Package: rtemis.core (via r-universe)

May 27, 2026

Title Core Utilities for the 'rtemis' Ecosystem

Version 0.2.0

Date 2026-05-27

Description Utilities used across packages of the 'rtemis' ecosystem. Includes the msg() messaging system and the fmt() formatting system. Provides a library of 'S7' properties, test_* functions that return logical values, check_* functions that throw informative errors, and clean_* functions that return validated and coerced values. This code began as part of the 'rtemis' package (<doi:10.32614/CRAN.package.rtemis>).

License GPL (>= 3)

URL https://www.rtemis.org, https://github.com/rtemis-org/rtemis.core

BugReports https://github.com/rtemis-org/rtemis.core/issues

Depends R (>= 4.1.0)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports data.table, methods, S7

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/testthat/parallel true

Config/roxygen2/version 8.0.0

Repository https://rtemis-org.r-universe.dev

Date/Publication 2026-05-27 18:24:21 UTC

RemoteUrl https://github.com/rtemis-org/rtemis.core

RemoteRef HEAD

RemoteSha 7263e24e92205a22bd25a4e9d23e6ec8afb4c7c4

Contents

`rtemis.core-package` **rtemis.core***: Rtemis Utilities*

Description

Core Utilities for rtemis R Packages

Author(s)

Maintainer: E.D. Gennatas <gennatas@gmail.com> (ORCID) [copyright holder]

Authors:

- E.D. Gennatas <gennatas@gmail.com> (ORCID) [copyright holder]

See Also

Useful links:

- https://www.rtemis.org
- https://github.com/rtemis-org/rtemis.core
- Report bugs at https://github.com/rtemis-org/rtemis.core/issues

abbreviate_class *Abbreviate object class name*

Description

Abbreviate object class name

Usage

```
abbreviate_class(x, n = 4L)
```

Arguments

x	Object.
n	Integer: Minimum abbreviation length.

Value

Character: Abbreviated class wrapped in angle brackets.

Author(s)

EDG

Examples

```
abbreviate_class(iris)
abbreviate_class(iris, n = 3)
```

abort *Dual-channel error signal*

Description

Signals a condition AND optionally writes a styled event line to the operator console. The two channels carry complementary information so nothing is duplicated:

Usage

```
abort(..., class = NULL, parent = NULL, verbosity = NULL, package = NULL)
```

Arguments

`...`	Message components, concatenated with no separator.
`class`	Character vector: Additional condition classes (prepended to the base `c("rtemis_error", "error", "condition")`).
`parent`	Condition or NULL: Wrapped parent condition. Its message is echoed to the console (when verbosity allows) and stored on the signalled condition as `$parent`.
`verbosity`	Integer or NULL: Overrides `get_verbosity()` for the console echo. Set to `0L` to suppress the echo without affecting the signalled condition.
`package`	Character or NULL: Package name for verbosity override.

Details

- **Console echo** (when verbosity allows): the most-specific condition class plus the caller bracket - a structured "what failed, where" line. Falls back to `"rtemis_error"` when `class = NULL` so the line is always namespace-tagged and recognizable as ours.
- **Condition** `$message`: the corrective human-readable text passed via `...`. Plain text with all ANSI escapes stripped, so it is safe to serialize into JSON, HTML, or any other ANSI-unaware sink (e.g. browser-side error display), and is what `conditionMessage()` / R's default error printer / `tryCatch(error = ...)` handlers see.

Use `class` to add wire-protocol-specific condition classes that callers can catch via `tryCatch()`. The base classes `"rtemis_error"`, `"error"`, and `"condition"` are always added.

The condition also carries `$trace` - a `pairlist` of `sys.calls()` captured at the abort site, with `abort()`'s own frame trimmed. Unlike base R's `traceback()` (which only sees `.Traceback`, populated only when an error reaches the top-level uncaught), `$trace` survives `tryCatch()` and travels with the condition - so server-side handlers can ship the stack to a browser-side debug pane, or callers can call `format_trace()` to print it.

Value

Does not return - always signals a condition via `stop()`.

Author(s)

EDG

Examples

```
## Not run:
abort("Could not parse ", "hyperparameters", ".",
      class = "rtemislive_invalid_params")

## End(Not run)
```

ansi256_to_hex *Convert ANSI 256 color code to HEX*

Description

Convert ANSI 256 color code to HEX

Usage

```
ansi256_to_hex(code)
```

Arguments

code	Integer: ANSI 256 color code (0-255).

Value

Character: HEX color string.

Author(s)

EDG

Examples

```
ansi256_to_hex(1)
```

bold *Make text bold*

Description

A `fmt()` convenience wrapper for making text bold.

Usage

```
bold(text, output_type = c("ansi", "html", "plain"))
```

Arguments

text	Character: Text to make bold
output_type	Character: Output type ("ansi", "html", "plain")

Value

Character: Formatted text with bold styling

Author(s)

EDG

Examples

```
message(bold("This is bold!"))
```

`bounded_double_property` *Create a bounded double S7 property*

Description

Returns a `new_property()` for a double scalar constrained to a given interval. Useful for bounds not covered by the pre-built properties.

Usage

```
bounded_double_property(
  lower = -Inf,
  upper = Inf,
  lower_open = FALSE,
  upper_open = FALSE,
  nullable = FALSE
)
```

Arguments

`lower`	Numeric scalar. Lower bound. Default `-Inf`.
`upper`	Numeric scalar. Upper bound. Default `Inf`.
`lower_open`	Logical scalar. If `TRUE`, lower bound is exclusive `(lower, ...]`. Default `FALSE`.
`upper_open`	Logical scalar. If `TRUE`, upper bound is exclusive `[..., upper)`. Default `FALSE`.
`nullable`	Logical scalar. If `TRUE`, `NULL` is also accepted. Default `FALSE`.

Value

An S7 property object.

Author(s)

EDG

Examples

```
# Learning rate in (0, 1]
lr_prop <- bounded_double_property(0, 1, lower_open = TRUE)
```

character_scalar *Non-empty character scalar S7 property*

Description

S7 property accepting a single non-NA, non-empty (after trimming whitespace) string.

Usage

```
character_scalar
```

Value

An S7 property object.

Author(s)

EDG

check_character *Check character*

Description

Check character

Usage

```
check_character(x, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

x	Vector to check.
allow_null	Logical: If TRUE, NULL values are allowed and return early.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if check fails.

Author(s)

EDG

Examples

```
check_character("papaya")
# Throws error:
try(check_character(42L))
```

`check_character_scalar` *Check character scalar*

Description

Check character scalar

Usage

```
check_character_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

`x`	Character: Value to check. Must be a single non-NA, non-empty string.
`arg_name`	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_character_scalar("hello")
# Throw error:
try(check_character_scalar(""))
try(check_character_scalar(NA_character_))
try(check_character_scalar(c("a", "b")))
```

check_data.table *Check data.table*

Description

Check data.table

Usage

```
check_data.table(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Object to check.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if input is not a data.table, returns x invisibly otherwise.

