

Package: rmdtools (via r-universe)

October 4, 2026

Type Package

Title Tools for RMarkdown

Version 0.2.7

Date 2019-06-22

URL <https://github.com/skranz/rmdtools>

Author Sebastian Kranz

Maintainer Sebastian Kranz <sebastian.kranz@uni-ulm.de>

Description Tools for RMarkdown

License GPL >= 2.0

Depends restorepoint, dplyr, stringtools, htmltools, codeUtils,
markdown, commonmark, knitr

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev

Repository <https://skranz.r-universe.dev>

Date/Publication 2026-04-07 06:48:17 UTC

RemoteUrl <https://github.com/skranz/rmdtools>

RemoteRef master

RemoteSha 399fa7001c3860894dd8eb5aad0122495a1979e2

Contents

adapt.hf.blocks	3
adapt.if.blocks	4
add.code.ui	4
args.to.chunk.header	4
cat.rmd.params	5
cat.whisker.params	5
chunk.opt.string.to.list	5
compile.rmd	6
eval.chunk.like.whisker	6

eval.chunks.in.text	7
eval.placeholder	7
eval.placeholders	8
eval.rmd.blocks.in.text	8
eval.whiskers.in.text	9
fast_df	9
find.dot.blocks	9
find.rmd.blocks	10
find.rmd.chunks	10
find.rmd.nested	11
get.levels.parents	11
get.levels.parents.by.types	12
get.start.end.levels	12
get.types.inds	13
hideShowButton	13
html.table	14
html.template	14
import.yaml	15
inline.dependencies	15
inline.dependency	15
knit.chunk	16
knit.rmd	17
knit.rmd.in.temp	17
make.chunk.info	17
make.placeholder.info	18
make.whisker.info	18
makeDependenciesRelativeToPackage	18
makeDependencyRelativeToPackage	19
mark_utf8	19
md2html	19
normalize.id	20
parse.arg.str	20
parse.block.args	21
parse.chunk.args	21
parse.chunk.names	21
parse.hashdot.yaml	22
paste.whiskers	22
print.yaml	22
random.string	23
read.as.utf8	23
read.yaml	23
readUtf8	24
remove.quotes	24
remove.rmd.chunks	24
render.compiled.rmd	25
render.deps.as.singletons.tags	25
render.rmd.in.temp	26
render.value	26

replace.if.blocks	26
replace.whiskers	27
reverse.whisker.placeholders	27
rmd.between.start.end.lines	28
rmd.blocks.to.placeholders	28
rmd.chunks.to.placeholders	29
rmd.whiskers.to.placeholders	29
sc	29
simple.html.page	30
split.by.line.start	30
table.knit_print.data.frame	31
transformTagDependencies	31
view.html	32
view.rmd	32
whisker_print	33
whisker_print.default	33
writeUtf8	33
Index	34

adapt.hf.blocks	<i>Adapt header and footer of if blocks for output format and parse already the if condition for faster runtime evaluation</i>
-----------------	--

Description

Adapt header and footer of if blocks for output format and parse already the if condition for faster runtime evaluation

Usage

```
adapt.hf.blocks(
  txt,
  block.df = NULL,
  out.type = "html",
  only.types = c("if", "note"),
  ...
)
```

adapt.if.blocks	<i>Adapt header and footer of if blocks for output format and parse already the if condition for faster runtime evaluation</i>
-----------------	--

Description

Adapt header and footer of if blocks for output format and parse already the if condition for faster runtime evaluation

Usage

```
adapt.if.blocks(
  txt,
  block.df = NULL,
  out.type = "html",
  only.types = "if",
  ...
)
```

add.code.ui	<i>Add code button with div to an ui</i>
-------------	--

Description

Add code button with div to an ui

Usage

```
add.code.ui(
  ui = NULL,
  code,
  show_code,
  code.highlight = TRUE,
  print_remove_code_note = TRUE
)
```

args.to.chunk.header	<i>Create a Rmd chunk header line from a list of arguments</i>
----------------------	--

