Package 'gluedown'

January 14, 2020

Title Wrap Vectors in Markdown Formatting

Version 1.0.2

Description Ease the transition between R vectors and markdown text. With 'gluedown' and 'rmarkdown', users can create traditional vectors in R, glue those strings together with the markdown syntax, and print those formatted vectors directly to the document. This package primarily uses GitHub Flavored Markdown (GFM), an offshoot of the unambiguous CommonMark specification by John MacFarlane (2019) <https://spec.commonmark.org/>.

License GPL-3

URL https://kiernann.com/gluedown/, https://github.com/kiernann/gluedown/

BugReports https://github.com/kiernann/gluedown/issues

Depends R (>= 3.3)

Imports glue (>= 1.3.1)

Suggests covr (>= 3.3.2), dplyr (>= 0.8.3), httr (>= 1.4.1), knitr (>= 1.25), markdown (>= 1.1), rmarkdown (>= 1.16), rvest (>= 0.3.2), spelling (>= 2.1), stringr (>= 1.4.0), testthat (>= 2.1.0)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.0.2

NeedsCompilation no

Author Kiernan Nicholls [aut, cre]

Maintainer Kiernan Nicholls <kiernann@protonmail.com>

Repository CRAN

Date/Publication 2020-01-14 05:50:02 UTC

R topics documented:

`gluedown` *gluedown: A package to format character vectors with markdown.*

Description

The `gluedown` package helps transition from R's powerful vectors to formatted markdown text. The functions use `glue::glue()` to wrap character vectors in valid markdown syntax. In combination with the `knitr` package, this allows users to directly print R vectors as formatted text for improved clarity and readability.

Glue wrappers

The md_*() functions return `glue` objects, which are returned using `cat()` by default. This allows users to both manipulate the formatted strings as they would with any character vector and still present the string to the user when an `knitr` chunk option is set to return code results `'asis'`.

Other wrappers

The `md_table()` and `md_convert()` functions wrap around `knitr::kable()` and `markdown::markdownToHTML()` respectively. The later allows users to convert md_*() outputs to HTML fragments.

`md_autolink` *Markdown autolink*

Description

Take a character vector and wrap each element in < and > to return a glue vector of autolink text. This inline is rendered as the <href> HTML tag.

Usage

```
md_autolink(url)
```

Arguments

`url`	A character vector of absolute URLs.

Details

Autolinks are absolute URIs and email addresses inside < and >. They are parsed as links, with the URL or email address as the link label.

A URI autolink consists of <, followed by an absolute URI followed by >. It is parsed as a link to the URI, with the URI as the link's label.

An absolute URI, for these purposes, consists of a scheme followed by a colon (:) followed by zero or more characters other than ASCII whitespace and control characters, <, and >. If the URI includes these characters, they must be percent-encoded (e.g. %20 for a space).

For purposes of this spec, a scheme is any sequence of 2–32 characters beginning with an ASCII letter and followed by any combination of ASCII letters, digits, or the symbols plus ("+"), period (".."), or hyphen ("-").

Value

A `glue` vector of length equal to `x`.

See Also

Other inline functions: `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_escape()`, `md_hardline()`, `md_image()`, `md_issue()`, `md_italic()`, `md_link()`, `md_softline()`, `md_strike()`, `md_text()`

Examples

```
md_autolink("http://foo.bar.baz")
```

`md_blank` *Markdown blank line*

Description

Create a blank line between other markdown block-level elements.

Usage

```
md_blank()
```

Details

Blank lines between block-level elements are ignored, except for the role they play in determining whether a list is tight or loose.

Blank lines at the beginning and end of the document are also ignored.

Value

A `glue` vector of length one containing two newline characters.

See Also

Other leaf block functions: `md_chunk()`, `md_fence()`, `md_heading()`, `md_indent()`, `md_paragraph()`, `md_reference()`, `md_rule()`, `md_setext()`, `md_table()`

Examples

```
md_blank
```

`md_bold` *Markdown bold emphasis*

Description

Take a character vector and wrap each element in double asterisks to create a glue vector of bold emphasis text. This inline is rendered as the <stong> HTML tag.

Usage

```
md_bold(x)
```

Arguments

x	The text to be emphasized in bold.

Details

A double ** or __ can open or close emphasis... Emphasis begins with a delimiter that can open emphasis and ends with a delimiter that can close emphasis, and that uses the same character (__ or **) as the opening delimiter.

Value

A glue vector of length equal to x.

