

Package: `wyz.code.rdoc` (via `r-universe`)

October 17, 2024

Type Package

Title Wizardry Code Offensive Programming R Documentation

Version 1.1.19

Author Fabien Gelineau <neonira@gmail.com>

Maintainer Fabien Gelineau <neonira@gmail.com>

Description Allows to generate on-demand or by batch, any R documentation file, whatever is kind, data, function, class or package. It populates documentation sections, either automatically or by considering your input. Input code could be standard R code or offensive programming code. Documentation content completeness depends on the type of code you use. With offensive programming code, expect generated documentation to be fully completed, from a format and content point of view. With some standard R code, you will have to activate post processing to fill-in any section that requires complements. Produced manual page validity is automatically tested against R documentation compliance rules. Documentation language proficiency, wording style, and phrasal adjustments remains your job.

Encoding UTF-8

LazyData true

License GPL-3

Depends R (>= 4.0)

Imports methods, data.table (>= 1.11.8), tidyr,
wyz.code.offensiveProgramming (>= 1.1.22), stringr (>= 1.4.0),
R6 (>= 2.4.0), crayon (>= 1.3.4), digest (>= 0.6.23)

Suggests testthat, knitr, rmarkdown

RoxygenNote 7.0.2

VignetteBuilder knitr

URL https://neonira.github.io/offensiveProgrammingBook_v1.2.2/

NeedsCompilation no

Date/Publication 2021-10-06 07:00:02 UTC

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

RemoteUrl <https://github.com/cran/wyz.code.rdoc>

RemoteRef HEAD

RemoteSha 983f4cc651fe63f909a03caaa1089ca7254a49a0

Contents

auditDocumentationFiles	3
beautify	3
completeManualPage	4
computeDocumentationStatistics	5
convertExamples	6
dummy	7
escapeContent	8
family	9
generateEnc	9
generateEnumeration	10
generateMarkup	11
generateOptionLink	12
generateOptionSexpr	13
generateParagraph	14
generateParagraph2NL	15
generateParagraphCR	16
generateReference	17
generateS3MethodSignature	18
generateSection	19
generateTable	20
GenerationContext-class	21
getStandardSectionNames	22
identifyReplacementVariables	23
InputContext-class	24
interpretResults	26
ManualPageBuilder-class	27
opRdocInformation	29
ProcessingContext-class	29
produceAllManualPagesFromObject	31
produceDocumentationFile	32
produceManualPage	33
producePackageLink	34
rdocKeywords	35
sentenseize	36
shortcuts	37
verifyDocumentationFile	38

Index

39

`auditDocumentationFiles`*Audit Documentation Files*

Description

Audit documentation files from a set of folders

Usage

```
auditDocumentationFiles(folder_s_1m)
```

Arguments

`folder_s_1m` An length-1 or more vector of existing folder names

Value

Provides a named list with two entries named `correct` and `incorrect`. All entries are file names.

Incorrect entries are the ones that has length issues as detected by function [computeDocumentationStatistics](#).

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

See Also

Function [verifyDocumentationFile](#) allows to check documentation content using standard R function [tools:checkRd](#).

Examples

```
auditDocumentationFiles('man')
```

`beautify`*Beautify R documentation content*

Description

R documentation beautifying functions

Usage

```
beautify(escapeBraces_b_1 = FALSE)
```

Arguments

escapeBraces_b_1
A single boolean value, allowing to escape braces also

Value

A named list of R vectorized functions. See examples below.
Content provided to function will be processed by function [generateMarkup](#).

Author(s)

Fabien Gelineau <neonira@gmail.com>
Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
b <- beautify()
cat('length', length(b), '\n')
#25

cat(names(b), '\n')
# acronym bold cite code dQuote email emph enc env figure file format kbd link
# option pkg preformatted samp source sQuote strong url var verb codelink

x <- 'some content'
cat(x, ':', b$code(x), '\n')
#some content : \code{some content}
```

completeManualPage *Complete Manual Page*

Description

Complete a manual page

Usage

```
completeManualPage(filename_s_1, processingContext_o,
                    add_b_1 = TRUE, verbosity_b_1 = FALSE)
```

Arguments

filename_s_1 A single string value that is a file name
processingContext_o
a single processing object. See [codeProcessingContext](#). Only postProcessing_1 parameter is considered by this function.
add_b_1 a boolean flag. When TRUE add content to existing content. Otherwise patch content.
verbosity_b_1 a single boolean flag, to set or unset verbosity

Value

This function adds or patches on-demand sections of a manual page file.

You may consider twice prior using this function. It is a convenience that aims to sustain your productivity. You may get very quick results using it, but at the probable cost of non reproducibility in comparison with manual pages produced using function [ManualPageBuilder](#).

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
f <- function() {}
ic <- InputContext(NULL, 'f')
p <- produceManualPage(ic)
# WARNING: File /tmp/Rtmpvk4BG5/f.Rd
# checkRd: (5) /tmp/Rtmpvk4BG5/f.Rd:0-9: Must have a \description
```

```
completeManualPage(p$context$filename,
  ProcessingContext(postProcessing_1 = list(
    details = function(content_s) 'some more details',
    concept = function(content_s) 'yet another concept'
  )), verbosity = TRUE
)
# adding details
# adding concept
# [1] TRUE
```

computeDocumentationStatistics

Compute Documentation Statistics

Description

Compute documentation statistics, providing section length in lines and identifying too long lines.

