Package 'parsermd'

May 9, 2026

Title Formal Parser and Related Tools for R Markdown Documents

Version 0.2.0

Description An implementation of a formal grammar and parser for R Markdown documents using the Boost Spirit X3 library. It also includes a collection of high level functions for working with the resulting abstract syntax tree.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports purrr, Rcpp, cli (>= 2.5.0), checkmate, readr, tidyr, dplyr, tibble, yaml, withr, rmarkdown, pillar, rlang, magrittr, tidyselect (>= 1.2.0), fs, quarto, S7

RoxygenNote 7.3.2

SystemRequirements C++17

LinkingTo Rcpp, BH

Suggests testthat (>= 3.0.0), knitr, styler, stringr

Config/testthat/edition 3

VignetteBuilder quarto

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

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

NeedsCompilation yes

Author Colin Rundel [aut, cre]

Maintainer Colin Rundel <rundel@gmail.com>

Repository CRAN

Date/Publication 2025-08-19 15:30:02 UTC

Contents

`as_ast` *Convert an object into an* `rmd_ast`.

Description

Currently only supports conversion of `rmd_tibble` objects back to `rmd_ast`.

Usage

```
as_ast(x, ...)
```

Arguments

x	Object to convert
...	Unused, for extensibility.

Value

Returns an rmd_ast object.

Examples

```
parse_rmd(system.file("examples/hw01.Rmd", package="parsermd")) %>%
  as_tibble() %>%
  as_ast()
```

---

as_document *Convert an* rmd_ast*,* rmd_tibble*, or any ast node into text.*

Description

Convert an rmd_ast, rmd_tibble, or any ast node into text.

Usage

```
as_document(x, padding = "", collapse = NULL, use_yaml_opts = TRUE, ...)
```

Arguments

x	rmd_ast, rmd_tibble, or parsermd node object.
padding	Padding to add between nodes when assembling the text.
collapse	If not NULL, use value to collapse lines.
use_yaml_opts	Logical. Whether to use YAML format for chunk options (default TRUE).
...	Passed to to_ast() when converting rmd_collection or qmd_collection.

Value

Returns a character vector.

chunk_options *Get and set code chunk options*

Description

Helper functions for obtaining or changing chunk options within an rmd object.

Usage

```
rmd_set_options(x, ...)

rmd_get_options(x, ..., defaults = list(), yaml_style = TRUE)
```

Arguments

x	An rmd_ast, rmd_tibble, or any rmd ast node object.
...	Either a collection of named values for the setter or a character values of the option names for the getter.
defaults	A named list of default values for the options.
yaml_style	logical, if TRUE (default) return option names in YAML style (with hyphens), if FALSE return normalized style (with dots)

Value

rmd_set_options returns the modified version of the original object.

rmd_get_options returns a list of the requested options (or all options if none are specified). Non-chunk nodes return NULL.

Examples

```
rmd = parse_rmd(system.file("examples/minimal.Rmd", package = "parsermd"))

str(rmd_get_options(rmd))
str(rmd_get_options(rmd, "include"))

# Get options in YAML style (default) vs normalized style
chunk = rmd_chunk("r", "test", options = list(`fig-width` = 8, eval = TRUE))
rmd_get_options(chunk, yaml_style = TRUE)   # fig-width
rmd_get_options(chunk, yaml_style = FALSE)  # fig.width

rmd_set_options(rmd, include = TRUE)
```

inline_code_utils *Inline code detection and extraction utilities*

Description

Functions for detecting and extracting inline code chunks from AST nodes after the initial parsing phase.

