 \newcommand\markdownMakeOther{%
6225 \count0=128\relax
6226 \loop
6227 \catcode\count0=11\relax
6228 \advance\count0 by 1\relax
6229 \ifnum\count0<256\repeat}%
```

## 3.4 ConT<sub>E</sub>Xt Implementation

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

The ConT<sub>E</sub>Xt implementation redefines the plain T<sub>E</sub>X logging macros (see Section 3.2.1 on page 161) to use the ConT<sub>E</sub>Xt `\writestatus` macro.

```
6230 \def\markdownInfo#1{\writestatus{markdown}{#1.}}%
6231 \def\markdownWarning#1{\writestatus{markdown\space warn}{#1.}}%
6232 \def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
6233 \do\#\do\~\do_ \do\% \do\~}%
6234 \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<sup>A</sup>T<sub>E</sub>X package.

```
6235 \def\markdownMakeOther{%
6236 \count0=128\relax
6237 \loop
6238 \catcode\count0=11\relax
6239 \advance\count0 by 1\relax
6240 \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<sub>E</sub>Xt.

```
6241 \catcode`|=12}%
```

### 3.4.1 Typesetting Markdown

The `\startmarkdown` and `\stopmarkdown` macros are implemented using the `\markdownReadAndConvert` macro.

In Knuth's T<sub>E</sub>X, trailing spaces are removed very early on when a line is being put to the input buffer. [10, sec. 31]. According to Eijkhout [11, 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)T<sub>E</sub>X, but ConT<sub>E</sub>Xt MkIV funnels all input through its own input handler. This makes it possible to suppress the removal of trailing spaces in ConT<sub>E</sub>Xt MkIV and therefore to insert hard line breaks into markdown text.

```
6242 \ifx\startluacode\undefined % MkII
6243 \begingroup
6244 \catcode`\|=0%
6245 \catcode`\|=12%
6246 |gdef|startmarkdown{%
6247 |markdownReadAndConvert{\stopmarkdown}%
6248 {|stopmarkdown}}%
6249 |gdef|stopmarkdown{%
6250 |markdownEnd}%
6251 |endgroup
6252 \else % MkIV
6253 \startluacode
6254 document.markdown_buffering = false
6255 local function preserve_trailing_spaces(line)
6256 if document.markdown_buffering then
6257 line = line:gsub("[\t][\t]$", "\t\t")
6258 end
6259 return line
6260 end
6261 resolvers.installinputlinehandler(preserve_trailing_spaces)
6262 \stoptluacode
6263 \begingroup
6264 \catcode`\|=0%
6265 \catcode`\|=12%
6266 |gdef|startmarkdown{%
6267 |ctxlua{document.markdown_buffering = true}%
6268 |markdownReadAndConvert{\stopmarkdown}%
6269 {|stopmarkdown}}%
6270 |gdef|stopmarkdown{%
6271 |ctxlua{document.markdown_buffering = false}%
6272 |markdownEnd}%
6273 |endgroup
6274 \fi
```

### 3.4.2 Token Renderer Prototypes

The following configuration should be considered placeholder.

