Package: htmltomarkdown (via r-universe)

June 27, 2026

Title High-performance HTML to Markdown converter

Version 3.8.0

Description High-performance HTML to Markdown converter Rust bindings generated with extendr.

URL https://github.com/xberg-io/html-to-markdown

BugReports https://github.com/xberg-io/html-to-markdown/issues

License MIT

Depends R (>= 4.2)

Imports jsonlite

Suggests testthat (>= 3.0.0), withr, roxygen2, lintr, styler

SystemRequirements Cargo (Rust's package manager), rustc (>= 1.91)

Config/rextendr/version 0.4.2

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

Config/pak/sysreqs libclang-dev

Repository https://xberg-io.r-universe.dev

Date/Publication 2026-06-27 11:34:46 UTC

RemoteUrl https://github.com/xberg-io/html-to-markdown

RemoteRef main

RemoteSha ef432d15d22a37de2cbd0a3159000b68acf14f09

RemoteSubdir packages/r

Contents

`AnnotationKind` *The type of an inline text annotation*

Description

Uses internally tagged representation (`"annotation_type": "bold"`) for JSON serialization.

Usage

```
AnnotationKind
```

CodeBlockStyle *Create a CodeBlockStyle enum value*

Description

Returns the default CodeBlockStyle variant.

Usage

```
CodeBlockStyle()
```

Value

A CodeBlockStyle enum value

---

conversion_options *Create a ConversionOptions list for generated bindings*

Description

All parameters default to `NULL`, which means the Rust default is used. Pass named arguments to override individual settings.

Usage

```
conversion_options(
  heading_style = NULL,
  list_indent_type = NULL,
  list_indent_width = NULL,
  bullets = NULL,
  strong_em_symbol = NULL,
  escape_asterisks = NULL,
  escape_underscores = NULL,
  escape_misc = NULL,
  escape_ascii = NULL,
  code_language = NULL,
  autolinks = NULL,
  default_title = NULL,
  br_in_tables = NULL,
  compact_tables = NULL,
  highlight_style = NULL,
  extract_metadata = NULL,
```

```
  whitespace_mode = NULL,
  strip_newlines = NULL,
  wrap = NULL,
  wrap_width = NULL,
  convert_as_inline = NULL,
  sub_symbol = NULL,
  sup_symbol = NULL,
  newline_style = NULL,
  code_block_style = NULL,
  keep_inline_images_in = NULL,
  preprocessing = NULL,
  encoding = NULL,
  debug = NULL,
  strip_tags = NULL,
  preserve_tags = NULL,
  skip_images = NULL,
  url_escape_style = NULL,
  link_style = NULL,
  output_format = NULL,
  include_document_structure = NULL,
  extract_images = NULL,
  max_image_size = NULL,
  capture_svg = NULL,
  infer_dimensions = NULL,
  max_depth = NULL,
  exclude_selectors = NULL,
  visitor = NULL
)
```

Arguments

`heading_style`	Heading style to use in Markdown output (ATX # or Setext underline)
`list_indent_type`	How to indent nested list items (spaces or tab)
`list_indent_width`	Number of spaces (or tabs) to use for each level of list indentation
`bullets`	Bullet character(s) to use for unordered list items (e.g. `"-"`, `"*"`)
`strong_em_symbol`	Character used for bold/italic emphasis markers (`*` or `_`)
`escape_asterisks`	Escape `*` characters in plain text to avoid unintended bold/italic
`escape_underscores`	Escape _ characters in plain text to avoid unintended bold/italic
`escape_misc`	Escape miscellaneous Markdown metacharacters (`[]()#` etc.) in plain text
`escape_ascii`	Escape ASCII characters that have special meaning in certain Markdown dialects
`code_language`	Default language annotation for fenced code blocks that have no language hint

`autolinks` Automatically convert bare URLs into Markdown autolinks

`default_title` Emit a default title when no `<title>` tag is present

`br_in_tables` Render `<br>` elements inside table cells as literal line breaks

`compact_tables` Emit tables without column padding (compact GFM format)