Usage

```
rmd_has_inline_code(x, engine = NULL)

rmd_extract_inline_code(x, flatten = FALSE)
```

Arguments

x	An AST node, list of nodes, or character vector
engine	character vector, optional glob patterns for matching inline code engine names. If NULL (default), matches any inline code.
flatten	Return a flat list inline codes if TRUE

Value

- `rmd_has_inline_code()`: logical vector indicating which nodes contain inline code
- `rmd_extract_inline_code()`: list of inline code objects found in the content

parse_collection *Parse a collection of R Markdown or Quarto document*

Description

[Experimental]

Recursively searches a directory for R Markdown or Quarto documents and parses them into a collection of `rmd_ast` objects

Usage

```
parse_qmd_collection(
  dir = "./",
  pattern = "*.qmd",
  all = FALSE,
  recurse = TRUE,
  regex = FALSE
)
```

```
parse_rmd_collection(
  dir = "./",
  pattern = "*.Rmd",
  all = FALSE,
  recurse = TRUE,
  regex = FALSE
)
```

Arguments

dir	Directory to search
pattern	Pattern to match files, defaults to glob syntax
all	Search includes hidden files
recurse	Search recusively within dir
regex	Treat pattern as a regular expression syntax for pattern

Value

Returns a tibble object with columns for document name, path, and ast.

Examples

```
parse_rmd_collection(system.file("examples/", package="parsermd"))
```

---

parse_rmd *Parse an R Markdown or Quarto document*

Description

Documents are parsed into an rmd_ast object.

Usage

```
parse_rmd(rmd)

parse_qmd(qmd)
```

Arguments

rmd	Either the path to an Rmd file or a character vector containing the contents of a R Markdown document.
qmd	Either the path to an qmd file or a character vector containing the contents of a Quarto document.

Value

Returns a `rmd_ast` object.

Examples

```
parse_rmd(system.file("examples/hw01.Rmd", package="parsermd"))
```

---

render *Render* `parsermd` *objects*

Description

Object contents are converted to a character vector and written to a temporary directory before rendering via `quarto::quarto_render()` or `rmarkdown::render()`.

Note that this function has the potential to overwrite existing output files (e.g. `.html`, `.pdf`, etc).

Usage

```
render(x, name = NULL, ..., engine = c("quarto", "rmarkdown"))
```

Arguments

`x`	Object to render, e.g. a `rmd_ast`, `rmd_tibble`, `character` vector, etc.
`name`	Name of the output file, if not given it will be inferred from the name of `x`.
`...`	Any additional arguments to be passed to `quarto::quarto_render()` or `rmarkdown::render()`
`engine`	The rendering engine to use, either "quarto" or "rmarkdown".

Value

Returns the results of the render function.

---

rmd_ast_append *Append or preprend nodes to an ast*

Description

Functions for adding nodes to the beginning or end of an ast.

Usage

```
rmd_ast_append(x, ...)

rmd_ast_prepend(x, ...)
```

Arguments

x	An object containing an rmd_ast of some kind, e.g. rmd_ast, rmd_tibble, or rmd_collection.
...	A collections of ast nodes to append or prepend.

Value

An object of the same class as x

---

rmd_check_template *Check an Rmd against a template*

Description

This function compares the provided Rmd against a template and reports on discrepancies (e.g. missing or unmodified components).

Usage

```
rmd_check_template(rmd, template, ...)
```

Arguments

rmd	The rmd to be check, can be an rmd_ast, rmd_tibble, or text that can be handled by parse_rmd.
template	rmd_template object from rmd_template().
...	Unused, for extensibility.

Value

Invisibly returns TRUE if the rmd matches the template, FALSE otherwise.

Examples

```
tmpl = parse_rmd(system.file("examples/hw01.Rmd", package = "parsermd")) %>%
  rmd_select(by_section(c("Exercise *", "Solution"))) %>%
  rmd_template(keep_content = TRUE)

rmd_check_template(
  system.file("examples/hw01-student.Rmd", package = "parsermd"),
  tmpl
)
```

rmd_chunk *Code chunk node*

Description

S7 class representing an executable code chunk

Usage

```
rmd_chunk(
  engine = "r",
  label = "",
  options = list(),
  code = character(),
  indent = "",
  n_ticks = 3L
)
```

Arguments

engine	Character. Language engine
label	Character. Chunk label
options	List. Combined chunk options (traditional and YAML)
code	Character vector. Code lines
indent	Character. Indentation
n_ticks	Integer. Number of backticks

rmd_classes_s7 *S7 Class Definitions for RMD Nodes*

Description

S7 classes for representing R Markdown AST nodes with automatic validation. All classes inherit from the base rmd_node class and provide type-safe object creation with built-in validation of properties.