See Also

Other inline functions: md_autolink(), md_code(), md_convert(), md_disallow(), md_escape(), md_hardline(), md_image(), md_issue(), md_italic(), md_link(), md_softline(), md_strike(), md_text()

Examples

```
md_bold("Example")
md_bold(state.name)
```

---

md_bullet *Markdown bullet list*

Description

take a character vector and return a glue vector of valid bullet list items. When printed together, these bullet list items create a bullet list. This container block is rendered as the <ul> HTML tag, with each element of the vector creating a separate <li> tag.

Usage

```
md_bullet(x, marker = c("*", "-", "+"))
```

Arguments

x	The vector of bullet point list items.
marker	The bullet list marker to use; one of -, +, or *.

Details

A list is a sequence of one or more list items of the same type. The list items may be separated by any number of blank lines.

Two list items are of the same type if they begin with a list marker of the same type. Two list markers are of the same type if (a) they are bullet list markers using the same character (-, +, or *)...

Value

A glue vector with length equal to x.

See Also

Other container block functions: md_list(), md_order(), md_quote(), md_task()

Examples

```
md_bullet(state.name[1:5])
md_bullet(sample(state.name, 5), marker = "+")
```

---

md_chunk *Markdown code block*

Description

Take a character vector of lines and return a glue vector

Usage

```
md_chunk(x, type = c("tick", "tilde", "indent"), ...)
```

Arguments

x	A character vector of lines to be wrapped concatenated into a single block, possibly created by readLines() or deparse().
type	The type of code block to be created. Either "tick", "tilde" (which call md_fence()) or "indent" (which calls md_indent()).
...	Arguments to be passed to md_fence() or md_indent().

Details

Turn a character vector of lines into a single code block either indented or fenced in tildes or backticks. This markdown leaf block can be rendered as nested HTML <code> and <pre> tags. This function either calls md_fence() or md_indent() based on the type argument.

Value

A glue object of length 1, with elements of x formatted via md_fence() or md_indent().

See Also

Other leaf block functions: md_blank(), md_fence(), md_heading(), md_indent(), md_paragraph(), md_reference(), md_rule(), md_setext(), md_table()

Examples

```
md_chunk("$ sudo apt install r-base-dev", info = "bash")
md_chunk(c("library(ggplot2)", "ggplot(mpg)+", "geom_point(aes(displ, hwy))"), type = "tilde")
md_indent(
  n = c(4, 4, 6),
  x = c(
    "library(dplyr)",
    "starwars %>%",
    "filter(species == 'Droid')"
  )
)
```

---

md_code *Markdown code span*

Description

Take a character vector and wrap each element in backticks to create a glue vector of inline code spans. This inline is rendered as a <code> HTML tag.

Usage

```
md_code(x)
```

Arguments

x	The text to be formatted as fixed-width inline code.

Details

A backtick string is a string of one or more backtick characters that is neither preceded nor followed by a backtick.

A code span begins with a backtick string and ends with a backtick string of equal length. The contents of the code span are the characters between the two backtick strings, normalized in the following ways: * First, line endings are converted to spaces. * If the resulting string both begins and ends with a space character, but does not consist entirely of space characters, a single space character is removed from the front and back. This allows you to include code that begins or ends with backtick characters, which must be separated by whitespace from the opening or closing backtick strings.

Value

A glue vector of length equal to x.

See Also

Other inline functions: md_autolink(), md_bold(), md_convert(), md_disallow(), md_escape(), md_hardline(), md_image(), md_issue(), md_italic(), md_link(), md_softline(), md_strike(), md_text()

Examples

```
md_code("ex_var")
md_code(state.name[1:3])
```

`md_convert` *Convert markdown to HTML*

Description

Take a character vector of valid markdown text and pass it to `markdown::markdownToHTML()` to create a glue vector of HTML fragments. Primarily used to test that md_*() functions create vectors that meet the GFM spec and can be rendered as HTML.

Usage

```
md_convert(x, frag = TRUE, disallow = TRUE)
```

Arguments

`x`	A character vector of *markdown* text to be converted.
`frag`	logical; Whether only a single HTML fragment should be returned. `TRUE` by default.
`disallow`	logical; Should `md_disallow()` be called on the converted output?

Details

GFM enables the `tagfilter` extension, where the following HTML tags will be filtered when rendering HTML output...

