Package 'erify'

June 30, 2024

Type Package

Title Check Arguments and Generate Readable Error Messages

Version 0.6.0

Author Renfei Mao

Maintainer Renfei Mao <renfeimao@gmail.com>

Description Provides several validator functions for checking if arguments passed by users have valid types, lengths, etc. and for generating informative and well-formatted error messages in a consistent style. Also provides tools for users to create their own validator functions. The error message style used is adopted from <https://style.tidyverse.org/error-messages.html>.

License MIT + file LICENSE

URL https://github.com/flujoo/erify, https://flujoo.github.io/erify/

Encoding UTF-8

RoxygenNote 7.3.1

Imports glue

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2024-06-30 09:30:05 UTC

Contents

`back_quote` *Back Quote Object*

Description

Convert an R object to character and add back quotations.

Usage

```
back_quote(x, recursive = TRUE, as_double = TRUE)
```

Arguments

`x`	An R object.
`recursive`	Optional. `TRUE` or `FALSE` which indicates if to back quote each item of `x` or to back quote `x` as a whole, when `x` is a vector. The default value is `TRUE`.
`as_double`	Optional. `TRUE` or `FALSE` which indicates if to differentiate between type double and integer. The default value is `TRUE`, which means integers are handled as doubles.

Value

A character vector.

Examples

```
back_quote(1:3)

back_quote(1:3, recursive = FALSE)

back_quote(1:3, as_double = FALSE)

back_quote(NULL)

back_quote(list(c, 1:3, "a"))
```

check_binary_classes *Check Binary Operator's Arguments' Classes*

Description

Check if the arguments of a binary operator have valid classes, and if not, generate an error message.

Usage

```
check_binary_classes(
  x,
  y,
  valid_x,
  valid_y = NULL,
  operator = NULL,
  commutative = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  ...
)
```

Arguments

x, y	The argument to check, which can be any object.
valid_x, valid_y	A character vector which contains the valid classes. valid_y is assigned valid_x, if not specified.
operator	Optional. A single character which represents the binary operator.
commutative	TRUE or FALSE which indicates if arguments x and y can be swapped around. The default value is TRUE.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. glue::glue() syntax can be used, see "Examples" section. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in `check_type()` for how to customize error message and how to add and retrieve additional arguments.

`vignette("erify")` for a gentle introduction to this package.

Examples

```
## Not run:
x <- 1
class(x) <- c("a", "b")

y <- 2
class(y) <- c("c", "d")

check_binary_classes(x, y, c("d", "e"))
check_binary_classes(x, y, c("d", "e"), operator = "+")

check_binary_classes(x, y, c("d", "e"), c("a", "f"))
check_binary_classes(x, y, c("d", "e"), c("a", "f"), commutative = FALSE)

# customize error message with `glue::glue()` syntax
check_binary_classes(
  x, y, c("d", "e"),
  specific = "Left: {feature_x[1]}, {feature_x[2]}.",
  supplement = "Right: {feature_y[1]}, {feature_y[2]}."
)

## End(Not run)
```

---

check_bool *Check If Argument Is Single Logical*

Description

Check if an argument is `TRUE` or `FALSE`, and if not, generate an error message.

Usage

```
check_bool(
  x,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  ...
)
```

Arguments

x	The argument to check, which can be any object.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
x <- TRUE
check_bool(x)

## Not run:
# `x` must have type logical
x <- 1
check_bool(x)

# `x` must have length 1
x <- c(TRUE, FALSE)
check_bool(x)

# `x` must not be `NA`
x <- NA
check_bool(x)

## End(Not run)
```

check_class *Check Argument's Class*

Description

Check if an argument has valid class, and if not, generate an error message.

Usage

```
check_class(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  ...
)
```

Arguments

x	The argument to check, which can be any object.
valid	A character vector which contains valid classes.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. glue::glue() syntax can be used, see "Examples" section. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
x <- 1
class(x) <- c("a", "b")

check_class(x, c("a", "c"))

## Not run:
check_class(x, c("c", "d"))

# customize error message with `glue::glue()` syntax
specific <- "Unbelievable! The first class of `{name}` is {feature[1]}."
check_class(x, c("c", "d"), specific = specific)

## End(Not run)
```

`check_classes` *Check Each Item's Class*

Description

Check if each item of an argument has valid class, and if not, generate an error message.

