

Package: ktools (via r-universe)

October 8, 2026

Title Tools for Tracing, Shiny Routing, and JavaScript Dependencies

Version 0.9.10

Description Provides utility tools designed to homogenize development patterns across projects. This includes specialized tracing utilities, mechanisms for generating unique identifiers, JavaScript dependency management, targeting unique input from multiple HTML tags within Shiny apps and more.

License GPL (>= 3)

URL <https://thekangaroofactory.github.io/ktools/>

BugReports <https://github.com/thekangaroofactory/ktools/issues>

Depends R (>= 4.1.0)

Imports lifecycle, dplyr, lubridate, htmltools, shiny, jsonlite, DT

Encoding UTF-8

Roxygen list(markdown = TRUE)

Suggests testthat (>= 3.0.0), quarto

VignetteBuilder quarto

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

Repository <https://thekangaroofactory.r-universe.dev>

Date/Publication 2026-10-08 10:45:52 UTC

RemoteUrl <https://github.com/thekangaroofactory/ktools>

RemoteRef HEAD

RemoteSha 698654536c8e4bf3c0830f51b3f586916bdb70c3

Contents

action_link	2
catl	4
copy_template	5
count_lines	6
create_data	7
date_range	7
filter_date	8
getNsFunction	9
getTimestamp	9
has_date	10
incat	11
input_decode	11
match.list	12
match.option	13
onclick_event	13
scan_code	15
seq_timestamp	15
simple_ui	16
source_code	17
toupper_words	17
toupperfirst	18
trace_level	18
update_r_environ	19
use_plot_theme	20
use_r_environ	20
uuid	21
with_js	22
Index	23

action_link	Action Link Input(s)
-------------	----------------------

Description

Wrapper function around shiny::actionLink() to produce multiple action links with unique ids but same target input.

Usage

```
action_link(  
  id,  
  pattern = "action_link",  
  label = "",  
  icon = NULL,  
  target,
```

```

    value = "this.id",
    namespace = NULL,
    event = TRUE,
    as_character = FALSE,
    ...
  )

```

Arguments

id	a vector or list of ids.
pattern	the pattern to generate the actionLink ids.
label	the label of the link.
icon	an optional icon (passed to shiny::actionLink()).
target	the name of the target input.
value	an optional value to pass to the target input.
namespace	an option character string, for the id of the module.
event	a logical if shiny should treat the click as an event.
as_character	a logical if the shiny.tag should be returned or a character value.
...	Named attributes to pass to the shiny::actionLink() function.

Details

The actionLink inputId will be computed based on namespace, pattern and id so that it should be unique across different module instances.

All produced actionLinks will target the same input entry with different values. By default, it will send the id of the link, which could be used a character string to decode.

This behavior can be tuned to send a list (see [onclick_event\(\)](#)). When value is a list, the id will be added to that list so that one can detect what link was clicked.

When the target input belongs to a shiny module server, the id of the module server instance should be passed to namespace.

Value

a list or a vector of length 1.

See Also

[onclick_event\(\)](#), [shiny::actionLink\(\)](#)

Examples

```

## Not run:
# basic use case (single actionLink):
action_link(id = "foo", label = "click", target = "on_click")

# Send tag id to the target input:

```

```

# input$on_click will get "action_link_1", "action_link_2", "action_link_3"
action_link(id = c(1,2,3), label = "click", target = "on_click", value = NULL)

# Send specific pattern to the target input:
# input$on_click will get "do_this_1", "do_this_2" or "do_this_3"
action_link(id = c(1,2,3), label = "click", target = "on_click", pattern = "do_this")

# Send specific value to the target input:
# input$on_click will get "do_this" no matter what link is clicked
action_link(id = c(1,2,3), label = "click", target = "on_click", value = "do_this")

# Send a list of values to the target input:
# input$on_click will get list(a = 1, id = 1) after clicking on the first link
action_link(id = c(1,2,3), label = "click", target = "on_click", value = list(a = 1))

# Use with Shiny module:
action_link(id = c(1,2,3), label = "click", target = "on_click",
            pattern = "do_this", namespace = "module_id")

## End(Not run)

```

catl

Conditional Output

Description

This wrapper function aims at enabling a trace level mechanism that relies on the [cat](#) function.