Value

A `glue` vector of length 1 containing HTML tags.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_disallow()`, `md_escape()`, `md_hardline()`, `md_image()`, `md_issue()`, `md_italic()`, `md_link()`, `md_softline()`, `md_strike()`, `md_text()`

Examples

```
md_convert(x = md_bold("test"))
```

`md_disallow` *Disallow certain raw HTML*

Description

Take a character vector of raw HTML text (possibly via `md_convert()`) and disallow certain tags by replacing < with <.

Usage

```
md_disallow(html)
```

Arguments

`html`	A character vector of *markdown* text to be converted.

Details

GFM enables the tagfilter extension, where the following HTML tags will be filtered when rendering HTML output:

- <title>
- <textarea>
- <style>
- <xmp>
- <iframe>
- <noembed>
- <noframes>
- <script>
- <plaintext>

Filtering is done by replacing the leading < with the entity <. These tags are chosen in particular as they change how HTML is interpreted in a way unique to them (i.e. nested HTML is interpreted differently), and this is usually undesireable (sic) in the context of other rendered Markdown content.

All other HTML tags are left untouched.

Value

A `glue` vector of length 1 containing HTML tags.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_escape()`, `md_hardline()`, `md_image()`, `md_issue()`, `md_italic()`, `md_link()`, `md_softline()`, `md_strike()`, `md_text()`

Examples

```
md_disallow("<title>GitHub Flavored Markdown Spec</title>")
```

`md_escape` *Backslash escape all punctuation*

Description

Take a character vector containing punctuation and return a glue vector with every punctuation mark prepended with double escape backslashes.

Usage

```
md_escape(x)
```

Arguments

`x`	A character vector of strings containing punctuation that might accidentally be considered markdown syntax.

Details

When trying to format text containing markdown syntax characters, it's necessary to "escape" those characters so that they are ignored by formatting.

Any ASCII punctuation character may be backslash-escaped... Escaped characters are treated as regular characters and do not have their usual Markdown meanings.

Value

A character string with all [:punct:] properly escaped with prepended backslashes.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_hardline()`, `md_image()`, `md_issue()`, `md_italic()`, `md_link()`, `md_softline()`, `md_strike()`, `md_text()`

Examples

```
md_escape("# six seasons and a movie")
```

`md_fence` *Markdown fenced code block*

Description

Turn a character vector of lines into a single code block with lines bookended with a code fence of backticks or tildes. This markdown leaf block can be rendered as HTML <code> tags inside <pre> tags.

Usage

```
md_fence(x, char = c("`", "~"), info = "r")
```

Arguments

`x`	A character vector of lines to be wrapped concatenated into a single block, possibly created by possibly created by `readLines()` or `deparse()`.
`char`	The character to use in the code fence; either backtick characters... or tildes (~). Defaults to backticks.
`info`	The info string text to follow the initial code fence, typically a code for the language of the lines of `x`. Defaults to `r`.

Details

A code fence is a sequence of at least three consecutive backtick characters ... or tildes (~). (Tildes and backticks cannot be mixed.) A fenced code block begins with a code fence, indented no more than three spaces.

The line with the opening code fence may optionally contain some text following the code fence; this is trimmed of leading and trailing whitespace and called the info string...

The content of the code block consists of all subsequent lines, until a closing code fence of the same type as the code block began with (backticks or tildes), and with at least as many backticks or tildes as the opening code fence...

A fenced code block may interrupt a paragraph, and does not require a blank line either before or after.

The content of a code fence is treated as literal text, not parsed as inlines. The first word of the info string is typically used to specify the language of the code sample, and rendered in the class attribute of the code tag. However, this spec does not mandate any particular treatment of the info string (see the `info` argument).

Value

A character vector wrapped on either side by code fences.

See Also

Other leaf block functions: `md_blank()`, `md_chunk()`, `md_heading()`, `md_indent()`, `md_paragraph()`, `md_reference()`, `md_rule()`, `md_setext()`, `md_table()`

Examples

```
md_fence(deparse(sd))
md_fence(c("library(dplyr)", "starwars %>%", "  filter(species == 'Droid')"))
```

`md_hardline` *Markdown hard line breaks*

Description

Take a character vector and return a collapsed glue vector with each original element separated by two spaces and a newline. This inline is rendered with a
 HTML tag.

Usage

```
md_hardline(...)
```

Arguments

`...`	Any number of character vectors.

