Package: md4r (via r-universe)

September 17, 2024

Title Markdown Parser Implemented using the 'MD4C' Library

Version 0.5.2.1

Description Provides an R wrapper for the 'MD4C' (Markdown for 'C') library. Functions exist for parsing markdown ('CommonMark' compliant) along with support for other common markdown extensions (e.g. 'GitHub' flavored markdown, 'LaTeX' equation support, etc.). The package also provides a number of higher level functions for exploring and manipulating markdown abstract syntax trees as well as translating and displaying the documents.

License MIT + file LICENSE

URL https://rundel.github.io/md4r/, https://github.com/rundel/md4r

BugReports https://github.com/rundel/md4r/issues

Imports checkmate, cli, glue, purrr, Rcpp, stringr, textutils, tibble

Suggests diffmatchpatch, styler, testthat (>= 3.0.0), withr

LinkingTo Rcpp

Config/testthat/edition 3

Copyright John MacFarlane, RStudio, PBC; Martin Mitáš, Colin Rundel

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://rundel.r-universe.dev

RemoteUrl https://github.com/rundel/md4r

RemoteRef HEAD

RemoteSha c57eaefdb4e0c031bbbfcba73b9bf1c127df39e9

Contents

`flags` *Markdown parser flags*

Description

The `md4c` library supports a number of markdown variants / options. The parsing of these is controlled by flags passed to the parser. The following functions provide commonly used utilities for these flags.

Usage

```
flags_available()

flags_describe()

flags_used(md)
```

Arguments

`md`	Markdown ast object

Value

`flags_available()` returns a character vector of available flags accepted by `parse_md()`.

`flags_describe()` returns a tibble with columns `flag` and `description` describing each flag.

`flags_used()` returns a character vector of flags used in a parsed markdown document.

Examples

```
flags_available()

flags_describe()

md_file = system.file("examples/commonmark.md", package = "md4r")
md = parse_md(md_file)
flags_used(md)
```

md_block *Tools for creating block nodes*

Description

These functions are used to create block nodes. Blocks nodes are used to represent block level elements in markdown.

Usage

```
md_block_doc(..., flags = "MD_DIALECT_COMMONMARK")

md_block_quote(...)

md_block_html(...)

md_block_p(...)

md_block_hr(...)

md_block_code(..., info = "", lang = "", fence_char = "`")

md_block_ul(..., tight = 1, mark = "*")

md_block_ol(..., tight = 1L, start = 1, mark_delimiter = ".")

md_block_li(..., is_task = 0, task_mark = " ")

md_block_h(..., level)

md_block_table(..., col_count, body_row_count, head_row_count = 1)

md_block_thead(...)

md_block_tbody(...)

md_block_tr(...)

md_block_th(..., align = c("default", "left", "center", "right"))

md_block_td(..., align = c("default", "left", "center", "right"))
```

Arguments

...	Child nodes that will be contained within the block - all must inherit from md_node.
flags	Used by md_block_doc to specify the document parser flag(s).

info	Used by md_block_code nodes to specify the code block info string.
lang	Used by md_block_code nodes to specify the code block language.
fence_char	Used by md_block_code nodes to specify the code block fence character.
tight	Used by md_block_ul or md_block_ol nodes to specify whether the list is tight or loose.
mark	Used by md_block_ul nodes to specify the list marker.
start	Used by md_block_ol nodes to specify the start number of the list.
mark_delimiter	Used by md_block_ol nodes to specify the delimiter between the value and text. Only "." and ")" are allowed.
is_task	Used by md_block_li nodes to specify whether the item is a task.
task_mark	Used by md_block_li task nodes to specify the task mark character. Only " ", "x", and "X" are allowed.
level	Used by md_block_h nodes to specify the heading level.
col_count	Used by md_block_table nodes to specify the number of columns.
body_row_count	Used by md_block_table nodes to specify the number of body rows.
head_row_count	Used by md_block_table nodes to specify the number of header rows. Should only be 1.
align	Used by md_block_td or md_block_th nodes to specify the alignment of the table contents.

Value

Returns a list with a class of specified type along with md_span and md_node.

See Also

md_span, md_text

---

md_span *Tools for creating span nodes*

Description

These functions are used to create span nodes. Span nodes are used to represent inline elements in markdown and include things like links, images, code, emphasized text, strong text, and more.

Usage

```
md_span_em(...)

md_span_strong(...)

md_span_a(..., href, title)

md_span_img(..., src, title)

md_span_code(...)

md_span_del(...)

md_span_latexmath(...)

md_span_latexmath_display(...)

md_span_wikilink(..., target)

md_span_u(...)
```

Arguments

`...`	Child nodes that will be contained within the span - all must inherit from `md_node`.
`href`	Used by `md_span_a` nodes to provide the `href` attribute.
`title`	Used by `md_span_a` or `md_span_img` nodes to provide a `title` attribute.
`src`	Used by `md_span_img` nodes to provide the `src` attribute.
`target`	Used by `md_span_wikilink` nodes to provide the `target` attribute.

Value

Returns a list with a class of specified type along with `md_span` and `md_node`.

See Also

md_block, md_text

`md_text` *Tools for creating text nodes*

Description

These functions are used to create text nodes. Text nodes are used to represent textual elements in markdown.

Usage

```
md_text_normal(x)

md_text_nullchar()

md_text_br()

md_text_softbr()

md_text_entity(x)

md_text_code(x)

md_text_html(x)

md_text_latexmath(x)
```

Arguments

x	Text content of the node.

Value

Returns a character vector with a class of specified type along with `md_text` and `md_node`.

See Also

md_block, md_span

---

parse_md *Parse markdown*

Description

Parse either a literal markdown string or a markdown file given a path. Different dialects and features are supported via the `flags` argument. See `flags_describe()` for possible flags and their usage. `parse_md()` defaults parsing using the commonmark spec while `parse_gfm()` uses the GitHub flavored markdown spec.

Usage

```
parse_md(md, flags = "MD_DIALECT_COMMONMARK")

parse_gfm(md, flags = "MD_DIALECT_GITHUB")
```

Arguments

md	Character. Either literal string of markdown or a path to a markdown file.
flags	Character vector. Dialect flags used by the parser.

Value

Both functions return a markdown ast, a list with the md_block_doc class.

Examples

```
parse_md(system.file("examples/commonmark.md", package = "md4r"))

parse_gfm(system.file("examples/github.md", package = "md4r"))
```

---

to_html *Convert to html*

Description

Coverts a markdown object (full ast or node) to HTML.

Usage

```
to_html(md, ...)
```

Arguments

md	Markdown object
...	Unused, for extensibility.

Value

Returns a character vector of HTML lines representing the markdown object.

Examples

```
md_file = system.file("examples/commonmark.md", package = "md4r")
md = parse_md(md_file)
cat(to_html(md), sep="\n")
```

to_md *Convert to markdown*

Description

Coverts a markdown object (full ast or node) to markdown text.

Usage

```
to_md(md, ...)
```

Arguments

md	Markdown object
...	Unused, for extensibility.

Value

Returns a character vector of markdown lines representing the markdown object.

Examples

```
md_file = system.file("examples/commonmark.md", package = "md4r")
md = parse_md(md_file)
cat(to_md(md), sep="\n")
```

Index