Package: doubt (via r-universe)

November 4, 2024

Title Enable Operators Containing the '?' Symbol

Version 0.1.0

Description Overload utils::'?' to build unary and binary operators from existing functions, piping operators of different precedence, and flexible syntaxes.

Depends R (>= 3.1.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports utils, methods, unglue

Suggests testthat (>= 2.1.0), covr

Roxygen list(markdown = TRUE)

Repository https://moodymudskipper.r-universe.dev

RemoteUrl https://github.com/moodymudskipper/doubt

RemoteRef HEAD

RemoteSha 917baa8e609a8293a688179d4f03cde18f01b111

Contents

? *Modified question mark operator*

Description

? was modified to allow definition of new operators (unary, binary or n-ary). We refer to those as "dubious" operators, both as a reference to the package name and to emphasize the fact that they're not parsed as proper operators. . Standard usage as documented in `?utils::Question` still works.

Usage

```
`?`(e1, e2)
```

Arguments

e1	lhs
e2	rhs

dubious operators

Every accessible function, custom defined or base/packaged, can be called as an infix operator, for example `1:5 %%intersect? 3:7` is equivalent to `intersect(1:5, 3:7)`. In that case, `%%intersect?` will have the precedence of `%%`, which is the most intuitive, but any precedence including and below unary `+` can be used, for instance `*intersect?` will have the precedence of `*`.

Unary operators can be used to, for instance `~head? x` is the same as `head(x)`. This form can also be used with several arguments, but in this case we need to write `~head? {x ; n)` for instance, which is convenient to go to the next line without the need of a comma.

dubious pipes

We can pipe with a chosen precedence by using a dubious pipe, for instance `x + y ~saveRDS? file` will save `x + y`, not just `x`

We can pipe with a chosen precedence by using a dubious pipe, for instance `x + y ~saveRDS? file` will save `x + y`, not just `x`

dubious syntaxes

defining `"?add: ({x})({y})" <- "{x} + {y}"` will allow us to call `?add: (a)(b)` to add `a` and `b`.

Examples

```
cars +head? 2
+head? cars
+head? {
  cars
  2}
```

`register_dubious_syntaxes` *Register Dubious Syntaxes*

Description

To use a dubious syntax in a package, use this function in the definition of `.onAttach`

Usage

```
register_dubious_syntaxes(syntaxes)
```

Arguments

`syntaxes`	a character vector of the syntaxes to support

Examples

```
## Not run:
# define your syntax as you would define a normal function
`?add> {x} : {y}` <- function(x, y) x + y

# register the syntax in your .onAttach definition
.onAttach <- function(libname, pkgname) {
doubt::register_dubious_syntaxes("?add> {x} : {y}")
invisible()
}

## End(Not run)
```

Index