Package ‘intoo’

March 6, 2020

Title Minimal Language-Like Extensions

Version 0.4.0

Date 2020-03-07

License GPL (>= 2)

Maintainer Abby Spurdle <spurdle.a@gmail.com>

Author Abby Spurdle [cre, aut], Emil Bode [ctb]

URL https://sites.google.com/site/spurdlea/r

Description Binary operators (%$%, %$%<-, %@% and %@%<-) to set and get object attributes and environment members, convenience functions for constructing named lists, S3 objects and self-referencing function objects, and functions for printing objects, particularly suitable for printing function objects bundled with data.

Depends methods

Suggests vectools

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-06 21:40:02 UTC

R topics documented:

1_operators *Attribute and Environment Member Operators*

Description

Binary operators to get and set object attributes and environment members.

Usage

```
#object attributes
object %$% name
object %$% name <- value

#environment members
object %@% name
object %@% name <- value
```

Arguments

`object`	An object.
`name`	An attribute's name.
`value`	An attribute's value.

Details

The attribute operators are equivalent to calling the attr and attr<- functions.

Note that currently, they need to be inside parenthesis, for subsetting and function calls.

Also note that attribute names need to be quoted inside R packages.

Examples

```
#simple object
object <- 0

#set attributes
object %$% A <- 1
object %$% B <- 2
object %$% A %$% I <- 10
object %$% A %$% J <- 20

#get attributes
object %$% A
object %$% B
object %$% A %$% I
object %$% A %$% J

#quoted version
```

```
#(for R packages)
object %$% "A"
object %$% "B"

#this doesn't work
#object %$% A [1]

#however, this does work
(object %$% A) [1]

#function object
#(using lexical scope)
g <- function ()
{   A <- 2
    B <- 2
    function (x)
        A * B + x
}
f <- g ()

#get environment members
f %@% A
f %@% B
```

`2_constructor_utilities` *Constructor Utilities*

Description

Functions for constructing named lists and S3 objects.

Usage

```
LIST (...)
EXTEND (object, class, ...)
```

Arguments

`object`	An object.
`class`	String, giving the class name. Can be a character vector, in which case, subclasses precede superclasses. Also, can be NULL or missing.
`...`	A list of name=value pairs, or objects.

Details

The LIST function constructs/returns a named list, automatically naming it's elements, if names aren't supplied.

The EXTEND function constructs/returns an S3 object, by prepending the (optional) new class to the existing classes, and then assigning attributes.

Note that it's possible to combine the with/EXTEND functions, to take attribute values from a named list.

Examples

```
object <- x <- y <- z <- 1

LIST (x, y, z=0, k=z)

EXTEND (object, "something", x, y, z=0, k=z)
```

3_self-referencing_utilities
Self-Referencing Utilities

Description

Convenience functions to get the system function, its attributes or environment.

Usage

```
THIS ()

THAT ()
THEN ()
```

Details

These functions are wrappers for standard R functions.

THIS returns the system function, that is, the current function that's being called.

THAT returns the system function's attributes and THEN returns the system function's environment. Note that THAT and THEN are contractions of TH(is) AT(tributes) and TH(is) (EN)vironment.

They need to be evaluated inside the function where they're defined.
(This is important, please refer to the examples).

It may be useful to combine the with/THAT functions.
(Also, please refer to the examples).

In general, the THEN function isn't necessary.
(Because R uses lexical scoping, and functions can be called with standard syntax, using objects contained in their environments).

Examples

```
#good
#(THIS evaluated *before* do.something)
f <- function (x)
{   this <- THIS ()
    do.something (this, x)
}

#bad
#(THIS evaluated *in* do.something)
f <- function (x)
    do.something (THIS (), x)

#combined with/THAT example
f <- function (x)
{   . <- THAT ()
    with (.,
        do.something (attr.1, attr.2, x) )
}
```

`4_print_objects` *Print Object Information*

Description

Functions for printing objects, compactly but informatively.

Usage

```
object.model (object, ...,
    value=FALSE,
    private.attributes=TRUE, public.attributes=TRUE,
    attribute.values=value, comment=FALSE, n=3)

object.summary (object,...,
    value=TRUE,
    private.attributes=FALSE, public.attributes=TRUE,
    attribute.values=value, comment=TRUE, n=6)
```

Arguments

`object`	An object.
`value`	If true, print the object’s value.
`private.attributes`	If true, print private attributes, regarded as attributes starting with a period.
`public.attributes`	If true, print public attributes, regarded as attributes not starting with a period.

`attribute.values`	If true, print the attributes' values.
`comment`	If true, print the object's comment attribute, if applicable.
`n`	Number of elements, lines or rows to print.
`...`	Ignored.

Details

These functions are the same, except for their defaults.

By default, the object.model function prints a compact representation of an object.
And by default, the object.summary function prints a semi-compact representation of an object.

Output contains non-recursive information regarding attributes.
In general, language-related attributes (such as names) are excluded from the list of attributes.

The object.summary function has output intermediate between the str and head functions, and is particularly suitable for printing functions bundled with data.

Examples

```
#function object
#with private and public attributes
f <- function (x)
{   . = THAT ()
    with (.,
        paste (A, B, x, sep=":") )
}
f <- EXTEND (f, "concat", .X=10, A="A", B="B")

#compact print
object.model (f)

#semi-compact print
object.summary (f)
```

`5_other_list_utilities` *Unpack Named Lists*

Description

Copy elements from a named list into the current environment.

Usage

```
UNPACK (x)
```

Arguments

`x`	A named list.

Details

This function is a wrapper for the list2env function.

Examples

```
f <- function ()
    list (x=1, y=2)

UNPACK (f () )
x
y
```

Index