Details

A line break (not in a code span or HTML tag) that is preceded by two or more spaces and does not occur at the end of a block is parsed as a hard line break (rendered in HTML as a
 tag)

Value

A `glue` vector with elements of `...` separated by two trailing spaces and a single newline.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_escape()`, `md_image()`, `md_issue()`, `md_italic()`, `md_link()`, `md_softline()`, `md_strike()`, `md_text()`

Examples

```
# compare the following
md_bold(c("One", "Two"))
md_hardline(md_bold(c("One", "Two")), md_italic("Three"))
```

`md_heading` *Markdown ATX headings*

Description

Turn a character vector into a vector of valid markdown ATX headings. These markdown leaf blocks can be rendered as the <h1> through <h6> HTML tags. See `md_setext()` to create setext (underlined) headings.

Usage

```
md_heading(x, level = 1)
```

Arguments

`x`	A character vector of heading text.
`level`	A numeric vector of use to determine the number of heading hash characters to preceed each element of `x`. The heading level is equal to the number of # characters in the opening sequence.

Details

An ATX heading consists of a string of characters, parsed as inline content, between an opening sequence of 1–6 unescaped # characters and an optional closing sequence of any number of unescaped # characters. The opening sequence of # characters must be followed by a space or by the end of line. The optional closing sequence of #s must be preceded by a space and may be followed by spaces only. The opening # character may be indented 0-3 spaces. The raw contents of the heading are stripped of leading and trailing spaces before being parsed as inline content. The heading level is equal to the number of # characters in the opening sequence.

Value

A `glue` vector of headings with length equal to `x`.

See Also

Other leaf block functions: `md_blank()`, `md_chunk()`, `md_fence()`, `md_indent()`, `md_paragraph()`, `md_reference()`, `md_rule()`, `md_setext()`, `md_table()`

Examples

```
md_heading("Overview")
md_heading(x = c("One", "Two"), level = 1:2)
md_heading(x = c("Installation", "Usage"), level = 2)
```

`md_image` *Markdown image links*

Description

Take character vectors of alternative text, image link destinations, and optional titles and return single glue vector of valid markdown inline image links. This inline is rendered as the <img> HTML tag.

Usage

```
md_image(url, alt = "", title = NULL)
```

Arguments

`url`	A character vector of link destination (URL) strings.
`alt`	A character vector of alternative text that can be used to refer to an image.
`title`	The optional title of the link.

Details

Syntax for images is like the syntax for links, with one difference. Instead of link text, we have an image description. The rules for this are the same as for link text, except that (a) an image description starts with ![rather than [, and (b) an image description may contain links. An image description has inline elements as its contents. When an image is rendered to HTML, this is standardly used as the image's `alt` attribute.

Value

A `glue` vector of collapsed alternative text and associated URLs.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_escape()`, `md_hardline()`, `md_issue()`, `md_italic()`, `md_link()`, `md_softline()`, `md_strike()`, `md_text()`

Examples

```
if (file.exists("man/figures/logo.png")) md_image("man/figures/logo.png")
md_image("http://hexb.in/hexagons/eff.png")
md_image("http://hexb.in/hexagons/eff.png", "EFF Hex Sticker")
md_image("http://hexb.in/hexagons/eff.png", "EFF Hex Sticker", "Logo")
```

`md_indent` *Markdown indented code block*

Description

Turn a character vector of lines into a single code block with each line indented four spaces. This markdown leaf block can be rendered as nested HTML <code> and <pre> tags. This is the code block format required by legacy Reddit-flavored Markdown.

Usage

```
md_indent(x, n = 4)
```

Arguments

`x`	A character vector of lines to be wrapped concatenated into a single block, possibly created by `readLines()` or `deparse()`.
`n`	A numeric vector

Details

An indented code block is composed of one or more indented chunks separated by blank lines. An indented chunk is a sequence of non-blank lines, each indented four or more spaces. The contents of the code block are the literal contents of the lines, including trailing line endings, minus four spaces of indentation. An indented code block has no info string.

Value

A `glue` object of length 1, with the elements of `x` preceded with 4 spaces and separated by a newline.

See Also

Other leaf block functions: `md_blank()`, `md_chunk()`, `md_fence()`, `md_heading()`, `md_paragraph()`, `md_reference()`, `md_rule()`, `md_setext()`, `md_table()`

Examples

```
md_indent(deparse(md_bold))
```

`md_issue` *Markdown GitHub issue*

Description

Take a character vector and numeric vector and concatenate them into a glue vector of valid GitHub issue autolinks (username/repo#issue).