Usage

```
computeDocumentationStatistics(filename_s_1, maxLineLength_pi_1 = 100L)
```

Arguments

filename_s_1 A single string value
maxLineLength_pi_1
 A single positive integer value

Details

Wherever a `line_length_issue` is not NA, you should correct the faulty line by editing the file. Not doing so will very probably imply a failure during check package procedure execution.

Value

A data.table with three columns.

<code>keywords</code>	the section names embedded in the file
<code>lines</code>	the number of lines for the section
<code>line_length_issue</code>	the line numbers where issues are found or NA.

Note

This function should be use when prior package delivery, to ensure documentation lines meet the R documentation specifications.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
# computeDocumentationStatistics('myfile.Rd')
```

<code>convertExamples</code>	<i>Convert Examples</i>
------------------------------	-------------------------

Description

Generate example section content from R code.

Usage

```
convertExamples(examples_l, captureOutput_b_1n = TRUE,
                mode_s_1n = c(NA, "donttest", "dontrun", "dontshow")[1])
```

Arguments

<code>examples_l</code>	An unconstrained list of functions
<code>captureOutput_b_1n</code>	a length-1 or N boolean vector
<code>mode_s_1n</code>	An length-1 or N vector of string values taken amongst valid values. See examples below.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```

someComputation <- function(numberAsString_s_1) {
  suppressWarnings(sum(as.integer(strsplit(numberAsString_s_1, '')[[1]]), na.rm = TRUE))
}

examples <- list(
  function() {
    someComputation("145")
  },
  function() {
    someComputation("1547215")
  },
  function() {
    someComputation(NA_character_)
  },
  function() {
    invisible(someComputation("0x145ABC"))
  }
)

cat(convertExamples(examples, TRUE, c(NA_character_, 'donttest', 'dontrun', 'dontshow')))
# ----- example 1 -----
# someComputation("145")
# 10

# \donttest{
# ----- example 2 -----
someComputation("1547215")
# 25
# }
# \dontrun{
# ----- example 3 -----
# someComputation(NA)
# 0
# }
# \dontshow{
# ----- example 4 -----
#invisible(someComputation("0x145ABC"))
# }"

```

Description

Data set to be used as example for demo purpose.

Usage

```
dummy
```

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Source

Data set generated by NEONIRA

escapeContent

Escape Specials Characters In Content

Description

Manage characters to be escaped in R documentation text

Usage

```
escapeContent(content_s_1, escapeBraces_b_1 = FALSE)
```

Arguments

`content_s_1` A single string value that is the content to consider

`escapeBraces_b_1`

A single boolean value, allowing also to escape braces

Value

A single string with character '@' and '%' escaped.

When `escapeBraces_b_1` is set, characters '{' and '}' are also escaped.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
escapeContent('www@xxx.com')
# "www@xxx.com"

escapeContent('\code{ x % y }', TRUE)
# "\code{ x \% y \}"
```

family	<i>Data set family</i>
--------	------------------------

Description

Data set to be used as example for demo purpose.

Usage

```
family
```

Author(s)

Fabien Gelineau <neonira@gmail.com>
 Maintainer: Fabien Gelineau <neonira@gmail.com>

Source

Data set generated by NEONIRA

generateEnc	<i>Generate Enc</i>
-------------	---------------------

Description

Generate special markup for encoding text in R documentation

Usage

```
generateEnc(entries_l)
```

Arguments

entries_l An unconstrained named list of listwith text and ascii names, holding respectively international text and ASCII equivalent.

Value

A list of strings

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
generateEnc(list(list(text = 'Français', ascii = 'Français')))
# [[1]]
# [1] "\enc{Français}{Français}"

generateEnc(list(list(text = 'é', ascii = 'e'), list(text = 'è', ascii = 'e')))
# [[1]]
# [1] "\enc{é}{e}"

# [[2]]
# [1] "\enc{è}{e}"
```

generateEnumeration *Generate Enumeration*

Description

Generate enumeration for R documentation

Usage

```
generateEnumeration(entries_s, itemize_b_1 = FALSE)
```

Arguments

`entries_s` An unconstrained vector of strings
`itemize_b_1` A single boolean value

Value

A special character vector to mimic either `enumerate` or `itemize` accordingly to R documentation specification.