`highlight_style`
Style used for `<mark>` / highlighted text (e.g. `==text==`)

`extract_metadata`
Populate `result.metadata` with `<head>` / `<meta>` extraction

`whitespace_mode`
Controls how whitespace sequences are normalised in the converted output

`strip_newlines` Strip all newlines from the output, producing a single-line result

`wrap` Wrap long lines at `wrap_width` characters

`wrap_width` Maximum output line width in characters when `wrap` is `true` (default `80`)

`convert_as_inline`
Treat the entire document as inline content (no block-level wrappers)

`sub_symbol` Markdown notation for subscript text (e.g. `"~"`)

`sup_symbol` Markdown notation for superscript text (e.g. `"^"`)

`newline_style` How to encode hard line breaks (`<br>`) in Markdown

`code_block_style`
Style used for fenced code blocks (backticks or tilde)

`keep_inline_images_in`
HTML tag names whose `<img>` children are kept inline instead of block

`preprocessing` Options for the HTML pre-processing pass applied before conversion begins

`encoding` Expected character encoding of the input HTML (default `"utf-8"`)

`debug` Emit debug information during conversion

`strip_tags` HTML tag names whose content is stripped from the output entirely

`preserve_tags` HTML tag names that are preserved verbatim in the output

`skip_images` Skip conversion of `<img>` elements (omit images from output)

`url_escape_style`
URL encoding strategy for link and image destinations

`link_style` Link rendering style (inline or reference)

`output_format` Target output format (Markdown, plain text, etc.)

`include_document_structure`
Include structured document tree in result

`extract_images` Extract inline images from data URIs and SVGs

`max_image_size` Maximum decoded image size in bytes (default 5MB)

`capture_svg` Capture SVG elements as images

`infer_dimensions`
Infer image dimensions from data

`max_depth` Maximum DOM traversal depth. `None` means unlimited

`exclude_selectors`
CSS selectors for elements to exclude entirely (element + all content)

`visitor` (feature-gated) Optional visitor for custom traversal logic

Value

A named list suitable for the `options` argument of `convert()`.

`ConversionOptions` *Main conversion options for HTML to Markdown conversion*

Description

Use `ConversionOptions::builder()` to construct, or `Default::default()` for defaults.

Usage

```
ConversionOptions
```

Fields

`heading_style` Heading style to use in Markdown output (ATX # or Setext underline).

`list_indent_type` How to indent nested list items (spaces or tab).

`list_indent_width` Number of spaces (or tabs) to use for each level of list indentation.

`bullets` Bullet character(s) to use for unordered list items (e.g. `"-"`, `"*"`).

`strong_em_symbol` Character used for bold/italic emphasis markers (`*` or `_`).

`escape_asterisks` Escape `*` characters in plain text to avoid unintended bold/italic.

`escape_underscores` Escape `_` characters in plain text to avoid unintended bold/italic.

`escape_misc` Escape miscellaneous Markdown metacharacters (`[]()#` etc.) in plain text.

`escape_ascii` Escape ASCII characters that have special meaning in certain Markdown dialects.

`code_language` Default language annotation for fenced code blocks that have no language hint.

`autolinks` Automatically convert bare URLs into Markdown autolinks.

`default_title` Emit a default title when no `<title>` tag is present.

`br_in_tables` Render `<br>` elements inside table cells as literal line breaks.

`compact_tables` Emit tables without column padding (compact GFM format).

`highlight_style` Style used for `<mark>` / highlighted text (e.g. `==text==`).

`extract_metadata` Populate `result.metadata` with `<head>` / `<meta>` extraction (title, description, Open

`whitespace_mode` Controls how whitespace sequences are normalised in the converted output.

`strip_newlines` Strip all newlines from the output, producing a single-line result.

`wrap` Wrap long lines at `wrap_width` characters.