Description

Create a Rmd chunk header line from a list of arguments

Usage

```
args.to.chunk.header(args = list(), type = "r", label = args$label)
```

cat.rmd.params	<i>Scan all used block and whisker parameters in an .rmd file and create a template for a list</i>
----------------	--

Description

Scan all used block and whisker parameters in an .rmd file and create a template for a list

Usage

```
cat.rmd.params(
  file = NULL,
  text = readLines(file, warn = FALSE),
  use.blocks = TRUE,
  use.whiskers = TRUE
)
```

cat.whisker.params	<i>cat.whisker params</i>
--------------------	---------------------------

Description

cat.whisker params

Usage

```
cat.whisker.params(file = NULL, text = readLines(file, warn = FALSE))
```

chunk.opt.string.to.list	<i>Translates a chunk header to a list of its option</i>
--------------------------	--

Description

Translates a chunk header to a list of its option

Usage

```
chunk.opt.string.to.list(str, keep.name = FALSE)
```

Arguments

str	the chunk header as written in the rmd file
keep.name	shall the chunk name be kept?

code.compile.rmd
Main function to compile rmd to html

Description

Main function to compile rmd to html

Usage

```
compile.rmd(
  file = NULL,
  text = readLines(file, warn = FALSE),
  envir = list(),
  if.blocks = c("ph", "render", "ignore")[1],
  blocks = c("ph", "render", "ignore")[1],
  whiskers = c("ph", "render", "render.as.text", "ignore")[1],
  chunks = c("ph", "knit", "render", "ignore")[1],
  start.line = "<!-- START -->",
  end.line = "<!-- END -->",
  set.utf8 = TRUE,
  out.type = "html",
  fragment.only = FALSE,
  whiskers.call.list = NULL,
  blocks.call.list = NULL,
  add.info = TRUE,
  use.commonmark = FALSE
)
```

eval.chunk.like.whisker

Render a knitr chunk in the same way as a whisker (taking into some chunk options, like results="asis")

Description

Render a knitr chunk in the same way as a whisker (taking into some chunk options, like results="asis")

Usage

```
eval.chunk.like.whisker(
  code,
  call = NULL,
  options = NULL,
  env = parent.frame()
)
```

eval.chunks.in.text	<i>Render all knitr chunks in the same way as a whisker (taking into some chunk options, like results="asis")</i>
---------------------	---

Description

Render all knitr chunks in the same way as a whisker (taking into some chunk options, like results="asis")

Usage

```
eval.chunks.in.text(rmd, envir = parent.frame())
```

eval.placeholder	<i>Evaluate a placeholder and return its value</i>
------------------	--

Description

Evaluate a placeholder and return its value

Usage

```
eval.placeholder(  
  ph,  
  envir = parent.frame(),  
  chunks = "knit",  
  dir = getwd(),  
  out.type = "html",  
  cr = NULL,  
  on.error = c("null", "error", "stop")[1],  
  use.commonmark = TRUE,  
  ...  
)
```

eval.placeholders	<i>Evaluate placeholders in compiled rmd</i>
-------------------	--

Description

Evaluate placeholders in compiled rmd

Usage

```
eval.placeholders(  
  cr = NULL,  
  envir = parent.frame(),  
  ph = cr$ph,  
  type = NULL,  
  on.error = c("null", "error", "stop")[1],  
  out.type = first.none.null(cr$out.type, "html"),  
  ...  
)
```

eval.rmd.blocks.in.text	<i>compile all given types of rmd blocks</i>
-------------------------	--