Set `itemize_b_1` to TRUE if you want an item list, instead of an enumeration.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
generateEnumeration(LETTERS[1:3])
# [1] "\enumerate{\item A\n\item B\n\item C}"

generateEnumeration(LETTERS[1:3], TRUE)
# [1] "\itemize{\item A\n\item B\n\item C}"
```

generateMarkup *Generate R documentation atomic piece*

Description

Generate R documentation atomic pieces, managing various parameters to fulfil R documentation requirements.

Usage

```
generateMarkup(content_s, keyword_s_1 = NA_character_,
               content2_s = NA_character_,
               inline_b_1 = TRUE, useSpace_b_1 = FALSE,
               escapeBraces_b_1 = FALSE,
               content3_s = NA_character_)
```

Arguments

content_s	a vector of strings, that is the content to consider
keyword_s_1	a R documentation keyword. See rdockKeywords .
content2_s	a vector of strings that is a second content, useful with some keywords that require two members
inline_b_1	a single boolean expressing if the result should be printed on a single line or not?
useSpace_b_1	a single boolean asking for space insertion. When dealing with documentation keywords that requires two members, some may require a space in between to work properly. This parameters allows you to ask for this.
escapeBraces_b_1	when TRUE, braces characters are escaped
content3_s	a vector of strings that is a third content, useful with some keywords that require three members

Details

Very convenient function, to customize your R documentation output.

Might be used programmatically to generate pieces or full documentation.

Tested thoroughly with zero, one, two and three contents to cover all the markups of R documentation.

See examples below.

Value

A single string, containing one or several lines of text. Provided content is processed by function [escapeContent](#).

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#) to know more about R documentation requirements.

See Also

Refer to [escapeContent](#).

Examples

```
# 0. zero content example
print(generateMarkup(keyword = 'R'))
# "\\R"

# 1. one content example
print(generateMarkup('a title', 'title'))
# "\\title{a title}"

# 2. Two contents examples
print(generateMarkup('https://neonira.github.io/offensiveProgrammingBook/',
  'href', 'Offensive Programming Book'))
# "\\href{https://neonira.github.io/offensiveProgrammingBook/}{Offensive Programming Book}"

print(generateMarkup('a', 'item', 'description of a', useSpace_b_1 = TRUE))
# "\\item{a} {description of a}"

print(generateMarkup('a', 'item', 'description of a', useSpace_b_1 = FALSE))
# "\\item{a}{description of a}"

# 3. Three contents example
print(generateMarkup('content_1', 'ifelse', 'content_2', content3_s = 'content_3'))
# "\\ifelse{content_1}{content_2}{content_3}"
```

generateOptionLink *Generate Option Link*

Description

Generate cross reference in R documentation

Usage

```
generateOptionLink(options_s_1, topicName_s_1, escapeBraces_b_1 = FALSE)
```

Arguments

options_s_1 A single string value that is generally a package name
topicName_s_1 A single string value that is the generally a function name
escapeBraces_b_1
 A single boolean value, asking to escape braces

Value

A single string, containing one option link. See references and examples below.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#) section 2.5, to know more about using cross references in R documentation.

Examples

```
# Typical use case
generateOptionLink('myPackage', 'myFunction')
#[1] "\\link[myPackage]{myFunction}"

# Refer to reference R documentation for following case
generateOptionLink('=terms.object', 'terms')
#[1] "\\link[=terms.object]{terms}"
```

generateOptionSexpr *Generate Option Sexpr*

Description

Generation option Sexpr in R documentation

Usage

```
generateOptionSexpr(options_s_1, topicName_s_1, escapeBraces_b_1 = FALSE)
```

Arguments

options_s_1 A single string value that is generally R code used to set expression options
 topicName_s_1 A single string value that is the generally R code
 escapeBraces_b_1
 A single boolean value, asking to escape braces

Value

A single string, containing one option Sexpr. See references and examples below.

Author(s)

Fabien Gelineau <neonira@gmail.com>
 Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#) section 2.12, to know more about using Sexpr handling in R documentation.

Examples

```
generateOptionSexpr('echo=TRUE', 'x <- 1')
#[1] "\\Sexpr[echo=TRUE]{x <- 1}"
```

generateParagraph	<i>Generate Paragraph</i>
-------------------	---------------------------

Description

Generate paragraph, collating provided contents with given string.

Usage

```
generateParagraph(..., collapse_s_1 = "\n", addFinalSeparator_b_1 = FALSE)
```

Arguments

... additional arguments, content to be collated.
 collapse_s_1 The string to be used to collate content
 addFinalSeparator_b_1
 A single boolean value. When TRUE, a final separator will be added to generated content.

Value

A single string, with possibly many new line character embedded.

Author(s)

Fabien Gelineau <neonira@gmail.com>
Maintainer: Fabien Gelineau <neonira@gmail.com>

See Also

Functions [generateParagraph2NL](#) and [generateParagraphCR](#).

Examples

```
generateParagraph(LETTERS[1:3])  
# "A\nB\nC"  
  
generateParagraph(LETTERS[1:3], addFinalSeparator_b_1 = TRUE)  
# "A\nB\nC\n"
```

`generateParagraph2NL` *Function generateParagraph2NL*

Description

Generate paragraph, collating provided contents with double new line.

Usage

```
generateParagraph2NL(..., addFinalSeparator_b_1 = FALSE)
```

Arguments

... additional arguments.
addFinalSeparator_b_1
 A single boolean value, asking to add an extraneous new line at the end of the
 computed string.

Value

A single string, with possibly many new line character embedded.

Author(s)

Fabien Gelineau <neonira@gmail.com>
Maintainer: Fabien Gelineau <neonira@gmail.com>

See Also

Functions [generateParagraph](#) and [generateParagraphCR](#).

Examples