```
6275 \def\markdownRendererLineBreakPrototype{\blank}%
6276 \def\markdownRendererLeftBracePrototype{\textbraceleft}%
6277 \def\markdownRendererRightBracePrototype{\textbraceright}%
6278 \def\markdownRendererDollarSignPrototype{\textdollar}%
6279 \def\markdownRendererPercentSignPrototype{\percent}%
6280 \def\markdownRendererUnderscorePrototype{\textunderscore}%
6281 \def\markdownRendererCircumflexPrototype{\textcircumflex}%
6282 \def\markdownRendererBackslashPrototype{\textbackslash}%
6283 \def\markdownRendererTildePrototype{\textasciitilde}%
6284 \def\markdownRendererPipePrototype{\char`|}%
6285 \def\markdownRendererLinkPrototype#1#2#3#4{%
6286 \useURL[#1][#3][[#4]#1\footnote[#1]{\ifx\empty#4\empty\else#4:
6287 \fi}\tt<\hyphenatedurl{#3}>}}%
6288 \usemodule[database]
6289 \defineseparatedlist
6290 [MarkdownConTeXtCSV]
6291 [separator={,},
6292 before=\bTABLE,after=\eTABLE,
6293 first=\bTR,last=\eTR,
6294 left=\bTD,right=\eTD]
6295 \def\markdownConTeXtCSV{csv}
6296 \def\markdownRendererContentBlockPrototype#1#2#3#4{%
6297 \def\markdownConTeXtCSV@arg{#1}%
6298 \ifx\markdownConTeXtCSV@arg\markdownConTeXtCSV
6299 \placetable[] [tab:#1]{#4}{%
6300 \processeparatedfile[MarkdownConTeXtCSV][#3]}%
6301 \else
6302 \markdownInput{#3}%
6303 \fi}%
6304 \def\markdownRendererImagePrototype#1#2#3#4{%
6305 \placefigure[] [fig:#1]{#4}{\externalfigure[#3]}%
6306 \def\markdownRendererUlBeginPrototype{\startitemize}%
6307 \def\markdownRendererUlBeginTightPrototype{\startitemize[packed]}%
6308 \def\markdownRendererUlItemPrototype{\item}%
6309 \def\markdownRendererUlEndPrototype{\stopitemize}%
6310 \def\markdownRendererUlEndTightPrototype{\stopitemize}%
6311 \def\markdownRendererOlBeginPrototype{\startitemize[n]}%
6312 \def\markdownRendererOlBeginTightPrototype{\startitemize[packed,n]}%
6313 \def\markdownRendererOlItemPrototype{\item}%
6314 \def\markdownRendererOlItemWithNumberPrototype#1{\sym{#1.}}%
6315 \def\markdownRendererOlEndPrototype{\stopitemize}%
6316 \def\markdownRendererOlEndTightPrototype{\stopitemize}%
6317 \definedescription
6318 [MarkdownConTeXtDlItemPrototype]
```

```

6319 [location=hanging,
6320 margin=standard,
6321 headstyle=bold]%
6322 \definestartstop
6323 [MarkdownConTeXtDlPrototype]
6324 [before=\blank,
6325 after=\blank]%
6326 \definestartstop
6327 [MarkdownConTeXtDlTightPrototype]
6328 [before=\blank\startpacked,
6329 after=\stoppacked\blank]%
6330 \def\markdownRendererDlBeginPrototype{%
6331 \startMarkdownConTeXtDlPrototype}%
6332 \def\markdownRendererDlBeginTightPrototype{%
6333 \startMarkdownConTeXtDlTightPrototype}%
6334 \def\markdownRendererDlItemPrototype#1{%
6335 \startMarkdownConTeXtDlItemPrototype{#1}}%
6336 \def\markdownRendererDlItemEndPrototype{%
6337 \stopMarkdownConTeXtDlItemPrototype}%
6338 \def\markdownRendererDlEndPrototype{%
6339 \stopMarkdownConTeXtDlPrototype}%
6340 \def\markdownRendererDlEndTightPrototype{%
6341 \stopMarkdownConTeXtDlTightPrototype}%
6342 \def\markdownRendererEmphasisPrototype#1{\em#1}%
6343 \def\markdownRendererStrongEmphasisPrototype#1{\bf#1}%
6344 \def\markdownRendererBlockQuoteBeginPrototype{\startquotation}%
6345 \def\markdownRendererBlockQuoteEndPrototype{\stopquotation}%
6346 \def\markdownRendererInputVerbatimPrototype#1{\typefile{#1}}%
6347 \def\markdownRendererInputFencedCodePrototype#1#2{%
6348 \ifx\relax#2\relax
6349 \typefile{#1}%
6350 \else

```

The code fence infostrng is used as a name from the ConT<sub>E</sub>Xt `\definetyping` macro. This allows the user to set up code highlighting mapping as follows:

```

\definetyping [latex]
\setuptyping [latex] [option=TEX]

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

```

~~~

```
\stopmarkdown
\stoptext
```

```
6351 \typefile[#2] []{#1}%
6352 \fi}%
6353 \def\markdownRendererHeadingOnePrototype#1{\chapter{#1}}%
6354 \def\markdownRendererHeadingTwoPrototype#1{\section{#1}}%
6355 \def\markdownRendererHeadingThreePrototype#1{\subsection{#1}}%
6356 \def\markdownRendererHeadingFourPrototype#1{\subsubsection{#1}}%
6357 \def\markdownRendererHeadingFivePrototype#1{\subsubsubsection{#1}}%
6358 \def\markdownRendererHeadingSixPrototype#1{\subsubsubsubsection{#1}}%
6359 \def\markdownRendererHorizontalRulePrototype{%
6360 \blackrule[height=1pt, width=\hsize]}%
6361 \def\markdownRendererFootnotePrototype#1{\footnote{#1}}%
6362 \stopmodule\protect
```

There is a basic implementation of tables.