Description

compile all given types of rmd blocks

Usage

```
eval.rmd.blocks.in.text(  
  txt,  
  to = "html",  
  envir = parent.frame(),  
  call.list = NULL,  
  block.df = NULL,  
  only.types = NULL,  
  ignore.types = "if",  
  use.del.rows.na = FALSE,  
  replace.funs = NULL  
)
```

eval.whiskers.in.text *evaluate whiskers and replace them in the text*

Description

evaluate whiskers and replace them in the text

Usage

```
eval.whiskers.in.text(
  str,
  envir = parent.frame(),
  signif = getOption("whiskerSignifDigits"),
  round = getOption("whiskerRoundDigits"),
  add.params = TRUE,
  whiskers.call.list = NULL
)
```

fast_df *Create a data_frame (tibble) considerably faster*

Description

columns must all be named and cannot be computed from earlier columns

Usage

```
fast_df(...)
```

find.dot.blocks *Find blocks that have only a starting line of the form*

Description

#. type arguments

Usage

```
find.dot.blocks(
  txt,
  dot.levels = NULL,
  dot.start = "#. ",
  single.line.level = Inf
)
```

find.rmd.blocks	<i>Find all rmd blocks that start with a line '#< ...' and end with a line '#>'</i>
-----------------	---

Description

Find all rmd blocks that start with a line '#< ...' and end with a line '#>'

Usage

```
find.rmd.blocks(txt)
```

Arguments

txt	the rmd code, separated into lines
-----	------------------------------------

Value

A data_frame with the columns start, end, type, arg.str or NULL if no block was found

find.rmd.chunks	<i>Find the start and end lines of all rmd chunks</i>
-----------------	---

Description

Find the start and end lines of all rmd chunks

Usage

```
find.rmd.chunks(rmd, add.code = FALSE)
```

Arguments

rmd	the rmd code as character vector, one element per line
-----	--

find.rmd.nested	<i>find blocks, chunks and dot blocks and add nesting info</i>
-----------------	--

Description

find blocks, chunks and dot blocks and add nesting info

Usage

```
find.rmd.nested(txt, dot.levels = NULL)
```

Arguments

txt	the rmd source as character vector, each line one element
dot.levels	a list that describes the level of dot block types

Value

a data.frame

get.levels.parents	<i>Find the parent index of each row given a vector levels that describes the nestedness</i>
--------------------	--

Description

Find the parent index of each row given a vector levels that describes the nestedness

Usage

```
get.levels.parents(levels, is.parent.type = NULL)
```

Arguments

levels	integer vector of levels must start with lowest level and increase by 1 or decrease to an integer number
is.parent.type	NULL or a optional boolean vector of those rows that are of a type for which children shall be found.

Value

a vector of parent indices or 0 for most outer levels

```
get.levels.parents.by.types
```

Find the parent index of each row given a vector levels that describes the nestedness

Description

Find the parent index of each row given a vector levels that describes the nestedness

Usage

```
get.levels.parents.by.types(
  levels,
  types,
  parent.types = setdiff(unique(types), c(NA))
)
```

Arguments

levels	integer vector of levels must start with lowest level and increase by 1 or decrease to an integer number
types	character vector of types
parent.types	the parent.types that shall be characterized

Value

a matrix of parent type indices with length(levels) rows and length(parent.types) columns. If there is no parent type, we enter a 0.

```
get.start.end.levels
```

Find the levels given ordered start and end positions of possible nested blocks

Description

Find the levels given ordered start and end positions of possible nested blocks

Usage

```
get.start.end.levels(start, end, start.level = 1L)
```

Arguments

start	vector of start positions of the blocks
end	vector of end positions of the blocks
start.level	the initial level

Value

a vector of levels of each blocks, more deeply nested blocks have higher levels

get.types.inds	<i>Gets for each element the index of its type</i>
----------------	--

Description

Gets for each element the index of its type

Usage

```
get.types.inds(types)
```

Arguments

types	a character vector of types
-------	-----------------------------

hideShowButton	<i>A button that toggles whether content in a div is displayed or not</i>
----------------	---

