

# Package ‘hypertext’

April 17, 2026

**Title** 'HTML' Element Construction

**Version** 1.2.0

**Description** Provides a deterministic, framework-agnostic Domain-Specific Language for building 'HTML' nodes and rendering them to a string.

**License** MIT + file LICENSE

**URL** <https://github.com/sigflux/hypertext>

**BugReports** <https://github.com/sigflux/hypertext/issues>

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Kennedy Mwavu [aut, cre] (ORCID:  
<<https://orcid.org/0009-0006-3157-7234>>),  
Sigflux [cph, fnd]

**Maintainer** Kennedy Mwavu <[mwavukennedy@gmail.com](mailto:mwavukennedy@gmail.com)>

**Repository** CRAN

**Date/Publication** 2026-04-17 20:50:02 UTC

## Contents

doctype . . . . .	2
raw_html . . . . .	2
render . . . . .	3
tag . . . . .	4
tags . . . . .	5
tag_list . . . . .	6

<b>Index</b>	<b>7</b>
--------------	----------

---

doctype	<i>Render the <code>&lt;!DOCTYPE html&gt;</code> declaration</i>
---------	--

---

### Description

Convenience wrapper around `raw_html()` that returns the HTML5 document-type declaration. Useful when building a full page.

### Usage

```
doctype()
```

### Value

A "hypertext.raw" object containing `<!DOCTYPE html>`.

### Examples

```
page <- tag_list(  
  doctype(),  
  tags$html(  
    tags$head(tags$title("Home")),  
    tags$body(tags$h1("Welcome"))  
  )  
)  
render(page)
```

---

raw_html	<i>Mark a string as raw HTML</i>
----------	----------------------------------

---

### Description

Wraps one or more character strings so that `render()` outputs them verbatim, **without** escaping HTML special characters.

### Usage

```
raw_html(...)
```

### Arguments

... Character strings of raw HTML /// Optional.

### Details

This is useful for injecting pre-rendered HTML, inline `<script>/<style>` blocks, SVG markup, or any content that should not be entity-escaped.

**Value**

A character vector of class "hypertext.raw".

**Examples**

```
raw_html("<script>alert('hi')</script>")
render(tags$div(raw_html("<em>already formatted</em>")))
```

---

render	<i>Render an HTML node tree to a character string</i>
--------	---

---

**Description**

Converts a `hypertext.tag` object (and all its descendants) into an HTML string. Text children are escaped; nested `hypertext.tag` children are rendered recursively.

**Usage**

```
render(x, ...)

## S3 method for class 'hypertext.tag'
render(x, file = "", write_mode = c("overwrite", "append"), ...)

## Default S3 method:
render(x, file = "", write_mode = c("overwrite", "append"), ...)

## S3 method for class 'hypertext.raw'
render(x, file = "", write_mode = c("overwrite", "append"), ...)

## S3 method for class 'list'
render(x, file = "", write_mode = c("overwrite", "append"), ...)
```

**Arguments**

x	hypertext.tag object, a string, or a list of these <i>///</i> Required.
...	Further arguments passed from or to other methods.
file	String <i>///</i> Optional. Path to file to print to.
write_mode	String <i>///</i> Optional. Either "overwrite" (default) which overwrites the contents of file, or "append" which appends the HTML string to file.

**Details**

When `file` is provided, the rendered HTML is written to the specified file via `cat()` and the HTML string is returned invisibly.

**Value**

A single character string of HTML.

**Examples**

```
page <- tags$html(
  tags$head(
    tags$title("Home")
  ),
  tags$body(
    tags$h1("Welcome")
  )
)

# return HTML as a string:
render(page)

## Not run:
# write to a file:
render(page, file = "index.html")

## End(Not run)
```

---

tag

---

*Create an HTML element node*


---

**Description**

Low-level constructor. Named arguments become attributes; unnamed arguments become children (text or nested nodes).

**Usage**

```
tag(tag_name, ..., tag_type = c("normal", "void"))
```

**Arguments**

tag_name	String /// Required. The HTML element name.
...	Attributes (named) and children (unnamed) /// Optional.
tag_type	String /// Optional. Either "normal" (default) for a standard element with children, or "void" for a self-closing element whose children are ignored.

**Value**

List of class "hypertext.tag" with components tag, attrs, children, and tag\_type.

**Examples**

```
# web component
tag(tag_name = "calcite-action-bar", layout = "horizontal")

# custom element with children
tag(
  tag_name = "my-card",
  class = "shadow",
  tag(tag_name = "my-card-header", "Title"),
  tag(tag_name = "my-card-body", "Content")
)

# custom void element
tag(tag_name = "my-icon", name = "home", tag_type = "void")
```

tags

*HTML tag functions***Description**

A named list of functions, one per HTML5 element. Access individual tags with `tags$div(...)`, `tags$p(...)`, etc.

**Usage**

```
tags
```

**Format**

An object of class `list` of length 115.

**Details**

Named arguments become HTML attributes; unnamed arguments become child nodes or text content.

**Boolean / valueless attributes**

Pass `NA` or `TRUE` as the attribute value to produce a valueless attribute (e.g. `disabled`). `FALSE` or `NULL` suppresses the attribute entirely.

**Multi-value attributes**

Character vectors of length `> 1` are collapsed with a space, so `class = c("a", "b")` renders as `class="a b"`.

**Examples**

```
tags$p(class = "lead", "Hello, world!")
render(tags$div(id = "app", tags$h1("Title")))
```

---

tag_list	<i>Create a tag list (sibling container)</i>
----------	--

---

**Description**

Useful when you want to collect sibling nodes into a single object without wrapping them in a parent element.

**Usage**

```
tag_list(...)
```

**Arguments**

...                   Tags, text, or other renderable objects /// Optional.

**Value**

A list of class "hypertext.tag.list".

**Examples**

```
t1 <- tag_list(tags$p("one"), tags$p("two"))
render(t1)
```

# Index

\* **datasets**  
tags, 5

cat(), 3

doctype, 2

raw\_html, 2  
raw\_html(), 2  
render, 3  
render(), 2

tag, 4  
tag\_list, 6  
tags, 5