Usage

```
md_issue(repo, num)
```

Arguments

`repo`	A character vector in the format `"user/rep"`.
`num`	The issue or pull number *without* hash symbol.

Details

Within conversations on GitHub, references to issues and pull requests are automatically converted to shortened links.

Value

A character vector which GitHub can automatically hyperlink.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_escape()`, `md_hardline()`, `md_image()`, `md_italic()`, `md_link()`, `md_softline()`, `md_strike()`, `md_text()`

Other markdown extensions: `md_strike()`, `md_table()`

Examples

```
md_issue("kiernann/gluedown", 1:5)
```

`md_italic` *Markdown italic emphasis*

Description

Take a character vector and wrap each element in single underscores to create a glue vector of italic emphasis text. This inline is rendered as the HTML tag.

Usage

```
md_italic(x)
```

Arguments

`x`	The text to be emphasized in italics.

Details

A single * or _ can open or close emphasis... Emphasis begins with a delimiter that can open emphasis and ends with a delimiter that can close emphasis, and that uses the same character (_ or *) as the opening delimiter.

Value

A `glue` vector of length equal to `x`.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_escape()`, `md_hardline()`, `md_image()`, `md_issue()`, `md_link()`, `md_softline()`, `md_strike()`, `md_text()`

Examples

```
md_italic("Example")
md_italic(state.name)
```

`md_link` *Markdown inline link (6.6)*

Description

Take character vectors of link texts, link destinations, and optional titles and return single glue vector of valid markdown inline links. This inline is rendered as the <href> HTML tag.

Usage

```
md_link(x, url = NULL, title = NULL)
```

Arguments

`x`	either: (1) A *named* vector, with names set to the hyperlink text and elements set to the accompanying URL; or (2) a simple character vector of text with another vector of URLs passed to the `url` argument.
`url`	The URL to lead to.
`title`	The optional title of the link.

Details

A link contains link text (the visible text), a link destination (the URI that is the link destination), and optionally a link title. There are two basic kinds of links in Markdown. In inline links the destination and title are given immediately after the link text.

A link text consists of a sequence of zero or more inline elements enclosed by square brackets ([and])...

An inline link consists of a link text followed immediately by a left parenthesis (, optional whitespace, an optional link destination, an optional link title separated from the link destination by whitespace, optional whitespace, and a right parenthesis). The link's text consists of the inlines contained in the link text (excluding the enclosing square brackets). The link's URI consists of the link destination, excluding enclosing <...> if present, with backslash-escapes in effect as described above. The link's title consists of the link title, excluding its enclosing delimiters, with backslash-escapes in effect as described above.

Value

A `glue` vector of collapsed display text and associated URLs.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_escape()`, `md_hardline()`, `md_image()`, `md_issue()`, `md_italic()`, `md_softline()`, `md_strike()`, `md_text()`

Examples

```
md_link("tidyverse", "https://www.tidyverse.org/")
md_link(c(CRAN = "https://cran.r-project.org/"))
```

`md_list` *Markdown generic list*

Description

Turn a character vector into a valid markdown list block. This is a generic function that calls `md_bullet()`, `md_order()`, or `md_task()` depending on what string is provided in the `type` argument.

Usage

```
md_list(x, type = c("bullet", "ordered", "task"), ...)
```

Arguments

`x`	A character vector of list items.
`type`	The type of list to create; either bullet, ordered, or task.
`...`	Arguments passed to the appropriate list type function.

Value

A `glue` vector with length equal to `x`.

See Also

Other container block functions: `md_bullet()`, `md_order()`, `md_quote()`, `md_task()`

Examples

```
md_list(state.name[1:5], type = "bullet", marker = "+")
md_list(state.name[6:10], type = "ordered", marker = ")")
md_list(state.name[11:15], type = "task", check = 3:5)
```

`md_order` *Markdown ordered list*

Description

take a character vector and return a glue vector of valid ordered list items. When printed together, these ordered list items create a ordered list. This container block is rendered as the <ol> HTML tag, with each element of the vector creating a separate <li> tag.

Usage

```
md_order(x, marker = c(".", ")"), seq = TRUE, pad = TRUE)
```

Arguments

`x`	The vector of numbered list items.
`marker`	The ordered list marker following each arabic digits; either . or).
`seq`	logical; Should sequential numbers be used? Defaults to `TRUE`. If `FALSE`, each element will be preceded by the number one; many markdown engines will automatically render repeated ones as a sequential list.
`pad`	logical; If sequential numbers are used, should they be padded with zeroes on the left to match the width of the greatest number? Defaults to `TRUE`.