Description

A button that toggles whether content in a div is displayed or not

Usage

```
hideShowButton(  
  id,  
  label,  
  content = NULL,  
  div.id = NULL,  
  shown = FALSE,  
  div.class = NULL,  
  ...  
)
```

html.table	<i>own function to print a compact html table from a data frame with option to select row</i>
------------	---

Description

own function to print a compact html table from a data frame with option to select row

Usage

```
html.table(  
  df,  
  sel.row = NULL,  
  col.names = TRUE,  
  row.names = FALSE,  
  border = TRUE,  
  bg.color = c("#dddddd", "#ffffff"),  
  font.size = "80%",  
  round.digits = 8,  
  signif.digits = 8,  
  col.tooltips = NULL,  
  ...  
)
```

html.template	<i>Variant of htmltools::htmlTemplate</i>
---------------	---

Description

Variant of htmltools::htmlTemplate

Usage

```
html.template(  
  file = NULL,  
  html = NULL,  
  envir = parent.frame(),  
  document_ = "auto"  
)
```

import.yaml	<i>A more convenient yaml importer</i>
-------------	--

Description

A more convenient yaml importer

Usage

```
import.yaml(  
  file = NULL,  
  text = NULL,  
  verbose = FALSE,  
  keep.quotes = FALSE,  
  quote.char = "__QUOTE__",  
  catch.error = TRUE,  
  check.by.row = FALSE,  
  utf8 = TRUE  
)
```

inline.dependencies	<i>Inline dependencies with local file references</i>
---------------------	---

Description

Inline dependencies with local file references

Usage

```
inline.dependencies(deps, mustWork = TRUE)
```

inline.dependency	<i>Inline a dependency with local file references</i>
-------------------	---

Description

Inline a dependency with local file references

Usage

```
inline.dependency(dependency, mustWork = TRUE)
```

knit.chunk

Knits a single rmd knitr chunk Has several options. The output can be given as html, shiny tag or markdown eval_mode = "sculpt" shows shows the final output ommiting all intermediate results and messages deps.action = "add" adds back dependencies to html or shiny tag. Important for rendering shiny widgets deps.action = "relative" also changes absolute dependencies to relative dependencies based on a package. That is important if we want to save the generated shiny tag and use it as part of a shiny application on another computer.

Description

Knits a single rmd knitr chunk Has several options. The output can be given as html, shiny tag or markdown eval_mode = "sculpt" shows shows the final output ommiting all intermediate results and messages deps.action = "add" adds back dependencies to html or shiny tag. Important for rendering shiny widgets deps.action = "relative" also changes absolute dependencies to relative dependencies based on a package. That is important if we want to save the generated shiny tag and use it as part of a shiny application on another computer.

Usage

```
knit.chunk(
  text,
  envir = parent.frame(),
  fragment.only = TRUE,
  quiet = TRUE,
  encoding = getOption("encoding"),
  html.table = TRUE,
  out.type = c("html", "shiny")[1],
  knit.dir = getwd(),
  use.commonmark = TRUE,
  deps.action = c("add", "relative", "ignore")[1],
  args = NULL,
  eval_mode = c("knit", "sculpt", "sculpt_asis", "eval", "html")[1],
  show_code = c("no", "note", "open_note", "note_after", "open_note_after", "before",
    "after")[1],
  code.highlight = use.commonmark,
  add.meta.attr = TRUE,
  remove.empty.html.comments = TRUE,
  ...
)
```

knit.rmd	<i>Knits the rmd txt</i>
----------	--------------------------

Description

Knits the rmd txt

Usage

```
knit.rmd(
  text,
  envir = parent.frame(),
  fragment.only = TRUE,
  quiet = TRUE,
  encoding = getOption("encoding"),
  html.table = TRUE,
  out.type = "html",
  use.commonmark = FALSE
)
```

knit.rmd.in.temp	<i>Knits the rmd txt inside a temporary directory instead of the current wd</i>
------------------	---

Description

Does not create /figure subfolder in current wd

Usage

```
knit.rmd.in.temp(text, envir = parent.frame(), ...)
```

make.chunk.info	<i>Parse a n rmd chunk and store info in a list</i>
-----------------	---