Usage

```
catl(..., level = 1, debug = Sys.getenv("TRACE_LEVEL"), newline = TRUE)
```

Arguments

...	R objects as defined in cat
level	a numeric value indicating the level of trace
debug	an optional (usually omitted) value to indicate the level of trace to be displayed
newline	a logical. If TRUE (default), new line "\n" will be added

Details

The goal is to provide a trace level mechanism that can be disabled in a production context (as printing output to the console or file is costly).

The different levels are left flexible.

By default, debug is set to `Sys.getenv("TRACE_LEVEL")`. If the environment variable is not set, no output will be produced (silent mode). When the variable is set, the function will pass calls with `level <= debug` to the original [cat](#) function.

The [trace_level](#) function can be used to activate a specific trace level.

Value

None (invisible NULL).

Examples

```
# silent unless TRACE_LEVEL is set
catl("Awesome message here")

# force print no matter if TRACE_LEVEL is set
catl("Awesome message here", debug = 1)

# demonstrate trace levels
catl("Awesome message here", level = 2, debug = 1)
catl("Awesome message here", level = 1, debug = 2)
```

copy_template	<i>Copy Package Template Files.</i>
---------------	-------------------------------------

Description

Copy Package Template Files.

Usage

```
copy_template(template, pkg = "ktools", path = getwd(), filename = NULL)
```

Arguments

template	the name of the template file to be copied.
pkg	the name of the package where to find the template.
path	destination path where to copy the template (default = working directory).
filename	an optional name for the copy.

Details

By default, the function addresses templates delivered along with this package (i.e. pkg = "ktools").

When filename is not provided, the copy will have same name as the template without the "template_" prefix pattern.

An error will be thrown if the template file is not found.

Value

a logical (see [file.copy\(\)](#))

Examples

```
## Not run:
copy_template(template = "template_shiny_server.R", path = "shinyapp", filename = "my_server.R")

## End(Not run)
```

count_lines

*Code Stats***Description**

Builds a basic code counts report.

Usage

```
count_lines(con, verbose = FALSE)
```

Arguments

con a connection object or a character string (passed to [readLines](#)).

verbose a logical (FALSE by default) if counts should be printed to the console.

Details

The report will contain the following information:

- total
- code
- comment
- documentation
- spacing

Code lines ending with a comment are counted as code lines (hence ignored from comments).

Value

a named vector.

See Also

[scan_code\(\)](#)

Examples

```
infile <- tempfile()

lines <- "' This\n#'\n\n# comment\np <- 1\n\nif(p==1) print(p)\n"

writeLines(lines, infile)
count_lines(infile)
```

`create_data`*Create Data*

Description

Build an empty data.frame from a vector of colClasses.

Usage

```
create_data(colClasses)
```

Arguments

`colClasses` a named vector of classes.

Details

The names of the vector will be used to name the columns and the values to cast the columns to the given classes.

If NA is given as input for colClasses, then read.table() will return an empty data.frame (0 obs. of 0 variables)

Value

a data.frame.

Examples

```
create_data(colClasses = c(id = "numeric", date = "Date", comment = "character"))
```

`date_range`*Date Range*

Description

Compute specific range between two dates.

[Deprecated]

Usage

```
date_range(min, max, type = "this_year")
```

Arguments

`min` a Date for the lower value.

`max` a Date for the higher value.

`type` an optional character vector, to set the strategy (default = "this_year").

Value

a Date vector c(start, end).

Examples

```
## Not run:
date_range(min = Sys.Date()-365, max = Sys.Date()+365, type = "this_year")

## End(Not run)
```

filter_date

Filter by Date Range

Description

Filter a data.frame by specific date range.

Usage

```
filter_date(x, colname = NULL, ref = Sys.Date(), unit = "month")
```

Arguments

x	a data.frame with a Date or POSIXct column.
colname	an optional character string, the name of the column to use in filter.
ref	an optional Date value to be used as a reference for the date range.
unit	a string to compute the date range (default = "month") (see round_date).

Details

The purpose of this function is to make data navigation between date range easier. By default, it will filter the input data.frame to keep dates belonging to the current month.

When colname = NULL (the default), it will look for a 'date' column in x.