Details

A list is a sequence of one or more list items of the same type. The list items may be separated by any number of blank lines.

Two list items are of the same type if they begin with a list marker of the same type. Two list markers are of the same type if (b) they are ordered list numbers with the same delimiter (either . or)).

A list is an ordered list if its constituent list items begin with ordered list markers, and a bullet list if its constituent list items begin with bullet list markers.

The start number of an ordered list is determined by the list number of its initial list item. The numbers of subsequent list items are disregarded.

Value

A `glue` vector with length equal to `x`.

See Also

Other container block functions: `md_bullet()`, `md_list()`, `md_quote()`, `md_task()`

Examples

```
md_order(state.name[1:5])
md_order(sample(state.name, 5), marker = ")")
md_order(sample(state.name, 5), seq = FALSE)
```

---

md_paragraph *Markdown paragraphs breaks*

Description

Take a character vector and return a glue vector of paragraphs separated by double newlines. This leaf block is rendered as distinct <p> HTML tags.

Usage

```
md_paragraph(...)
```

Arguments

...	Any number of character vectors.

Details

A sequence of non-blank lines that cannot be interpreted as other kinds of blocks forms a paragraph. The contents of the paragraph are the result of parsing the paragraph's raw content as inlines. The paragraph's raw content is formed by concatenating the lines and removing initial and final whitespace... Paragraphs can contain multiple lines, but no blank lines.

Value

A glue vector with elements of ... separated by two newlines.

See Also

Other leaf block functions: md_blank(), md_chunk(), md_fence(), md_heading(), md_indent(), md_reference(), md_rule(), md_setext(), md_table()

Examples

```
md_paragraph(stringr::sentences[1:3])
```

`md_quote` *Markdown block quotes*

Description

Take a character vector and prepend each element in a greater-than symbol to create a glue vector of block quote markdown text. This inline is rendered as a <blockquote> HTML tag.

Usage

```
md_quote(x)
```

Arguments

`x`	The character vector of quotes.

Details

A block quote marker consists of 0-3 spaces of initial indent, plus (a) the character > together with a following space, or (b) a single character > not followed by a space.

The following rules define block quotes:

1. **Basic case.** If a string of lines *Ls* constitute a sequence of blocks *Bs*, then the result of prepending a block quote marker to the beginning of each line in *Ls* is a block quote containing *Bs*.
2. **Laziness.** If a string of lines *Ls* constitute a block quote with contents *Bs*, then the result of deleting the initial block quote marker from one or more lines in which the next non-whitespace character after the block quote marker is paragraph continuation text is a block quote with *Bs* as its content. Paragraph continuation text is text that will be parsed as part of the content of a paragraph, but does not occur at the beginning of the paragraph.
3. **Consecutiveness.** A document cannot contain two block quotes in a row unless there is a blank line between them.

Nothing else counts as a block quote.

Value

A character vector with a greater-than symbol (>) prepended to each element.

See Also

Other container block functions: `md_bullet()`, `md_list()`, `md_order()`, `md_task()`

Examples

```
md_quote("Give me liberty, or give me death!")
md_quote(stringr::sentences[1:3])
```

`md_reference` *Markdown link reference*

Description

Take character vectors of link texts, link destinations, and optional titles and return single glue vector of valid markdown link references. This markdown leaf block then uses the `label` placed *elsewhere* in a markdown document to render <href> HTML tags.

Usage

```
md_reference(label, url, title = NULL)
```

Arguments

`label`	A link label that is referenced elsewhere in the document.
`url`	The URL to hyperlink the referenced text with.
`title`	An *optional* link title; defaults to `NULL`.

Details

A full reference link (6.6) consists of a link text immediately followed by a link label that matches a link reference definition elsewhere in the document...

A link reference definition consists of a link label, indented up to three spaces, followed by a colon (`:`), optional whitespace (including up to one line ending), a link destination, optional whitespace (including up to one line ending), and an optional link title, which if it is present must be separated from the link destination by whitespace. No further non-whitespace characters may occur on the line.

A link reference definition does not correspond to a structural element of a document. Instead, it defines a label which can be used in reference links and reference-style images elsewhere in the document. Link reference definitions can come either before or after the links that use them.

Value

A single `glue` vector of length equal to that of `label` and `url`, with elements the concatenated arguments.