Description

Parse a n rmd chunk and store info in a list

Usage

```
make.chunk.info(txt, id = NULL)
```

```
make.placeholder.info
```

Make a info for a placeholder object

Description

Make a info for a placeholder object

Usage

```
make.placeholder.info(txt, type, form)
```

```
make.whisker.info
```

Parse a whisker and create meta info

Description

Parse a whisker and create meta info

Usage

```
make.whisker.info(txt, add.variables = FALSE)
```

```
makeDependenciesRelativeToPackage
```

Change in a lst of dependencies absolute paths in scr\$file to a relative path based on a package

Description

Change in a lst of dependencies absolute paths in scr\$file to a relative path based on a package

Usage

```
makeDependenciesRelativeToPackage(deps)
```

makeDependencyRelativeToPackage

Change in an dependency an absolute path in scr\$file to a relative path based on a package

Description

Change in an dependency an absolute path in scr\$file to a relative path based on a package

Usage

```
makeDependencyRelativeToPackage(dep)
```

mark_utf8

Recursively encode strings in list as UTF-8

Description

Recursively encode strings in list as UTF-8

Usage

```
mark_utf8(x)
```

md2html

Own markdown to html converter that interfaces commonmark

Description

Own markdown to html converter that interfaces commonmark

Usage

```
md2html(
  text,
  fragment.only = TRUE,
  options = c("use_xhtml", "mathjax", if (include.images) "base64_images" else NULL,
    "highlight_code"),
  include.images = TRUE,
  smart = FALSE,
  use.commonmark = FALSE,
  ...
)
```

Arguments

text	the variable containing the markdown text
fragment.only	only a fragment or should html headers be added
options	options to markdownToHTML. These are the default options without smarty-pants
smart	smart punctuation (relevant for commonmark)
use.commonmark	use commonmark instead of markdownToHTML (no mathjax and included images)...

Value

The created HTML as a string

normalize.id	<i>Normalize an id to letters that are allowed</i>
--------------	--

Description

Normalize an id to letters that are allowed

Usage

```
normalize.id(str, allowed = c(letters, LETTERS, 0:9, "_"), subst = "_")
```

parse.arg.str	<i>Parse an arg.str as list</i>
---------------	---------------------------------

Description

Parse an arg.str as list

Usage

```
parse.arg.str(arg.str)
```

parse.block.args	<i>Parse the name of an rmd block</i>
------------------	---------------------------------------

Description

Parse the name of an rmd block

Usage

```
parse.block.args(
  header,
  arg.str = NULL,
  add.type = TRUE,
  type = "",
  allow.unquoted.title = FALSE
)
```

parse.chunk.args	<i>Parse the name of a knitr chunk and its arguments</i>
------------------	--

Description

Parse the name of a knitr chunk and its arguments

Usage

```
parse.chunk.args(header, arg.str = NULL)
```

parse.chunk.names	<i>Take a vector of chunk header lines and returns the chunk names</i>
-------------------	--

Description

Take a vector of chunk header lines and returns the chunk names

Usage

```
parse.chunk.names(header)
```

Arguments

header	the chunk headers
--------	-------------------

parse.hashdot.yaml	<i>Parse a hashdot yaml string</i>
--------------------	------------------------------------

Description

Parse a hashdot yaml string

Usage

```
parse.hashdot.yaml(txt, hashdot = "#. ", ...)
```

paste.whiskers	<i>Subtitutes whiskers and pastes string parts together</i>
----------------	---

Description

Unlike replace whiskers, returns a string vector if whiskers evaluate as vectors Cannot deal with nested whiskers