Usage

```
rmd_node()

rmd_ast(nodes = list())

rmd_yaml(yaml = list())
```

Arguments

nodes	List of rmd_node objects for the AST container
yaml	List containing YAML frontmatter content

Details

The following S7 classes are available for creating R Markdown AST nodes:

Core Classes:

- `rmd_node()` - Abstract base class for all R Markdown AST nodes. This is the parent class for all specific node types and should not be instantiated directly.
- `rmd_ast()` - Container for multiple nodes representing a complete document AST.

Content Nodes:

- `rmd_yaml()` - YAML frontmatter header containing document metadata.
- `rmd_chunk()` - Code chunks with executable code in various engines (R, Python, etc.).
- `rmd_raw_chunk()` - Raw code chunks that are not executed.
- `rmd_markdown()` - Plain markdown text content.
- `rmd_heading()` - Section headings at various levels (1-6).

Code and Inline Elements:

- `rmd_code_block()` - Fenced code blocks without execution.
- `rmd_code_block_literal()` - Code blocks with literal {{...}} attributes.
- `rmd_inline_code()` - Inline code spans within markdown text.
- `rmd_shortcode()` - Quarto/Pandoc shortcodes for special functionality.
- `rmd_span()` - Generic inline spans with attributes.

Structural Elements:

- `rmd_fenced_div_open()` - Opening tags for fenced divs (:::).
- `rmd_fenced_div_close()` - Closing tags for fenced divs (:::).

See Also

rmd_node_utilities for utility functions that work with these S7 objects

rmd_code_block *Markdown code block node*

Description

S7 class representing a fenced code block

Usage

```
rmd_code_block(
  id = character(),
  classes = character(),
  attr = character(),
  code = character(),
  indent = "",
  n_ticks = 3L
)
```

Arguments

id	Character vector. HTML ID (length 0 or 1)
classes	Character vector. CSS classes
attr	Named character vector. Key-value attributes (keys as names)
code	Character vector. Code lines
indent	Character. Indentation
n_ticks	Integer. Number of backticks

rmd_code_block_literal *Code block literal node*

Description

S7 class representing a code block with {{...}} attributes

Usage

```
rmd_code_block_literal(
  attr = "",
  code = character(),
  indent = "",
  n_ticks = 3L
)
```

Arguments

attr	Character. Raw attribute content from {{...}}
code	Character vector. Code lines
indent	Character. Indentation
n_ticks	Integer. Number of backticks

---

rmd_fenced_div_close *Closing fenced div node*

Description

S7 class representing the closing of a fenced div

Usage

```
rmd_fenced_div_close()
```

---

rmd_fenced_div_open *Opening fenced div node*

Description

S7 class representing the opening of a fenced div

Usage

```
rmd_fenced_div_open(
  id = character(),
  classes = character(),
  attr = character()
)
```

Arguments

id	Character vector. HTML ID (length 0 or 1)
classes	Character vector. CSS classes
attr	Named character vector. Key-value attributes (keys as names)

`rmd_fenced_div_wrap` *Wrap selected nodes with fenced divs*

Description

This function wraps selected nodes in an `rmd_ast` with fenced div opening and closing tags. The selection is implemented using the same approach as `rmd_select()` which enables a variety of useful syntax for selecting nodes.

The function checks if the selected indices form a single continuous range. If this is not the case then an error will be thrown. If wrapping multiple discontinous ranges of nodes is desired then the `allow_multiple` can be set to `TRUE`.

Usage

```
rmd_fenced_div_wrap(
  x,
  ...,
  open = rmd_fenced_div_open(),
  allow_multiple = FALSE
)
```

Arguments

`x`	Rmd object, e.g. `rmd_ast` or `rmd_tibble`.
`...`	One or more unquoted expressions separated by commas for node selection. Uses the same syntax as `rmd_select()`.
`open`	An `rmd_fenced_div_open` node that defines the opening fenced div. Defaults to `rmd_fenced_div_open()`.
`allow_multiple`	Logical. If `FALSE` (default), throws an error when selected indices are discontinuous. If `TRUE`, allows wrapping multiple discontinuous ranges separately.