`wrap_width` Maximum output line width in characters when `wrap` is `true` (default `80`).

`convert_as_inline` Treat the entire document as inline content (no block-level wrappers).

`sub_symbol` Markdown notation for subscript text (e.g. `"~"`).

`sup_symbol` Markdown notation for superscript text (e.g. `"^"`).

`newline_style` How to encode hard line breaks (`<br>`) in Markdown.

`code_block_style` Style used for fenced code blocks (backticks or tilde).

`keep_inline_images_in` HTML tag names whose `<img>` children are kept inline instead of block.

`preprocessing` Options for the HTML pre-processing pass applied before conversion begins.

`encoding` Expected character encoding of the input HTML (default `"utf-8"`).

`debug` Emit debug information during conversion.

`strip_tags` HTML tag names whose content is stripped from the output entirely.

`preserve_tags` HTML tag names that are preserved verbatim in the output.

`skip_images` Skip conversion of `<img>` elements (omit images from output).

`url_escape_style` URL encoding strategy for link and image destinations.

`link_style` Link rendering style (inline or reference).

`output_format` Target output format (Markdown, plain text, etc.).

`include_document_structure` Include structured document tree in result.

`extract_images` Extract inline images from data URIs and SVGs.

`max_image_size` Maximum decoded image size in bytes (default 5MB).

`capture_svg` Capture SVG elements as images.

`infer_dimensions` Infer image dimensions from data.

`max_depth` Maximum DOM traversal depth. `None` means unlimited. When set, subtrees beyond this depth are

`exclude_selectors` CSS selectors for elements to exclude entirely (element + all content).

`visitor` Optional visitor for custom traversal logic.

`ConversionOptionsUpdate`

Partial update for `ConversionOptions`

Description

Uses `Option<T>` fields for selective updates. Bindings use this to construct options from language-native types. Prefer `ConversionOptionsBuilder` for Rust code.

Usage

```
ConversionOptionsUpdate
```

Fields

`heading_style` Optional override for `ConversionOptions::heading_style`.

`list_indent_type` Optional override for `ConversionOptions::list_indent_type`.

`list_indent_width` Optional override for `ConversionOptions::list_indent_width`.

`bullets` Optional override for `ConversionOptions::bullets`.

`strong_em_symbol` Optional override for `ConversionOptions::strong_em_symbol`.

`escape_asterisks` Optional override for `ConversionOptions::escape_asterisks`.

`escape_underscores` Optional override for `ConversionOptions::escape_underscores`.

`escape_misc` Optional override for `ConversionOptions::escape_misc`.

`escape_ascii` Optional override for `ConversionOptions::escape_ascii`.

`code_language` Optional override for `ConversionOptions::code_language`.

`autolinks` Optional override for `ConversionOptions::autolinks`.

`default_title` Optional override for `ConversionOptions::default_title`.

`br_in_tables` Optional override for `ConversionOptions::br_in_tables`.

`compact_tables` Optional override for `ConversionOptions::compact_tables`.

`highlight_style` Optional override for `ConversionOptions::highlight_style`.

`extract_metadata` Optional override for `ConversionOptions::extract_metadata`.

`whitespace_mode` Optional override for `ConversionOptions::whitespace_mode`.

`strip_newlines` Optional override for `ConversionOptions::strip_newlines`.

`wrap` Optional override for `ConversionOptions::wrap`.

`wrap_width` Optional override for `ConversionOptions::wrap_width`.

`convert_as_inline` Optional override for `ConversionOptions::convert_as_inline`.

`sub_symbol` Optional override for `ConversionOptions::sub_symbol`.

`sup_symbol` Optional override for `ConversionOptions::sup_symbol`.

`newline_style` Optional override for `ConversionOptions::newline_style`.

`code_block_style` Optional override for `ConversionOptions::code_block_style`.

`keep_inline_images_in` Optional override for `ConversionOptions::keep_inline_images_in`.

`preprocessing` Optional override for `ConversionOptions::preprocessing`.