Usage

```
check_classes(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  ...
)
```

Arguments

`x`	The argument to check, which must be a list.
`valid`	A character vector which contains valid classes.
`name`	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument `x` is captured automatically.
`general`	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
`specific`	Optional. A single character which gives a detailed description of the error. `glue::glue()` syntax can be used, see "Examples" section. By default, this is generated automatically.

supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
# argument to check
arg <- lapply(1:10, function(x) {class(x) <- c("a", "b"); x})

check_classes(arg, "a")

## Not run:
check_classes(arg, c("x", "y"))

## End(Not run)
```

---

check_content *Check Argument's Content*

Description

Check if an argument is from some given choices or satisfies some requirement, and if not, generate an error message.

Usage

```
check_content(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  as_double = TRUE,
  ...
)
```

Arguments

x	The argument to check, which can be any object.
valid	can be 1. a function, which takes x as argument and returns TRUE or FALSE, 2. an expression, which contains x and evaluates to TRUE or FALSE, 3. a string of R code, which evaluates to TRUE or FALSE, or 4. a non-empty atomic vector, which contains the valid choices.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
as_double	Optional. TRUE or FALSE which indicates if to differentiate between type double and integer. The default value is TRUE, which means integers are handled as doubles.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
valid <- c(1, 2, 3)

x <- 2L
check_content(x, valid)

## Not run:
# `x` must have the same type with `valid`
x <- "a"
check_content(x, valid)

# `x` must have length 1
x <- c(1, 2)
```

```
check_content(x, valid)

# differentiate between type double and integer
x <- 2L
check_content(x, valid, as_double = FALSE)

# `valid` can be a function
check_content(x, is.na, general = "`x` must be `NA`.")

# `valid` can be a string of R code
check_content(x, "is.na(x)", general = "`x` must be `NA`.")

## End(Not run)
```

---

check_contents *Check Each Item's Content*

Description

Check if each item of an argument is from some given choices or satisfies some requirement, and if not, generate an error message.

Usage

```
check_contents(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  as_double = TRUE,
  ...
)
```

Arguments

x	The argument to check, which can be any object.
valid	can be 1. a function, which takes x as argument and returns TRUE or FALSE, 2. an expression, which contains x and evaluates to TRUE or FALSE, 3. a string of R code, which evaluates to TRUE or FALSE, or 4. a non-empty atomic vector, which contains the valid choices.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.

general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
as_double	Optional. TRUE or FALSE which indicates if to differentiate between type double and integer. The default value is TRUE, which means integers are handled as doubles.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
## Not run:
x <- c(1, 2, 3)

check_contents(x, c(4, 5))

general = "Each item of `x` must be `NA`."

# `valid` can be a function or R code
check_contents(x, is.na, general = general)
check_contents(x, "is.na(x_i)", general = general)

## End(Not run)
```

---

check_interval *Check If Argument Is in Interval*

Description

Check if an argument is a number in an interval, and if not, generate an error message.

Usage

```
check_interval(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  ...
)
```

Arguments

x	The argument to check, which can be any object.
valid	A numeric vector of length 2, which represents the valid closed interval. If valid is an integer vector, x must also be an integer. valid can contain NA. For example, c(1, NA) means x must be no less than 1.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. glue::glue() syntax can be used, see "Examples" section. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

vignette("erify") for a gentle introduction to this package.

Examples

```
x <- 3.3

check_interval(x, c(1, 5))

## Not run:
check_interval(x, c(1L, 5L))
check_interval(x, c(4, NA))
check_interval(x, c(NA, 2))
```

```
## End(Not run)
```

`check_intervals` *Check If Each Item Is in Interval*

Description

Check if each item of an argument is a number in an interval, and if not, generate an error message.

Usage

```
check_intervals(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  ...
)
```

Arguments

`x`	The argument to check.
`valid`	A numeric vector of length 2, which represents the valid closed interval. If `valid` is an integer vector, `x` must also be an integer. `valid` can contain `NA`. For example, `c(1, NA)` means `x` must be no less than 1.
`name`	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument `x` is captured automatically.
`general`	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
`specific`	Optional. A single character which gives a detailed description of the error. `glue::glue()` syntax can be used, see "Examples" section. By default, this is generated automatically.
`supplement`	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see `throw()`. By default, this is left empty.
`...`	Optional. Additional arguments which can be retrieved with `tryCatch()`.

Value

returns an invisible `NULL` if the argument is valid, or generates an error message.

See Also

"Examples" section in `check_type()` for how to customize error message and how to add and retrieve additional arguments.

`vignette("erify")` for a gentle introduction to this package.

Examples

```
x <- c(1, 3, 5)
check_intervals(x, c(0, 6))

## Not run:
check_intervals(x, c(2, 4))

## End(Not run)
```

`check_length` *Check Argument's Length*

Description

Check if an argument has valid length, and if not, generate an error message.

Usage