Value

Returns the modified Rmd object with selected nodes wrapped in the fenced div(s).

Examples

```
rmd = parse_rmd(system.file("examples/hw01.Rmd", package = "parsermd"))

rmd_fenced_div_wrap(rmd, "plot-dino":"cor-dino")

rmd_fenced_div_wrap(rmd, has_type("rmd_chunk"), allow_multiple=TRUE)

rmd_fenced_div_wrap(
  rmd, has_type("rmd_chunk"),
  open = rmd_fenced_div_open(classes = ".note"),
  allow_multiple = TRUE
```

```
)
```

`rmd_heading` *Heading node*

Description

S7 class representing a markdown heading

Usage

```
rmd_heading(name = character(0), level = integer(0))
```

Arguments

`name`	Character. Heading text
`level`	Integer. Heading level (1-6)

`rmd_inline_code` *Inline code node*

Description

S7 class representing inline code

Usage

```
rmd_inline_code(
  engine = "",
  code = "",
  braced = FALSE,
  start = -1L,
  length = -1L
)
```

Arguments

`engine`	Character. Language engine
`code`	Character. Code content
`braced`	Logical. Whether code is braced
`start`	Integer. Start position
`length`	Integer. Length

`rmd_insert` *Insert nodes at specified locations*

Description

This function inserts nodes into an `rmd_ast` at specified locations relative to selected nodes. The selection is implemented using the same approach as `rmd_select()` which enables a variety of useful syntax for selecting nodes.

The function checks if the selected indices form continuous ranges and can either insert before or after that range. `allow_multiple` parameter can be used to allow insertion based on multiple discontinuous ranges.

Usage

```
rmd_insert(
  x,
  ...,
  nodes,
  location = c("before", "after"),
  allow_multiple = FALSE
)
```

Arguments

`x`	Rmd object, e.g. `rmd_ast` or `rmd_tibble`, to insert the nodes.
`...`	One or more unquoted expressions separated by commas for node selection. Uses the same syntax as `rmd_select()`.
`nodes`	Nodes to insert. Can be a single rmd node object, a list of rmd node objects, or an `rmd_ast` object.
`location`	Character. Either "before" or "after" to specify where to insert relative to the selected nodes. "before" inserts before the first node of each selected range, "after" inserts after the last node of each selected range.
`allow_multiple`	Logical. If `FALSE` (default), throws an error when selected indices are discontinuous. If `TRUE`, allows inserting at multiple discontinuous locations.

Value

Returns the modified Rmd object with nodes inserted at the specified locations.

Examples

```
rmd = parse_rmd(system.file("examples/hw01.Rmd", package = "parsermd"))

new_nodes = list(
  rmd_markdown(lines = "This is a comment"),
  rmd_chunk(engine = "r", code = "# New code")
```

```
)
rmd_insert(rmd, "plot-dino", nodes = new_nodes, location = "after")

new_heading = rmd_heading(name = "Analysis", level = 2L)
rmd_insert(rmd, has_type("rmd_chunk"), nodes = new_heading,
           location = "before", allow_multiple = TRUE)

rmd_insert(rmd, c(1, 3, 5),
  nodes = rmd_markdown(lines = "Separator"),
  location = "after",
  allow_multiple = TRUE)
```

---

rmd_markdown *Markdown text node*

Description

S7 class representing markdown text content

Usage

```
rmd_markdown(lines = character(0))
```

Arguments

lines	Character vector. Markdown text lines

---

rmd_modify *Modify nodes of an Rmd ast*

Description

This function applies a function to selected nodes of an `rmd_ast` or `rmd_tibble`. The selection is implemented using the same approach as `rmd_select()` which enables a variety of useful syntax for selecting nodes from the ast.

The function `.f` must return a valid rmd node object (e.g., `rmd_chunk`, `rmd_heading`, etc.). The results are validated to ensure they maintain the proper structure and class.

