017_metaprogramming.Rmd

last updated 27 April 2023

Contents

{make small}

What is Symbol?

- Lisp calls it symbol
- S calls it name
- Symbols refer to R objects. The name of any R object is usually a symbol. Symbols can be created through the functions as.name and quote. They naturally appear as atoms of parsed expressions, try e.g. as.list(quote(x y)). Ref: R Language 2.1.3.1

```


## x is a double
x  <- 5

## turn x into a symbol
    s  <- as.symbol(x)
    is.symbol(s)
```

Create with as.symbol

```
      ## [1] TRUE
```

```
    typeof(s)
```

```
      ## [1] "symbol"
```

```
    storage.mode(s)
```

```
      ## [1] "symbol"
```

```
    mode(s)
```

```
      ## [1] "name"
```

```
    if (F) eval(s) #error
```

```
p  <- parse(text = "{
## x is a double
x  <- 5
## turn x into a symbol
    s  <- as.symbol(x)
    is.symbol(s)
    typeof(s)
    storage.mode(s)
    mode(s)

    if (F) eval(s) #error
}"
)
```

```


f  <- function() {}

is.symbol(f)
```

functions ???

```
## [1] FALSE
```

```
# [1] FALSE

q  <- quote(f)
is.symbol(q)
```

```
## [1] TRUE
```

```
# [1] TRUE

### FAIL
name  <- as.name(f)
```

```
## Error in as.vector(x, "symbol"): cannot coerce type 'closure' to vector of type 'symbol'
```

```
s  <- as.symbol(f)
```

```
## Error in as.vector(x, "symbol"): cannot coerce type 'closure' to vector of type 'symbol'
```

```


y  <- 5
name  <- as.name(y)
name
```

Create with as.name

```
## `5`
```

```
# `5`
is.symbol(name)
```

```
## [1] TRUE
```

```
# [1] TRUE
storage.mode(name)
```

```
## [1] "symbol"
```

```
typeof(name)
```

```
## [1] "symbol"
```

```
eval(name)  #error
```

```
## Error in eval(name): object '5' not found
```

```


z  <- 5
name  <- quote(z)
# z
typeof(name)
```

Create using quote

```
## [1] "symbol"
```

```
# [1] "symbol"

eval(name)
```

```
## [1] 5
```

```
# [1] 5
```

```


y  <- 5
name  <- as.name(y)

sapply(list(is.symbol,
            typeof,
            mode,
            storage.mode),
       do.call, list(quote(name)))
```

Future shortcut

```
## [1] "TRUE"   "symbol" "name"   "symbol"
```

```
# [1] "TRUE"   "symbol" "name"   "symbol"
```

```
knitr::knitr_exit()
```

```
## Error: 'knitr_exit' is not an exported object from 'namespace:knitr'
```

rlang::expr | rlang::enexpr

```
rlang::expr(a+b+10)
```

```
## a + b + 10
```

```
expr(some_fun(a))
```

```
## Error in expr(some_fun(a)): could not find function "expr"
```

```
capture_it  <- function(x) expr(x)
capture_it(a+b+10)  #x
```

```
## Error in expr(x): could not find function "expr"
```

```
capture_it2  <- function(x) enexpr(x)
capture_it2(a+b+10) # returns a+b+10
```

```
## Error in enexpr(x): could not find function "enexpr"
```

```
capture_it2(caller_env())
```

```
## Error in enexpr(x): could not find function "enexpr"
```

```
# STUDY f
f  <- expr(f(x=1, y=2))
```

```
## Error in expr(f(x = 1, y = 2)): could not find function "expr"
```

```
typeof(f)
```

```
## [1] "closure"
```

```
class(f)
```

```
## [1] "function"
```

```
attributes(f)
```

```
## $srcref
## function() {}
```

```
f   # just its value
```

```
## function() {}
```

```
f[[1]]  # its name?
```

```
## Error in f[[1]]: object of type 'closure' is not subsettable
```

```
f[[2]]
```

```
## Error in f[[2]]: object of type 'closure' is not subsettable
```

```
f[[3]]
```

```
## Error in f[[3]]: object of type 'closure' is not subsettable
```

```
as_list(f)
```

```
## Error in as_list(f): could not find function "as_list"
```

```
# f[[4]]
```

lobstr::ast

```
lobstr::ast(x+y)
```

```
##  `+`
##  x
##  y
```

```
lobstr::ast(f(x,y))
```

```
##  f
##  x
##  y
```

```
lobstr::ast(f(x, g(y)))
```

```
## █─f
## ├─x
## └─█─g
##   └─y
```

```
lobstr::ast(f(x, g(h(z)), h(z)))
```

```
## █─f
## ├─x
## ├─█─g
## │ └─█─h
## │   └─z
## └─█─h
##   └─z
```

```
ast( ff  <- expr(f(x,y)))
```

```
## Error in ast(ff <- expr(f(x, y))): could not find function "ast"
```

rlang::call2() create function (given a tree)

```
# must quote
rlang::call2("f", "y", "z")
```

```
## f("y", "z")
```

```
rlang::call2("f", 1, 2  )
```

```
## f(1, 2)
```

```
# inverse!
ast(call2("f", 1, 2 ))
```

```
## Error in ast(call2("f", 1, 2)): could not find function "ast"
```

```
#
# error, nice try
# call2(ast(f(x,y)))
```

alternative to call2 to develop code, use !! unquote

```
# quote (makes expression?)
xx  <- expr(x+x)
```

```
## Error in expr(x + x): could not find function "expr"
```

```
yy   <- expr(y + y)
```

```
## Error in expr(y + y): could not find function "expr"
```

```
# create R code: (uquote expression?)
expr(!!xx/!!yy)
```

```
## Error in expr(!!xx/!!yy): could not find function "expr"
```

```
# inverse!
ff  <- expr(f(x,y))
```

```
## Error in expr(f(x, y)): could not find function "expr"
```

```
expr(!!ff)
```

```
## Error in expr(!!ff): could not find function "expr"
```

```
# create code

f  <- function(var){
    var  <- enexpr(var)  # capture
    expr(3*(!!var))  # do something, and capture this.
}
f(x)
```

```
## Error in enexpr(var): could not find function "enexpr"
```

```
f(x+y)
```

```
## Error in enexpr(var): could not find function "enexpr"
```

```
f(f(x))
```

```
## Error in enexpr(var): could not find function "enexpr"
```

```
f(1+3+x)
```

```
## Error in enexpr(var): could not find function "enexpr"
```

```
# Hadley's example:
cv <- function(var) {
  var <- enexpr(var)
  expr(sd(!!var) / mean(!!var))
}
```

```
file <- "017_metaprogramming.Rmd"
file  <- normalizePath(file)
file


rmarkdown::render( file,
                    output_format = "pdf_document",
                    output_dir="~/Downloads/print_and_delete")
)
```

```
## Error: <text>:9:1: unexpected ')'
## 8:                     output_dir="~/Downloads/print_and_delete")
## 9: )
##    ^
```