```
generateParagraph2NL(LETTERS[1:3])  
# "A\nB\nC"
```

```
generateParagraph2NL(LETTERS[1:3], addFinalSeparator_b_1 = TRUE)  
# "A\nB\nC\n"
```

generateParagraphCR *Function generateParagraphCR*

Description

Data set to be used as example for demo purpose.

Usage

```
generateParagraphCR(..., addFinalSeparator_b_1 = FALSE)
```

Arguments

```
...                    additional arguments.  
addFinalSeparator_b_1  
                      A single boolean value
```

Value

A single string, with possibly many embedded '\cr' character sequences.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Source

Data set generated by NEONIRA

See Also

Functions [generateParagraph](#) and [generateParagraph2NL](#).

Examples

```
generateParagraphCR(LETTERS[1:3])  
# "A\crB\crC"
```

```
generateParagraphCR(LETTERS[1:3], addFinalSeparator_b_1 = TRUE)  
# "A\crB\crC\cr"
```

generateReference	<i>Generate Reference</i>
-------------------	---------------------------

Description

Generate text to standardize references.

Usage

```
generateReference(data_l)
```

Arguments

data_l An unconstrained list

Value

A single string, containing the generated reference text. Can be appended several times to elaborate a multiple reference text.

See references and examples below.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#) to know more about using web references in R documentation.

Examples

```
generateReference(  
  list(url = 'https://neonira.github.io/offensiveProgrammingBook/',  
        label = 'Offensive Programming Book')  
)  
# "Refer to  
# \href{https://neonira.github.io/offensiveProgrammingBook/}{Offensive Programming Book}."
```

`generateS3MethodSignature`*Generate S3 method signature*

Description

Function to create easily function signature from an S3 class

Usage

```
generateS3MethodSignature(methodName_s_1, className_s_1, argumentNames_s)
```

Arguments

`methodName_s_1` a single string that is the function/method name to consider
`className_s_1` a single string that is the class name to consider
`argumentNames_s`
a vector of strings that are the function/method argument names

Value

A single string.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#) to know more about R documentation requirements.

Examples

```
print(generateReference(  
  list(url = 'https://neonira.github.io/offensiveProgrammingBook/',  
        label = 'Offensive Programming Book')  
))  
# provides following result  
# "Refer to \href{https://neonira.github.io/offensiveProgrammingBook/}{Offensive Programming Book}."
```

generateSection	<i>Generate Section</i>
-----------------	-------------------------

Description

Generate R documentation section

Usage

```
generateSection(sectionName_s_1, content_s)
```

Arguments

sectionName_s_1	A single string value
content_s	An unconstrained vector of string values

Value

A single string, containing the generated reference text. Can be appended several times to elaborate a multiple reference text.

See references and examples below.

Note

This function should not be used directly unless you need to write your own manual page generation program.

To generate a manual page directly, you would better use [produceManualPage](#).

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#) to know more about using web references in R documentation.

Refer to [Parsing Rd files](#) by Duncan Murdoch.

Examples

```
generateSection('concept', 'meta programming')  
# "\\concept{meta programming}"
```

generateTable	<i>Generate Table</i>
---------------	-----------------------

Description

Generate table format in R documentation

Usage

```
generateTable(content_dt, alignement_s_1 = NA_character_,
              numberRows_b_1 = FALSE, showHeader_b_1 = TRUE)
```

Arguments

content_dt A data.table to be use a source data

alignement_s_1 A single string value, expressing the column alignment directive

numberRows_b_1 A single boolean value. Set it when you want data rows to be automatically numbered.

showHeader_b_1 A single boolean value. Set it when you want column names to be displayed.

Value

A single string, containing potentially many embedded formatting strings.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#) section Lists and Tables .

Examples

```
library(data.table)
dt <- data.table::data.table(x = runif(3), y = letters[1:3])

generateTable(dt)
# "\tabular{ll}{\n0.975343016441911 \tab a \cr\n
# 0.647014946676791 \tab b \cr\n0.576294980244711 \tab c \cr\n}"

generateTable(dt, numberRows_b_1 = TRUE)
# "\tabular{rll}{\n1 \tab 0.11690619844012 \tab a \cr\n
# 2 \tab 0.467709563905373 \tab b \cr\n3 \tab 0.957075224025175 \tab c \cr\n}"
```

 GenerationContext-class

Generation Context

Description

Define a generation context to produce a manual page

Usage

```

GenerationContext(targetFolder_s_1 = tempdir(),
                  overwrite_b_1 = FALSE,
                  verbosity_b_1 = FALSE,
                  useMarkers_b_1 = FALSE
                  )

```

Arguments

`targetFolder_s_1` a single string that is the target folder to store produced manual pages. Must exist.

`overwrite_b_1` a single boolean value, allowing to overwrite an existing manual page.

`verbosity_b_1` a single boolean verbosity flag. Turn on for interactive use. Keep off for programmatic usage.

`useMarkers_b_1` A single boolean value, specifying if sections should be generated with special markers whenever possible.

Value

An object instance of class `GenerationContext` based on `environment`.

Information

Environment fields:

⇒ <code>overwrite_b_1</code>	logical
⇒ <code>self</code>	environment
⇒ <code>targetFolder_s_1</code>	character
⇒ <code>useMarkers_b_1</code>	logical
⇒ <code>verbosity_b_1</code>	logical

offensive programming - semantic naming: Class name compliance is TRUE.

offensive programming - function return types: Class owns no function return type instrumentation.

offensive programming - test case definitions: Class owns no test case definitions.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

See Also

Class [InputContext](#) class [ProcessingContext](#) class and class [ManualPageBuilder](#).

Examples

