

Package ‘Q7’

October 12, 2022

Title Types and Features for Object Oriented Programming

Version 0.1.0

Description Construct message-passing style objects with types and features. Q7 types uses composition instead of inheritance in creating derived types, allowing defining any code segment as feature and associating any feature to any object. Compared to R6, Q7 is simpler and more flexible, and is more friendly in syntax.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends magrittr

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Siqu Zhang [aut, cre] (<<https://orcid.org/0000-0003-2617-5776>>),
Bayer HealthCare [fnd]

Maintainer Siqu Zhang <iqis.gnahz@gmail.com>

Repository CRAN

Date/Publication 2020-11-25 10:00:06 UTC

R topics documented:

clone	2
clone.Q7instance	2
extend	3
feature	4
feature_generic	5
implement	6
is	6
list2inst	7
localize	8
merge	8
type	9

Index**11**

clone	<i>Clone</i>
-------	--------------

Description

Clone

Usage

clone(...)

Arguments

... dot-dot-dot

clone.Q7instance	<i>Clone an Instance</i>
------------------	--------------------------

Description

Clone an Instance

Usage

```
## S3 method for class 'Q7instance'
clone(inst, deep = TRUE, ...)
```

Arguments

inst	Q7 object instance
deep	to copy nested object instances recursively; Boolean
...	dot-dot-dot

Value

Q7 object instance

Examples

```
Type1 <- type(function(num){
  print_num <- function(){
    base::print(num)
  }
})
myType1 <- Type1(1)
myType1$print_num()
myType1_clone <- clone(myType1)
myType1_clone$print_num()
```

extend	<i>Extend a Type upon a (Proto)type</i>
--------	---

Description

Used only inside a type definition

Usage

```
extend(prototype)
```

Arguments

prototype Q7type; function

Value

localized Q7type; function

Examples

```
Type1 <- type(function(arg1){
  val1 <- arg1
  get_val1 <- function(){
    val1
  }
}, "Type1")

Type2 <- type(function(arg1, arg2){
  extend(Type1)(arg1)
  val2 <- arg2
  get_val2 <- function(){
    val2
  }
}, "Type2")
```

```
myType2 <- Type2("foo", "bar")

myType2$get_val1()
myType2$get_val2()
```

feature

Create an Object Feature

Description

Create an Object Feature

Usage

```
feature(expr)
```

Arguments

expr expression

Value

a Q7 feature

Examples

```
Type1 <- type(function(num){})

hasMagic <- feature({
  change_number <- function(){
    num + 1
  }
})

myType1 <- Type1(1) %>% hasMagic()
myType1$change_number()

# Use S3 method dispatch for different behaviors
hasMagic <- feature_generic(s3 = "hasMagic")

hasMagic.Type1 <- feature({
  change_number <- function(){
    num + 1
  }
})
```

```
hasMagic.Type2 <- feature({
  change_number <- function(){
    num - 1
  }
})

Type1 <- type(function(num){},
              s3 = "Type1") %>%
  hasMagic()

Type2 <- type(function(num){},
              s3 = "Type2") %>%
  hasMagic()

myType1 <- Type1(1)
myType1$change_number()

myType2 <- Type2(1)
myType2$change_number()
```

feature_generic

Create a Generic Feature

Description

Use this function when you need to create more than one methods for Q7 types with different S3 classes. The s3 field and the feature's name should be the same.

Usage

```
feature_generic(s3, ...)
```

Arguments

s3	S3 Class of the feature
...	dot-dot-dot

Value

a generic Q7 feature

See Also

[feature](#)

implement *Implement any Feature for an Object*

Description

Implement any Feature for an Object

Usage

```
implement(obj, feat)
```

Arguments

obj	Q7 object (type or instance)
feat	Q7 feature or expression

Value

Q7 object (type or instance)

Examples

```
Type1 <- type(function(num){})  
  
myType1 <- Type1(1) %>% implement({  
  change_number <- function(){  
    num + 1  
  }  
})  
  
myType1$change_number()
```

is *Is it a Q7 Type, Instance or Feature?*

Description

Is it a Q7 Type, Instance or Feature?

Usage

```
is_type(x)  
  
is_instance(x)  
  
is_feature(x)
```

Arguments

x object

Value

Boolean

list2inst *Build a Q7 Object Instance from a List*

Description

Build a Q7 Object Instance from a List

Usage

```
list2inst(x, s3 = "default", parent = parent.frame(), ...)
```

Arguments

x list
s3 S3 class name of the instance
parent parent environment of the instance
... dot-dot-dot

Value

Q7 object instance

Examples

```
my_data <- list(a = 1,  
              add_to_a = function(value){  
                  .my$a <- a + value  
              })  
  
myDataObject <- list2inst(my_data)  
  
myDataObject$a  
myDataObject$add_to_a(20)  
myDataObject$a
```

localize	<i>Make a Localized Copy of a Q7 Type or Instance</i>
----------	---

Description

Make a Localized Copy of a Q7 Type or Instance

Usage

```
localize(obj, envir = parent.frame())
```

Arguments

obj	Q7 type or instance
envir	environment

Value

function

merge	<i>Merge all Members of Two Instances</i>
-------	---

Description

All public and private members of instance 2 will be copied to instance 1, overwriting any of the same names.

Usage

```
merge(inst1, inst2)
```

Arguments

inst1	instance to move members to
inst2	instance to move members from

Value

Q7 instance, with environment identity of inst1 and members from both instances.

Examples

```

Screamer <- type(function(words){
  scream <- function(){
    paste0(paste(words,
                 collapse = " "),
           "!!!")
  }
})

Whisperer <- type(function(words){
  whisper <- function(){
    paste0("shhhhhh....",
          paste(words,
                collapse = " "),
          "...")
  }
})

p1 <- Screamer("I love you")
p1$scream()

p2 <- Whisperer("My parents came back")
p2$whisper()

p1 <- p1 %>% merge(p2)

# note the the "word" for both methods became that of p2
p1$whisper()
p1$scream()

```

type	<i>Create a Q7 Type</i>
------	-------------------------

Description

Create a Q7 Type

Usage

```

type(x = function() {
}, s3 = "Q7default")

```

Arguments

x	function or expression; becomes the definition of the object
s3	S3 class for the object; necessary when using S3 generic functions

Value

Q7 type; function

Examples

```
Adder <- type(function(num1, num2){  
  add_nums <- function(){  
    num1 + num2  
  }  
})
```

```
myAdder <- Adder(1, 2)  
myAdder$add_nums()
```

Index

clone, 2
clone.Q7instance, 2

extend, 3

feature, 4, 5
feature_generic, 5

implement, 6
is, 6
is_feature (is), 6
is_instance (is), 6
is_type (is), 6

list2inst, 7
localize, 8

merge, 8

type, 9