Author(s)

EDG

Examples

```
check_data.table(data.table::as.data.table(iris))
# Throws error:
try(check_data.table(iris))
```

check_dependencies **rtemis.core** *internal: Dependencies check*

Description

Checks if dependencies can be loaded; names missing dependencies if not.

Usage

```
check_dependencies(..., verbosity = 0L)
```

Arguments

...	List or vector of strings defining namespaces to be checked
verbosity	Integer: Verbosity level. Note: An error will always printed if dependencies are missing. Setting this to FALSE stops it from printing "Dependencies check passed".

Value

Called for side effects. Aborts and prints list of missing dependencies, if any.

Author(s)

EDG

Examples

```
check_dependencies("base")
# Throws error:
try(check_dependencies("zlorbglorb"))
```

`check_double_scalar` *Check double scalar*

Description

Check double scalar

Usage

```
check_double_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

`x`	Numeric: Value to check. Must be a single non-NA number (integer inputs are accepted).
`arg_name`	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_double_scalar(3.14)
check_double_scalar(1L)
# Throw error:
try(check_double_scalar(NA_real_))
try(check_double_scalar(c(1.0, 2.0)))
```

check_enum *Check if value is in set of allowed values*

Description

Checks if a value is in a set of allowed values, and throws an error if not.

Usage

```
check_enum(x, allowed_values, arg_name = deparse(substitute(x)))
```

Arguments

x	Value to check.
allowed_values	Vector of allowed values.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if x is not in allowed_values, returns x invisibly otherwise.

Author(s)

EDG

Examples

```
check_enum("apple", c("apple", "banana", "cherry"))
# Throws error:
try(check_enum("granola", c("croissant", "bagel", "scramble")))
```

check_float_neg1_1 *Check float -1 <= x <= 1*

Description

Check float -1 <= x <= 1

Usage

```
check_float_neg1_1(x, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric vector.
allow_null	Logical: If TRUE, NULL values are allowed and return early.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_float_neg1_1(c(-1, 0, 1))
# Throws error:
try(check_float_neg1_1(c(-1.5, 0, 1.5)))
```

`check_float01exc` *Check float between 0 and 1, exclusive*

Description

Check float between 0 and 1, exclusive

Usage

```
check_float01exc(x, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

`x`	Numeric vector.
`allow_null`	Logical: If TRUE, NULL values are allowed and return early.
`arg_name`	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_float01exc(c(0.2, 0.7))
# Throws error:
try(check_float01exc(c(0, 0.5, 1)))
```

check_float01inc *Check float between 0 and 1, inclusive*

Description

Check float between 0 and 1, inclusive

Usage

```
check_float01inc(x, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric vector.
allow_null	Logical: If TRUE, NULL values are allowed and return early.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_float01inc(0.5)
```

check_float0pos *Check float greater than or equal to 0*

Description

Checks if an input is a numeric vector containing non-negative (>= 0) values and no NAs. It is designed to validate function arguments.

Usage

```
check_float0pos(x, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric vector.
allow_null	Logical: If TRUE, NULL values are allowed and return early.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_float0pos(c(0, 0.5, 1))
# Allows integers since they are numeric and can be coerced to double without loss of information
check_float0pos(c(0L, 1L))
# Throws error:
try(check_float0pos(c(-1.5, 0, 1.5)))
```

---

check_floatpos *Check positive float*

Description

Check positive float

Usage

```
check_floatpos(x, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric vector.
allow_null	Logical: If TRUE, NULL values are allowed and return early.
arg_name	Character: Name of the variable for error messages.

Details

Checking with `is.numeric()` allows integer inputs as well, which should be ok since it is unlikely the function that consumes this will enforce double type only, but instead is most likely to allow implicit coercion from integer to numeric.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_floatpos(c(0.5, 1.5))
# Allows integers since they are numeric and can be coerced to double without loss of information
check_floatpos(c(1L, 3L))
# Throws error:
try(check_floatpos(c(-1.5, 0.5, 1.5)))
```

---

check_floatpos1 *Check float in (0, 1]*

Description

Check float in (0, 1]

Usage

```
check_floatpos1(x, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric vector.
allow_null	Logical: If TRUE, NULL values are allowed and return early.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_floatpos1(c(0.5, 1))
# Throw error:
try(check_floatpos1(c(0, 0.7)))
try(check_floatpos1(c(0.5, 1.5)))
```

check_inherits *Check class of object*

Description

Check class of object

Usage

```
check_inherits(x, cl, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

x	Object to check.
cl	Character: class to check against.
allow_null	Logical: If TRUE, NULL values are allowed and return early.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_inherits("papaya", "character")
# These will throw errors:
try(check_inherits(c(1, 2.5, 3.2), "integer"))
try(check_inherits(iris, "list"))
```

check_integer_scalar *Check integer scalar*

Description

Check integer scalar

Usage

```
check_integer_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric: Value to check. Must be a single non-NA whole number.
arg_name	Character: Argument name to use in error messages.

Details

Accepts any single numeric value that is a whole number. Integer-typed inputs (1L) and double-typed whole numbers (1, 100) are both accepted for user convenience.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_integer_scalar(5L)
check_integer_scalar(100)
# Throw error:
try(check_integer_scalar(1.5))
try(check_integer_scalar(NA_integer_))
```

---

check_logical *Check logical*

Description

Check logical

Usage

```
check_logical(x, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

x	Vector to check.
allow_null	Logical: If TRUE, NULL values are allowed and return early.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_logical(c(TRUE, FALSE))
# Throws error:
try(check_logical(c(0, 1)))
```

---

check_logical_scalar *Check logical scalar*

Description

Check logical scalar

Usage

```
check_logical_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Logical: Value to check. Must be a single non-NA TRUE or FALSE.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_logical_scalar(TRUE)
check_logical_scalar(FALSE)
# Throw error:
try(check_logical_scalar(NA))
try(check_logical_scalar(1L))
try(check_logical_scalar(c(TRUE, FALSE)))
```

check_nonneg_double_scalar *Check non-negative double scalar*

Description

Check non-negative double scalar

Usage

```
check_nonneg_double_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric: Value to check. Must be a single finite number greater than or equal to zero.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_nonneg_double_scalar(0)
check_nonneg_double_scalar(5)
# Throw error:
try(check_nonneg_double_scalar(-0.001))
try(check_nonneg_double_scalar(Inf))
```

check_nonneg_double_vector *Check non-negative double vector*

Description

Check non-negative double vector

Usage

```
check_nonneg_double_vector(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric: Value to check. Must be a non-empty vector with all elements finite, greater than or equal to zero, and no NAs.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_nonneg_double_vector(c(0, 1, 2.5))
# Throw error:
try(check_nonneg_double_vector(c(-1, 0, 1)))
try(check_nonneg_double_vector(c(1, Inf)))
```

---

check_numeric *Check numeric*

Description

Checks that x is numeric. Uses `is.numeric()`, which accepts both "double" and "integer" inputs - the right semantics for "is this a number?". Prefer this over `check_inherits(x, "numeric")`, which rejects integers because their literal class is `"integer"`, not `"numeric"` (a long-standing R/S3 quirk). This trips up wire-format callers: `jsonlite::fromJSON("1")` returns an integer, and the value would then fail `inherits(., "numeric")` despite being a perfectly good number.

