Package: glossary (via r-universe)

November 1, 2024

Title Glossaries for Markdown and Quarto Documents

Date 2023-11-06

Version 1.0.0.9003

Description Add glossaries to markdown and quarto documents by tagging individual words. Definitions can be provided inline or in a separate file.

License CC BY 4.0

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports kableExtra, knitr, markdown, rvest, xml2, yaml

Suggests covr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/debruine/glossary, https://debruine.github.io/glossary/

BugReports https://github.com/debruine/glossary/issues

VignetteBuilder knitr

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

RemoteUrl https://github.com/debruine/glossary

RemoteRef HEAD

RemoteSha 05e4a61063e4e83340c89d85f65f354dd1e6cf94

Contents

`add_to_quarto` *Add glossary helper files to a quarto book*

Description

Adds the necessary helper files to an existing quarto book project and gives instructions to edit the _quarto.yml file accordingly.

Usage

```
add_to_quarto(
  quarto_dir = ".",
  css_path = "glossary.css",
  glossary_path = "glossary.yml",
  script_path = "_setup.R"
)
```

Arguments

`quarto_dir`	The base directory for your quarto project
`css_path`	The relative path to the css file you want to append the glossary styles to (creates a new file if it doesn't exist), using the quarto_dir as a base directory
`glossary_path`	The relative path to the glossary file, using the quarto_dir as a base directory; if this file does not exist, one will be created
`script_path`	The relative path to a pre-chapter script, using the quarto_dir as a base directory; set to FALSE to omit this step

Details

The _quarto.yml file is not edited for you because there is currently no way to do this that doesn't remove your formatting and comments from the file.

Since quarto books render each chapter in a separate environment, it is helpful to have a pre-chapter script that does any common setup. The code below will be added to a new or existing pre-chapter script, and this script sourced in the .Rprofile for this project to allow for a persistent glossary (this project .Rprofile will be run instead of your global .Rrofile). Set `script_path` to `FALSE` to handle this on your own.

```
library(glossary)
glossary_path("glossary.yml")
glossary_persistent(TRUE)
```

Value

No return value, called for side effects.

`glossary` *Display glossary entry*

Description

Display a glossary term with an optional popup of the definition, and add the term to the table created by `glossary_table`. This function is mainly meant to be used via inline R in R Markdown or quarto documents, e.g.:

`` `r glossary("Alpha")` `` does not always have to equal .05.

Usage

```
glossary(
  term,
  display = term,
  def = NULL,
  add_to_table = TRUE,
  show = c("term", "def"),
  popup = glossary_popup(),
  path = glossary_path()
)
```

Arguments

`term`	The glossary term to link to, can contain spaces
`display`	The text to display (if different than the term)
`def`	The short definition to display on hover and in the glossary table; if NULL, this will be looked up from the file in the `path` argument
`add_to_table`	whether to add to the table created by `glossary_table`
`show`	whether to show the term or just the definition
`popup`	whether to show the popup on "click" or "hover" (or "none"); set default with `glossary_popup`
`path`	the path to the glossary file, or NULL for local definitions; set default with `glossary_path`

Details

If the path is set to "psyteachr", the glossary term will link to the PsyTeachR glossary. Set `show = "def"` to just show the definition.

Value

character string

Examples

```
# set glossary path to example file
path <- system.file("glossary.yml", package = "glossary")
glossary_path(path)

glossary("alpha")
glossary("alpha", "$\\alpha$")
glossary("alpha", def = "The first letter of the Greek alphabet")
glossary("alpha", show = "term")
glossary("alpha", show = "def")
```

`glossary_add` *Add a definition*

Description

Write a term and definition to an existing glossary file.

