

Package ‘markdown’

June 4, 2024

Type Package

Title Render Markdown with 'commonmark'

Version 1.13

Description Render Markdown to full and lightweight HTML/LaTeX documents with the 'commonmark' package. This package has been superseded by 'litedown'.

Depends R (>= 2.11.1)

Imports utils, commonmark (>= 1.9.0), xfun (>= 0.38)

Suggests knitr, rmarkdown (>= 2.18), yaml, RCurl

License MIT + file LICENSE

URL <https://github.com/rstudio/markdown>

BugReports <https://github.com/rstudio/markdown/issues>

RoxygenNote 7.3.1

Encoding UTF-8

NeedsCompilation no

Author Yihui Xie [aut, cre] (<<https://orcid.org/0000-0003-0645-5666>>),

JJ Allaire [aut],

Jeffrey Horner [aut],

Henrik Bengtsson [ctb],

Jim Hester [ctb],

Yixuan Qiu [ctb],

Kohske Takahashi [ctb],

Adam November [ctb],

Nacho Caballero [ctb],

Jeroen Ooms [ctb],

Thomas Leeper [ctb],

Joe Cheng [ctb],

Andrzej Oles [ctb],

Posit Software, PBC [cph, fnd]

Maintainer Yihui Xie <xie@yihui.name>

Repository CRAN

Date/Publication 2024-06-04 20:50:02 UTC

Contents

markdown-package	2
html_format	3
mark	4
markdown_options	5
rpubsUpload	7
smartypants	8

Index	10
--------------	-----------

markdown-package	<i>Markdown rendering for R</i>
------------------	---------------------------------

Description

Markdown is a plain-text formatting syntax that can be converted to XHTML or other formats. This package provides wrapper functions (mainly `mark()`) based on the **commonmark** package.

Author(s)

Maintainer: Yihui Xie <xie@yihui.name> ([ORCID](#))

Authors:

- JJ Allaire
- Jeffrey Horner

Other contributors:

- Henrik Bengtsson [contributor]
- Jim Hester [contributor]
- Yixuan Qiu [contributor]
- Kohske Takahashi [contributor]
- Adam November [contributor]
- Nacho Caballero [contributor]
- Jeroen Ooms [contributor]
- Thomas Leeper [contributor]
- Joe Cheng [contributor]
- Andrzej Oles [contributor]
- Posit Software, PBC [copyright holder, funder]

See Also

Useful links:

- <https://github.com/rstudio/markdown>
- Report bugs at <https://github.com/rstudio/markdown/issues>

Description

Convenience functions for R Markdown v2 users.

Usage

```
html_format(  
  meta = NULL,  
  template = NULL,  
  options = NULL,  
  keep_md = FALSE,  
  keep_tex = FALSE,  
  latex_engine = "xelatex"  
)
```

```
latex_format(  
  meta = NULL,  
  template = NULL,  
  options = NULL,  
  keep_md = FALSE,  
  keep_tex = FALSE,  
  latex_engine = "xelatex"  
)
```

Arguments

meta, template, options

Arguments to be passed to [mark\(\)](#).

keep_md, keep_tex

Whether to keep the intermediate ‘.md’ and ‘.tex’ files generated from ‘.Rmd’.

latex_engine

The LaTeX engine to compile ‘.tex’ to ‘.pdf’. This argument and keep_tex are for latex_format() only, and ignored in html_format().

Details

We refer to this **markdown** package plus **knitr** as “R Markdown v1”, and the **rmarkdown** package as “R Markdown v2”. The former uses **commonmark** to convert Markdown, and the latter uses Pandoc. However, the latter also accept custom converting tools in addition to Pandoc. The output formats here provide the custom converting function [mark\(\)](#) to **rmarkdown**, so that users can take advantage of [rmarkdown::render\(\)](#) and the Knit button in RStudio. It is absolutely not necessary to rely on **rmarkdown**. The only point is convenience. If you do not use [rmarkdown::render\(\)](#) or the Knit button, you can definitely just call [markdown::mark\(\)](#) directly.

mark

*Render Markdown to an output format***Description**

Render Markdown to an output format via the **commonmark** package. The function `mark_html()` is a shorthand of `mark(format = 'html', template = TRUE)`, and `mark_latex()` is a shorthand of `mark(format = 'latex', template = TRUE)`.