`encoding` Optional override for `ConversionOptions::encoding`.

`debug` Optional override for `ConversionOptions::debug`.

`strip_tags` Optional override for `ConversionOptions::strip_tags`.

`preserve_tags` Optional override for `ConversionOptions::preserve_tags`.

`skip_images` Optional override for `ConversionOptions::skip_images`.

`url_escape_style` Optional override for `ConversionOptions::url_escape_style`.

`link_style` Optional override for `ConversionOptions::link_style`.

`output_format` Optional override for `ConversionOptions::output_format`.

`include_document_structure` Optional override for `ConversionOptions::include_document_structure`.

`extract_images` Optional override for `ConversionOptions::extract_images`.

`max_image_size` Optional override for `ConversionOptions::max_image_size`.

`capture_svg` Optional override for `ConversionOptions::capture_svg`.

`infer_dimensions` Optional override for `ConversionOptions::infer_dimensions`.

`max_depth` Optional override for `ConversionOptions::max_depth`.

`exclude_selectors` Optional override for `ConversionOptions::exclude_selectors`.

`visitor` Optional override for `ConversionOptions::visitor`.

`convert` *Convert HTML to Markdown, returning a* `ConversionResult` *with content, metadata, images,*

Description

and warnings.

Usage

```
convert(html, options = ConversionOptions$default())
```

Arguments

`html`	— the HTML string to convert.
`options`	— optional conversion options. Defaults to `ConversionOptions::default`.

Value

ConversionResult object (list with class attribute).

Errors

Returns an error if HTML parsing fails or if the input contains invalid UTF-8.

DocumentMetadata *Document-level metadata extracted from* <head> *and top-level elements*

Description

Contains all metadata typically used by search engines, social media platforms, and browsers for document indexing and presentation.

Usage

```
DocumentMetadata
```

Fields

`title` Document title from <title> tag

`description` Document description from <meta name="description"> tag

`keywords` Document keywords from <meta name="keywords"> tag, split on commas

`author` Document author from <meta name="author"> tag

`canonical_url` Canonical URL from <link rel="canonical"> tag

`base_href` Base URL from <base href=""> tag for resolving relative URLs

`language` Document language from `lang` attribute

`text_direction` Document text direction from `dir` attribute

`open_graph` Open Graph metadata (og:* properties) for social media Keys like "title", "description", "image",

`twitter_card` Twitter Card metadata (twitter:* properties) Keys like "card", "site", "creator", "title",

`meta_tags` Additional meta tags not covered by specific fields Keys are meta name/property attributes, values

GridCell *A single cell in a table grid*

Description

A single cell in a table grid

Usage

```
GridCell
```

Fields

`content` The text content of the cell.

`row` 0-indexed row position.

`col` 0-indexed column position.

`row_span` Number of rows this cell spans (default 1).

`col_span` Number of columns this cell spans (default 1).

`is_header` Whether this is a header cell (<th>).

HeaderMetadata *Header element metadata with hierarchy tracking*

Description

Captures heading elements (h1-h6) with their text content, identifiers, and position in the document structure.

Usage

```
HeaderMetadata
```

Fields

`level` Header level: 1 (h1) through 6 (h6)

`text` Normalized text content of the header

`id` HTML id attribute if present

`depth` Document tree depth at the header element

`html_offset` Byte offset in original HTML document

HeadingStyle *Create a HeadingStyle enum value*

Description

Returns the default HeadingStyle variant.

Usage

```
HeadingStyle()
```

Value

A HeadingStyle enum value

HighlightStyle *Create a HighlightStyle enum value*

Description

Returns the default HighlightStyle variant.

Usage

```
HighlightStyle()
```

Value

A HighlightStyle enum value

ImageMetadata *Image metadata with source and dimensions*

Description

Captures <img> elements and inline <svg> elements with metadata for image analysis and optimization.

