

# Package: teal.code (via r-universe)

August 26, 2024

**Type** Package

**Title** Code Storage and Execution Class for 'teal' Applications

**Version** 0.5.0

**Date** 2024-01-10

**Description** Introduction of 'qenv' S4 class, that facilitates code execution and reproducibility in 'teal' applications.

**License** Apache License 2.0

**URL** <https://insightsengineering.github.io/teal.code/>,  
<https://github.com/insightsengineering/teal.code>

**BugReports** <https://github.com/insightsengineering/teal.code/issues>

**Depends** methods, R (>= 4.0)

**Imports** checkmate (>= 2.1.0), grDevices, lifecycle (>= 0.2.0), rlang (>= 1.1.0)

**Suggests** cli (>= 3.4.0), knitr (>= 1.42), magrittr (>= 1.5), rmarkdown (>= 2.19), shiny (>= 1.6.0), testthat (>= 3.1.5)

**VignetteBuilder** knitr

**RdMacros** lifecycle

**Config/Needs/verdepcheck** mllg/checkmate, r-lib/lifecycle, r-lib/rlang, rstudio/shiny, r-lib/styler, r-lib/cli, yihui/knitr, tidyverse/magrittr, rstudio/rmarkdown, r-lib/testthat

**Config/Needs/website** insightsengineering/nesttemplate

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Collate** 'qenv-class.R' 'qenv-errors.R' 'qenv-concat.R'  
'qenv-constructor.R' 'qenv-eval\_code.R' 'qenv-get\_code.R'  
'qenv-get\_env.R' 'qenv-get\_var.R' 'qenv-get\_warnings.R'  
'qenv-join.R' 'qenv-show.R' 'qenv-within.R'  
'teal.code-package.R' 'utils.R'

**Repository** <https://insightsengineering.r-universe.dev>

**RemoteUrl** <https://github.com/insightsengineering/teal.code>

**RemoteRef** v0.5.0

**RemoteSha** 74d490ed2c34346d8b8cdf683e81b54b055e4a8f

## Contents

|                            |           |
|----------------------------|-----------|
| concat . . . . .           | 2         |
| dev_suppress . . . . .     | 3         |
| get_env . . . . .          | 4         |
| get_var . . . . .          | 4         |
| get_warnings . . . . .     | 5         |
| join . . . . .             | 6         |
| qenv . . . . .             | 8         |
| show,qenv-method . . . . . | 11        |
| <b>Index</b>               | <b>12</b> |

---

|        |                                     |
|--------|-------------------------------------|
| concat | <i>Concatenate two qenv objects</i> |
|--------|-------------------------------------|

---

### Description

Combine two qenv objects by simple concatenate their environments and the code.

### Usage

```
concat(x, y)
```

### Arguments

|   |        |
|---|--------|
| x | (qenv) |
| y | (qenv) |

### Details

We recommend to use the `join` method to have a stricter control in case `x` and `y` contain duplicated bindings and code. RHS argument content has priority over the LHS one.

### Value

qenv object.

## Examples

```
q <- qenv()
q1 <- eval_code(q, expression(iris1 <- iris, mtcars1 <- mtcars))
q2 <- q1
q1 <- eval_code(q1, "iris2 <- iris")
q2 <- eval_code(q2, "mtcars2 <- mtcars")
qq <- concat(q1, q2)
get_code(qq)
```

---

|              |  |
|--------------|--|
| dev_suppress | <i>Suppresses plot display in the IDE by opening a PDF graphics device</i> |
|--------------|--|

---

## Description

This function opens a PDF graphics device using `grDevices::pdf` to suppress the plot display in the IDE. The purpose of this function is to avoid opening graphic devices directly in the IDE.

## Usage

```
dev_suppress(x)
```

## Arguments

x lazy binding which generates the plot(s)

## Details

The function uses `base::on.exit` to ensure that the PDF graphics device is closed (using `grDevices::dev.off`) when the function exits, regardless of whether it exits normally or due to an error. This is necessary to clean up the graphics device properly and avoid any potential issues.

## Value

No return value, called for side effects.