See Also

Other leaf block functions: `md_blank()`, `md_chunk()`, `md_fence()`, `md_heading()`, `md_indent()`, `md_paragraph()`, `md_rule()`, `md_setext()`, `md_table()`

Examples

```
md_reference("tv", "https://www.tidyverse.org/", "tidyverse")
md_reference(label = 1:2, url = c("https://one.org", "https://two.com"))
```

md_rule *Markdown horizontal rule (4.1)*

Description

Create a glue vector of characters used to represent a *thematic break*. This markdown leaf block is rendered as the <hr> HTML tag.

Usage

```
md_rule(char = c("*", "-", "_"), n = 3, space = FALSE)
```

Arguments

char	The type of rule; either: -, _, or *. Defaults to *.
n	The width of the rule; an integer indicating number of times to repeat each character. Defaults to the minimum of 3.
space	logical or numeric; How many spaces to place between each char. Defaults to FALSE, which places 0 spaces.

Details

A line consisting of 0-3 spaces of indentation, followed by a sequence of three or more matching -, _, or * characters, each followed optionally by any number of spaces or tabs, forms a thematic break.

Value

A repeated-character glue vector with length 1.

See Also

Other leaf block functions: md_blank(), md_chunk(), md_fence(), md_heading(), md_indent(), md_paragraph(), md_reference(), md_setext(), md_table()

Examples

```
md_rule()
md_rule("_")
md_rule(n = 10)
md_rule(space = TRUE)
```

`md_setext` *Markdown Setext headings (4.3)*

Description

Turn a character vector into a vector of valid markdown Setext headings. These markdown leaf blocks can be rendered as the <h1> and <h2> tags *only*.

Usage

```
md_setext(x, level = 1, width = TRUE)
```

Arguments

`x`	A character vector of heading text.
`level`	An numeric vector of all either 1 or 2 to determine whether level 1 headings are created with `=` or level two with `-`. If less levels are provided than headings, `level` will be repeated via `glue::glue()`.
`width`	logical or integer; if `TRUE` the width will be automatically determined by the width of the longest line in `x`. If an integer, the setext underline will be that wide.

Details

A setext heading consists of one or more lines of text, each containing at least one non-whitespace character, with no more than 3 spaces indentation, followed by a setext heading underline. The lines of text must be such that, were they not followed by the setext heading underline, they would be interpreted as a paragraph: they cannot be interpretable as a code fence, ATX heading, block quote, thematic break, list item, or HTML block.

A setext heading underline is a sequence of `=` characters or a sequence of `-` characters, with no more than 3 spaces indentation and any number of trailing spaces. If a line containing a single `-` can be interpreted as an empty list items, it should be interpreted this way and not as a setext heading underline.

The heading is a level 1 heading if `=` characters are used in the setext heading underline, and a level 2 heading if `-` characters are used. The contents of the heading are the result of parsing the preceding lines of text as CommonMark inline content.

In general, a setext heading need not be preceded or followed by a blank line. However, it cannot interrupt a paragraph, so when a setext heading comes after a paragraph, a blank line is needed between them.

Value

A `glue` vector of headings with length equal to `x`.

See Also

Other leaf block functions: `md_blank()`, `md_chunk()`, `md_fence()`, `md_heading()`, `md_indent()`, `md_paragraph()`, `md_reference()`, `md_rule()`, `md_table()`

Examples

```
md_setext("Overview")
md_setext("This is a setext\nheading", level = 2)
md_setext(c("one", "two", "three", "four"), level = c(1, 2))
md_setext("Installation", level = 2, width = 55)
```

`md_softline` *Markdown soft line breaks*

Description

Take a character vector and return a glue vector of separated by a single newline. This inline is rendered as single <p> HTML tags.

Usage

```
md_softline(...)
```

Arguments

`...`	Any number of character vectors.

Details

A regular line break (not in a code span or HTML tag) that is not preceded by two or more spaces or a backslash is parsed as a softbreak. (A softbreak may be rendered in HTML either as a line ending or as a space. The result will be the same in browsers. In the examples here, a line ending will be used.)

Value

A `glue` vector with elements of `...` separated by a single newline.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_escape()`, `md_hardline()`, `md_image()`, `md_issue()`, `md_italic()`, `md_link()`, `md_strike()`, `md_text()`

Examples

```
# compare the following
md_bold(c("One", "Two"))

md_softline(md_bold(c("One", "Two")))
```

`md_strike` *Markdown strikethrough (extension)*

Description

Take a character vector and wrap each element in tildes to create a glue vector of strikethrough text. This inline is rendered as the <strike> HTML tag.