Usage

```
glossary_add(term, def, path = glossary_path(), replace = FALSE)
```

Arguments

`term`	The term to define
`def`	The definition to add
`path`	The path to the glossary file; set default with `glossary_path`
`replace`	Whether to replace an existing definition

Value

NULL; Called for side effects

Examples

```
# make a new glossary file
path <- tempfile("glossary", fileext = ".yml")
glossary_path(path, create = TRUE)

# add an entry for "joins"
glossary_add("joins", "Ways to combine data from two tables")

# now you can access the definition
glossary("joins")
```

glossary_add_to_table *Add terms and definitions to table*

Description

Add terms and definitions to table

Usage

```
glossary_add_to_table(gloss)
```

Arguments

gloss	a named list(term = def)

Value

NULL; called for side effects

glossary_load_all *Load all definitions*

Description

Load all the definitions in a glossary file, usually for subsequent display using glossary_table

Usage

```
glossary_load_all(path = glossary_path())
```

Arguments

path	The path to the glossary file; set default with glossary_path

Value

NULL; Called for side effects

Examples

```
demo_glossary <- system.file("glossary.yml", package = "glossary")
glossary_load_all(demo_glossary)

glossary_table(FALSE) # get table as a data frame
```

glossary_path *Set or get the default glossary path*

Description

Set or get the default glossary path

Usage

```
glossary_path(path, create = FALSE)
```

Arguments

path	the path to the glossary file, or NULL for local definitions
create	create a new glossary file if it doesn't exist

Value

path string if path is NULL

Examples

```
path <- glossary_path() # get current path

# create (if doesn't exist) and set path
newpath <- tempfile("glossary", fileext = ".yml")
glossary_path(newpath, create = TRUE)

# set path (assumes file exists)
glossary_path(path)
```

glossary_persistent *Set or get the path to a persistent table object*

Description

Quarto books render each chapter in a separate environment, so you need to set a persistent table if you want to display all glossary items in a table in a separate chapter. Set the persistent table at the top of each chapter, after loading the glossary package, and it will automatically add any glossary entries to the persistent table.

Usage

```
glossary_persistent(path)
```

Arguments

path	the path to the persistent table, or TRUE for the default table name ("glossary-persistent.yml"), or FALSE for no persistence

Value

path to persistent table object, or FALSE if not set

Examples

```
p <- glossary_persistent() # get current path

# set default persistent table path
glossary_persistent(TRUE)

# set non-default path
glossary_persistent(p)
```

---

glossary_popup *Set or get the default popup type*

Description

Set or get the default popup type

Usage

```
glossary_popup(popup)
```

Arguments

popup	If NULL, get the current default popup type, or set to one of "click", "hover", or "none"

Value

string if popup is NULL

Examples

```
# get current popup style
popstyle <- glossary_popup()

# change popup to click style
glossary_popup("click")

# change back to original popup style
glossary_popup(popstyle)
```

glossary_reset *Reset glossary table*

Description

Resets the list that collects glossary entries for the table.

Usage

```
glossary_reset()
```

Value

NULL; Called for side effects

Examples

```
glossary_reset()
```

glossary_style *Create CSS styles for glossary entries*

Description

Set the color and style of the linked in-text terms and pop-up definitions. Colors should be a valid CSS color string, such as "purple" or "#FF0000".

Usage

```
glossary_style(
  color = "purple",
  text_decoration = "underline",
  def_bg = "#333",
  def_color = "white",
  inline = TRUE
)
```

Arguments

color	Text color of the linked term
text_decoration	Style of the linked term; a valid CSS text-decoration string, such as "none", underline" or "red wavy underline"
def_bg	Background color of the definition pop-up
def_color	Text color of the definition pop-up
inline	If TRUE, includes tags and uses cat() to output the style, if FALSE, omits tags and returns text

Value

A CSS style string

Examples

```
glossary_style("#003366", "underline")
```

`glossary_table` *Display glossary table*

Description

All terms defined with `glossary` (since the last call to `glossary_reset`) are added to a list, which this function displays using kable (or outputs as a data frame).

Usage

```
glossary_table(as_kable = TRUE)
```

Arguments

`as_kable`	if the output should be a kableExtra table or a data frame

Value

kable table or data frame

Examples

```
glossary_reset()
# add a definition to the table
glossary("term", def = "definition", path = NULL)

glossary_table() # show table as kable
glossary_table(FALSE) # or as a data frame
```

Index