```
6363 \newcount\markdownConTeXtRowCounter
6364 \newcount\markdownConTeXtRowTotal
6365 \newcount\markdownConTeXtColumnCounter
6366 \newcount\markdownConTeXtColumnTotal
6367 \newtoks\markdownConTeXtTable
6368 \newtoks\markdownConTeXtTableFloat
6369 \def\markdownRendererTablePrototype#1#2#3{%
6370 \markdownConTeXtTable={}%
6371 \ifx\empty#1\empty
6372 \markdownConTeXtTableFloat={%
6373 \the\markdownConTeXtTable}%
6374 \else
6375 \markdownConTeXtTableFloat={%
6376 \placetable{#1}{\the\markdownConTeXtTable}}%
6377 \fi
6378 \begingroup
6379 \setupTABLE[r][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]
6380 \setupTABLE[c][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]
6381 \setupTABLE[r][1][topframe=on, bottomframe=on]
6382 \setupTABLE[r][#1][bottomframe=on]
6383 \markdownConTeXtRowCounter=0%
6384 \markdownConTeXtRowTotal=#2%
6385 \markdownConTeXtColumnTotal=#3%
6386 \markdownConTeXtRenderTableRow}
6387 \def\markdownConTeXtRenderTableRow#1{%
6388 \markdownConTeXtColumnCounter=0%
6389 \ifnum\markdownConTeXtRowCounter=0\relax
6390 \markdownConTeXtReadAlignments#1%
6391 \markdownConTeXtTable={\bTABLE}}%
```

```

6392 \else
6393   \markdownConTeXtTable=\expandafter{%
6394     \the\markdownConTeXtTable\bTR}%
6395   \markdownConTeXtRenderTableCell#1%
6396   \markdownConTeXtTable=\expandafter{%
6397     \the\markdownConTeXtTable\eTR}%
6398 \fi
6399 \advance\markdownConTeXtRowCounter by 1\relax
6400 \ifnum\markdownConTeXtRowCounter>\markdownConTeXtRowTotal\relax
6401   \markdownConTeXtTable=\expandafter{%
6402     \the\markdownConTeXtTable\eTABLE}%
6403   \the\markdownConTeXtTableFloat
6404   \endgroup
6405   \expandafter\gobbleoneargument
6406 \fi\markdownConTeXtRenderTableRow}
6407 \def\markdownConTeXtReadAlignments#1{%
6408   \advance\markdownConTeXtColumnCounter by 1\relax
6409   \if#1d%
6410     \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
6411   \fi\if#1l%
6412     \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
6413   \fi\if#1c%
6414     \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=middle]
6415   \fi\if#1r%
6416     \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=left]
6417   \fi
6418   \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
6419     \expandafter\gobbleoneargument
6420   \fi\markdownConTeXtReadAlignments}
6421 \def\markdownConTeXtRenderTableCell#1{%
6422   \advance\markdownConTeXtColumnCounter by 1\relax
6423   \markdownConTeXtTable=\expandafter{%
6424     \the\markdownConTeXtTable\bTD#1\eTD}%
6425   \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
6426     \expandafter\gobbleoneargument
6427   \fi\markdownConTeXtRenderTableCell}
6428 \def\markdownRendererTickedBox{${\boxtimes$}
6429 \def\markdownRendererHalfTickedBox{${\boxdot$}
6430 \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_EXbook*. 3rd ed. Vol. A. Computers & Typesetting. Reading, MA: Addison-Wesley, 1986. ix, 479. ISBN: 0-201-13447-0.
- [5] Till Tantau, Joseph Wright, and Vedran Miletić. *The Beamer class*. Feb. 10, 2021. URL: <https://mirrors.ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf> (visited on 02/11/2021).
