

# Package ‘mod’

May 9, 2026

**Type** Package

**Title** Lightweight and Self-Contained Modules for Code Organization

**Version** 0.1.3

**Description** Creates modules inline or from a file. Modules can contain any R object and be nested. Each module have their own scope and package ``search path" that does not interfere with one another or the user's working environment.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**URL** <https://github.com/iqis/mod>

**BugReports** <https://github.com/iqis/mod/issues>

**Suggests** testthat (>= 2.1.0), covr

**NeedsCompilation** no

**Author** Siqu Zhang [aut, cre]

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

**Repository** CRAN

**Date/Publication** 2019-08-23 10:40:02 UTC

## Contents

as_module . . . . .	2
drop . . . . .	2
is_module . . . . .	3
is_thing . . . . .	4
module . . . . .	4
name . . . . .	5
print.module . . . . .	6
provide . . . . .	6
refer . . . . .	7

require . . . . .	8
thing . . . . .	9
use . . . . .	9
[.thing . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

---

as_module	<i>Use a Package as if a Module</i>
-----------	-------------------------------------

---

### Description

Use a Package as if a Module

### Usage

```
as_module(package)
```

### Arguments

package            name of a package; character

### Value

a module that contains a package's exported objects

### Examples

```
tcltk <- as_module("tcltk")
ls(tcltk)

tcltk$is.tclObj(NULL)
```

---

drop	<i>Drop a Module</i>
------	----------------------

---

### Description

Detach a named module from the search path. If no arguments is supplied, detach the most recently attached module.

### Usage

```
drop(name)
```

**Arguments**

name                    name of the module to exit from; character

**Value**

TRUE if successful; invisible

**Examples**

```
use(mod::ule({
  a <- 1
}), as = "my_module")

use(mod::ule({
  b <- 2
}), as = "my_other_module")

search()

# by name
drop("my_module")

# and at the head position
drop()

search()
```

---

is\_module

*Test if an Object is a Module*

---

**Description**

Test if an Object is a Module

**Usage**

```
is_module(x)
```

**Arguments**

x                    An object

**Value**

TRUE if the object is a module, FALSE otherwise

---

is_thing	<i>Test if an Object is a Thing</i>
----------	-------------------------------------

---

**Description**

Test if an Object is a Thing

**Usage**

```
is_thing(x)
```

**Arguments**

x	an object
---	-----------

**Value**

TRUE if the object is a thing, FALSE otherwise

---

module	<i>Make a Module</i>
--------	----------------------

---

**Description**

Institute a module object inline or from a file. `mod::ule()` is a useful shorthand for `module()` when this package is not attached.

**Usage**

```
module(..., parent = parent.frame(), lock = TRUE,
  expose_private = FALSE)
```

```
ule(..., parent = parent.frame(), lock = TRUE,
  expose_private = FALSE)
```

```
acquire(module, parent = baseenv(), lock = TRUE,
  expose_private = FALSE)
```

**Arguments**

...	module expression
parent	the enclosing environment
lock	lock the environment; logical
expose_private	expose the private environment as <code>'..private..'</code> ; logical
module	module object, or path to a module file

**Value**

an environment of class module containing defined objects

**Examples**

```
# from file
module_path <- system.file("misc", "example_module.R", package = "mod")
example_module <- acquire(module_path)

example_module$e(123)

# inline
my_module <- mod::ule({
  a <- 1
  .a <- 2
  f <- function(){.a}
})

my_module$a
my_module$f
```

---

name	<i>Name a Module</i>
------	----------------------

---

**Description**

Name a Module

**Usage**

```
name(name)
```

**Arguments**

name            the name of the module; character

**Value**

the input

---

 print.module

*Print a Module*


---

### Description

Print a Module

### Usage

```
## S3 method for class 'module'
print(x, ...)
```

### Arguments

x	an object
...	dot-dot-dot, ignored

### Value

the object itself; invisible

---

 provide

*Provide Objects from a Module*


---

### Description

Can only be used inside a module expression. If this function is used, only the names included as argument are public. If not used, every name in the module will be public.