```
GenerationContext()
```

```
GenerationContext(overwrite = TRUE, verbosity = TRUE)
```

```
getStandardSectionNames
```

Get Standard Section Names

Description

Get R documentation standard section names

Usage

```
getStandardSectionNames(sort_b_1 = FALSE)
```

Arguments

`sort_b_1` A single boolean value. Result is sorted when set to TRUE.

Value

A vector of type characters, expressing section names.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#).

Examples

```

getStandardSectionNames()
# [1] "name"          "docType"      "alias"        "title"        "description"
# [6] "usage"         "arguments"    "details"      "value"        "custom_section"
# [11] "references"    "author"       "note"         "seealso"      "examples"
# [16] "keyword"      "concept"      "encoding"     "synopsis"     "Rdversion"
# [21] "RdOpts"       "Sexpr"

```

```

identifyReplacementVariables
      Identify Replacement Variables

```

Description

Identify replacement variables in the generated manual page to ease their substitutions.

Usage

```
identifyReplacementVariables(filename_s)
```

Arguments

`filename_s` An unconstrained vector of string values that are the source filenames

Value

When producing a manual page using [produceManualPage](#) function, under format-driven mode, sections will be generated with a very simple content based on format 'XXX_???' to ease post processing substitutions and hand-crafted replacements.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

See Also

Function [produceManualPage](#) and class [GenerationContext](#).

Examples

```
# identifyReplacementVariables('myfile.Rd')
```

 InputContext-class *Input Context*

Description

Environment class InputContext. Defines and eases input context management.

Usage

```
InputContext(object_o_1,
             methodName_s_1 = NA_character_,
             packageName_s_1 = NA_character_,
             dataFilename_s_1 = NA_character_
            )
```

Arguments

object_o_1 a single object or NULL

methodName_s_1 a single string value that is the method name to consider

packageName_s_1
 a single string that is the target package name to use

dataFilename_s_1
 a single string that is the data file name

Value

An object instance.

Information**Environment fields:**

↪ beautifier	list
↪ class_kind	character
↪ class_name	character
↪ data_name	NULL
↪ dataFilename_s_1	character
↪ file_name	character
↪ hack_description	logical
↪ instrumentationLevel	list
↪ kind	double
↪ kinds	character
↪ methodName_s_1	character
↪ number_replacements	integer
↪ object_o_1	list
↪ packageName_s_1	character
↪ self	environment

⇒ typeFactory_o_1 environment
 ⇒ use_markers logical

Environment methods:

→ buildMethodName()
 → generateConditionalMarker(force_b_1 = FALSE)
 → generateConditionalMarker(Generatorforce_b_1 = FALSE)
 → getFilename()
 → getKind()
 → getName()
 → markerGenerator()
 → produceAlias()
 → produceArguments()
 → produceAuthor()
 → produceConcept()
 → produceCustom_section()
 → produceDescription()
 → produceDetails()
 → produceDocType()
 → produceEncoding()
 → produceExamples()
 → produceFormat()
 → produceKeyword()
 → produceName()
 → produceNote()
 → produceRdOpts()
 → produceRdversion()
 → produceReferences()
 → produceSeealso()
 → produceSexpr()
 → produceSource()
 → produceSynopsis()
 → produceTitle()
 → produceUsage()
 → produceValue()
 → retrieveStrategy()
 → setUseMarkers(value_b_1)

offensive programming - semantic naming: Class name compliance is TRUE.

offensive programming - function return types: Class owns no function return type instrumentation.

offensive programming - test case definitions: Class owns no test case definitions.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

See Also

Class [GenerationContext](#) class [ProcessingContext](#) class and class [ManualPageBuilder](#).

Examples

```
ic1 <- InputContext(NULL)

ic2 <- InputContext(NULL, 'append', package = 'my.package.name')
```

interpretResults	<i>Interpret Results</i>
------------------	--------------------------

Description

Interpret results of function [ManualPageBuilder](#)

Usage

```
interpretResults(manualPageGenerationResults_l)
```

Arguments

manualPageGenerationResults_l
A list resulting from [ManualPageBuilder](#) function.

Details

This function checks for presence of content that should be present in a well formatted and documented function manual page. It provides hints. You could follow those hints to produce great documentation.

Value

Provides output that allows to know which sections has been generated and which sections are missing or probably missing.

Good practice

When producing a manual page using [ManualPageBuilder](#), keeping the result in a R variable allows you to interpret this result at any time in the future. This is helpful when working incrementally to produce a fully automated generation scheme for a given manual page. See examples below.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
ic <- InputContext(NULL, 'append', packageName_s_1 = 'wyz.code.rdoc')

res <- produceManualPage(ic)
# WARNING: File /tmp/RtmpYIampA/append.Rd
# checkRd: (5) /tmp/RtmpYIampA/append.Rd:0-19: Must have a \description

interpretResults(res)
# filename is /tmp/RtmpYIampA/append.Rd [OVERWRITTEN]
# generated 8 sections: name, alias, title, usage, arguments, author, keyword, encoding
# missing 3 sections: description, value, examples
# probably missing 1 section: details
```

ManualPageBuilder-class

Manual Page Builder

Description

Environment class ManualPageBuilder. Creates manual pages according to the given context.