Usage

```
check_numeric(x, allow_null = TRUE, arg_name = deparse(substitute(x)))
```

Arguments

x	Vector to check.
allow_null	Logical: If TRUE, NULL values are allowed and return early.
arg_name	Character: Name of the variable for error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_numeric(1L)
check_numeric(1.5)
check_numeric(c(1, 2, 3))
# Throws error:
try(check_numeric("1"))
try(check_numeric(TRUE))
```

`check_optional_character_scalar` *Check optional character scalar*

Description

Check optional character scalar

Usage

```
check_optional_character_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

`x`	Optional Character: Value to check. Must be `NULL` or a single non-NA, non-empty string.
`arg_name`	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_character_scalar(NULL)
check_optional_character_scalar("hello")
# Throw error:
try(check_optional_character_scalar(""))
try(check_optional_character_scalar(c("a", "b")))
```

check_optional_double_scalar *Check optional double scalar*

Description

Check optional double scalar

Usage

```
check_optional_double_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Optional Numeric: Value to check. Must be NULL or a single non-NA number.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_double_scalar(NULL)
check_optional_double_scalar(2.5)
# Throw error:
try(check_optional_double_scalar(NA_real_))
```

check_optional_integer_scalar *Check optional integer scalar*

Description

Check optional integer scalar

Usage

```
check_optional_integer_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Optional Numeric: Value to check. Must be NULL or a single non-NA whole number.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_integer_scalar(NULL)
check_optional_integer_scalar(10L)
# Throw error:
try(check_optional_integer_scalar(1.5))
```

---

check_optional_logical_scalar
Check optional logical scalar

Description

Check optional logical scalar

Usage

```
check_optional_logical_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Optional Logical: Value to check. Must be NULL or a single non-NA TRUE or FALSE.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_logical_scalar(NULL)
check_optional_logical_scalar(FALSE)
# Throw error:
try(check_optional_logical_scalar(NA))
```

`check_optional_nonneg_double_scalar` *Check optional non-negative double scalar*

Description

Check optional non-negative double scalar

Usage

```
check_optional_nonneg_double_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

`x`	Optional Numeric: Value to check. Must be `NULL` or a single finite number greater than or equal to zero.
`arg_name`	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_nonneg_double_scalar(NULL)
check_optional_nonneg_double_scalar(0)
# Throw error:
try(check_optional_nonneg_double_scalar(-1))
```

check_optional_nonneg_double_vector

Check optional non-negative double vector

Description

Check optional non-negative double vector

Usage

```
check_optional_nonneg_double_vector(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Optional Numeric: Value to check. Must be NULL or a non-empty vector with all elements finite, greater than or equal to zero, and no NAs.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_nonneg_double_vector(NULL)
check_optional_nonneg_double_vector(c(0, 1, 5))
# Throw error:
try(check_optional_nonneg_double_vector(c(-1, 0)))
```

check_optional_pos_double_scalar

Check optional positive double scalar

Description

Check optional positive double scalar

Usage

```
check_optional_pos_double_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Optional Numeric: Value to check. Must be NULL or a single finite number strictly greater than zero.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_pos_double_scalar(NULL)
check_optional_pos_double_scalar(2.5)
# Throw error:
try(check_optional_pos_double_scalar(0))
```

---

check_optional_pos_double_vector

Check optional positive double vector

Description

Check optional positive double vector

Usage

```
check_optional_pos_double_vector(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Optional Numeric: Value to check. Must be NULL or a non-empty vector with all elements finite, strictly greater than zero, and no NAs.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_pos_double_vector(NULL)
check_optional_pos_double_vector(c(0.5, 2))
# Throw error:
try(check_optional_pos_double_vector(c(0, 1)))
```

`check_optional_pos_integer_scalar` *Check optional positive integer scalar*

Description

Check optional positive integer scalar

Usage

```
check_optional_pos_integer_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

`x`	Optional Numeric: Value to check. Must be `NULL` or a single non-NA whole number greater than zero.
`arg_name`	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_pos_integer_scalar(NULL)
check_optional_pos_integer_scalar(5L)
# Throw error:
try(check_optional_pos_integer_scalar(0))
```

check_optional_prob_scalar *Check optional probability scalar*

Description

Check optional probability scalar

Usage

```
check_optional_prob_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Optional Numeric: Value to check. Must be NULL or a single finite number in $[0, 1]$.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_prob_scalar(NULL)
check_optional_prob_scalar(0.5)
# Throw error:
try(check_optional_prob_scalar(2.0))
```

check_optional_prob_vector *Check optional probability vector*

Description

Check optional probability vector

Usage

```
check_optional_prob_vector(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Optional Numeric: Value to check. Must be NULL or a non-empty vector with all elements in $[0, 1]$ and no NAs.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_prob_vector(NULL)
check_optional_prob_vector(c(0.2, 0.8))
# Throw error:
try(check_optional_prob_vector(c(0.5, 2)))
```

---

check_optional_scalar_character
Check Optional Scalar Character

Description

Check Optional Scalar Character

Usage

```
check_optional_scalar_character(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Optional Character: Value to check.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects.

