Package ‘functional’

July 22, 2025

Type Package

Title Curry, Compose, and other higher-order functions

Version 0.6

Date 2014-07-15

Author Peter Danenberg <pcd@roxygen.org>

Maintainer Peter Danenberg <pcd@roxygen.org>

Description Curry, Compose, and other higher-order functions

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-16 07:38:04

Contents

Compose — *Compose an arbitrary number of functions.*

Description

My Happy Hacking keyboard gave out during the writing of this procedure; moment of silence, please.

Usage

```
Compose(...)
```

Arguments

...	the functions to be composed

Value

A composed function

Examples

```
car <- function(list) list[[1]]
cdr <- function(list) list[2:length(list)]
cadr <- Compose(cdr, car)
stopifnot(cadr(c(1,2,3)) == 2)
```

Curry — *Pre-specify a procedures named parameters, returning a new procedure.*

Description

Thanks, Byron Ellis. https://stat.ethz.ch/pipermail/r-devel/2007-November/047318.html

Usage

```
Curry(FUN, ...)
```

Arguments

FUN	the function to be curried
...	the determining parameters

Value

A new function partially determined

Examples

```
double <- Curry(`*`, e1=2)
stopifnot(double(4) == 8)
```

`CurryL` *Lazy curry; thanks, Jamie! <https://github.com/klutometis/R-functional/issues/1>*

Description

Lazy curry; thanks, Jamie! <https://github.com/klutometis/R-functional/issues/1>

Usage

```
CurryL(FUN, ...)
```

Arguments

`FUN`	the function to be curried
`...`	the determining parameters

Examples

```
# day is not defined; thanks, Jamie Folson.
CurryL(function(...) match.call(),
       x=5,
       y=as.Date(day))(z=as.Date(day,"%Y"))
```

`Identity` *Identity function.*

Description

Is concatenation benign?

Usage

```
Identity(...)
```

Arguments

`...`	tautological arguments

Value

The tautologized arguments, concatenated

Examples

```
list.copy <- function(list)
  Reduce(Identity, list)

list <- c(1, 2, 3)
stopifnot(list.copy(list) == list)
```

`multi.argument.Compose`

Composition with multiple arguments.

Description

Thanks, Alexander Davis!

Usage

```
multi.argument.Compose(...)
```

Arguments

`...`	the functions to be composed

Value

A composed function

Examples

```
f <- function(x, y) x+y
g <- function(x) x*2
stopifnot(multi.argument.Compose(f, g)(1,1) == 4)
```

`Negate` *Negate a function; borrowed from src/library/base/R/funprog.R for pre-2.7 Rs.*

Description

Negate a function; borrowed from src/library/base/R/funprog.R for pre-2.7 Rs.

Usage

```
Negate(f)
```

Arguments

`f`	the function to be negated

Value

The negated function

Examples

```
is.even <- function(a) a%%2 == 0
is.odd <- Negate(is.even)
stopifnot(Reduce(`&&`, Map(is.odd, c(1, 3, 5))))
```

`Swap` *Thanks, Gabor; see <http://stackoverflow.com/a/23726989>: swaps the first two arguments in a function.*

Description

Thanks, Gabor; see <http://stackoverflow.com/a/23726989>: swaps the first two arguments in a function.

Usage

```
Swap(f)
```

Arguments

`f`	The function whose arguments to swap

Value

A function with swapped arguments

Index