- [6] Vít Novotný. *L^AT_EX 2_ε no longer keys packages by pathnames*. Feb. 20, 2021. URL: <https://github.com/latex3/latex2e/issues/510> (visited on 02/21/2021).
- [7] Geoffrey M. Poore. *The minted Package. Highlighted source code in L^AT_EX*. July 19, 2017. URL: <https://mirrors.ctan.org/macros/latex/contrib/minted/minted.pdf> (visited on 09/01/2020).
- [8] Roberto Ierusalimsky. *Programming in Lua*. 3rd ed. Rio de Janeiro: PUC-Rio, 2013. xviii, 347. ISBN: 978-85-903798-5-0.
- [9] Johannes Braams et al. *The L^AT_EX 2_ε Sources*. Apr. 15, 2017. URL: <https://mirrors.ctan.org/macros/latex/base/source2e.pdf> (visited on 01/08/2018).
- [10] Donald Ervin Knuth. *T_EX: The Program*. Vol. B. Computers & Typesetting. Reading, MA: Addison-Wesley, 1986. xvi, 594. ISBN: 0-201-13437-7.
- [11] Victor Eijkhout. *T_EX by Topic. A T_EXnician's Reference*. Wokingham, England: Addison-Wesley, Feb. 1, 1992. 307 pp. ISBN: 0-201-56882-0.

Index

| | |
|------------------------------------|--------------------|
| <code>\author</code> | 190 |
| <code>\autocites</code> | 181 |
| <code>blankBeforeBlockquote</code> | 8 |
| <code>blankBeforeCodeFence</code> | 8 |
| <code>blankBeforeHeading</code> | 8 |
| <code>breakableBlockquotes</code> | 9 |
| <code>cacheDir</code> | 7, 11, 25, 26, 159 |
| <code>citationNbsps</code> | 9 |
| <code>citations</code> | 9, 37 |

| | |
|---------------------------------------|------------------------|
| <code>\cite</code> | 181 |
| <code>\citep</code> | 181 |
| <code>\citet</code> | 181 |
| <code>codeSpans</code> | 10 |
| <code>compactdesc</code> | 4 |
| <code>compactenum</code> | 4 |
| <code>compactitem</code> | 4 |
| <code>contentBlocks</code> | 10 |
| <code>contentBlocksLanguageMap</code> | 10, 113 |
| <code>convert</code> | 166 |
| <code>\csvautotabular</code> | 4 |
|
 | |
| <code>\date</code> | 190 |
| <code>defaultOptions</code> | 7, 22, 109, 139 |
| <code>\definetyping</code> | 196 |
| <code>definitionLists</code> | 11, 32 |
| <code>\directlua</code> | 3, 25, 171 |
|
 | |
| <code>eagerCache</code> | 11 |
| <code>\enableregime</code> | 193 |
| <code>\endmarkdown</code> | 41 |
| <code>entities.char_entity</code> | 108 |
| <code>entities.dec_entity</code> | 108 |
| <code>entities.hex_entity</code> | 108 |
| <code>escape</code> | 111, 111, 112 |
| <code>escape_citation</code> | 111, 111 |
| <code>escape_minimal</code> | 111, 111 |
| <code>escape_uri</code> | 111, 111 |
| <code>escaped_chars</code> | 111, 111 |
| <code>escaped_citation_chars</code> | 110, 111 |
| <code>escaped_minimal_strings</code> | 110, 111 |
| <code>escaped_uri_chars</code> | 110, 111 |
| <code>expandtabs</code> | 140 |
|
 | |
| <code>fencedCode</code> | 12, 34, 175 |
| <code>\filecontents</code> | 166 |
| <code>finalizeCache</code> | 8, 11, 12, 13, 25, 159 |
| <code>footnotes</code> | 13, 37 |
| <code>frozenCacheCounter</code> | 13, 159, 161 |
| <code>frozenCacheFileName</code> | 8, 12, 26, 159 |
|
 | |
| <code>hardLineBreaks</code> | 13 |
| <code>hashEnumerators</code> | 14 |

| | |
|--|------------------------------------|
| headerAttributes | 14, 17 |
| html | 14, 38 |
| hybrid | 15, 19, 29, 47, 111, 112, 141, 172 |
|
 | |
| \includegraphics | 4 |
| inlineFootnotes | 15 |
| \input | 23, 59, 168, 171 |