Usage

```
ManualPageBuilder(inputContext_o_1,
                  processingContext_o_1 = ProcessingContext(),
                  generationContext_o_1 = GenerationContext())
```

Arguments

`inputContext_o_1`
The input context object to consider for generation. See [InputContext](#).

`processingContext_o_1`
The processing context object to consider for generation. See [ProcessingContext](#).

`generationContext_o_1`
The generation context object to consider for generation. See [GenerationContext](#).

Value

An object instance of class ManualPageBuilder.

Information**Environment fields:**

↳ colorizer	list
↳ generationContext_o_1	environment
↳ inputContext_o_1	environment

```

↳ processingContext_o_1  environment
↳ self                  environment
↳ strategy              list

```

Environment methods:

```

→ assembleManualPage(pieces_l)
→ buildManualPage()
→ documentContent()
→ getStrategy
→ interpretResults(result_l)

```

offensive programming - semantic naming: Class name compliance is TRUE.

offensive programming - function return types: Class owns no function return type instrumentation.

offensive programming - test case definitions: Class owns no test case definitions.

Note

As an end-user, you may prefer to use function [produceManualPage](#) as its usage is much more straightforward.

As a programmer, this class eases programming of your own manual page builder. See examples below.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

See Also

Class [InputContext](#) class [ProcessingContext](#) and class [GenerationContext](#).

Examples

```

ic <- InputContext(NULL, 'append', package = 'my.package.name')
m <- ManualPageBuilder(ic)
r <- m$buildManualPage()
interpretResults(r)

```

opRdocInformation *Offensive Programming R Documentation Information*

Description

List package functions and provide informations about their intended usage.

Usage

```
opRdocInformation()
```

Value

See [opInformation](#) value description.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
opRdocInformation()
```

ProcessingContext-class
Processing Context

Description

Environment class ProcessingContext. Defines and eases processing context management.

Usage

```
ProcessingContext(extraneous_l = list(), postProcessing_l = list())
```

Arguments

`extraneous_l` An unconstrained named list. Each entry will be turned into a R documentation section.

`postProcessing_l`
 An unconstrained named list. Each entry will trigger a post processing for the related R documentation section.

Details

If a post processing function returns NULL, related section will be removed from generated content. See examples below.

Post processing aims to put in action simple transformations, as changing letter cases, or applying simple beautifying technics. See [beautify](#).

Value

An object instance of class ProcessingContext.

Information**Environment fields:**

↳ extraneous_l	list
↳ postProcessing_l	list
↳ self	environment

Environment methods:

→ verifyExtraneous(extraneous_l)
 → verifyPostProcessing(postProcessing_l)

offensive programming - semantic naming: Class name compliance is TRUE.

offensive programming - function return types: Class owns no function return type instrumentation.

offensive programming - test case definitions: Class owns no test case definitions.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

See Also

Class [InputContext](#) class [GenerationContext](#) class and class [ManualPageBuilder](#).

Examples

```
pc <- ProcessingContext(
  extraneous_l = list(
    'my section' = "a special dedicace to neonira",
    keyword = 'documentation',
    concept = 'documentation generation'
  ),
  postProcessing_l = list(
    'my section' = function(content_s) {
```

```
      gsub('neonira', 'NEONIRA', content_s, fixed = TRUE)
    },
    author = function(content_s) { NULL } # destroy section
  )
)
```

produceAllManualPagesFromObject

Produce All Manual Pages From Object

Description

Produce object and methods manual pages from an object.

Usage

```
produceAllManualPagesFromObject(object_o_1,
                               processingContext_o_1 = ProcessingContext(),
                               generationContext_o_1 = GenerationContext(),
                               packageName_s_1 = NA_character_)
```

Arguments

`object_o_1` The single object you want to generate manual pages from.

`processingContext_o_1`
The processing context object to consider for generation. See [codeProcessingContext](#).

`generationContext_o_1`
The generation context object to consider for generation. See [GenerationContext](#).

`packageName_s_1`
a single string that is the target package name to consider for generation. Allows to generate automatically the author section content.

Note

This is an **EXPERIMENTAL** function. Prefer usage of function [produceManualPage](#) instead.

It generates reliable individual manual pages that taken all together are not fully compatible with R way to express documentation.

In particular, expect duplicated aliases to appear, and some name weirdness also.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

 produceDocumentationFile

Produce Documentation File

Description

Use this function to save documentation text into a documentation file.

Usage