## Examples

```
dev_suppress(plot(1:10))
```

---

|         |  |
|---------|--|
| get_env | <i>Access environment included in qenv</i> |
|---------|--|

---

**Description**

The access of environment included in `qenv@env` allows to e.g. list object names included in `qenv@env` slot.

**Usage**

```
get_env(object)
```

**Arguments**

object            (qenv)

**Value**

An environment stored in `qenv@env` slot.

**Examples**

```
q <- qenv()
q1 <- within(q, {
  a <- 5
  b <- data.frame(x = 1:10)
})
get_env(q1)
ls(get_env(q1))
```

---

|         |                             |
|---------|-----------------------------|
| get_var | <i>Get object from qenv</i> |
|---------|-----------------------------|

---

**Description**

Retrieve variables from the `qenv` environment.

**Usage**

```
get_var(object, var)
```

```
## S4 method for signature 'qenv'
x[[i]]
```

**Arguments**

object, x            (qenv)  
var, i                (character(1)) variable name.

**Value**

The value of required variable (var) within qenv object.

**Examples**

```
q <- qenv()
q1 <- eval_code(q, code = quote(a <- 1))
q2 <- eval_code(q1, code = "b <- a")
get_var(q2, "b")
q2[["b"]]
```

---

|              |                                      |
|--------------|--------------------------------------|
| get_warnings | <i>Get warnings from qenv object</i> |
|--------------|--------------------------------------|

---

**Description**

Retrieve all warnings raised during code evaluation in a qenv.

**Usage**

```
get_warnings(object)
```

**Arguments**

object                (qenv)

**Value**

character containing warning information or NULL if no warnings.

**Examples**

```
data_q <- qenv()
data_q <- eval_code(data_q, "iris_data <- iris")
warning_qenv <- eval_code(
  data_q,
  bquote(p <- hist(iris_data[, .("Sepal.Length")], ff = ""))
)
cat(get_warnings(warning_qenv))
```

---

|      |                          |
|------|--------------------------|
| join | <i>Join qenv objects</i> |
|------|--------------------------|

---

### Description

Checks and merges two qenv objects into one qenv object.

### Usage

```
join(x, y)
```

### Arguments

|   |        |
|---|--------|
| x | (qenv) |
| y | (qenv) |

### Details

Any common code at the start of the qenvs is only placed once at the start of the joined qenv. This allows consistent behavior when joining qenvs which share a common ancestor. See below for an example.

There are some situations where `join()` cannot be properly performed, such as these three scenarios:

1. Both qenv objects contain an object of the same name but are not identical.

Example:

```
x <- eval_code(qenv(), expression(mtcars1 <- mtcars))
y <- eval_code(qenv(), expression(mtcars1 <- mtcars['wt']))

z <- join(x, y)
# Error message will occur
```

In this example, `mtcars1` object exists in both `x` and `y` objects but the content are not identical. `mtcars1` in the `x` qenv object has more columns than `mtcars1` in the `y` qenv object (only has one column).

2. `join()` will look for identical `@id` values in both qenv objects. The index position of these `@ids` must be the same to determine the evaluation order. Otherwise, `join()` will throw an error message.

Example:

```
common_q <- eval_code(qenv(), expression(v <- 1))
x <- eval_code(
  common_q,
  "x <- v"
)
y <- eval_code(
```

```

    common_q,
    "y <- v"
  )
z <- eval_code(
  y,
  "z <- v"
)
q <- join(x, y)
join_q <- join(q, z)
# Error message will occur

# Check the order of evaluation based on the id slot
shared_ids <- intersect(q@id, z@id)
match(shared_ids, q@id) # Output: 1 3
match(shared_ids, z@id) # Output: 1 2

```

The error occurs because the index position of identical @id between the two objects is not the same.