```
check_length(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  interval = NULL,
  ...
)
```

Arguments

`x`	The argument to check, which can be any object.
`valid`	A numeric vector which contains non-negative integers and `NA`, used with argument `interval` to indicate the valid lengths.
`name`	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument `x` is captured automatically.
`general`	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.

specific	Optional. A single character which gives a detailed description of the error. glue::glue() syntax can be used, see "Examples" section. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
interval	Optional. TRUE or FALSE which indicates if argument valid is interpreted as an interval or as single lengths. For example, c(1, 10) is interpreted as "larger than 1 and smaller than 10" if interval is TRUE, but as "1 or 10" if FALSE. NA can be used in valid when treated as interval. For example, c(0, NA) means "larger than 0". By default, interval is inferred from valid. For example, if valid has length unequal to 2, it's treated as single lengths.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
## Not run:
x <- c(1, 2)

# `valid` as interval
check_length(x, c(1, 3), interval = TRUE)
check_length(x, c(NA, 2))

# `valid` as single lengths
check_length(x, c(1, 3), interval = FALSE)

# customize error message with `glue::glue()` syntax
specific <- "Oh my god! `{name}`'s length is {feature}."
check_length(x, 3, specific = specific)

## End(Not run)
```

---

check_lengths *Check Each Item's Length*

Description

Check if each item of an argument has valid length, and if not, generate an error message.

Usage

```
check_lengths(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  interval = NULL,
  ...
)
```

Arguments

x	The argument to check, which must be a list.
valid	A numeric vector which contains non-negative integers and NA, used with argument interval to indicate the valid lengths.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. glue::glue() syntax can be used, see "Examples" section. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
interval	Optional. TRUE or FALSE which indicates if argument valid is interpreted as an interval or as single lengths. For example, c(1, 10) is interpreted as "larger than 1 and smaller than 10" if interval is TRUE, but as "1 or 10" if FALSE. NA can be used in valid when treated as interval. For example, c(0, NA) means "larger than 0". By default, interval is inferred from valid. For example, if valid has length unequal to 2, it's treated as single lengths.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
## Not run:
x <- list(1, c(1, 2), c(1, 2, 3))

check_lengths(x, c(1, NA))

specific = "Item {i} has length {feature}."
check_lengths(x, c(1, NA), specific = specific)

## End(Not run)
```

---

check_n *Check If Argument Is Single Natural Number*

Description

Check if an argument is a single natural number, and if not, generate an error message.

Can be used to check indices, for example.

Usage

```
check_n(
  x,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  zero = FALSE,
  ...
)

is_n(x, zero = FALSE)
```

Arguments

x	The argument to check, which can be any object.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.

zero	Optional. TRUE or FALSE which indicates if zero is acceptable. The default value is FALSE.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

check_n() returns an invisible NULL if the argument is valid, or it generates an error message.

is_n() returns TRUE or FALSE.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
x <- 1
check_n(x)

x <- 1L
check_n(x)

sapply(c(1, 2.1, 0, Inf, NA, -9), is_n)

## Not run:
# `x` must be a numeric
x <- "1"
check_n(x)

# `x` must have length 1
x <- 1:2
check_n(x)

# `x` must not be `NA`
x <- NA_integer_
check_n(x)

# `x` must be larger than 0
x <- -1
check_n(x)

# `x` must be an integer in a mathematical sense
x <- 1.1
check_n(x)

# make `0` acceptable
x <- 0
check_n(x)
check_n(x, zero = TRUE)
```

```
## End(Not run)
```

---

check_positive *Check If Argument Is Single Positive Number*

Description

Check if an argument is a single positive number, and if not, generate an error message.

Usage

```
check_positive(
  x,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  zero = FALSE,
  ...
)
```

Arguments

x	The argument to check, which can be any object.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
zero	Optional. TRUE or FALSE which indicates if zero is acceptable. The default value is FALSE.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
x <- 1.1
check_positive(x)

x <- 1L
check_positive(x)

## Not run:
# `x` must be a numeric
x <- "1"
check_positive(x)

# `x` must have length 1
x <- 1:2
check_positive(x)

# `x` must not be `NA`
x <- NA_integer_
check_positive(x)

# `x` must be larger than 0
x <- -1
check_positive(x)

# make `0` acceptable
x <- 0
check_positive(x)
check_positive(x, zero = TRUE)

## End(Not run)
```

---

check_string *Check If Argument Is Single Character*

Description

Check if an argument is a single character. and if not, generate an error message.

Can be used to check argument names, for example.

Usage

```
check_string(
  x,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  ...
)
```

```
is_string(x)
```

Arguments

x	The argument to check, which can be any object.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
x <- "a"
check_string(x)

## Not run:
# `x` must have type character
x <- c
check_string(x)

# `x` must have length 1
x <- c("a", "b")
check_string(x)

# `NA_character_` is not acceptable
x <- NA_character_
check_string(x)

## End(Not run)
```

check_type *Check Argument's Type*

Description

Check if an argument has valid type, and if not, generate an error message.