Usage

```
paste.whiskers(
  str,
  values = parent.frame(),
  eval = TRUE,
  signif.digits = NULL,
  error.val = "`Error`",
  empty.val = NULL,
  whisker.start = "{{",
  whisker.end = "}}",
  sep = "",
  collapse = NULL,
  return.list = FALSE
)
```

print.yaml	<i>Prints list read from a yaml file</i>
------------	--

Description

Prints list read from a yaml file

Usage

```
## S3 method for class 'yaml'
print(obj)
```

random.string	Create n random strings of length nchar each
---------------	--

Description

Create n random strings of length nchar each

Usage

```
random.string(n = 1, nchar = 14)
```

read.as.utf8	Read a text file and convert to UTF-8
--------------	---------------------------------------

Description

Read a text file and convert to UTF-8

Usage

```
read.as.utf8(file, sep.lines = TRUE, warn = FALSE, ...)
```

read.yaml	Reads a yaml file and returns as a list
-----------	---

Description

Reads a yaml file and returns as a list

Usage

```
read.yaml(  
  file = NULL,  
  verbose = FALSE,  
  keep.quotes = TRUE,  
  quote.char = "__QUOTE__",  
  text = NULL,  
  catch.error = TRUE,  
  check.by.row = FALSE,  
  space.after.colon = FALSE,  
  utf8 = TRUE  
)
```

readUtf8	<i>Read a text file that was saved in UTF-8 format</i>
----------	--

Description

Read a text file that was saved in UTF-8 format

Usage

```
readUtf8(file, sep.lines = TRUE, warn = FALSE, ...)
```

remove.quotes	<i>Remove quotes from strings</i>
---------------	-----------------------------------

Description

Remove quotes from strings

Usage

```
remove.quotes(str, quotes = c("'", "\""))
```

remove.rmd.chunks	<i>Removes the rmd chunks with the given chunk names from rmd code</i>
-------------------	--

Description

Removes the rmd chunks with the given chunk names from rmd code

Usage

```
remove.rmd.chunks(rmd, chunk.names)
```

Arguments

rmd	the rmd code as character vector, one element per line
chunk.names	the list of rmd chunks that shall be removed

render.compiled.rmd	<i>Render a compiled rmd</i>
---------------------	------------------------------

Description

Render a compiled rmd

Usage

```
render.compiled.rmd(  
  cr = NULL,  
  txt = cr$body,  
  envir = parent.frame(),  
  fragment.only = FALSE,  
  chunks = c("knit", "eval")[1],  
  out.type = if (is.null(cr$out.type)) "html" else cr$out.type,  
  use.print = "none",  
  overwrite.values = FALSE,  
  on.error = "error",  
  use.commonmark = TRUE  
)
```

render.deps.as.singletons.tags

*Render a list of dependencies as a list of singleton head tags TO DO:
Some dependencies may use local file names... Those dependencies
must be fully inlined...*

Description

Render a list of dependencies as a list of singleton head tags TO DO: Some dependencies may use local file names... Those dependencies must be fully inlined...

Usage

```
render.deps.as.singletons.tags(deps, inline.local.files = TRUE)
```

<code>render.rmd.in.temp</code>	<i>Render with RMarkdown::render the rmd txt inside a temporary directory instead of the current wd</i>
---------------------------------	---

Description

Does not create /figure subfolder in current wd

Usage

```
render.rmd.in.temp(text, envir = parent.frame(), quiet = TRUE, ...)
```

<code>render.value</code>	<i>Render a value in a format specified by out.type</i>
---------------------------	---

Description

Render a value in a format specified by out.type

Usage

```
render.value(val, out.type = "html", ...)
```

<code>replace.if.blocks</code>	<i>extract #< if blocks from a rmd txt</i>
--------------------------------	---

Description

extract #< if blocks from a rmd txt

Usage

```
replace.if.blocks(
  txt,
  envir = parent.frame(),
  call.list = NULL,
  block.df = NULL,
  warn.if.na = TRUE,
  del.rows.na = FALSE,
  if.df = NULL
)
```

replace.whiskers	<i>replace whiskers using a list of values, with several options</i>
------------------	--