Value

a filtered data.frame.

Examples

```
# -- this month (the default)
filter_date(data.frame(date = Sys.Date() - runif(n = 10, min = -50, max = 50)))

# -- previous month
filter_date(data.frame(date = Sys.Date() - runif(n = 10, min = -50, max = 50)),
ref = Sys.Date() - as.integer(format(Sys.Date(), "%d")))
```

getNsFunction	<i>Package Function in Do Call</i>
---------------	------------------------------------

Description

Trick to solve the use of :: for package functions in do.call()

Usage

```
getNsFunction(x)
```

Arguments

x	the name of the function, most probably as pkg::fun()
---	---

Value

the exported function name in namespace pkg

Examples

```
## Not run:  
getNsFunction("pkg::fun")  
  
## End(Not run)
```

getTimestamp	<i>Compute Timestamp</i>
--------------	--------------------------

Description

Compute a numeric timestamp.

Usage

```
getTimestamp(k = 1000, digits = 0, silent = FALSE)
```

Arguments

k	a numeric (default = 1000), used as a multiplication factor.
digits	an integer (default = 0) to round the value.
silent	an optional (default = FALSE) logical. If TRUE, no traces will go to the console.

Details

By default (k = 1000), the function returns a numeric up to the millisecond.
Output should not be used as a unique id if users / systems may call it more than one time per millisecond.

Value

a numeric.

See Also

[uuid\(\)](#)

Examples

```
# compute up to the second timestamp
getTimestamp(k = 1, digits = 0)

# compute up to the millisecond timestamp
getTimestamp(k = 1000)
getTimestamp()
```

has_date	<i>Find Date Column(s)</i>
----------	----------------------------

Description

Detects Date and/or POSIXct column(s) in a data.frame.

Usage

```
has_date(x, single = TRUE)
```

Arguments

- x a data.frame object.
- single a logical (default TRUE) if it should return only the first column.

Value

the name of the column(s) or NA if no date column is found.

Examples

```
has_date(data.frame(date = Sys.Date()))
has_date(data.frame(date = Sys.Date(), update = Sys.Date()), single = FALSE)
has_date(data.frame(date = Sys.Date(), update = Sys.Date()), single = TRUE)
```

incat	<i>Indented Cat</i>
-------	---------------------

Description**[Experimental]****Usage**

```
incat(x, sep = "")
```

Arguments

x	R objects.
sep	a character vector of strings to append after each element (default is "").

Details

When the function detects a direct call (i.e. not from another function), then it basically just do cat otherwise it finds the calling function name and print indented output based on callstack position.

Value

None (invisible NULL).

Examples

```
# default
incat("direct call")

# call from different levels
foo <- function() incat("inside foo")
bar <- function() {incat("inside bar"); foo()}

foo()
bar()
```

input_decode	<i>Decode Input Value</i>
--------------	---------------------------

Description

Split an input value into a named vector.

Usage

```
input_decode(x)
```

Arguments

x the (input) value to decode.

Details

The function is used together with [action_link\(\)](#) / [onclick_event\(\)](#) when their value is left to the HTML tag id.

The expected pattern for x is "namespace-action_value" or "action_value". More complex use cases should be covered using a list value in [action_link\(\)](#).

Value

a named vector c(namespace, action, value) or c(action, value).

Examples

```
input_decode("module1-add_leg_896")
input_decode("add_leg_896")
```

match.list

Match Lists

Description

Matches a named list against a reference.

Usage

```
match.list(x, reference)
```

Arguments

x a named list to check.
reference a named list to use as a reference.

Details

The function will drop any element from x that does not fit with a name found in reference and add missing ones.

It will also check that elements in x have same classes as matching element in reference.

Value

a list.

Examples

```
match.list(x = list(foo = 1), reference = list(foo = 0, bar = "dummy"))
```

`match.option`*Match Options*

Description

Matches a named list against the one expected in function arguments.

Usage

```
match.option(fun, arg, value)
```

Arguments

<code>fun</code>	the reference of the function, without ().
<code>arg</code>	the name of the argument to test.
<code>value</code>	the value to test.

Value

The list of options, including missing ones.