3. The usage of temporary variable in the code expression could cause join() to fail.

Example:

```

common_q <- qenv()
x <- eval_code(
  common_q,
  "x <- numeric(0)
  for (i in 1:2) {
    x <- c(x, i)
  }"
)
y <- eval_code(
  common_q,
  "y <- numeric(0)
  for (i in 1:3) {
    y <- c(y, i)
  }"
)
q <- join(x,y)
# Error message will occur

# Check the value of temporary variable i in both objects
x@env$i # Output: 2
y@env$i # Output: 3

```

join() fails to provide a proper result because of the temporary variable i exists in both objects but has different value. To fix this, we can set i <- NULL in the code expression for both objects.

```

common_q <- qenv()
x <- eval_code(
  common_q,

```

```

    "x <- numeric(0)
    for (i in 1:2) {
      x <- c(x, i)
    }
    # dummy i variable to fix it
    i <- NULL"
  )
y <- eval_code(
  common_q,
  "y <- numeric(0)
  for (i in 1:3) {
    y <- c(y, i)
  }
  # dummy i variable to fix it
  i <- NULL"
)
q <- join(x,y)

```

**Value**

qenv object.

**Examples**

```

q <- qenv()
q1 <- eval_code(q, expression(iris1 <- iris, mtcars1 <- mtcars))
q2 <- q1
q1 <- eval_code(q1, "iris2 <- iris")
q2 <- eval_code(q2, "mtcars2 <- mtcars")
qq <- join(q1, q2)
get_code(qq)

common_q <- eval_code(q, quote(x <- 1))
y_q <- eval_code(common_q, quote(y <- x * 2))
z_q <- eval_code(common_q, quote(z <- x * 3))
join_q <- join(y_q, z_q)
# get_code only has "x <- 1" occurring once
get_code(join_q)

```

---

qenv

*Code tracking with qenv object*


---

**Description**

**[Stable]**

Create a qenv object and evaluate code in it to track code history.



**Usage**

```

qenv()

new_qenv(env = new.env(parent = parent.env(.GlobalEnv)), code = character())

eval_code(object, code)

get_code(object, deparse = TRUE, ...)

## S3 method for class 'qenv'
within(data, expr, ...)

```

**Arguments**

|         |  |
|---------|--|
| env     | <b>[Deprecated]</b> (environment) Environment being a result of the code evaluation. |
| code    | (character or language) code to evaluate. If character, comments are retained.       |
| object  | (qenv)   |
| deparse | (logical(1)) flag specifying whether to return code as character or expression.      |
| ...     | see Details  |
| data    | (qenv)   |
| expr    | (expression) to evaluate. Must be inline code, see Using language objects...         |

**Details**

`qenv()` instantiates a `qenv` with an empty environment. Any changes must be made by evaluating code in it with `eval_code` or `within`, thereby ensuring reproducibility.

`new_qenv()` (**[Deprecated]** and not recommended) can instantiate a `qenv` object with data in the environment and code registered.

`eval_code` evaluates given code in the `qenv` environment and appends it to the code slot. Thus, if the `qenv` had been instantiated empty, contents of the environment are always a result of the stored code.

`get_code` retrieves the code stored in the `qenv`. ... passes arguments to methods.

`within` is a convenience function for evaluating inline code inside the environment of a `qenv`. It is a method for the base generic that wraps `eval_code` to provide a simplified way of passing code. `within` accepts only inline expressions (both simple and compound) and allows for injecting values into `expr` through the ... argument: as `name:value` pairs are passed to ..., `name` in `expr` will be replaced with `value`.

**Value**

`qenv` and `new_qenv` return a `qenv` object.

`eval_code` returns a `qenv` object with `expr` evaluated or `qenv.error` if evaluation fails.

`get_code` returns the traced code (from `@code` slot) in the form specified by `deparse`.

`within` returns a `qenv` object with `expr` evaluated or `qenv.error` if evaluation fails.

### Using language objects with within

Passing language objects to `expr` is generally not intended but can be achieved with `do.call`. Only single expressions will work and substitution is not available. See examples.