Usage

```
mark(
  file = NULL,
  output = NULL,
  text = NULL,
  format = c("html", "latex"),
  options = NULL,
  template = FALSE,
  meta = list()
)

mark_html(..., template = TRUE)

mark_latex(..., template = TRUE)
```

Arguments

<code>file</code>	Path to an input file. If not provided, it is presumed that the <code>text</code> argument will be used instead. This argument can also take a character vector of Markdown text directly. To avoid ambiguity in the latter case, a single character string input will be treated as a file if the file exists. If a string should be treated as Markdown text when it happens to be a file path, wrap it in <code>I()</code> .
<code>output</code>	Output file path. If not character, the results will be returned as a character vector. If not specified and the input is a file path, the output file path will have the same base name as the input file, with an extension corresponding to the <code>format</code> argument, e.g., <code>mark('foo.md', format = 'latex')</code> will generate an output file <code>'foo.tex'</code> by default.
<code>text</code>	A character vector of the Markdown text. By default, it is read from <code>file</code> .
<code>format</code>	An output format supported by commonmark , e.g., <code>'html'</code> , <code>'man'</code> , and <code>'text'</code> , etc. See the <code>markdown_*</code> renderers in commonmark .
<code>options</code>	Options to be passed to the renderer. See <code>markdown_options()</code> for details. This argument can take either a character vector of the form <code>" +option1 option2-option3"</code> (use <code>+</code> or a space to enable an option, and <code>-</code> to disable an option), or a list of the form <code>list(option1 = value1, option2 = value2, ...)</code> . A string <code>" +option1"</code> is equivalent to <code>list(option1 = TRUE)</code> , and <code>" -option2"</code> means <code>list(option2 = FALSE)</code> . Options that do not take logical values must be specified via a list, e.g., <code>list(width = 30)</code> .

template	Path to a template file. The default value is <code>getOption('markdown.FORMAT.template', markdown::pkg_file('resources', 'markdown.FORMAT'))</code> where <code>FORMAT</code> is the output format name (<code>html</code> or <code>latex</code>). It can also take a logical value: <code>TRUE</code> means to use the default template, and <code>FALSE</code> means to generate only a fragment without using any template.
meta	A named list of metadata. Elements in the metadata will be used to fill out the template by their names and values, e.g., <code>list(title = ...)</code> will replace the <code>\$title\$</code> variable in the template. See the Section “YAML metadata” in the vignette <code>vignette('intro', package = 'markdown')</code> for supported variables.
...	Arguments to be passed to <code>mark()</code> .

Value

Invisible `NULL` when output is to a file, otherwise a character vector of the rendered output.

See Also

The spec of GitHub Flavored Markdown: <https://github.github.com/gfm/>

Examples

```
library(markdown)
mark(c("Hello _World_!", "", "Welcome to **markdown**."))
# a few corner cases
mark(character(0))
mark("")
# if input happens to be a file path but should be treated as text, use I()
mark(I("This is *not* a file.md"))
# that's equivalent to
mark(text = "This is *not* a file.md")

mark_html("Hello _World_!", template = FALSE)
# write HTML to an output file
mark_html("_Hello_, **World**!", output = tempfile())

mark_latex("Hello _World_!", template = FALSE)
```

markdown_options

Markdown rendering options

Description

A list of all options to control Markdown rendering. Options that are enabled by default are marked by a `+` prefix, and those disabled by default are marked by `-`.

Usage

```
markdown_options()
```

Details

See vignette('intro', package = 'markdown') for the full list of options and their documentation.

Value

A character vector of all available options.