Usage

```
rmd_modify(x, .f, ...)
```

Arguments

x	Rmd object, e.g. rmd_ast or rmd_tibble.
.f	A function to apply to the selected nodes. Must return a valid rmd node object.
...	Selection arguments (unnamed) and function arguments (named). Unnamed arguments are used for node selection using tidyselect syntax. Named arguments are passed to the function .f.

Value

Returns the modified Rmd object (either rmd_ast or rmd_tibble depending on input). Only the selected nodes are modified by applying .f, while unselected nodes remain unchanged.

Examples

```
rmd = parse_rmd(system.file("examples/hw01.Rmd", package = "parsermd"))

# Modify specific chunks by label
f = function(node) { # Add a comment to the chunk
  node@code = c("# Modified chunk", node@code)
  node
}
rmd_modify(rmd, .f = f, "plot-dino") |>
  rmd_select("plot-dino") |>
  as_document() |>
  cat(sep="\n")

# Modify all chunks with named arguments passed to function
f = function(node, prefix = "## ") {
  node@code = paste0(prefix, node@code)
  node
}
rmd_modify(rmd, f, has_type("rmd_chunk"), prefix = "# ") |>
 rmd_select(has_type("rmd_chunk")) |>
 as_document() |>
 cat(sep="\n")
```

---

rmd_node_sections *Find the sections for each rmd object node*

Description

Uses the section headings of an rmd object to identify the hierarchical structure of the document.

Usage

```
rmd_node_sections(x, levels = 1:6, drop_na = FALSE)
```

Arguments

x	An rmd object, e.g. rmd_ast or rmd_tibble.
levels	Limit which section heading levels to return.
drop_na	Should NA sections be dropped.

Value

A list of section names for each node.

---

rmd_node_utilities *rmd node utility functions*

Description

Functions for extracting information for Rmd nodes.

Usage

```
rmd_node_label(x)

rmd_node_label(x) <- value

## Default S3 replacement method:
rmd_node_label(x) <- value

## S3 replacement method for class 'rmd_chunk'
rmd_node_label(x) <- value

rmd_node_type(x)

rmd_node_length(x)

rmd_node_content(x)

rmd_node_attr(x, attr)

rmd_node_engine(x)

rmd_node_options(x, yaml_style = TRUE)

rmd_node_code(x)

rmd_node_options(x) <- value

## Default S3 replacement method:
rmd_node_options(x) <- value
```

```
## S3 replacement method for class 'rmd_chunk'
rmd_node_options(x) <- value

rmd_node_attr(x, attr) <- value

## Default S3 replacement method:
rmd_node_attr(x, attr) <- value

## S3 replacement method for class 'rmd_node'
rmd_node_attr(x, attr) <- value

rmd_node_content(x) <- value

## Default S3 replacement method:
rmd_node_content(x) <- value

## S3 replacement method for class 'rmd_chunk'
rmd_node_content(x) <- value

## S3 replacement method for class 'rmd_raw_chunk'
rmd_node_content(x) <- value

## S3 replacement method for class 'rmd_markdown'
rmd_node_content(x) <- value

## S3 replacement method for class 'rmd_code_block'
rmd_node_content(x) <- value

## S3 replacement method for class 'rmd_code_block_literal'
rmd_node_content(x) <- value

rmd_node_set_label(x, value)

rmd_node_set_options(x, ...)

rmd_node_set_content(x, value)

rmd_node_set_attr(x, attr, value)
```

Arguments

x	An rmd object, e.g. `rmd_ast` or `rmd_tibble`.
value	The new value to assign (for assignment functions).
attr	Attribute name to extract or set.
yaml_style	logical, if `TRUE` (default) return option names in YAML style (with hyphens), if `FALSE` return normalized style (with dots)

... For rmd_node_set_options(), named arguments that will be converted to a list of options to assign.