| isdir | 3 |
| iterlines | 140 |
|
 | |
| jekyllData | 3, 15, 34–36 |
| \jobname | 26 |
|
 | |
| languages_json | 113, 113 |
| larsers | 140 |
|
 | |
| \maketitle | 192 |
| \markdown | 41 |
| markdown | 41, 41, 42, 174 |
| markdown* | 4, 41, 41, 42, 174 |
| \markdown_jekyll_data_concatenate_address:NN | 189 |
| \markdown_jekyll_data_pop: | 190 |
| \markdown_jekyll_data_push:nN | 190 |
| \markdown_jekyll_data_push_address_segment:n | 188 |
| \markdown_jekyll_data_set_keyval:NN | 191 |
| \markdown_jekyll_data_set_keyvals:nn | 191 |
| \markdown_jekyll_data_update_address_tls: | 189 |
| \markdownBegin | 24, 24, 25, 40, 41, 59 |
| \markdownEnd | 24, 24, 25, 40, 41, 59 |
| \markdownError | 39, 39 |
| \markdownExecute | 170 |
| \markdownExecuteDirect | 170, 170 |
| \markdownExecuteShellEscape | 169, 170 |
| \markdownFrozenCacheCounter | 161, 162, 172, 173 |
| \markdownIfOption | 166 |
| \markdownInfo | 39 |
| \markdownInput | 24, 25, 41, 42, 172, 174 |
| \markdownInputFileStream | 166 |
| \markdownInputPlainTeX | 174 |
| \markdownLuaExecute | 169, 170, 171, 172 |
| \markdownLuaOptions | 163, 166 |
| \markdownMakeOther | 40, 193 |
| \markdownMode | 3, 26, 40, 40, 169, 171 |

| | |
|---|--|
| <code>\markdownOptionCacheDir</code> | 3, 26, 51, 166, 176 |
| <code>\markdownOptionErrorTempFileName</code> | 26, 171 |
| <code>\markdownOptionFinalizeCache</code> | 25, 25, 51, 161 |
| <code>\markdownOptionFrozenCache</code> | 8, 12, 25, 25, 46, 47, 51, 162, 175, 176 |
| <code>\markdownOptionFrozenCacheFileName</code> | 25, 26 |
| <code>\markdownOptionHelperScriptFileName</code> | 25, 26, 27, 170, 171 |
| <code>\markdownOptionHybrid</code> | 51 |
| <code>\markdownOptionInputTempFileName</code> | 26, 167, 168 |
| <code>\markdownOptionOutputDir</code> | 26 |
| <code>\markdownOptionOutputTempFileName</code> | 26, 171 |
| <code>\markdownOptionSmartEllipses</code> | 51 |
| <code>\markdownOptionStripPercentSigns</code> | 28, 167 |
| <code>\markdownOptionTightLists</code> | 177 |
| <code>\markdownOutputFileStream</code> | 166 |
| <code>\markdownPrepare</code> | 165 |
| <code>\markdownReadAndConvert</code> | 40, 166, 174, 194 |
| <code>\markdownReadAndConvertProcessLine</code> | 168, 168 |
| <code>\markdownReadAndConvertStripPercentSigns</code> | 167 |
| <code>\markdownReadAndConvertTab</code> | 166 |
| <code>\markdownRendererBlockQuoteBegin</code> | 34 |
| <code>\markdownRendererBlockQuoteEnd</code> | 34 |
| <code>\markdownRendererCite</code> | 37, 37 |
| <code>\markdownRendererCodeSpan</code> | 29 |
| <code>\markdownRendererCodeSpanPrototype</code> | 58 |
| <code>\markdownRendererContentBlock</code> | 30, 30 |
| <code>\markdownRendererContentBlockCode</code> | 30 |
| <code>\markdownRendererContentBlockOnlineImage</code> | 30 |
| <code>\markdownRendererDlBegin</code> | 32 |
| <code>\markdownRendererDlBeginTight</code> | 19, 32 |
| <code>\markdownRendererDlDefinitionBegin</code> | 33 |
| <code>\markdownRendererDlDefinitionEnd</code> | 33 |
| <code>\markdownRendererDlEnd</code> | 33 |
| <code>\markdownRendererDlEndTight</code> | 19, 33 |
| <code>\markdownRendererDlItem</code> | 33 |
| <code>\markdownRendererDlItemEnd</code> | 33 |
| <code>\markdownRendererEllipsis</code> | 17, 28 |
| <code>\markdownRendererEmphasis</code> | 33, 55 |