### Usage

```
provide(...)
```

### Arguments

...	name of any object to be accessible by user; name or character
-----	--

### Value

NULL; invisible

**Examples**

```

mod_a <- mod::ule({
  # names included in provide() are public, however...
  mod:::provide(var,.var, ..var)
  # It is suggested to omit mod::: when using
  var <- 1
  .var <- 2
  ..var <- 3 # objects denoted by .. prefix are always private.
  another_var <- 4 # objects not included in provide() are also private.
})

mod_b <- mod::ule({
  # if no call to provide(), all objects are public, except...
  var <- 1
  .var <- 2
  ..var <- 3 # objects denoted by .. prefix are always private.
})

ls(mod_a)
ls(mod_b)

```

---

refer

*Copy Bindings from a Module to Another*


---

**Description**

Can only be used inside a module expression. Makes reference to objects from one module, with specified filters.

**Usage**

```
refer(..., include = c(), exclude = c(), prefix = "", sep = ".")
```

**Arguments**

...	names of modules; dot-dot-dot
include	names to include; character
exclude	names to excludde; character
prefix	prefix to names; character
sep	separator between prefix and names; character

**Value**

NULL; invisible

**Examples**

```

mod_a <- mod::ule(number <- 1)
mod_b <- mod::ule(number <- 2)

mod_c <- mod::ule({
  mod::refer(mod_a, mod_b, prefix = .)
  # It is suggested to omit mod:: when using
  number <- mod_a.number + mod_b.number
})

mod_c$number

```

---

require	<i>Load/Attach Package to Local Search Path</i>
---------	---

---

**Description**

Can only be used in a module expression. Emulates the effect of `base::require()` in its containing module, making functions and their chain of environment available. Masks `base::require()` inside a module context.

**Usage**

```
require(package)
```

**Arguments**

package            name of the package; name or character

**Value**

NULL; invisible

**Examples**

```

mod_tcl <- mod::ule({
  mod::require(tcltk)
  # It is suggested to omit mod:: when using
  f <- tcl
})

identical(mod_tcl$f, tcltk::tcl)

```

---

thing	<i>Make a Thing</i>
-------	---------------------

---

**Description**

A "thing" is a special object made based on a module. Contains an active binding, defined with the 'dot' argument.

**Usage**

```
thing(..., dot, parent = parent.frame(), lock = TRUE,
      expose_private = FALSE)
```

**Arguments**

...	module expression
dot	function expression used for active binding to '.'
parent	the enclosing environment
lock	lock the environment; logical
expose_private	expose the private environment as '..private..'; logical

**Value**

a module containing an active binding

**Examples**

```
my_thing <- mod::thing({
  a <- 1
}, dot = function() a)

my_thing$.

my_thing[]
```

---

use	<i>Load/Attach a Module to the Search Path</i>
-----	--

---

**Description**

Load/Attach a Module to the Search Path

**Usage**

```
use(module, as, parent = baseenv(), lock = TRUE,
     expose_private = FALSE)
```

**Arguments**

module	module object, or path to a module file
as	name when attached to search; character
parent	the enclosing environment
lock	lock the environment; logical
expose_private	expose the private environment as ‘..private.’; logical

**Value**

TRUE if successful; invisible

**Examples**

```
module_path <- system.file("misc", "example_module.R", package = "mod")
example_module <- acquire(module_path)

# Attach module object to search path
use(example_module)
# or directly from file
use(module_path, "example_module")
```

---

[.thing

*Invoke the Active Binding in a Thing*

---

**Description**

Invoke the Active Binding in a Thing

**Usage**

```
## S3 method for class 'thing'
x[...]
```

**Arguments**

x	a thing
...	dot-dot-dot, ignored

**Value**

the return value of the active binding in a thing

# Index

[.thing, 10

acquire (module), 4

as\_module, 2

drop, 2

is\_module, 3

is\_thing, 4

module, 4

name, 5

print.module, 6

provide, 6

refer, 7

require, 8

thing, 9

ule (module), 4

use, 9