Examples

```
# all available options
markdown::markdown_options()

library(markdown)

# toc example
mkd <- c("# Header 1", "p1", "## Header 2", "p2")

cat(mark(mkd, options = "+number_sections"))
cat(mark(mkd, options = "+number_sections+toc"))

# hard_wrap example
cat(mark("foo\nbar\n"))
cat(mark("foo\nbar\n", options = "hard_wrap"))

# latex math example
mkd <- c(
  "`$x$` is inline math  $x!$ ", "", "Display style:", "", " $x + y$ ", "",
  "\\begin{eqnarray}
  a^2+b^2 & = & c^2\\\\
  \\sin^2(x)+\\cos^2(x) & = & 1
  \\end{eqnarray}"
)

cat(mark(mkd))
cat(mark(mkd, options = "-latex_math"))

# tables example (need 4 spaces at beginning of line here)
cat(mark("
First Header | Second Header
----- | -----
Content Cell | Content Cell
Content Cell | Content Cell
"))

# but not here
cat(mark("
First Header | Second Header
----- | -----
Content Cell | Content Cell
Content Cell | Content Cell
", options = '-table'))
```

```

# autolink example
cat(mark("https://www.r-project.org/"))
cat(mark("https://www.r-project.org/", options = "-autolink"))

# strikethrough example
cat(mark("~~awesome~~"))
cat(mark("~~awesome~~", options = "-strikethrough"))

# superscript and subscript examples
cat(mark("2^10^"))
cat(mark("2^10^", options = "-superscript"))
cat(mark("H~2~0"))
cat(mark("H~2~0", options = "-subscript"))

# code blocks
cat(mark('```r\n1 + 1;\n```'))
cat(mark('```{r}\n1 + 1;\n```'))
cat(mark('```{r .js}\n1 + 1;\n```'))
cat(mark('```{r .js #foo}\n1 + 1;\n```'))
cat(mark('```{r .js #foo style="color:red;"}\n1 + 1;\n```'))
cat(mark('```\n```{r, echo=TRUE}\n1 + 1;\n```'))

# raw blocks
cat(mark('```{=html}\n<p>raw HTML</p>\n```'))
cat(mark('```{=latex}\n<p>raw HTML</p>\n```'))

# skip_html tags
mkd = '<style>a {}</style><script type="text/javascript">console.log("No!");</script>\n[Hello](#)'
cat(mark(mkd))
# TODO: wait for https://github.com/r-lib/commonmark/issues/15 to be fixed
# cat(mark(mkd, options = "tagfilter"))

```

rpubsUpload

Upload an HTML file to RPubs

Description

This function uploads an HTML file to rpubs.com. If the upload succeeds a list that includes an id and continueUrl is returned. A browser should be opened to the continueUrl to complete publishing of the document. If an error occurs then a diagnostic message is returned in the error element of the list.

Usage

```

rpubsUpload(
  title,
  htmlFile,
  id = NULL,
  properties = list(),

```

```
method = getOption("rpubs.upload.method", "auto")
)
```

Arguments

title	The title of the document.
htmlFile	The path to the HTML file to upload.
id	If this upload is an update of an existing document then the id parameter should specify the document id to update. Note that the id is provided as an element of the list returned by successful calls to rpubsUpload.
properties	A named list containing additional document properties (RPubs doesn't currently expect any additional properties, this parameter is reserved for future use).
method	Method to be used for uploading. "internal" uses a plain http socket connection; "curl" uses the curl binary to do an https upload; "rcurl" uses the RCurl package to do an https upload; and "auto" uses the best available method searched for in the following order: "curl", "rcurl", and then "internal". The global default behavior can be configured by setting the rpubs.upload.method option (the default is "auto").

Value

A named list. If the upload was successful then the list contains a `id` element that can be used to subsequently update the document as well as a `continueUrl` element that provides a URL that a browser should be opened to in order to complete publishing of the document. If the upload fails then the list contains an `error` element which contains an explanation of the error that occurred.

Examples

```
## Not run:
# upload a document
result <- rpubsUpload("My document title", "Document.html")
if (!is.null(result$continueUrl))
  browseURL(result$continueUrl) else stop(result$error)

# update the same document with a new title
updateResult <- rpubsUpload("My updated title", "Document.html", result$id)

## End(Not run)
```

 smartypants

Convert some ASCII strings to HTML entities

Description

Transform ASCII strings (c) (copyright), (r) (registered trademark), (tm) (trademark), and fractions n/m into *smart* typographic HTML entities.

Usage

```
smartypants(text)
```

Arguments

text A character vector of the Markdown text.

Value

A character vector of the transformed text.

Examples

```
cat(smartypants("1/2 (c)\n"))
```

Index

`html_format`, 3

`I()`, 4

`latex_format (html_format)`, 3

`mark`, 4

`mark()`, 2, 3

`mark_html (mark)`, 4

`mark_latex (mark)`, 4

`markdown (markdown-package)`, 2

`markdown-package`, 2

`markdown_*`, 4

`markdown_options`, 5

`markdown_options()`, 4

`rmarkdown::render()`, 3

`rpubsUpload`, 7

`smartypants`, 8