| <code>\markdownRendererFootnote</code> | 37 |
| <code>\markdownRendererHalfTickedBox</code> | 28 |
| <code>\markdownRendererHeadingFive</code> | 36 |
| <code>\markdownRendererHeadingFour</code> | 36 |
| <code>\markdownRendererHeadingOne</code> | 36 |

| | |
|---|----------------|
| <code>\markdownRendererHeadingSix</code> | 36 |
| <code>\markdownRendererHeadingThree</code> | 36 |
| <code>\markdownRendererHeadingTwo</code> | 36 |
| <code>\markdownRendererHorizontalRule</code> | 37 |
| <code>\markdownRendererImage</code> | 30 |
| <code>\markdownRendererImagePrototype</code> | 58 |
| <code>\markdownRendererInlineHtmlComment</code> | 38 |
| <code>\markdownRendererInputFencedCode</code> | 34 |
| <code>\markdownRendererInputVerbatim</code> | 34 |
| <code>\markdownRendererInterblockSeparator</code> | 28 |
| <code>\markdownRendererJekyllDataBegin</code> | 34 |
| <code>\markdownRendererJekyllDataBoolean</code> | 35 |
| <code>\markdownRendererJekyllDataEmpty</code> | 36 |
| <code>\markdownRendererJekyllDataEnd</code> | 34 |
| <code>\markdownRendererJekyllDataMappingBegin</code> | 35 |
| <code>\markdownRendererJekyllDataMappingEnd</code> | 35 |
| <code>\markdownRendererJekyllDataNumber</code> | 35 |
| <code>\markdownRendererJekyllDataSequenceBegin</code> | 35 |
| <code>\markdownRendererJekyllDataSequenceEnd</code> | 35 |
| <code>\markdownRendererJekyllDataString</code> | 36 |
| <code>\markdownRendererLineBreak</code> | 28 |
| <code>\markdownRendererLink</code> | 29, 55 |
| <code>\markdownRendererNbsp</code> | 29 |
| <code>\markdownRendererOlBegin</code> | 31 |
| <code>\markdownRendererOlBeginTight</code> | 19, 31 |
| <code>\markdownRendererOlEnd</code> | 32 |
| <code>\markdownRendererOlEndTight</code> | 19, 32 |
| <code>\markdownRendererOlItem</code> | 17, 32 |
| <code>\markdownRendererOlItemEnd</code> | 32 |
| <code>\markdownRendererOlItemWithNumber</code> | 17, 32 |
| <code>\markdownRendererStrongEmphasis</code> | 34 |
| <code>\markdownRendererTable</code> | 37 |
| <code>\markdownRendererTextCite</code> | 37 |
| <code>\markdownRendererTickedBox</code> | 28 |
| <code>\markdownRendererUlBegin</code> | 30 |
| <code>\markdownRendererUlBeginTight</code> | 19, 31 |
| <code>\markdownRendererUlEnd</code> | 31 |
| <code>\markdownRendererUlEndTight</code> | 19, 31 |
| <code>\markdownRendererUlItem</code> | 31 |
| <code>\markdownRendererUlItemEnd</code> | 31 |
| <code>\markdownRendererUntickedBox</code> | 28 |
| <code>\markdownSetup</code> | 4, 43, 43, 174 |

| | |
|-------------------------------------|-------------------|
| <code>\markdownSetupSnippet</code> | 43, 43 |
| <code>\markdownWarning</code> | 39 |
| <code>new</code> | 6, 160 |
| <code>normalize_tag</code> | 139 |
| <code>os.execute</code> | 40, 170 |
| <code>\PackageError</code> | 173, 174 |
| <code>\PackageInfo</code> | 173, 174 |
| <code>\PackageWarning</code> | 173, 174 |
| <code>parsers</code> | 122 |
| <code>parsers.commented_line</code> | 124 |
| <code>\pdfshellescape</code> | 169 |
| <code>pipeTables</code> | 6, 16, 18, 37 |
| <code>preserveTabs</code> | 16, 18, 140 |
| <code>print</code> | 170, 171 |
| <code>reader</code> | 60, 122, 139, 140 |
| <code>reader->convert</code> | 158, 160 |
| <code>reader.new</code> | 139, 139 |
