Package ‘stringformattr’

October 14, 2022

Type    Package
Title   Dynamic String Formatting
Version 0.1.2
Author  Alexander Hoyle
Maintainer  Alexander Hoyle <alexander@alexanderhoyle.com>
Description  Pass named and unnamed character vectors into specified positions in strings. This represents an attempt to replicate some of python's string formatting.
License  MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 6.0.1
Imports stringr
NeedsCompilation no
Repository CRAN
Date/Publication 2017-12-17 06:51:48 UTC

R topics documented:

binary-string-concat .................................................. 2
format-string .......................................................... 2

Index 4
**Concatenate two strings.**

**Description**

%p% and %s% are wrappers for `paste0(..., collapse = '')` and `paste(..., collapse = '')`, respectively, which combine two character vectors.

**Usage**

```r
x %p% y
x %s% y
```

**Arguments**

- `x, y`: A character vector

**Examples**

```r
'the quick brown fox jum' %p% 'ped over the lazy dog'
```

```r
gen_sql <- function(column, table) "SELECT" %s% column %s% "FROM" %s% table
```

**Pass variables into strings**

**Description**

Pass variables into strings using pairs of curly brackets to identify points of insertion.

**Usage**

```r
string %f% args
```

**Arguments**

- `string`: A character vector
- `args`: A (possibly named) atomic vector
Examples

# order matters when not using a named vector
'the quick {} fox jumped {} the lazy {}' %f% c('brown', 'over', 'dog')

# use a named vector to insert values by referencing them in the string
gen_sql_query <- function(column, table, id){
    query <- "SELECT {col} FROM {tab} WHERE pk = {id}"
    query %f% c(col = column, tab = table, id = id)
}

gen_sql_query('LASTNAME', 'STUDENTS', '12345')

# `%f%` is vectorized
v <- c('{vegetable}', '{animal}', '{mineral}', '{animal} and {mineral}')
v %f% c(vegetable = 'carrot', animal = 'porpoise', mineral = 'salt')

# if the number of replacements is larger than the length of unnamed arguments,
# `%f%` will recycle the arguments (and give a warning)
c('{} {}', '{} {} {}', '{}') %f% c(0, 1)

# > "0 1" "0 1 0" "0"
Index

%f (format-string), 2
%p (binary-string-concat), 2
%s (binary-string-concat), 2

binary-string-concat, 2

format-string, 2