Usage

```
md_strike(x)
```

Arguments

`x`	A character vector of text to be striked through.

Details

GFM enables the strikethrough extension, where an additional emphasis type is available. Strikethrough text is any text wrapped in two tildes (~).

Value

A `glue` vector of length equal to `x`.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_escape()`, `md_hardline()`, `md_image()`, `md_issue()`, `md_italic()`, `md_link()`, `md_softline()`, `md_text()`

Other markdown extensions: `md_issue()`, `md_table()`

Examples

```
md_strike("Example")
md_strike(state.name[1:3])
```

md_table *Markdown tables (4.10 extension)*

Description

Take a list of data frames or matrices and pass them to `knitr::kable()` to create a glue vector containing valid markdown tables.

Usage

```
md_table(x, ...)
```

Arguments

x	An R object to be formatted as a table.
...	Arguments passed to `knitr::kable()`

Details

FM enables the `table` extension, where an additional leaf block type is available.

A table is an arrangement of data with rows and columns, consisting of a single header row, a delimiter row separating the header from the data, and zero or more data rows.

Each row consists of cells containing arbitrary text, in which inlines are parsed, separated by pipes (`|`). A leading and trailing pipe is also recommended for clarity of reading, and if there's otherwise parsing ambiguity. Spaces between pipes and cell content are trimmed. Block-level elements cannot be inserted in a table.

The delimiter row consists of cells whose only content are hyphens (`-`), and optionally, a leading or trailing colon (`:`), or both, to indicate left, right, or center alignment respectively (see `align` in `knitr::kable()`).

Value

A `glue` vector of length one, with each element or row of `x` separated by a newline.

See Also

Other leaf block functions: `md_blank()`, `md_chunk()`, `md_fence()`, `md_heading()`, `md_indent()`, `md_paragraph()`, `md_reference()`, `md_rule()`, `md_setext()`

Other markdown extensions: `md_issue()`, `md_strike()`

Examples

```
md_table(mtcars)
md_table(data.frame(x = LETTERS[1:3], y = 1:3), align = c("cc"))
```

`md_task` *Markdown task list (extension)*

Description

take a character vector and return a glue vector of valid bullet list items. When printed together, these bullet list items create a bullet list. This container block is rendered as the <ul> HTML tag, with each element of the vector creating a separate <li> tag. On venues supporting GitHub Flavored Markdown, this list will be specially rendered with the list item market replaces with a <input type="checkbox"> HTML tag.

Usage

```
md_task(x, check = NULL)
```

Arguments

`x`	A character vector of task list items.
`check`	A optional numeric vector of list elements which should be checked off.

Details

GFM enables the tasklist extension, where an additional processing step is performed on list items.

A task list item is a list item where the first block in it is a paragraph which begins with a task list item marker and at least one whitespace character before any other content.

A task list item marker consists of an optional number of spaces, a left bracket, either a whitespace character or the letter `x` in either lowercase or uppercase, and then a right bracket.

When rendered, the task list item marker is replaced with a semantic checkbox element; in an HTML output, this would be an <input type="checkbox"> element.

If the character between the brackets is a whitespace character, the checkbox is unchecked. Otherwise, the checkbox is checked.

This spec does not define how the checkbox elements are interacted with: in practice, implementors are free to render the checkboxes as disabled or inmutable elements, or they may dynamically handle dynamic interactions (i.e. checking, unchecking) in the final rendered document.

Value

A `glue` vector with length equal to `x`.

See Also

Other container block functions: `md_bullet()`, `md_list()`, `md_order()`, `md_quote()`

Examples

```
md_task(c("Wake up", "Eat Breakfast", "Brush Teeth"), check = c(1, 3))
```

`md_text` *Markdown textual content*

Description

Simple wrapper around `glue::as_glue()`. Take a character vector and return a glue vector.

Usage

```
md_text(x)
```

Arguments

`x`	A character vector.

Details

Any characters not given an interpretation by the [other] rules will be parsed as plain textual content.

Value

A `glue` vector.

See Also

Other inline functions: `md_autolink()`, `md_bold()`, `md_code()`, `md_convert()`, `md_disallow()`, `md_escape()`, `md_hardline()`, `md_image()`, `md_issue()`, `md_italic()`, `md_link()`, `md_softline()`, `md_strike()`

Examples

```
md_text("foo")
```

Index