Usage

```
ImageMetadata
```

Fields

`src` Image source (URL, data URI, or SVG content identifier)

`alt` Alternative text from alt attribute (for accessibility)

`title` Title attribute (often shown as tooltip)

`dimensions` Image dimensions as (width, height) if available

`image_type` Image type classification

`attributes` Additional HTML attributes

ImageType *Create a ImageType enum value*

Description

Returns the default ImageType variant.

Usage

```
ImageType()
```

Value

A ImageType enum value

LinkMetadata *Hyperlink metadata with categorization and attributes*

Description

Represents <a> elements with parsed href values, text content, and link type classification.

Usage

```
LinkMetadata
```

Fields

`href` The href URL value

`text` Link text content (normalized, concatenated if mixed with elements)

`title` Optional title attribute (often shown as tooltip)

`link_type` Link type classification

`rel` Rel attribute values (e.g., "nofollow", "stylesheet", "canonical")

`attributes` Additional HTML attributes

LinkStyle *Create a LinkStyle enum value*

Description

Returns the default LinkStyle variant.

Usage

```
LinkStyle()
```

Value

A LinkStyle enum value

LinkType *Create a LinkType enum value*

Description

Returns the default LinkType variant.

Usage

```
LinkType()
```

Value

A LinkType enum value

ListIndentType *Create a ListIndentType enum value*

Description

Returns the default ListIndentType variant.

Usage

```
ListIndentType()
```

Value

A ListIndentType enum value

NewlineStyle *Create a NewlineStyle enum value*

Description

Returns the default NewlineStyle variant.

Usage

```
NewlineStyle()
```

Value

A NewlineStyle enum value

NodeContent *The semantic content type of a document node*

Description

Uses internally tagged representation (`"node_type": "heading"`) for JSON serialization.

Usage

```
NodeContent
```

NodeContext *Context information passed to all visitor methods*

Description

Provides comprehensive metadata about the current node being visited, including its type, attributes, position in the DOM tree, and parent context.

Usage

```
NodeContext
```

Fields

`node_type` Coarse-grained node type classification

`tag_name` Raw HTML tag name (e.g., "div", "h1", "custom-element")

`attributes` All HTML attributes as key-value pairs

`depth` Depth in the DOM tree (0 = root)

`index_in_parent` Index among siblings (0-based)

`parent_tag` Parent element's tag name (None if root)

`is_inline` Whether this element is treated as inline vs block

`NodeType` *Create a NodeType enum value*

Description

Returns the default NodeType variant.

Usage

```
NodeType()
```

Value

A NodeType enum value

`OutputFormat` *Create a OutputFormat enum value*

Description

Returns the default OutputFormat variant.

Usage

```
OutputFormat()
```

Value

A OutputFormat enum value

PreprocessingOptions *HTML preprocessing options for document cleanup before conversion*

Description

HTML preprocessing options for document cleanup before conversion

Usage

```
PreprocessingOptions
```

Fields

`enabled` Enable HTML preprocessing globally

`preset` Preprocessing preset level (Minimal, Standard, Aggressive)

`remove_navigation` Remove navigation elements (nav, breadcrumbs, menus, sidebars)

`remove_forms` Remove form elements (forms, inputs, buttons, etc.)

PreprocessingOptionsUpdate *Partial update for* PreprocessingOptions

Description

This struct uses `Option<T>` to represent optional fields that can be selectively updated. Only specified fields (Some values) will override existing options; None values leave the corresponding fields unchanged when applied via `PreprocessingOptions::apply_update`.

Usage

```
PreprocessingOptionsUpdate
```

Fields

`enabled` Optional global preprocessing enablement override

`preset` Optional preprocessing preset level override (Minimal, Standard, Aggressive)

`remove_navigation` Optional navigation element removal override (nav, breadcrumbs, menus, sidebars)

`remove_forms` Optional form element removal override (forms, inputs, buttons, etc.)

PreprocessingPreset *Create a PreprocessingPreset enum value*

Description

Returns the default PreprocessingPreset variant.