Examples

```
# function expecting a list as argument
foo <- function(options = list(a = 1, b = 2)){

# match user input
match.option(foo, "options", list(a = 1))
```

`onclick_event`*Build onClick Event*

Description

This function helps building onclick event to pass complex instructions to a given listener. This is particularly useful when working with lists of links, most probably generated within an apply instruction over a data.frame.

Usage

```
onclick_event(target, value = "this.id", namespace = NULL, event = TRUE)
```

Arguments

target	the input to update on click.
value	the value to send to the input (see details).
namespace	the id of the module to target if so.
event	should Shiny treat it as an event.

Details

target defines the name of the input entry to update (i.e. input\$target).

value will be passed to the target input. By default, it will send the id of the HTML tag (ex: actionLink, actionButton).

This behavior can be tuned to send a more complex instruction. In particular, a list can be passed to the target input (see example).

Value

a character string.

See Also

[action_link\(\)](#)

Examples

```
## Not run:
# baseline:
# input$btn_clicked will get the id of the html tag
onclick_event(target = "btn_clicked")

# custom value:
# input$btn_clicked will get "foo"
onclick_event(target = "btn_clicked", value = "foo")

# custom list:
# input$btn_clicked will get the list
onclick_event(target = "btn_clicked", value = list(a = "foo", b = "bar"))

# -- shiny module:
# input$btn_clicked inside "module_id" will get 12
onclick_event(target = "btn_clicked", value = 12, namespace = "module_id")

## End(Not run)
```

`scan_code`*Scan Code*

Description

Generates a code report for a given path.

Usage

```
scan_code(path = getwd())
```

Arguments

`path` a path to scan.

Value

a data.frame.

See Also

[count_lines\(\)](#)

Examples

```
## Not run:  
scan_code()  
## End(Not run)
```

`seq_timestamp`*Sequence of Timestamps*

Description

[Deprecated]

Usage

```
seq_timestamp(n = 2)
```

Arguments

`n` the desired output vector length.

Value

a numeric vector of timestamps.

Examples

```
## Not run:  
seq_timestamp(n = 2)  
## End(Not run)
```

simple_ui

Shiny Outputs

Description

Wrapper around Shiny *Output functions.

Usage

```
simple_ui(id, output, type = "ui", ...)
```

Arguments

id	the id of the module.
output	the name of the output.
type	the type of output: "ui" (default), "text", "plot", "table".
...	other arguments to pass to the *Output function.

Details

id can be a vector of ids in case of sub module.

Value

an HTML tag.

Examples

```
simple_ui(id = "my_module", output = "my_output")
```

source_code	<i>Source Code</i>
-------------	--------------------

Description

Source all files from a given path.

Usage

```
source_code(path, verbose = FALSE)
```

Arguments

path	a character vector of the path to explore.
verbose	a logical value (default = FALSE). If TRUE, then details about sourced files will be sent to the console.

Details

Given path is explored with "*.R" extension pattern. Sub-directories will be explored as well.

Value

an integer, how many functions have been attached to the environment.

Examples

```
## Not run:  
source_code("./R")  
  
## End(Not run)
```

toupper_words	<i>Capitalize First Letters</i>
---------------	---------------------------------

Description

Capitalize First Letters

Usage

```
toupper_words(s, strict = FALSE)
```

Arguments

s	a character string.
strict	a logical (default = FALSE) if other characters should be forced to lower.

Value

a character string.

Examples

```
toupper_words("this is an example")
```

toupperfirst	<i>Capitalize First Letter</i>
--------------	--------------------------------

Description

Capitalize First Letter

Usage

```
toupperfirst(x)
```

Arguments

x a character vector, or an object that can be coerced to character by `as.character`.

Value

a character vector.

Examples

```
toupperfirst("capitalize first letter")
toupperfirst(1)
```

trace_level	<i>Trace Level</i>
-------------	--------------------

Description

A helper function to get/set the trace level printed by `catl()`.

Usage

```
trace_level(level = NULL)
```

Arguments

level an integer (or NULL).

Details

level is used to define the level of trace printed by `cat1()`. Use 0 to set the trace OFF or NULL to get the current trace level.

Value

Nothing or the active trace level.

See Also

`cat1()`

Examples

```
trace_level()
trace_level(2)
trace_level(0)
```

update_r_environ	<i>Update .Renviron File</i>
------------------	------------------------------

Description

Add an environment variable to the project.