Usage

```
check_type(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  ...
)
```

Arguments

x	The argument to check, which can be any object.
valid	A character vector which contains the valid types.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. glue::glue() syntax can be used, see "Examples" section. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

vignette("erify") for a gentle introduction to this package.

Examples

```
# argument to check
arg <- 10

# returns silently if the argument has valid type
check_type(arg, "double")

## Not run:
check_type(arg, "character")

# specify argument's name
check_type(arg, "character", name = "x")

# specify argument `specific` with `glue::glue()` syntax
specific <- "`{name}`'s type is {feature}, which is wrong."
check_type(arg, "character", specific = specific)

# specify argument `supplement`
supplement <- c("You're wrong.", i = "Check your code.")
check_type(arg, "character", supplement = supplement)

# turn off `specific`
check_type(arg, "character", specific = character())

## End(Not run)

# add and retrieve additional argument
tryCatch(
  {check_type(arg, "character", your_arg = "your data")},
  error = function(e) e$your_arg
)
```

---

check_types *Check Each Item's Type*

Description

Check if each item of an argument has valid type, and if not, generate an error message.

Usage

```
check_types(
  x,
  valid,
  name = NULL,
  general = NULL,
  specific = NULL,
  supplement = NULL,
  ...
)
```

Arguments

x	The argument to check, which must be a list.
valid	A character vector which contains the valid types.
name	A single character which gives the argument's name. The name is used in the error message. By default, the name of the argument passed to argument x is captured automatically.
general	Optional. A single character which is used to give a general statement of the error incurred. By default, this is generated automatically.
specific	Optional. A single character which gives a detailed description of the error. glue::glue() syntax can be used, see "Examples" section. By default, this is generated automatically.
supplement	Optional. A (named) character vector which gives some additional information about the error. The names are used to create bullets, see throw(). By default, this is left empty.
...	Optional. Additional arguments which can be retrieved with tryCatch().

Value

returns an invisible NULL if the argument is valid, or generates an error message.

See Also

"Examples" section in check_type() for how to customize error message and how to add and retrieve additional arguments.

vignette("erify") for a gentle introduction to this package.

Examples

```
## Not run:
# argument to check
arg <- as.list(1:10)

check_types(arg, "character")

# customize error message with `glue::glue()` syntax
specific <- "`{name}[[{i}]]` is an {feature}, oh my god!"
check_types(arg, "character",  specific = specific)

## End(Not run)
```

join *Connect Words with Conjunction*

Description

Connect given words with a conjunction, e.g. "and" and "or".

Usage

```
join(words, conjunction = "or")
```

Arguments

words	A vector of list whose items can be converted to characters.
conjunction	A single character which represents a conjunction word. The default value is "or".

Value

If has length 1 or less, words is returned. Or items of words are concatenated and returned.

Examples

```
words <- c("apple", "orange", "Pink Floyd")
join(words, "and")
```

throw *Generate and Signal Condition*

Description

Generate and signal a condition.

Usage

```
throw(general, specifics = NULL, env = NULL, as = "error", class = NULL, ...)
```

Arguments

general	A single character which gives a general statement of the condition.
specifics	Optional. A character vector which gives a list of details of the condition. If is character(0), throw() will return silently. If is a named vector, the names are used to create bullets. If the name is "x" or "i", the bullet will be colored and bold. The default name is "x". You can customize bullets with option erify.bullets.

env	Optional. An environment or named list which is used to evaluate the R code in the above arguments. See glue::glue().
as	Optional. "error", "warning" or "message" which indicates how to signal the condition. The default value is "error".
class	Optional. A character vector which assigns classes to the condition.
...	Optional. Additional arguments which are stored in the condition and can be retrieved with tryCatch().

Value

If specifics is character(0), returns an invisible NULL. Or signals an error, a warning, or a message.

Examples

```
general <- "You are wrong."

# returns silently
throw(general, character(0))

## Not run:
throw(general)

specifics <- c("Detail 1.", i = "Detail 2.")
throw(general, specifics)

# embed R code with glue syntax
throw("`x` is {x}.", env = list(x = 1))

## End(Not run)

# add and retrieve additional argument
tryCatch(
  { throw(general, arg = "I'm an additional argument.") },
  error = function(e) e$arg
)
```

Index