Author(s)

EDG

Examples

```
check_optional_scalar_character(NULL, "my_arg")
check_optional_scalar_character("hello", "my_arg")
# Throw error:
try(check_optional_scalar_character(c("hello", "world"), "my_arg"))
try(check_optional_scalar_character(123, "my_arg"))
```

`check_optional_unit_open_vector`

Check optional open-unit-interval vector

Description

Check optional open-unit-interval vector

Usage

```
check_optional_unit_open_vector(x, arg_name = deparse(substitute(x)))
```

Arguments

`x`	Optional Numeric: Value to check. Must be `NULL` or a non-empty vector with all elements strictly in $(0, 1)$ and no NAs.
`arg_name`	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_optional_unit_open_vector(NULL)
check_optional_unit_open_vector(c(0.1, 0.9))
# Throw error:
try(check_optional_unit_open_vector(c(0, 0.5)))
```

check_pos_double_scalar *Check positive double scalar*

Description

Check positive double scalar

Usage

```
check_pos_double_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric: Value to check. Must be a single finite number strictly greater than zero.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_pos_double_scalar(0.001)
check_pos_double_scalar(100)
# Throw error:
try(check_pos_double_scalar(0))
try(check_pos_double_scalar(-1))
try(check_pos_double_scalar(Inf))
```

check_pos_double_vector *Check positive double vector*

Description

Check positive double vector

Usage

```
check_pos_double_vector(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric: Value to check. Must be a non-empty vector with all elements finite, strictly greater than zero, and no NAs.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_pos_double_vector(c(0.1, 1, 10))
# Throw error:
try(check_pos_double_vector(c(0, 1)))
try(check_pos_double_vector(c(1, Inf)))
```

---

check_pos_integer_scalar

Check positive integer scalar

Description

Check positive integer scalar

Usage

```
check_pos_integer_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric: Value to check. Must be a single non-NA whole number greater than zero.
arg_name	Character: Argument name to use in error messages.

Details

Accepts any single numeric value that is a whole number strictly greater than zero. Integer-typed inputs (`1L`) and double-typed whole numbers (`1`, `100`) are both accepted for user convenience.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_pos_integer_scalar(1L)
check_pos_integer_scalar(10)
# Throw error:
try(check_pos_integer_scalar(0))
try(check_pos_integer_scalar(-1L))
try(check_pos_integer_scalar(1.5))
```

---

check_prob_scalar *Check probability scalar*

Description

Check probability scalar

Usage

```
check_prob_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric: Value to check. Must be a single finite number in $[0, 1]$.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_prob_scalar(0)
check_prob_scalar(0.5)
check_prob_scalar(1)
# Throw error:
try(check_prob_scalar(1.5))
try(check_prob_scalar(-0.1))
```

check_prob_vector *Check probability vector*

Description

Check probability vector

Usage

```
check_prob_vector(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric: Value to check. Must be a non-empty vector with all elements in $[0, 1]$ and no NAs.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_prob_vector(c(0, 0.5, 1))
# Throw error:
try(check_prob_vector(c(0.5, 1.5)))
try(check_prob_vector(c(0.5, NA)))
```

check_scalar_character *Check Scalar Character*

Description

Check Scalar Character

Usage

```
check_scalar_character(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Character: Value to check.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects.

Author(s)

EDG

Examples

```
check_scalar_character("hello", "my_arg")
# Throw error:
try(check_scalar_character(c("hello", "world"), "my_arg"))
try(check_scalar_character(123, "my_arg"))
```

---

check_scalar_logical *Check Scalar Logical*

Description

Check Scalar Logical

Usage

```
check_scalar_logical(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Logical: Value to check.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects.

Author(s)

EDG

Examples

```
check_scalar_logical(TRUE, "my_arg")
# Throw error:
try(check_scalar_logical(c(TRUE, FALSE), "my_arg"))
try(check_scalar_logical(NA, "my_arg"))
```

check_tabular *Check object is tabular*

Description

Checks if object is of class `data.frame`, `data.table`, or `tbl_df`.

Usage

```
check_tabular(x)
```

Arguments

x	Object to check.

Value

Called for side effects. Throws an error if input is not tabular, returns x invisibly otherwise.

Author(s)

EDG

Examples

```
check_tabular(iris)
check_tabular(data.table::as.data.table(iris))
# Throws error:
try(check_tabular(matrix(1:10, ncol = 2)))
```

check_unit_open_scalar *Check open-unit-interval scalar*

Description

Check open-unit-interval scalar

Usage

```
check_unit_open_scalar(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Numeric: Value to check. Must be a single finite number strictly in $(0, 1)$.
arg_name	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_unit_open_scalar(0.5)
# Throw error:
try(check_unit_open_scalar(0))
try(check_unit_open_scalar(1))
```

`check_unit_open_vector` *Check open-unit-interval vector*

Description

Check open-unit-interval vector

Usage

```
check_unit_open_vector(x, arg_name = deparse(substitute(x)))
```

Arguments

`x`	Numeric: Value to check. Must be a non-empty vector with all elements strictly in $(0, 1)$ and no NAs.
`arg_name`	Character: Argument name to use in error messages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

Examples

```
check_unit_open_vector(c(0.2, 0.5, 0.9))
# Throw error:
try(check_unit_open_vector(c(0, 0.5)))
try(check_unit_open_vector(c(0.5, 1)))
```

clean_colnames *Clean column names*

Description

Clean column names by replacing all spaces and punctuation with a single underscore

Usage

```
clean_colnames(x)
```

Arguments

x	Character vector or matrix with colnames or any object with `names()` method.

Value

Character vector.

Author(s)

EDG

Examples

```
clean_colnames(iris)
```

clean_int *Clean integer input*

Description

Clean integer input

Usage

```
clean_int(x, arg_name = deparse(substitute(x)))
```

Arguments

x	Double or integer vector to check.
arg_name	Character: Name of the variable for error messages.

Details

The goal is to return an integer vector. If the input is integer, it is returned as is. If the input is numeric, it is coerced to integer only if the numeric values are integers, otherwise an error is thrown.

Value

Integer vector

Author(s)

EDG

Examples

```
clean_int(6L)
clean_int(3)
# clean_int(12.1) # Error
clean_int(c(3, 5, 7))
# clean_int(c(3, 5, 7.01)) # Error
```

---

clean_names *Clean names*

Description

Clean character vector by replacing all symbols and sequences of symbols with single underscores, ensuring no name begins or ends with a symbol

Usage

```
clean_names(x, prefix_digits = "V_")
```

Arguments

x	Character vector.
prefix_digits	Character: prefix to add to names beginning with a digit. Set to NA to skip.

Value

Character vector.

Author(s)

EDG

Examples

```
x <- c("Patient ID", "_Date-of-Birth", "SBP (mmHg)")
x
clean_names(x)
```

`clean_posint` *Check positive integer*

Description

Check positive integer

Usage

```
clean_posint(x, allow_na = FALSE, arg_name = deparse(substitute(x)))
```

Arguments

`x`	Integer vector.
`allow_na`	Logical: If TRUE, NAs are excluded before checking. If FALSE (default), NAs trigger an error.
`arg_name`	Character: Name of the variable for error messages.

Value

Integer vector of positive values.

Author(s)

EDG

Examples

```
clean_posint(5)
```

col256 *Apply 256-color formatting*

Description

Apply 256-color formatting

Usage