```
produceDocumentationFile(filename_s_1, content_s, generationContext_o_1)
```

Arguments

filename_s_1 the target file name to use
 content_s An unconstrained vector of string values
 generationContext_o_1
 The generation context object to consider for generation. See [GenerationContext](#).

Classification

STRATUM \mapsto LAYER_1
 PHASING \mapsto BUILD
 INTENT \mapsto CONTENT_GENERATION

Note

From a end-user perspective, this function should only be used indirectly through a call to [produceManualPage](#) function.

Direct call is meaningful when crafting your own manual page builder code/program.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
fn <- tempfile()
p <- produceDocumentationFile(basename(fn), c(
  generateSection('name', 'alpha'),
  generateSection('alias', 'alpha'),
  generateSection('keyword', 'documentation generation')
),
  GenerationContext(dirname(fn)))

p
```



```

# $filename
# [1] "/tmp/RtmpSWZq4H/filee3c2700207f.Rd"
#
# $overwritten
# [1] TRUE

readLines(p$filename)
# [1] "\name{alpha}" "\alias{alpha}" "\keyword{documentation generation}"

```

produceManualPage	<i>Produce Manual Page</i>
-------------------	----------------------------

Description

Use this function to produce a manual page.

Usage

```

produceManualPage(inputContext_o_1,
                 processingContext_o_1 = ProcessingContext(),
                 generationContext_o_1 = GenerationContext())

```

Arguments

`inputContext_o_1`
The input context object to consider for generation. See [InputContext](#).

`processingContext_o_1`
The processing context object to consider for generation. See [ProcessingContext](#).

`generationContext_o_1`
The generation context object to consider for generation. See [GenerationContext](#).

Value

A list holding generation process information.

Use function [interpretResults](#) to get knowledge of generated parts and remaining issues.

Classification

STRATUM \mapsto LAYER_3
 PHASING \mapsto RUN
 INTENT \mapsto QUALITY_CONTROL

Author(s)

Fabien Gelineau <neonira@gmail.com>
 Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
ic <- InputContext(NULL, 'append', packageName_s_1 = 'wyz.code.rdoc')

res <- produceManualPage(ic)
# WARNING: File /tmp/RtmpYIampA/append.Rd
# checkRd: (5) /tmp/RtmpYIampA/append.Rd:0-19: Must have a \description

interpretResults(res)
# filename is /tmp/RtmpYIampA/append.Rd [OVERWRITTEN]
# generated 8 sections: name, alias, title, usage, arguments, author, keyword, encoding
# missing 3 sections: description, value, examples
# probably missing 1 section: details
```

producePackageLink *Produce Package Link*

Description

Generation package cross reference in R documentation

Usage

```
producePackageLink(packageName_s_1, topicName_s_1)
```

Arguments

`packageName_s_1` A single string value that represents generally a package name considered as source for the topic

`topicName_s_1` A single string value that is generally a documentation topic to link to

Value

A single string, containing the generated package link. See references and examples below.

Author(s)

Fabien Gelineau <neonira@gmail.com>
Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#) section 2.5, to know more about using cross references in R documentation.

See Also

See option link creation using function [generateOptionLink](#).

Examples

```
producePackageLink('tools', 'checkRd')  
#[1] "\\link{tools:checkRd}{tools:checkRd}"
```

rdocKeywords

R Documentation Keywords

Description

Provides all R documentation markup tags a.k.a keywords

Usage

```
rdocKeywords(asList_b_1 = FALSE)
```

Arguments

asList_b_1 A single boolean.

Value

A vector of type characters, containing all R documentation keywords, when parameter asList_b_1 is FALSE. Otherwise a list organizing this same content.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

References

Refer to [Writing R extensions](#).

Examples

```
rdocKeywords()
```

sentensize	<i>Create sentence</i>
------------	------------------------

Description

Create a sentence from given content

Usage

```
sentensize(x_s, ..., punctuationCharacter_s_1 = ".")
```

Arguments

x_s	An unconstrained vector of string values
...	additional arguments (should be convertible to character type).
punctuationCharacter_s_1	the punctuation character to use to end the sentence.

Details

Collate all provided arguments, then normalize spaces.
Finally, ensure capitalization of first letter and final colon.

Value

A single string.

Note

There is no way to ask for a different final punctuation mark. If you need to do so, either create your own helper function or simply [sub](#) provided result.

Author(s)

Fabien Gelineau <neonira@gmail.com>
Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
sentensize('a quick brown FOX jumps\tover', 'the lazy dog')
# "A quick brown FOX jumps over the lazy dog."

sentensize('a simple', ' question\t', punctuationCharacter_s_1 = '?')
# "A simple question?"
```

shortcuts	<i>Function shortcuts</i>
-----------	---------------------------

Description

Use this function to exploit prepared and customized shortcuts.

Usage

```
shortcuts(arguments_s = character(), doubleEscape_b_1 = TRUE)
```

Arguments

`arguments_s` A vector of function arguments you would like to get shortcuts for
`doubleEscape_b_1` A single boolean asking for double escape. On by default.

Value

A list with following names

<code>doc</code>	very common single R documentation keywords
<code>constants</code>	very common R constants used in R documentation
<code>types</code>	very common R types used in R documentation
<code>args</code>	function arguments ready to use in R documentation

The names are all turned to lowercase.