| <code>\shellescape</code> | 169 |
| <code>shiftHeadings</code> | 6, 16 |
| <code>slice</code> | 6, 14, 17, 109 |
| <code>smartEllipses</code> | 17, 28 |
| <code>\startmarkdown</code> | 59, 59, 194 |
| <code>startNumber</code> | 17, 32 |
| <code>status.shell_escape</code> | 169 |
| <code>\stopmarkdown</code> | 59, 59, 194 |
| <code>stripIndent</code> | 18, 140 |
| <code>tableCaptions</code> | 6, 18 |
| <code>taskLists</code> | 18, 28, 180 |
| <code>\tex.print</code> | 171 |
| <code>tex.print</code> | 170 |
| <code>texComments</code> | 19, 141 |
| <code>\textcites</code> | 181 |
| <code>tightLists</code> | 19, 31–33 |
| <code>\title</code> | 190 |
| <code>underscores</code> | 19 |
| <code>\url</code> | 4 |
| <code>\usepackage</code> | 41, 44 |

| | |
|---|----------|
| <code>\usetheme</code> | 44 |
| <code>util.cache</code> | 60 |
| <code>util.err</code> | 60 |
| <code>util.escaper</code> | 63 |
| <code>util.expand_tabs_in_line</code> | 61 |
| <code>util.flatten</code> | 62 |
| <code>util.intersperse</code> | 63 |
| <code>util.map</code> | 63 |
| <code>util.pathname</code> | 64 |
| <code>util.rope_last</code> | 62 |
| <code>util.rope_to_string</code> | 62 |
| <code>util.table_copy</code> | 61 |
| <code>util.walk</code> | 61, 62 |
| <code>\VerbatimInput</code> | 4 |
| <code>writer</code> | 60, 108 |
| <code>writer->active_headings</code> | 119, 119 |
| <code>writer->blockquote</code> | 117 |
| <code>writer->bulletlist</code> | 114 |
| <code>writer->citation</code> | 111 |
| <code>writer->citations</code> | 121 |
| <code>writer->code</code> | 112 |
| <code>writer->codeFence</code> | 117 |
| <code>writer->contentblock</code> | 114 |
| <code>writer->defer_call</code> | 122, 122 |
| <code>writer->definitionlist</code> | 116 |
| <code>writer->ellipsis</code> | 110 |
| <code>writer->emphasis</code> | 116 |
| <code>writer->eof</code> | 110 |
| <code>writer->escape</code> | 112, 112 |
| <code>writer->get_state</code> | 121 |
| <code>writer->heading</code> | 119 |
| <code>writer->hrule</code> | 110 |
| <code>writer->image</code> | 113 |
| <code>writer->inline_html_comment</code> | 115 |
| <code>writer->interblocksep</code> | 110 |
| <code>writer->is_writing</code> | 109, 109 |
| <code>writer->jekyllData</code> | 117 |
| <code>writer->linebreak</code> | 110 |
| <code>writer->link</code> | 112 |
| <code>writer->nbsp</code> | 109 |

| | |
|-------------------------------------|-----------------|
| <code>writer->note</code> | <i>121</i> |
| <code>writer->ollist</code> | <i>115</i> |
| <code>writer->pack</code> | <i>110, 159</i> |
| <code>writer->paragraph</code> | <i>110</i> |
| <code>writer->plain</code> | <i>109</i> |
| <code>writer->set_state</code> | <i>121</i> |
| <code>writer->slice_begin</code> | <i>109</i> |
| <code>writer->slice_end</code> | <i>109</i> |
| <code>writer->space</code> | <i>109</i> |
| <code>writer->string</code> | <i>111, 112</i> |
| <code>writer->strong</code> | <i>117</i> |
| <code>writer->suffix</code> | <i>109</i> |
| <code>writer->table</code> | <i>112</i> |
| <code>writer->checkbox</code> | <i>116</i> |
| <code>writer->uri</code> | <i>111</i> |
| <code>writer->verbatim</code> | <i>117</i> |
| <code>writer.new</code> | <i>108, 108</i> |
| <code>\writestatus</code> | <i>193</i> |