```
col256(text, col = "79", bg = FALSE, output_type = c("ansi", "html", "plain"))
```

Arguments

text	Character: Text to color
col	Character or numeric: Color (ANSI 256-color code, hex for HTML)
bg	Logical: If TRUE, apply as background color
output_type	Character: Output type ("ansi", "html", "plain")

Value

Character: Formatted text with 256-color styling

Author(s)

EDG

Examples

```
col256("Hello", col = 160, output_type = "ansi")
```

dbg *Debug log message*

Description

Muted log message gated at verbosity `>= 2L`. Use for development / troubleshooting output that should not appear in normal operation.

Usage

```
dbg(..., verbosity = NULL, package = NULL)
```

Arguments

...	Message components, concatenated with no separator.
verbosity	Integer or NULL: Overrides get_verbosity() when supplied.
package	Character or NULL: Package name for verbosity override.

Details

Named dbg() rather than debug() to avoid shadowing base::debug() - the R debugger entry point.

Value

Invisible NULL.

Author(s)

EDG

Examples

```
dbg("payload bytes: ", 1234L)
```

---

ddSci *Format Numbers for Printing*

Description

2 Decimal places, otherwise scientific notation

Usage

```
ddSci(x, decimal_places = 2, hi = 1e+06, as_numeric = FALSE)
```

Arguments

x	Vector of numbers
decimal_places	Integer: Return this many decimal places.
hi	Float: Threshold at or above which scientific notation is used.
as_numeric	Logical: If TRUE, convert to numeric before returning. This will not force all numbers to print 2 decimal places. For example: 1.2035 becomes "1.20" if as_numeric = FALSE, but 1.2 otherwise This can be helpful if you want to be able to use the output as numbers / not just for printing.

Details

Numbers will be formatted to 2 decimal places, unless this results in 0.00 (e.g. if input was .0032), in which case they will be converted to scientific notation with 2 significant figures. `ddSci` will return `0.00` if the input is exactly zero. This function can be used to format numbers in plots, on the console, in logs, etc.

Value

Formatted number

Author(s)

EDG

Examples

```
x <- .34876549
ddSci(x)
# "0.35"
x <- .00000000457823
ddSci(x)
# "4.6e-09"
```

`double_scalar` *Double scalar S7 property*

Description

S7 property accepting a single non-NA double value.

Usage

```
double_scalar
```

Value

An S7 property object.

Author(s)

EDG

enum *Create an enum S7 property*

Description

Returns a `new_property()` for a character scalar constrained to a fixed set of allowed values.

Usage

```
enum(values, default = NULL, nullable = FALSE)
```

Arguments

`values`	Character: Allowed values.
`default`	Optional Character: Default value.
`nullable`	Logical scalar. If `TRUE`, `NULL` is also accepted. Default `FALSE`.

Value

An S7 property object.

Author(s)

EDG

Examples

```
type_prop <- enum(c("string", "number", "boolean"), default = "string")
```

fmt *Text formatting*

Description

Formats text with specified color, styles, and background using ANSI escape codes or HTML, with support for plain text output.

Usage

```
fmt(
  x,
  col = NULL,
  bold = FALSE,
  italic = FALSE,
  underline = FALSE,
  thin = FALSE,
  muted = FALSE,
  bg = NULL,
  pad = 0L,
  output_type = c("ansi", "html", "plain")
)
```

Arguments

x	Character: Text to format.
col	Character: Color (hex code, named color, or NULL for no color).
bold	Logical: If TRUE, make text bold.
italic	Logical: If TRUE, make text italic.
underline	Logical: If TRUE, underline text.
thin	Logical: If TRUE, make text thin/light.
muted	Logical: If TRUE, make text muted/dimmed.
bg	Character: Background color (hex code, named color, or NULL).
pad	Integer: Number of spaces to pad before text.
output_type	Character: Output type ("ansi", "html", "plain").

Details

This function combines multiple formatting options into a single call, making it more efficient than nested function calls. It generates optimized ANSI escape sequences and clean HTML output.

Value

Character: Formatted text with specified styling.

Author(s)

EDG

Examples

```
# Simple color
fmt("Hello", col = "red")

# Bold red text
fmt("Error", col = "red", bold = TRUE)
```

```
# Multiple styles
fmt("Warning", col = "yellow", bold = TRUE, italic = TRUE)

# With background
fmt("Highlight", col = "white", bg = "blue", bold = TRUE)
```

`fmt_gradient` *Gradient text*

Description

Gradient text

Usage

```
fmt_gradient(x, colors, bold = FALSE, output_type = c("ansi", "html", "plain"))
```

Arguments

`x`	Character: Text to colorize.
`colors`	Character vector: Colors to use for the gradient.
`bold`	Logical: If TRUE, make text bold.
`output_type`	Character: Output type ("ansi", "html", "plain").

Value

Character: Text with gradient color applied.

Author(s)

EDG

Examples

```
fmt_gradient("Gradient Text", colors = c("blue", "red")) |> message()
```

format_trace *Pretty-print a captured call trace*

Description

Formats the $trace carried by an rtemis_error condition (see abort()) as a numbered, one-line-per-frame string. Most-recent frame at the bottom, matching base R's traceback() convention. Each frame is deparsed with a single-line cap so long calls stay readable; no styling is applied, so the output is safe for any sink (terminal, JSON, HTML).

Usage

```
format_trace(trace, max_width = 80L)
```

Arguments

trace	pairlist of calls, as captured by abort() on cond$trace. Passing the condition itself also works - the trace is extracted via cond$trace.
max_width	Integer: Max characters per deparsed line. Longer calls are truncated with a trailing ellipsis.

Value

Character scalar with one frame per \n-separated line, newest frame last. "" if the trace is empty or NULL.

Author(s)

EDG

Examples

```
## Not run:
cond <- tryCatch(
  check_numeric("oops"),
  error = identity
)
cat(format_trace(cond), "\n")

## End(Not run)
```

get_output_type *Get output type*

Description

Get output type for printing text.

Usage

```
get_output_type(output_type = c("ansi", "html", "plain"), filename = NULL)
```

Arguments

output_type	Character vector of output types.
filename	Optional Character: Filename for output.

Details

Exported as internal function for use by other rtemis packages.

Value

Character with selected output type.

Author(s)

EDG

Examples

```
get_output_type()
```

get_verbosity *Resolve the current logging verbosity*

Description

Reads `getOption("<package>.verbosity")` first when package is supplied, falling back to `getOption("rtemis.verbosi` and finally to `1L`. Levels: `0L` silent, `1L` info/warn/success/abort console echo, `2L` includes debug.

Usage

```
get_verbosity(package = NULL)
```

Arguments

package	Character or NULL: Optional package-specific override.

Value

Integer scalar verbosity level.

Author(s)

EDG

Examples