Usage

```
PreprocessingPreset()
```

Value

A PreprocessingPreset enum value

ProcessingWarning *A non-fatal diagnostic produced during HTML conversion*

Description

Warnings indicate that conversion completed but some content may have been handled differently than expected — for example, an image that could not be extracted, a truncated input, or malformed HTML that was repaired with best-effort parsing.

Usage

```
ProcessingWarning
```

Details

Conversion always succeeds (returns `ConversionResult`) even when warnings are present. Callers should inspect `warnings` and decide how to handle them based on their tolerance for partial results:

- **Logging pipelines**: emit each warning at `WARN` level and continue.
- **Strict pipelines**: treat any warning as a hard error by checking `result.warnings.is_empty()` before using the output.

See `WarningKind` for the full taxonomy of warning categories.

Fields

`message` Human-readable warning message.

`kind` The category of warning.

StructuredData *Structured data block (JSON-LD, Microdata, or RDFa)*

Description

Represents machine-readable structured data found in the document. JSON-LD blocks are collected as raw JSON strings for flexibility.

Usage

```
StructuredData
```

Fields

`data_type` Type of structured data (JSON-LD, Microdata, RDFa)

`raw_json` Raw JSON string (for JSON-LD) or serialized representation

`schema_type` Schema type if detectable (e.g., "Article", "Event", "Product")

StructuredDataType *Create a StructuredDataType enum value*

Description

Returns the default StructuredDataType variant.

Usage

```
StructuredDataType()
```

Value

A StructuredDataType enum value

TableData — *A top-level extracted table with both structured data and markdown representation*

Description

A top-level extracted table with both structured data and markdown representation

Usage

```
TableData
```

Fields

`grid` The structured table grid.

`markdown` The markdown rendering of this table.

TextAnnotation — *A styling or semantic annotation that applies to a byte range within a node's text*

Description

Unlike `DocumentNode`, which captures block-level structure (headings, paragraphs, etc.), a `TextAnnotation` describes inline-level markup — bold, italic, links, code spans, and similar — that spans a contiguous run of bytes inside `DocumentNode::content`'s text field.

Usage

```
TextAnnotation
```

Details

Byte offsets (`start..end`) are into the UTF-8 encoded text of the parent node. The range follows Rust slice conventions: `start` is inclusive and `end` is exclusive, so the annotated text is `text[start as usize..end as usize]`.

Multiple annotations on the same node can overlap (e.g. bold-italic text), and they are stored in the order they are encountered during DOM traversal.

See `AnnotationKind` for the full list of supported annotation types.

Fields

`start` Start byte offset (inclusive) into the parent node's text.

`end` End byte offset (exclusive) into the parent node's text.

`kind` The type of annotation.

TextDirection *Create a TextDirection enum value*

Description

Returns the default TextDirection variant.

Usage

```
TextDirection()
```

Value

A TextDirection enum value

UrlEscapeStyle *Create a UrlEscapeStyle enum value*

Description

Returns the default UrlEscapeStyle variant.

Usage

```
UrlEscapeStyle()
```

Value

A UrlEscapeStyle enum value

VisitResult *Result of a visitor callback*

Description

Allows visitors to control the conversion flow by either proceeding with default behavior, providing custom output, skipping elements, preserving HTML, or signaling errors.

Usage

```
VisitResult
```

Fields

`Continue` Continue with default conversion behavior

`Custom` Replace default output with custom markdown

`Skip` Skip this element entirely (don't output anything)

`PreserveHtml` Preserve original HTML (don't convert to markdown)

`Error` Stop conversion with an error

`WarningKind` *Create a WarningKind enum value*

Description

Returns the default WarningKind variant.

Usage

```
WarningKind()
```

Value

A WarningKind enum value

`WhitespaceMode` *Create a WhitespaceMode enum value*

Description

Returns the default WhitespaceMode variant.

Usage

```
WhitespaceMode()
```

Value

A WhitespaceMode enum value

Index