# Package 'super'

December 16, 2024

**Title** Interpreted String Literals

**Version** 0.0.4

**Description** An implementation of interpreted string literals. Based on the 'glue' package by Hester & Bryan (2024) <doi:10.32614/CRAN.package.glue> but with a focus on efficiency and simplicity at a cost of flexibility.

**License** MIT + file LICENSE

**URL** https://timtaylor.github.io/super/

**BugReports** https://github.com/TimTaylor/super/issues

**Depends** R (>= 3.6)

**Suggests** glue, litedown, microbenchmark, tinytest

**VignetteBuilder** litedown

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Config/build/compilation-database** true

**NeedsCompilation** yes

**Author** Tim Taylor [aut, cre] (<https://orcid.org/0000-0002-8587-7113>), Jim Hester [aut] (<https://orcid.org/0000-0002-2739-7082>), Jennifer Bryan [aut] (<https://orcid.org/0000-0002-6983-2759>), Posit Software, PBC [cph, fnd]

**Maintainer** Tim Taylor <tim.taylor@hiddenelephants.co.uk>

**Repository** CRAN

**Date/Publication** 2024-12-16 21:30:02 UTC

## Contents

glue *Format and interpolate a string*

## Description

Inputs enclosed by braces (e.g. {name}) are looked up in the provided environment (akin to calling get()). Single braces can be escaped by doubling them up. Variables are recycled to the length of the largest one.

glue() operates on the string as is.

glut() will trim the input prior to glueing.

## Usage

```
glue(x, env = parent.frame())

glut(x, env = parent.frame())
```

## Arguments

| | |
|---|---|
| x | [character string] |
| env | [environment] |
| | Where to look up the embraced input. |
| | Can be an environment or a list-like object that will be converted in the underlying function via list2env(). |

## Value

A character object.

## See Also

glue::glue_safe() and glue::glue_data_safe() on which which this function is an evolution.

## Examples

```
name <- "Fred"
age <- 50
cat(glue("My name is {name} and my age next year is {age}"))

# glut first trims the output
anniversary <- as.Date("1991-10-12")
cat(glut("
    My name is {name},
    my age next year is {age},
    my anniversary is {anniversary}.
"))

# single braces can be inserted by doubling them
```

```
glue("My name is {name}, not {{name}}.")

# List like objects can be used in place of an environment
dat <- cbind(car = rownames(mtcars), mtcars)
glue("{car} does {mpg} mpg.", dat)
```

---

`trim` *Trim a character vector*

---

## Description

Almost identical to `glue::trim()` save a slight difference in error handling for non-character input. This function trims a character vector according to the trimming rules used by glue. These follow similar rules to Python Docstrings, with the following features:

- Leading and trailing whitespace from the first and last lines is removed.
- A uniform amount of indentation is stripped from the second line on, equal to the minimum indentation of all non-blank lines after the first.
- Lines can be continued across newlines by using `\\`.

## Usage

```
trim(x)
```

## Arguments

| | |
|---|---|
| x | [character]. |

## Value

A character vector.

## See Also

`glue::trim()`.

## Examples

```
cat(trim("
    A formatted string
    Can have multiple lines
      with additional indentation preserved
    "))

cat(trim("
  \ntrailing or leading newlines can be added explicitly\n
  "))
```

```
cat(trim("
    A formatted string \\
    can also be on a \\
    single line
    "))
```

# Index