```
get_verbosity()
```

`highlight` *Highlight text*

Description

A `fmt()` convenience wrapper for highlighting text.

Usage

```
highlight(x, pad = 0L, output_type = c("ansi", "html", "plain"))
```

Arguments

`x`	Character: Text to highlight.
`pad`	Integer: Number of spaces to pad before text.
`output_type`	Character: Output type ("ansi", "html", "plain").

Value

Character: Formatted text with highlight.

Author(s)

EDG

Examples

```
message(highlight("This is highlighted!"))
```

info *Informational log message*

Description

Styled informational message, routed through `msg()` so it carries the shared datetime + caller prefix. Fires when verbosity is at least `1L`.

Usage

```
info(..., verbosity = NULL, package = NULL)
```

Arguments

`...`	Message components, concatenated with no separator.
`verbosity`	Integer or NULL: Overrides `get_verbosity()` when supplied.
`package`	Character or NULL: Package name for verbosity override lookup (e.g. `"rtemis.server"`).

Value

Invisible NULL.

Author(s)

EDG

Examples

```
info("Server started on port ", 8080L)
```

integer_scalar *Integer scalar S7 property*

Description

S7 property accepting a single non-NA integer value (must be `integer` type, e.g. `1L`).

Usage

```
integer_scalar
```

Value

An S7 property object.

Author(s)

EDG

`labelify` *Format text for label printing*

Description

Format text for label printing

Usage

```
labelify(
  x,
  underscores_to_spaces = TRUE,
  dotsToSpaces = TRUE,
  toLower = FALSE,
  toTitleCase = TRUE,
  capitalize_strings = c("id"),
  stringsToSpaces = c("\\$", "`")
)
```

Arguments

`x`	Character: Input
`underscores_to_spaces`	Logical: If TRUE, convert underscores to spaces.
`dotsToSpaces`	Logical: If TRUE, convert dots to spaces.
`toLower`	Logical: If TRUE, convert to lowercase (precedes `toTitleCase`). Default = FALSE (Good for getting all-caps words converted to title case, bad for abbreviations you want to keep all-caps)
`toTitleCase`	Logical: If TRUE, convert to Title Case. Default = TRUE (This does not change all-caps words, set `toLower` to TRUE if desired)
`capitalize_strings`	Character, vector: Always capitalize these strings, if present. Default = `"id"`
`stringsToSpaces`	Character, vector: Replace these strings with spaces. Escape as needed for `gsub`. Default = `"\\$"`, which formats common input of the type `data.frame$variable`

Value

Character vector.

Author(s)

EDG

Examples

```
x <- c("county_name", "total.cost$", "age", "weight.kg")
labelify(x)
```

`logical_scalar` *Logical scalar S7 property*

Description

S7 property accepting a single non-NA logical value.

Usage

```
logical_scalar
```

Value

An S7 property object.

Author(s)

EDG

`match_arg` *Match Arguments Ignoring Case*

Description

Match Arguments Ignoring Case

Usage

```
match_arg(x, choices)
```

Arguments

`x`	Character: Argument to match.
`choices`	Character vector: Choices to match against.

Value

Character: Matched argument.

Author(s)

EDG

Examples

```
match_arg("papaya", c("AppleExtreme", "SuperBanana", "PapayaMaster"))
```

`msg` *Message with provenance*

Description

Print message to output with a prefix including data and time, and calling function or full call stack

Usage

```
msg(
  ...,
  caller = NULL,
  call_depth = 1L,
  caller_id = 1L,
  newline_pre = FALSE,
  newline = TRUE,
  format_fn = plain,
  sep = " ",
  verbosity = 1L
)

msg0(
  ...,
  caller = NULL,
  call_depth = 1,
  caller_id = 1,
  newline_pre = FALSE,
  newline = TRUE,
  format_fn = plain,
  sep = "",
  verbosity = 1L
)
```

Arguments

`...`	Message to print
`caller`	Character: Name of calling function
`call_depth`	Integer: Print the system call path of this depth.
`caller_id`	Integer: Which function in the call stack to print
`newline_pre`	Logical: If TRUE begin with a new line.
`newline`	Logical: If TRUE end with a new line.
`format_fn`	Function: Formatting function to use on the message text.

sep	Character: Use to separate objects in ...
verbosity	Integer: Verbosity level of the message. If 0L, does not print anything and returns NULL, invisibly.

Details

If msg is called directly from the console, it will print [interactive>] in place of the call stack. msg0, similar to paste0, is msg(..., sep = "")

Value

If verbosity > 0L, returns a list with call, message, and date, invisibly, otherwise returns NULL invisibly.

Author(s)

EDG

Examples

```
msg("Hello")
```

---

msgdone *msgdone*

Description

msgdone

Usage

```
msgdone(caller = NULL, call_depth = 1, caller_id = 1, sep = " ")
```

Arguments

caller	Character: Name of calling function
call_depth	Integer: Print the system call path of this depth.
caller_id	Integer: Which function in the call stack to print
sep	Character: Use to separate objects in ...

Value

NULL invisibly

Author(s)

EDG

Examples

```
msgstart("Starting process...")
msgdone("Process complete")
```

`msgstart` *msgstart*

Description

msgstart

Usage

```
msgstart(..., newline_pre = FALSE, sep = "")
```

Arguments

`...`	Message to print
`newline_pre`	Logical: If TRUE begin with a new line.
`sep`	Character: Use to separate objects in `...`

Value

NULL invisibly

Author(s)

EDG

Examples

```
msgstart("Starting process...")
msgdone("Process complete.")
```

nonneg_double_scalar *Non-negative double scalar S7 property*

Description

S7 property accepting a single finite double greater than or equal to zero, i.e. in $[0, \infty)$.

Usage

```
nonneg_double_scalar
```

Value

An S7 property object.

Author(s)

EDG

nonneg_double_vector *Non-negative double vector S7 property*

Description

S7 property accepting a non-empty double vector with all elements finite, greater than or equal to zero, and no NAs.

Usage

```
nonneg_double_vector
```

Value

An S7 property object.

Author(s)

EDG

nonneg_integer_scalar *Non-negative integer scalar S7 property*

Description

S7 property accepting a single non-NA integer value greater than or equal to zero, i.e. in $[0, \infty)$ (e.g. `0L`, `1L`).

Usage

```
nonneg_integer_scalar
```

Value

An S7 property object.

Author(s)

EDG

optional *Create an optional S7 type*

Description

Creates an S7 union type that allows for the specified type or `NULL`.

Usage

```
optional(type)
```

Arguments

`type`	S7 base class or S7 class.

Details

This should be used when the S7 class already includes all the necessary validation for the non-NULL case. Otherwise, create a new S7 property with appropriate validator using `S7::new_property()`.

Value

An S7 union type that allows for the specified type or `NULL`.

Author(s)

EDG

Examples