Usage

```
update_r_environ(path = getwd(), key, value)
```

Arguments

path	the path to the .Renviron file (default is working directory).
key	a character string for the name of the variable.
value	the value to be assigned to the variable.

Value

a logical (TRUE if success, else FALSE)

See Also

`use_r_environ()`

Examples

```
## Not run:
update_r_environ(key = "MY_VAR", value = "my_value")

## End(Not run)
```

use_plot_theme	<i>Use Plot Theme</i>
----------------	-----------------------

Description

Create a theme function that can be used across a project.

Usage

```
use_plot_theme(path = getwd())
```

Arguments

path where to copy the template.

Value

a logical.

Examples

```
## Not run:  
use_plot_theme()  
  
## End(Not run)
```

use_r_environ	<i>Create .Renviron File</i>
---------------	------------------------------

Description

Create .Renviron File

Usage

```
use_r_environ(path = getwd())
```

Arguments

path the path where to create the file.

Details

By default, path is set to the working directory.

Value

a logical (see `file.create()`)

See Also

`update_r_environ()`

Examples

```
## Not run:
use_r_environ()

## End(Not run)
```

uuid	<i>Generate Uuid</i>
------	----------------------

Description

Compute unique identifier.

Usage

```
uuid()
```

Details

By default `getTimestamp()` gives a value up to the millisecond.

In some use cases, for example when a sequence of ids need to be generated grammatically, multiple identical values will be generated within the same millisecond.

This wrapper calls `getTimestamp()` with $k = 1000000$ to get a value up to the microsecond. This is usually enough to get unique value for example when it's called from applying a function over a list.

Value

a numeric.

Examples

```
uuid()

# showing what the function is trying to solve
any(duplicated(replicate(100, getTimestamp())))
any(duplicated(replicate(100, uuid())))
```

with_js	<i>JavaScript Dependency</i>
---------	------------------------------

Description

Checks whether a specific dependency is already inserted in the UI with this function, and performs the insert if necessary.

Usage

```
with_js(package, src, script, session = shiny::getDefaultReactiveDomain())
```

Arguments

package	the name of the package.
src	where to find the script (see details).
script	the script to include (relative to src).
session	the session object of the Shiny server.

Details

The function is designed to be called from another function creating/updating a widget that requires JavaScript. With this, the user does not need to manage the dependency manually.

When session is NULL, the function will return without doing anything. This is useful when the function is called from ui side. The JavaScript dependency will be attached next time the function is called from server side. This is because the session\$userData object cannot be updated from ui side.

Details about src can be found in [htmlDependency](#).

Value

NULL (invisibly)

Examples

```
## Not run:  
with_js(package = "ktag", src = "assets", script = "js/ktag.js",  
session = shiny::getDefaultReactiveDomain())  
  
## End(Not run)
```

Index

action_link, [2](#)
action_link(), [12](#), [14](#)

cat, [4](#)
catl, [4](#)
catl(), [18](#), [19](#)
copy_template, [5](#)
count_lines, [6](#)
count_lines(), [15](#)
create_data, [7](#)

date_range, [7](#)

file.copy(), [5](#)
file.create(), [21](#)
filter_date, [8](#)

getNsFunction, [9](#)
getTimestamp, [9](#)
getTimestamp(), [21](#)

has_date, [10](#)
htmlDependency, [22](#)

incat, [11](#)
input_decode, [11](#)

match.list, [12](#)
match.option, [13](#)

onclick_event, [13](#)
onclick_event(), [3](#), [12](#)

readLines, [6](#)
round_date, [8](#)

scan_code, [15](#)
scan_code(), [6](#)
seq_timestamp, [15](#)
shiny::actionLink(), [3](#)
simple_ui, [16](#)

source_code, [17](#)

toupper_words, [17](#)
toupperfirst, [18](#)
trace_level, [4](#), [18](#)

update_r_envirom, [19](#)
update_r_envirom(), [21](#)
use_plot_theme, [20](#)
use_r_envirom, [20](#)
use_r_envirom(), [19](#)
uuid, [21](#)
uuid(), [10](#)

with_js, [22](#)