Description

replace whiskers using a list of values, with several options

Usage

```
replace.whiskers(
  str,
  values = parent.frame(),
  eval = TRUE,
  signif.digits = NULL,
  vector.return.first = use.print == "none",
  pos = NULL,
  error.val = "`Error`",
  empty.val = NULL,
  use.whisker.render = FALSE,
  whisker.start = "{{",
  whisker.end = "}}",
  use.print = c("none", "knit", "whisker")[1]
)
```

reverse.whisker.placeholders	<i>Reverses whisker placeholders by their original whiskers</i>
------------------------------	---

Description

May be useful if blocks evaluate whiskers themselves or after rmd to html transformation

Usage

```
reverse.whisker.placeholders(txt, ph = cr$ph, cr = NULL)
```

```
rmd.between.start.end.lines
```

Extract rmd txt between start.line and end.line tag

Description

Extract rmd txt between start.line and end.line tag

Usage

```
rmd.between.start.end.lines(  
  txt,  
  start.line = "<!-- START -->",  
  end.line = "<!-- END -->",  
  return.start.end = FALSE  
)
```

```
rmd.blocks.to.placeholders
```

Replace blocks with placeholders of the form id

Description

TODO: Need to deal with nested blocks: replace from inner to outer

Usage

```
rmd.blocks.to.placeholders(  
  txt,  
  block.df = NULL,  
  whisker.prefix = "{{",  
  whisker.postfix = "}}",  
  del.rows.na = FALSE,  
  ignore.types = NULL,  
  only.types = NULL,  
  add.info = TRUE,  
  ...  
)
```

```
rmd.chunks.to.placeholders
```

Replace chunks with placeholders of the form id

Description

Replace chunks with placeholders of the form id

Usage

```
rmd.chunks.to.placeholders(  
  txt,  
  whisker.prefix = "{{",  
  whisker.postfix = "}}",  
  del.rows.na = FALSE,  
  add.info = TRUE,  
  id = NULL  
)
```

```
rmd.whiskers.to.placeholders
```

Replace whiskers with placeholders of the form id

Description

Replace whiskers with placeholders of the form id

Usage

```
rmd.whiskers.to.placeholders(  
  txt,  
  whisker.prefix = "{{",  
  whisker.postfix = "}}",  
  add.info = TRUE  
)
```

```
sc
```

Like paste0 but returns an empty vector if some string is empty

Description

Like paste0 but returns an empty vector if some string is empty

Usage

```
sc(..., sep = "", collapse = NULL)
```

simple.html.page	<i>A simple html page</i>
------------------	---------------------------

Description

A simple html page

Usage

```
simple.html.page(head, body)
```

split.by.line.start	<i>Splits a text vector into different blocks by a start line token</i>
---------------------	---

Description

Splits a text vector into different blocks by a start line token

Usage

```
## S3 method for class 'by.line.start'
split(
  txt,
  start.with = NULL,
  block.lines = NULL,
  add.start = TRUE,
  merge.lines = TRUE
)
```

Value

a data frame

`table.knit_print.data.frame`*knitr data.frame printer as nice HTML table with several options*

Description

knitr data.frame printer as nice HTML table with several options

Usage

```
table.knit_print.data.frame(  
  x,  
  table.max.rows = 100,  
  round.digits = 8,  
  signif.digits = 8,  
  html.data.frame = TRUE,  
  show.col.tooltips = TRUE,  
  col.tooltips = NULL,  
  output = "html",  
  options = NULL,  
  ...  
)
```

`transformTagDependencies`*Tag a shiny tag or a list of tags, and transform all dependencies with the function transform.fun which takes a list of dependencies as argument*

Description

Tag a shiny tag or a list of tags, and transform all dependencies with the function transform.fun which takes a list of dependencies as argument

