Package 'lisp'

February 20, 2015

Type Package

Encoding UTF-8

Title List-processing à la SRFI-1

Version 0.1

Date 2010-03-02

Author Peter Danenberg <pcd@roxygen.org>

Maintainer Peter Danenberg <pcd@roxygen.org>

Description Though SRFI-1 scopes both list-processing and higher-order
programming, we'll save some list-orthogonal functions for the
`functional' package; this is freely a mixture of
implementation and API.

License GPL (>= 2)

LazyLoad yes

Suggests RUnit

Collate 'lisp.R'

Repository CRAN

Date/Publication 2012-01-12 09:24:37

NeedsCompilation no

R topics documented:

caar 2
cadar 2
caddr 3
cadr 3
car 4
cdddr 4
cddr 5
cdr 5
cdrs 6
for.each 6

is.even 7
is.nil 7
is.odd 8
last 8
nil 8
pair.fold.right 9
pairwise 9
zip 10
zip.c 10
zip.list 11
zip.with.names 11

Index **12**

`caar` *Composite* `car/cdr`

Description

Composite `car/cdr`

Usage

```
caar(list)
```

Arguments

`list`	the list from which to extract

Value

The extracted elements

`cadar` *Composite* `car/cdr`

Description

Composite `car/cdr`

Usage

```
cadar(list)
```

Arguments

`list`	the list from which to extract

Value

The extracted elements

`caddr` *Composite* `car/cdr`

Description

Composite `car/cdr`

Usage

```
caddr(list)
```

Arguments

`list`	the list from which to extract

Value

The extracted elements

`cadr` *Composite* `car/cdr`

Description

Composite `car/cdr`

Usage

```
cadr(list)
```

Arguments

`list`	the list from which to extract

Value

The extracted elements

`car` *First element of a list*

Description

First element of a list

Usage

```
car(list)
```

Arguments

`list`	the list to first

Value

The first element

`cdddr` *Composite* `car/cdr`

Description

Composite `car/cdr`

Usage

```
cdddr(list)
```

Arguments

`list`	the list from which to extract

Value

The extracted elements

`cddr` *Composite* `car/cdr`

Description

Composite `car/cdr`

Usage

```
cddr(list)
```

Arguments

`list`	the list from which to extract

Value

The extracted elements

`cdr` *Return elements after the first of a list.*

Description

Return elements after the first of a list.

Usage

```
cdr(list)
```

Arguments

`list`	the list from which to extract

Value

The elements after the first, or `nil` if only one

`cdrs` *Try to get the* `cdrs`*; otherwise, return* `nil`*.*

Description

Try to get the `cdrs`; otherwise, return `nil`.

Usage

```
cdrs(...)
```

Arguments

`...`	lists to `cdr`

Value

the `cdr` of the lists

`for.each` *Apply* `f` *to the successive elements of* `...`*.*

Description

Apply `f` to the successive elements of `...`.

Usage

```
for.each(f, ...)
```

Arguments

`f`	the function to apply, whose arity should match the cardinality of `...`
`...`	lists upon which to apply `f` successively

Value

NULL

`is.even` *Is a number even?*

Description

Is a number even?

Usage

```
is.even(a)
```

Arguments

`a`	the number to test

Value

Whether the number is even

`is.nil` *Whether a list is empty.*

Description

Whether a list is empty.

Usage

```
is.nil(list)
```

Arguments

`list`	the list to test

Value

Whether the list is empty

`is.odd` *Is a number odd?*

Description

Is a number odd?

Usage

```
is.odd(a)
```

Arguments

`a`	the number to test

Value

Whether the number is odd

`last` *Last element in a list.*

Description

Last element in a list.

Usage

```
last(list)
```

Arguments

`list`	The list to last

`nil` *The empty list*

Description

The empty list

Usage

```
nil
```

Format

list()

`pair.fold.right` *pair-fold-right from SRFI-1.*

Description

pair-fold-right from SRFI-1.

Usage

```
pair.fold.right(f, nil, ...)
```

Arguments

`f`	function to apply over the list-tails
`nil`	the default value
`...`	the lists whose tails fold over

`pairwise` *Combine a list into pairwise elements; lists should be of the same length. In case of odd numbers of members, the last will be removed.*

Description

Combine a list into pairwise elements; lists should be of the same length. In case of odd numbers of members, the last will be removed.

Usage

```
pairwise(list)
```

Arguments

`list`	the list to be pairwise decomposed

Value

A list of pairwise elements

`zip` *Zip* n *lists together into tuplets of length* n.

Description

Zip *n* lists together into tuplets of length *n*.

Usage

```
zip(zipper, ...)
```

Arguments

`zipper`	the zipping function
`...`	the lists to be zipped

Value

A list of tuplets

`zip.c` *Zip using* `c`.

Description

Zip using `c`.

Usage

```
zip.c(...)
```

Arguments

`...`	the lists to be zipped

Value

A list of tuplets

See Also

`zip`

`zip.list` *Zip using* `list`.

Description

Zip using `list`.

Usage

```
zip.list(...)
```

Arguments

`...`	the lists to be zipped

Value

A list of tuplets

See Also

`zip`

`zip.with.names` *Do a less efficient zip whilst preserving names.*

Description

Do a less efficient zip whilst preserving names.

Usage

```
zip.with.names(...)
```

Arguments

`...`	lists to be zipped whilst preserving names

Index

*Topic **datasets**
    nil, 8

caar, 2
cadar, 2
caddr, 3
cadr, 3
car, 4
cdddr, 4
cddr, 5
cdr, 5
cdrs, 6

for.each, 6

is.even, 7
is.nil, 7
is.odd, 8

last, 8

nil, 8

pair.fold.right, 9
pairwise, 9

zip, *10*, 10, *11*
zip.c, 10
zip.list, 11
zip.with.names, 11