Value

- rmd_node_label() - returns a character vector of node labels, nodes without labels return NA.
- rmd_node_label<-() - assigns new labels to chunk nodes. For the setter, returns the modified object.
- rmd_node_type() - returns a character vector of node types.
- rmd_node_length() - returns an integer vector of node lengths (i.e. lines of code, lines of text, etc.), nodes without a length return NA.
- rmd_node_content() - returns the raw character vector(s) of node textual content (lines/code), nodes without content return NULL.
- rmd_node_attr() - returns the value of a given node attribute (S7 property), returns NULL if the attribute does not exist.
- rmd_node_engine() - returns a character vector of chunk engines, NA for all other node types.
- rmd_node_options() - returns a list of chunk node options (named list), NULL for all other node types. Option names are returned in YAML style (with hyphens) by default, or normalized style (with dots) if yaml_style = FALSE.
- rmd_node_options<-() - assigns new options to chunk nodes by merging with existing options. Takes a named list of options. For the setter, returns the modified object.
- rmd_node_attr<-() - assigns new attribute values to nodes. For the setter, returns the modified object.
- rmd_node_code() - returns a list of chunk node code (character vector), NULL for all other node types.
- rmd_node_set_label() - pipeable version of rmd_node_label<-() for setting node labels.
- rmd_node_set_options() - pipeable version of rmd_node_options<-() for setting chunk options.
- rmd_node_set_attr() - pipeable version of rmd_node_attr<-() for setting node attributes.
- rmd_node_content<-() - assigns new content to nodes. For the setter, returns the modified object.
- rmd_node_set_content() - pipeable version of rmd_node_content<-() for setting node content.

Examples

```
rmd = parse_rmd(system.file("examples/hw01.Rmd", package="parsermd"))

rmd_node_label(rmd)
rmd_node_type(rmd)
rmd_node_content(rmd)
rmd_node_attr(rmd, "level")
rmd_node_engine(rmd)
rmd_node_options(rmd)
rmd_node_code(rmd)
```

```
chunk = rmd_chunk("r", "example", code = "1 + 1")
rmd_node_label(chunk)
rmd_node_label(chunk) = "new_name"
rmd_node_label(chunk)

rmd_node_options(chunk) = list(eval = FALSE, echo = TRUE)
rmd_node_options(chunk)

rmd_node_attr(chunk, "engine") = "python"
rmd_node_attr(chunk, "engine")

rmd_node_content(chunk) = c("x = 2", "y = 3")
rmd_node_content(chunk)

chunk = rmd_chunk("r", "example", code = "1 + 1") |>
  rmd_node_set_label("new_label") |>
  rmd_node_set_options(eval = FALSE, echo = TRUE) |>
  rmd_node_set_content(c("a = 1", "b = 2"))

rmd_node_label(chunk)
rmd_node_options(chunk)
rmd_node_options(chunk, yaml_style = FALSE)  # get in normalized style
rmd_node_content(chunk)

chunk = rmd_chunk("r", "example", code = "x = 1") |>
  rmd_node_set_attr("engine", "python")

rmd_node_engine(chunk)
```

`rmd_raw_chunk` *Raw chunk node*

Description

S7 class representing a raw output chunk

Usage

```
rmd_raw_chunk(
  format = character(0),
  code = character(),
  indent = "",
  n_ticks = 3L
)
```

Arguments

`format`	Character. Output format

code	Character vector. Code lines
indent	Character. Indentation
n_ticks	Integer. Number of backticks

`rmd_select` *Select nodes of an Rmd ast*

Description

This function is implemented using `tidyselect::eval_select()` which enables a variety of useful syntax for selecting nodes from the ast.

Additionally, a number of `parsermd` helpers are available: `by_section()`, `has_type()`, `has_label()`, and `has_option()`.

Usage

```
rmd_select(x, ..., keep_yaml = TRUE)
```

Arguments

x	Rmd object, e.g. `rmd_ast` or `rmd_tibble`.
...	One or more unquoted expressions separated by commas. Chunk labels can be used as if they were positions in the data frame, so expressions like x:y can be used to select a range of nodes.
keep_yaml	Logical, whether to automatically include YAML nodes in the selection. If `TRUE` (default), equivalent to including `has_type("rmd_yaml")` in the selection.

Value

Returns a subset Rmd object (either `rmd_ast` or `rmd_tibble` depending on input).

See Also

rmd_select_helpers for helper functions to use with `rmd_select()`, including `by_section()`, `has_type()`, `has_label()`, and `has_option()`.