```
# Create an optional character type
optional(S7::class_character)
```

`optional_character_scalar`

Optional non-empty character scalar S7 property

Description

S7 property accepting `NULL` or a single non-NA, non-empty (after trimming whitespace) string.

Usage

```
optional_character_scalar
```

Value

An S7 property object.

Author(s)

EDG

`optional_double_scalar`

Optional double scalar S7 property

Description

S7 property accepting `NULL` or a single non-NA double value.

Usage

```
optional_double_scalar
```

Value

An S7 property object.

Author(s)

EDG

`optional_integer_scalar` *Optional integer scalar S7 property*

Description

S7 property accepting `NULL` or a single non-NA integer value (must be `integer` type, e.g. `1L`).

Usage

```
optional_integer_scalar
```

Value

An S7 property object.

Author(s)

EDG

`optional_logical_scalar` *Optional logical scalar S7 property*

Description

S7 property accepting `NULL` or a single non-NA logical value.

Usage

```
optional_logical_scalar
```

Value

An S7 property object.

Author(s)

EDG

optional_nonneg_double_scalar

Optional non-negative double scalar S7 property

Description

S7 property accepting `NULL` or a single finite double greater than or equal to zero.

Usage

```
optional_nonneg_double_scalar
```

Value

An S7 property object.

Author(s)

EDG

optional_nonneg_double_vector

Optional non-negative double vector S7 property

Description

S7 property accepting `NULL` or a non-empty double vector with all elements finite, greater than or equal to zero, and no NAs.

Usage

```
optional_nonneg_double_vector
```

Value

An S7 property object.

Author(s)

EDG

`optional_nonneg_integer_scalar` *Optional non-negative integer scalar S7 property*

Description

S7 property accepting `NULL` or a single non-NA integer value greater than or equal to zero.

Usage

```
optional_nonneg_integer_scalar
```

Value

An S7 property object.

Author(s)

EDG

`optional_pos_double_scalar` *Optional positive double scalar S7 property*

Description

S7 property accepting `NULL` or a single finite double strictly greater than zero.

Usage

```
optional_pos_double_scalar
```

Value

An S7 property object.

Author(s)

EDG

optional_pos_double_vector

Optional positive double vector S7 property

Description

S7 property accepting `NULL` or a non-empty double vector with all elements finite, strictly greater than zero, and no NAs.

Usage

```
optional_pos_double_vector
```

Value

An S7 property object.

Author(s)

EDG

optional_pos_integer_scalar

Optional positive integer scalar S7 property

Description

S7 property accepting `NULL` or a single non-NA integer value strictly greater than zero.

Usage

```
optional_pos_integer_scalar
```

Value

An S7 property object.

Author(s)

EDG

optional_prob_scalar *Optional probability scalar S7 property*

Description

S7 property accepting `NULL` or a single finite double in $[0, 1]$.

Usage

```
optional_prob_scalar
```

Value

An S7 property object.

Author(s)

EDG

optional_prob_vector *Optional probability vector S7 property*

Description

S7 property accepting `NULL` or a non-empty double vector with all elements in $[0, 1]$ and no NAs.

Usage

```
optional_prob_vector
```

Value

An S7 property object.

Author(s)

EDG

optional_unit_open_scalar

Optional open-unit-interval scalar S7 property

Description

S7 property accepting `NULL` or a single finite double strictly in $(0, 1)$.

Usage

```
optional_unit_open_scalar
```

Value

An S7 property object.

Author(s)

EDG

optional_unit_open_vector

Optional open-unit-interval vector S7 property

Description

S7 property accepting `NULL` or a non-empty double vector with all elements strictly in $(0, 1)$ and no NAs.

Usage

```
optional_unit_open_vector
```

Value

An S7 property object.

Author(s)

EDG

plain *Force plain text when using* `message()`

Description

Force plain text when using `message()`

Usage

```
plain(x)
```

Arguments

`x`	Character: Text to be output to console.

Value

Character: Text with ANSI escape codes removed.

Author(s)

EDG

Examples

```
message(plain("hello"))
```

pos_double_scalar *Positive double scalar S7 property*

Description

S7 property accepting a single finite double strictly greater than zero, i.e. in $(0, \infty)$.

Usage

```
pos_double_scalar
```

Value

An S7 property object.

Author(s)

EDG

pos_double_vector *Positive double vector S7 property*

Description

S7 property accepting a non-empty double vector with all elements finite, strictly greater than zero, and no NAs.

Usage

```
pos_double_vector
```

Value

An S7 property object.

Author(s)

EDG

pos_integer_scalar *Positive integer scalar S7 property*

Description

S7 property accepting a single non-NA integer value strictly greater than zero (e.g. 1L).

Usage

```
pos_integer_scalar
```

Value

An S7 property object.

Author(s)

EDG

printdf *Print data frame*

Description

Pretty print a data frame

Usage

```
printdf(
  x,
  pad = 0,
  spacing = 1,
  ddSci_dp = NULL,
  transpose = FALSE,
  justify = "right",
  colnames = TRUE,
  rownames = TRUE,
  column_fmt = highlight,
  row_fmt = gray,
  newline_pre = FALSE,
  newline = FALSE
)
```

Arguments

x	data frame
pad	Integer: Pad output with this many spaces.
spacing	Integer: Number of spaces between columns.
ddSci_dp	Integer: Number of decimal places to print using ddSci. Default = NULL for no formatting.
transpose	Logical: If TRUE, transpose x before printing.
justify	Character: "right", "left".
colnames	Logical: If TRUE, print column names.
rownames	Logical: If TRUE, print row names.
column_fmt	Formatting fn for printing column names.
row_fmt	Formatting fn for printing row names.
newline_pre	Logical: If TRUE, print a new line before printing data frame.
newline	Logical: If TRUE, print a new line after printing data frame.

Details

By design, numbers will not be justified, but using ddSci_dp will convert to characters, which will be justified. This is intentional for internal use.

Value

NULL invisibly

Author(s)

EDG

Examples

```
printdf(iris[1:6, ])
```

`printls` *Pretty print list*

Description

Pretty print a list (or data frame) recursively

Usage