### See Also

[base::within\(\)](#), [get\\_var\(\)](#), [get\\_env\(\)](#), [get\\_warnings\(\)](#), [join\(\)](#), [concat\(\)](#)

### Examples

```
# create empty qenv
qenv()

# create qenv with data and code (deprecated)
new_qenv(env = list2env(list(a = 1)), code = quote(a <- 1))
new_qenv(env = list2env(list(a = 1)), code = parse(text = "a <- 1", keep.source = TRUE))
new_qenv(env = list2env(list(a = 1)), code = "a <- 1")

# evaluate code in qenv
q <- qenv()
q <- eval_code(q, "a <- 1")
q <- eval_code(q, quote(library(checkmate)))
q <- eval_code(q, expression(assert_number(a)))

# retrieve code
get_code(q)
get_code(q, deparse = FALSE)

# evaluate code using within
q <- qenv()
q <- within(q, {
  i <- iris
})
q <- within(q, {
  m <- mtcars
  f <- faithful
})
q
get_code(q)

# inject values into code
q <- qenv()
q <- within(q, i <- iris)
within(q, print(dim(subset(i, Species == "virginica"))))
within(q, print(dim(subset(i, Species == species)))) # fails
within(q, print(dim(subset(i, Species == species))), species = "versicolor")
species_external <- "versicolor"
within(q, print(dim(subset(i, Species == species))), species = species_external)

# pass language objects
expr <- expression(i <- iris, m <- mtcars)
within(q, expr) # fails
```

```
do.call(within, list(q, expr))

exprlist <- list(expression(i <- iris), expression(m <- mtcars))
within(q, exprlist) # fails
do.call(within, list(q, do.call(c, exprlist)))
```

---

|                  |                            |
|------------------|----------------------------|
| show,qenv-method | <i>Display qenv object</i> |
|------------------|----------------------------|

---

### Description

Prints the qenv object.

### Usage

```
## S4 method for signature 'qenv'
show(object)
```

### Arguments

object (qenv)

### Value

object, invisibly.

### Examples

```
q <- qenv()
q1 <- eval_code(q, expression(a <- 5, b <- data.frame(x = 1:10)))
q1
```

# Index

`[[`, `qenv`-method (`get_var`), 4

`base::on.exit`, 3  
`base::within()`, 10

`concat`, 2  
`concat()`, 10  
`concat`, `qenv`, `qenv`-method (`concat`), 2  
`concat`, `qenv`, `qenv.error`-method (`concat`), 2  
`concat`, `qenv.error`, ANY-method (`concat`), 2

`dev_suppress`, 3

`eval_code` (`qenv`), 8  
`eval_code`, `qenv`, character-method (`qenv`), 8  
`eval_code`, `qenv`, expression-method (`qenv`), 8  
`eval_code`, `qenv`, language-method (`qenv`), 8  
`eval_code`, `qenv.error`, ANY-method (`qenv`), 8

`get_code` (`qenv`), 8  
`get_code`, `qenv`-method (`qenv`), 8  
`get_code`, `qenv.error`-method (`qenv`), 8  
`get_env`, 4  
`get_env()`, 10  
`get_env`, `qenv`-method (`get_env`), 4  
`get_env`, `qenv.error`-method (`get_env`), 4  
`get_var`, 4  
`get_var()`, 10  
`get_var`, `qenv`, character-method (`get_var`), 4  
`get_var`, `qenv.error`, ANY-method (`get_var`), 4

`get_warnings`, 5  
`get_warnings()`, 10  
`get_warnings`, NULL-method (`get_warnings`), 5

`get_warnings`, `qenv`-method (`get_warnings`), 5  
`get_warnings`, `qenv.error`-method (`get_warnings`), 5  
`grDevices::dev.off`, 3  
`grDevices::pdf`, 3

`join`, 6  
`join()`, 10  
`join`, `qenv`, `qenv`-method (`join`), 6  
`join`, `qenv`, `qenv.error`-method (`join`), 6  
`join`, `qenv.error`, ANY-method (`join`), 6

`new_qenv` (`qenv`), 8  
`new_qenv`, environment, character-method (`qenv`), 8  
`new_qenv`, environment, expression-method (`qenv`), 8  
`new_qenv`, environment, language-method (`qenv`), 8  
`new_qenv`, environment, missing-method (`qenv`), 8  
`new_qenv`, missing, missing-method (`qenv`), 8

`qenv`, 8

`show`, `qenv`-method, 11  
`show-qenv` (`show`, `qenv`-method), 11

`within.qenv` (`qenv`), 8