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
shortcuts(formalArgs(GenerationContext))
```

`verifyDocumentationFile`*Verify Documentation File*

Description

Verify documentation file compliance to R documentation scheme.

Usage

```
verifyDocumentationFile(filename_s_1)
```

Arguments

`filename_s_1` A single string value that is the filename holding R documentation to check

Value

Echoes on stdout status of documentation verification, as done by [tools:checkRd](#).

Author(s)

Fabien Gelineau <neonira@gmail.com>

Maintainer: Fabien Gelineau <neonira@gmail.com>

Examples

```
# verifyDocumentationFile("myfile.Rd")
```

Index

- * **datasets**
 - dummy, 7
 - family, 9
- * **documentation generation**
 - auditDocumentationFiles, 3
 - beautify, 3
 - completeManualPage, 4
 - computeDocumentationStatistics, 5
 - convertExamples, 6
 - dummy, 7
 - escapeContent, 8
 - family, 9
 - generateEnc, 9
 - generateEnumeration, 10
 - generateOptionSexpr, 13
 - generateParagraph, 14
 - generateParagraph2NL, 15
 - generateParagraphCR, 16
 - generateReference, 17
 - generateSection, 19
 - generateTable, 20
 - GenerationContext-class, 21
 - getStandardSectionNames, 22
 - identifyReplacementVariables, 23
 - InputContext-class, 24
 - interpretResults, 26
 - ManualPageBuilder-class, 27
 - opRdocInformation, 29
 - ProcessingContext-class, 29
 - produceAllManualPagesFromObject, 31
 - produceDocumentationFile, 32
 - produceManualPage, 33
 - producePackageLink, 34
 - rdocKeywords, 35
 - sentensize, 36
 - verifyDocumentationFile, 38
- * **documentation**
 - auditDocumentationFiles, 3
 - beautify, 3
 - completeManualPage, 4
 - computeDocumentationStatistics, 5
 - convertExamples, 6
 - escapeContent, 8
 - generateEnc, 9
 - generateEnumeration, 10
 - generateMarkup, 11
 - generateOptionSexpr, 13
 - generateReference, 17
 - generateS3MethodSignature, 18
 - generateSection, 19
 - generateTable, 20
 - GenerationContext-class, 21
 - getStandardSectionNames, 22
 - identifyReplacementVariables, 23
 - InputContext-class, 24
 - interpretResults, 26
 - ManualPageBuilder-class, 27
 - opRdocInformation, 29
 - ProcessingContext-class, 29
 - produceAllManualPagesFromObject, 31
 - produceDocumentationFile, 32
 - produceManualPage, 33
 - producePackageLink, 34
 - rdocKeywords, 35
 - sentensize, 36
 - verifyDocumentationFile, 38
- * **function**
 - generateParagraph, 14
 - generateParagraph2NL, 15
 - generateParagraphCR, 16
 - shortcuts, 37
- * **keywords**
 - generateMarkup, 11
 - generateS3MethodSignature, 18
- auditDocumentationFiles, 3

- beautify, [3](#), [30](#)
- completeManualPage, [4](#)
- computeDocumentationStatistics, [3](#), [5](#)
- convertExamples, [6](#)

- dummy, [7](#)

- environment, [21](#)
- escapeContent, [8](#), [12](#)

- family, [9](#)

- generateEnc, [9](#)
- generateEnumeration, [10](#)
- generateMarkup, [4](#), [11](#)
- generateOptionLink, [12](#), [34](#)
- generateOptionSexpr, [13](#)
- generateParagraph, [14](#), [15](#), [16](#)
- generateParagraph2NL, [15](#), [15](#), [16](#)
- generateParagraphCR, [15](#), [16](#)
- generateReference, [17](#)
- generateS3MethodSignature, [18](#)
- generateSection, [19](#)
- generateTable, [20](#)
- GenerationContext, [21](#), [23](#), [26–28](#), [30–33](#)
- GenerationContext
 (GenerationContext-class), [21](#)
- GenerationContext-class, [21](#)
- getStandardSectionNames, [22](#)

- identifyReplacementVariables, [23](#)
- InputContext, [22](#), [27](#), [28](#), [30](#), [33](#)
- InputContext (InputContext-class), [24](#)
- InputContext-class, [24](#)
- interpretResults, [26](#), [33](#)

- ManualPageBuilder, [5](#), [22](#), [26](#), [30](#)
- ManualPageBuilder
 (ManualPageBuilder-class), [27](#)
- ManualPageBuilder-class, [27](#)

- opInformation, [29](#)
- opRdocInformation, [29](#)

- ProcessingContext, [4](#), [22](#), [26–28](#), [31](#), [33](#)
- ProcessingContext
 (ProcessingContext-class), [29](#)
- ProcessingContext-class, [29](#)
- produceAllManualPagesFromObject, [31](#)

- produceDocumentationFile, [32](#)
- produceManualPage, [19](#), [23](#), [28](#), [31](#), [32](#), [33](#)
- producePackageLink, [34](#)

- rdocKeywords, [11](#), [35](#)

- sentensize, [36](#)
- shortcuts, [37](#)
- sub, [36](#)

- tools:checkRd, [3](#), [38](#)

- verifyDocumentationFile, [3](#), [38](#)