Usage

```
transformTagDependencies(tags, transform.fun = identity)
```

`view.html`*View an extended rmd file*

Description

View an extended rmd file

Usage

```
view.html(  
  file = NULL,  
  text = if (!is.null(file)) readLines(file, warn = FALSE) else NULL,  
  ui = NULL,  
  browser = rstudio::viewer,  
  dir = getwd()  
)
```

`view.rmd`*View an extended rmd file*

Description

View an extended rmd file

Usage

```
view.rmd(  
  file = NULL,  
  text = readLines(file, warn = FALSE),  
  envir = list(),  
  start.line = "<!-- START -->",  
  end.line = "<!-- END -->",  
  set.utf8 = TRUE,  
  knit = !chunks.like.whisker,  
  chunks.like.whisker = FALSE,  
  out.type = "shiny",  
  launch.browser = rstudio::viewer,  
  use.commonmark = FALSE  
)
```

whisker_print	<i>Print a whisker object</i>
---------------	-------------------------------

Description

Print a whisker object

Usage

```
whisker_print(x, ...)
```

whisker_print.default	<i>Need to implement different methods</i>
-----------------------	--

Description

Need to implement different methods

Usage

```
## Default S3 method:  
whisker_print(x, ...)
```

writeUtf8	<i>Write text file in UTF-8 format</i>
-----------	--

Description

Write text file in UTF-8 format

Usage

```
writeUtf8(x, file, bom = F)
```

Index

`adapt.hf.blocks`, 3
`adapt.if.blocks`, 4
`add.code.ui`, 4
`args.to.chunk.header`, 4

`cat.rmd.params`, 5
`cat.whisker.params`, 5
`chunk.opt.string.to.list`, 5
`compile.rmd`, 6

`eval.chunk.like.whisker`, 6
`eval.chunks.in.text`, 7
`eval.placeholder`, 7
`eval.placeholders`, 8
`eval.rmd.blocks.in.text`, 8
`eval.whiskers.in.text`, 9

`fast_df`, 9
`find.dot.blocks`, 9
`find.rmd.blocks`, 10
`find.rmd.chunks`, 10
`find.rmd.nested`, 11

`get.levels.parents`, 11
`get.levels.parents.by.types`, 12
`get.start.end.levels`, 12
`get.types.inds`, 13

`hideShowButton`, 13
`html.table`, 14
`html.template`, 14

`import.yaml`, 15
`inline.dependencies`, 15
`inline.dependency`, 15

`knit.chunk`, 16
`knit.rmd`, 17
`knit.rmd.in.temp`, 17

`make.chunk.info`, 17

`make.placeholder.info`, 18
`make.whisker.info`, 18
`makeDependenciesRelativeToPackage`, 18
`makeDependencyRelativeToPackage`, 19
`mark_utf8`, 19
`md2html`, 19

`normalize.id`, 20

`parse.arg.str`, 20
`parse.block.args`, 21
`parse.chunk.args`, 21
`parse.chunk.names`, 21
`parse.hashdot.yaml`, 22
`paste.whiskers`, 22
`print.yaml`, 22

`random.string`, 23
`read.as.utf8`, 23
`read.yaml`, 23
`readUtf8`, 24
`remove.quotes`, 24
`remove.rmd.chunks`, 24
`render.compiled.rmd`, 25
`render.deps.as.singletons.tags`, 25
`render.rmd.in.temp`, 26
`render.value`, 26
`replace.if.blocks`, 26
`replace.whiskers`, 27
`reverse.whisker.placeholders`, 27
`rmd.between.start.end.lines`, 28
`rmd.blocks.to.placeholders`, 28
`rmd.chunks.to.placeholders`, 29
`rmd.whiskers.to.placeholders`, 29

`sc`, 29
`simple.html.page`, 30
`split.by.line.start`, 30

`table.knit_print.data.frame`, 31
`transformTagDependencies`, 31

`view.html`, [32](#)

`view.rmd`, [32](#)

`whisker_print`, [33](#)

`whisker_print.default`, [33](#)

`writeUtf8`, [33](#)