```
printls(
  x,
  prefix = "",
  pad = 2L,
  item_format = bold,
  maxlength = 4L,
  center_title = TRUE,
  title = NULL,
  title_newline = TRUE,
  newline_pre = FALSE,
  format_fn_rhs = ddSci,
  print_class = TRUE,
  abbrev_class_n = 3L,
  print_df = FALSE,
  print_S4 = FALSE,
  limit = 12L
)
```

Arguments

`x`	list or object that will be converted to a list.
`prefix`	Character: Optional prefix for names.
`pad`	Integer: Pad output with this many spaces.
`item_format`	Formatting function for list item names.
`maxlength`	Integer: Maximum length of items to show using `headdot()` before truncating with ellipsis.

center_title	Logical: If TRUE, autopad title for centering, if present.
title	Character: Optional title to print before list.
title_newline	Logical: If TRUE, print title on new line.
newline_pre	Logical: If TRUE, print newline before list.
format_fn_rhs	Formatting function for right-hand side values.
print_class	Logical: If TRUE, print abbreviated class of object.
abbrev_class_n	Integer: Number of characters to abbreviate class names to.
print_df	Logical: If TRUE, print data frame contents, otherwise print n rows and columns.
print_S4	Logical: If TRUE, print S4 object contents, otherwise print class name.
limit	Integer: Maximum number of items to show. Use -1 for unlimited.

Details

Data frames in R began life as lists

Value

NULL invisibly

Author(s)

EDG

Examples

```
printls(list(a = 1:10, b = "Hello", c = list(d = 1, e = 2)), title = "A List")
```

---

prob_scalar *Probability scalar S7 property*

Description

S7 property accepting a single finite double in $[0, 1]$.

Usage

```
prob_scalar
```

Value

An S7 property object.

Author(s)

EDG

prob_vector *Probability vector S7 property*

Description

S7 property accepting a non-empty double vector with all elements in $[0, 1]$ and no NAs.

Usage

```
prob_vector
```

Value

An S7 property object.

Author(s)

EDG

repr *String representation*

Description

String representation

Usage

```
repr(x, ...)
```

Arguments

x	Object to represent as a string.
...	Additional arguments passed to methods.

Value

Character string representation of the object.

Author(s)

EDG

Examples

```
S7::method(repr, S7::class_character) <- function(x, ...) {
  paste0("<chr> \"", x, "\"")
}
cat(repr("hello"))
```

---

repr_ls *Show list as formatted string*

Description

Works exactly like printls, but instead of printing to console with cat, it outputs a single string, formatted using mformat, so that cat(repr_ls(x)) looks identical to printls(x) for any list x

Usage

```
repr_ls(
  x,
  prefix = "",
  pad = 2L,
  item_format = bold,
  maxlength = 4L,
  center_title = TRUE,
  title = NULL,
  title_newline = TRUE,
  newline_pre = FALSE,
  format_fn_rhs = ddSci,
  print_class = TRUE,
  abbrev_class_n = 3L,
  print_df = FALSE,
  print_S4 = FALSE,
  limit = 12L,
  output_type = c("ansi", "html", "plain")
)
```

Arguments

x	list or object that will be converted to a list.
prefix	Character: Optional prefix for names.
pad	Integer: Pad output with this many spaces.
item_format	Formatting function for items.
maxlength	Integer: Maximum length of items to show using `headdot()` before truncating with ellipsis.
center_title	Logical: If TRUE, autopad title for centering, if present.
title	Character: Title to print before list.

title_newline	Logical: If TRUE, print title on new line.
newline_pre	Logical: If TRUE, print newline before list.
format_fn_rhs	Formatting function for right-hand side of items.
print_class	Logical: If TRUE, print abbreviated class of object.
abbrev_class_n	Integer: Number of characters to abbreviate class names to.
print_df	Logical: If TRUE, print data frame contents, otherwise print n rows and columns.
print_S4	Logical: If TRUE, print S4 object contents, otherwise print class name.
limit	Integer: Maximum number of items to show.
output_type	Character: Output type for mformat ("ansi", "html", "plain").

Details

Exported as internal function for use by other rtemis packages.

Value

Character: Formatted string that can be printed with cat()

Author(s)

EDG

Examples

```
x <- list(
  a = 1:10,
  b = "Hello",
  c = list(
    d = 1,
    e = 2
  )
)
cat(repr_ls(x, title = "A List"))
```

---

rtemis_colors *rtemis Colors*

Description

A named vector of colors used in the rtemis ecosystem, provided as hex strings.

Usage

```
rtemis_colors
```

Value

Named character vector of hex color codes.

Author(s)

EDG

Examples

```
rtemis_colors[["teal"]]
```

`strip_ansi` *Strip ANSI escape sequences from a string*

Description

Removes the SGR / CSI escapes commonly produced by `fmt()` (and by any other tool that writes coloured terminal output). Safe to call on plain input - returns it unchanged.

Usage

```
strip_ansi(x)
```

Arguments

`x`	Character: Input.

Value

Character: `x` with ANSI escapes removed.

Author(s)

EDG

Examples

```
strip_ansi(fmt("hi", col = "red"))
```

success *Success log message*

Description

Styled success message. Fires when verbosity is at least 1L.

Usage

```
success(..., verbosity = NULL, package = NULL)
```

Arguments

...	Message components, concatenated with no separator.
verbosity	Integer or NULL: Overrides get_verbosity() when supplied.
package	Character or NULL: Package name for verbosity override.

Value

Invisible NULL.

Author(s)

EDG

Examples

```
success("Job ", "abc123", " complete")
```

test_inherits *Check class of object*

Description

Check class of object

Usage

```
test_inherits(x, cl)
```

Arguments

x	Object to check
cl	Character: class to check against

Value

Logical

Author(s)

EDG

Examples

```
test_inherits("papaya", "character") # TRUE
test_inherits(c(1, 2.5, 3.2), "integer")
test_inherits(iris, "list") # FALSE, compare to is_check(iris, is.list)
```

---

unit_open_scalar *Open-unit-interval scalar S7 property*

Description

S7 property accepting a single finite double strictly in $(0, 1)$.

Usage

```
unit_open_scalar
```

Value

An S7 property object.

Author(s)

EDG

---

unit_open_vector *Open-unit-interval vector S7 property*

Description

S7 property accepting a non-empty double vector with all elements strictly in $(0, 1)$ and no NAs.

Usage

```
unit_open_vector
```

Value

An S7 property object.

Author(s)

EDG

---

warn *Warning log message*

Description

Styled non-fatal message. By default emits a soft styled message via `msg()`; with `use_warning = TRUE`, calls `warning()` so callers can catch via `tryCatch(..., warning = handler)`.

Usage

```
warn(..., use_warning = FALSE, verbosity = NULL, package = NULL)
```

Arguments

`...`	Message components, concatenated with no separator.
`use_warning`	Logical: If TRUE, signal an R `warning` condition instead of (or in addition to) writing a styled message.
`verbosity`	Integer or NULL: Overrides `get_verbosity()` when supplied.
`package`	Character or NULL: Package name for verbosity override.

Value

Invisible NULL.

Author(s)

EDG

Examples

```
warn("Disk usage at ", 92L, "%")
```

Index