Examples

```
rmd = parse_rmd(system.file("examples/hw01.Rmd", package = "parsermd"))

rmd_select(rmd, "plot-dino", "cor-dino")
rmd_select(rmd, "plot-dino":"cor-dino")
rmd_select(rmd, `plot-dino`:`cor-dino`)

rmd_select(rmd, has_type("rmd_chunk"))

rmd_select(rmd, by_section(c("Exercise *", "Solution")))
```

`rmd_select_helpers` *Rmd selection helper functions*

Description

These functions are used in conjunction with `rmd_select()` to select nodes from an Rmd ast.

- `by_section()` - uses section selectors to select nodes.
- `has_type()` - selects all nodes that have the given type(s).
- `has_label()` - selects nodes with labels matching the given glob.
- `has_heading()` - selects heading nodes (only) with titles matching the given glob pattern(s).
- `has_option()` - selects nodes that have the given option(s) set.
- `has_shortcode()` - selects nodes containing shortcodes matching the given function name(s).
- `by_fenced_div()` - selects fenced div sections matching specified id, class, and/or attributes.

Usage

```
has_type(types)

by_section(sec_ref, keep_parents = TRUE)

has_label(label)

has_heading(heading)

has_code(code)

has_option(...)

has_shortcode(func_name = NULL)

has_inline_code(engine = NULL)

by_fenced_div(id = NULL, class = NULL, attr = NULL)
```

Arguments

`types`	Vector of character type names, e.g. `rmd_chunk`, `rmd_heading`, etc.
`sec_ref`	character vector, a section reference selector. See details below for further details on how these are constructed.
`keep_parents`	Logical, retain the parent headings of selected sections. Default: `TRUE`
`label`	character vector, glob patterns for matching chunk labels.
`heading`	character vector, glob patterns for matching heading titles.
`code`	character vector, regex patterns for matching chunk code line(s)

`...`	Either option names represented by a scalar string or a named argument with the form `opt = value` where `opt` is the option name and `value` is the value to be checked. For example `eval = TRUE` would check for the option `eval` being set to `TRUE`.
`func_name`	character vector, optional glob patterns for matching shortcode function names. If NULL (default), matches any shortcode.
`engine`	character vector, optional glob patterns for matching inline code engine names. If NULL (default), matches any inline code.
`id`	Character, optional ID to match (with or without # prefix)
`class`	Character vector, optional class names to match (with `.` prefix). All specified classes must be present in the fenced div (subset matching).
`attr`	Either a character vector of attribute names to check for existence, or a named list/vector where names are attribute names and values must match exactly.

Details

Section reference selectors:

Section reference selectors are a simplified version of CSS selectors that are designed to enable the selection nodes in a way that respects the implied hierarchy of a document's section headings.

They consist of a character vector of heading names where each subsequent value is assumed to be nested within the preceding value. For example, the section selector `c("Sec 1", "Sec 2")` would select all nodes that are contained within a section named `Sec 2` that is in turn contained within a section named `Sec 1` (or a section contained within a section named `Sec 1`, and so on).

The individual section names can be specified using wildcards (aka globbing patterns), which may match one or more sections within the document, e.g. `c("Sec 1", "Sec *")`. See `utils::glob2rx()` or wikipedia for more details on the syntax for these patterns.

Value

All helper functions return an integer vector of selected indexes.

See Also

`rmd_select()` for the main selection function that uses these helpers.

Examples

```
rmd = parse_rmd(system.file("examples/hw01.Rmd", package="parsermd"))

rmd_select(rmd, has_type("rmd_chunk"))

rmd_select(rmd, has_label("*dino"))

rmd_select(rmd, has_heading("Exercise *"))

rmd_select(rmd, has_option("message"))
rmd_select(rmd, has_option(message = FALSE))
rmd_select(rmd, has_option(message = TRUE))
```

```
rmd_select(rmd, has_shortcode())
rmd_select(rmd, has_shortcode("video"))

fdiv = parse_rmd(system.file("examples/fenced-divs.qmd", package="parsermd"))

rmd_select(fdiv, by_fenced_div())  # Select all fenced div pairs
rmd_select(fdiv, by_fenced_div(class = "note"))
rmd_select(fdiv, by_fenced_div(id = "special-section"))
rmd_select(fdiv, by_fenced_div(class = c("warning", "important")))
rmd_select(fdiv, by_fenced_div(attr = "icon"))
```

---

rmd_shortcode *Shortcode node*

Description

S7 class representing a shortcode function call

Usage

```
rmd_shortcode(
  func = character(0),
  args = character(),
  start = -1L,
  length = -1L
)
```

Arguments

func	Character. Function name
args	Character vector. Function arguments
start	Integer. Start position
length	Integer. Length

---

rmd_source *Source the code chunks of an Rmd document*

Description

This is the equivalent of the source() function for Rmd files or their resulting asts.

Usage

```
rmd_source(x, local = FALSE, ..., label_comment = TRUE, use_eval = TRUE)
```

Arguments

x	An Rmd document (e.g. rmd_ast, rmd_tibble, Rmd file path, etc.)
local	TRUE, FALSE or an environment, determining where the parsed expressions are evaluated. FALSE (the default) corresponds to the user's workspace (the global environment) and TRUE to the environment from which source is called.
...	Additional arguments passed to source.
label_comment	Attach chunk labels as comment before each code block.
use_eval	Use the eval chunk option to determine if code is included.

Value

Returns the result of source() for any R code chunks.

Examples

```
rmd_source(system.file("examples/minimal.Rmd", package = "parsermd"), echo=TRUE)
```

---

rmd_span *Span node*

Description

S7 class representing a span with attributes

Usage

```
rmd_span(
  text = "",
  id = character(),
  classes = character(),
  attr = character()
)
```

Arguments

text	Character. Span text content
id	Character vector. HTML ID (length 0 or 1)
classes	Character vector. CSS classes
attr	Named character vector. Additional attributes

`rmd_template` *Create a template from an* `rmd` *object.*

Description

Templates are objects which are meant to capture the structure of an R Markdown document and facilitate the comparison between the template and new Rmd documents, usually to ensure the structure and/or content matches sufficiently.

Usage

```
rmd_template(
  rmd,
  keep_content = FALSE,
  keep_labels = TRUE,
  keep_headings = FALSE,
  keep_yaml = FALSE,
  ...
)
```

Arguments

`rmd`	R Markdown document in the form of an `rmd_ast` or `rmd_tibble`.
`keep_content`	Should the template keep the document's content (markdown text and chunk code).
`keep_labels`	Should the template keep the document's code chunk labels.
`keep_headings`	Should the template keep the document's headings.
`keep_yaml`	Should the template keep the document's yaml.
`...`	Unused, for extensibility.

Value

Returns an `rmd_template` object, which is a derived tibble containing relevant structural details of the document.

Examples

```
rmd = parse_rmd(system.file("examples/hw01.Rmd", package="parsermd"))

rmd_select(rmd, by_section(c("Exercise *", "Solution"))) %>%
  rmd_template()
```

shortcode_utils *Shortcode detection and extraction utilities*

Description

Functions for detecting and extracting shortcodes from AST nodes after the initial parsing phase.

Usage

```
rmd_has_shortcode(x, func_name = NULL)

rmd_extract_shortcodes(x, flatten = FALSE)
```

Arguments

x	An AST node, list of nodes, or character vector
func_name	character vector, optional glob patterns for matching shortcode function names. If NULL (default), matches any shortcode.
flatten	Return a flat list shortcodes if TRUE

Value

- `rmd_has_shortcode()`: logical vector indicating which nodes contain shortcodes
- `rmd_extract_shortcodes()`: list of shortcode objects found in the content

span_utils *Span detection and extraction utilities*

Description

Functions for detecting and extracting spans from AST nodes after the initial parsing phase.

Usage

```
rmd_has_span(x)

rmd_extract_spans(x, flatten = FALSE)
```

Arguments

x	An AST node, list of nodes, or character vector
flatten	Return a flat list of spans if TRUE

Value

- `rmd_has_span()`: logical vector indicating which nodes contain spans
- `rmd_extract_spans()`: list of span objects found in the content

Index