Package 'flow'

August 13, 2021

Title View and Browse Code Using Flow Diagrams

Version 0.0.2

Description Visualize as flow diagrams the logic of functions, expressions or scripts in a static way or when running a call, and ease debugging. Advanced features include analogs to 'debug' and 'debugonce' to target specific functions to draw, an utility to draw the calls used in the tests of the package in a markdown report, and an utility to draw all the functions of one package in a markdown report.

License GPL-3

Encoding UTF-8

Suggests testthat (>= 2.1.0), covr, knitr, rmarkdown

Imports nomnoml, utils, htmlwidgets, rstudioapi, webshot, styler

RoxygenNote 7.1.1.9001

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Antoine Fabri [aut, cre]

Maintainer Antoine Fabri <antoine.fabri@gmail.com>

Repository CRAN

Date/Publication 2021-08-13 09:00:04 UTC

R topics documented:

flow_debug *Debug With Flow Diagrams*

Description

These functions are named after the base functions debug(), undebug() and debugonce(). flow_debug() will call flow_run(), with the same additional arguments, on all the following calls to f() until flow_undebug() is called. flow_debugonce() will only call flow_run() on the next call to f().

Usage

```
flow_debug(
  f,
  prefix = NULL,
  code = TRUE,
  narrow = FALSE,
  truncate = NULL,
  swap = TRUE,
  out = NULL,
  browse = FALSE
)

flow_debugonce(
  f,
  prefix = NULL,
  code = TRUE,
  narrow = FALSE,
  truncate = NULL,
  swap = TRUE,
  out = NULL,
  browse = FALSE
)

flow_undebug(f)
```

Arguments

f	function to debug
prefix	prefix to use for special comments in our code used as block headers, must start with "#", several prefixes can be provided
code	Whether to display the code in code blocks or only the header, to be more compact, if NA, the code will be displayed only if no header is defined by special comments
narrow	TRUE makes sure the diagram stays centered on one column (they'll be longer but won't shift to the right)
truncate	maximum number of characters to be printed per line

swap	whether to change `var <-if(cond) expr` into `if(cond) var <-expr` so the diagram displays better
out	a path to save the diagram to. Special values "html", "htm", "png", "pdf", "jpg" and "jpeg" can be used to export the objec to a temp file of the relevant format and open it, if a regular path is used the format will be guessed from the extension.
browse	whether to debug step by step (block by block), can also be a vector of block ids, in this case `browser()` calls will be inserted at the start of these blocks

Details

By default, unlike `debug()` and `debugonce()`, `flow_debug()` and `flow_debugonce()` don't trigger a debugger but only draw diagrams, this is consistent with `flow_run()`'s defaults. To browse through the code, use the `browse` argument.

Value

These functions return `NULL` invisibly (called for side effects)

---

flow_doc *Draw Flow Diagrams for an Entire Package*

Description

Draw Flow Diagrams for an Entire Package

Usage

```
flow_doc(
  pkg = NULL,
  prefix = NULL,
  code = TRUE,
  narrow = FALSE,
  truncate = NULL,
  swap = TRUE,
  out = NULL,
  engine = c("nomnoml", "plantuml")
)
```

Arguments

pkg	package name as a string
prefix	prefix to use for special comments in our code used as block headers, must start with `"#"`, several prefixes can be provided
code	Whether to display the code in code blocks or only the header, to be more compact, if `NA`, the code will be displayed only if no header is defined by special comments

narrow	TRUE makes sure the diagram stays centered on one column (they'll be longer but won't shift to the right)
truncate	maximum number of characters to be printed per line
swap	whether to change var <-if(cond) expr into if(cond) var <-expr so the diagram displays better
out	path to output (.html or .md), if left NULL a temp *html* file will be created and opened.
engine	either "nomnoml" (default) or "plantuml" (experimental), if the latter, arguments prefix, narrow, and code

Details

if pkg and out are both left NULL, a vignette diagrams.md will be built in the root, so that pkgdown::build_site will use it as an additional page. See also the vignette *"Build reports to document functions and unit tests"*.

Value

Returns NULL invisibly (called for side effects).

---

flow_draw *Draw Diagram From Debugger*

Description

flow_draw() should only be used in the debugger triggered by a call to flow_run(), or following a call to flow_debug() or flow_debugonce(). d is an active binding to flow_draw(), it means you can just type d (without parentheses) instead of flow_draw().

Usage

```
flow_draw()

d
```

Details

d was designed to look like the other shortcuts detailed in ?browser, such as f, c etc... It differs however in that it can be overridden. For instance if the function uses a variable d or that a parent environment contains a variable d, flow::d won't be found. In that case you will have to use flow_draw().

If d or flow_draw() are called outside of the debugger they will return NULL silently.

Value

Returns NULL invisibly (called for side effects)

`flow_test` *Build Report From Tests*

Description

Build a markdown report from test scripts, showing the paths taken in tested functions, and where they fail if they do. See also the vignette *"Build reports to document functions and unit tests"*.

Usage

```
flow_test(
  prefix = NULL,
  code = TRUE,
  narrow = FALSE,
  truncate = NULL,
  swap = TRUE,
  out = NULL,
  failed_only = FALSE
)
```

Arguments

`prefix`	prefix to use for special comments in our code used as block headers, must start with `"#"`, several prefixes can be provided
`code`	Whether to display the code in code blocks or only the header, to be more compact, if `NA`, the code will be displayed only if no header is defined by special comments
`narrow`	`TRUE` makes sure the diagram stays centered on one column (they'll be longer but won't shift to the right)
`truncate`	maximum number of characters to be printed per line
`swap`	whether to change `var <-if(cond) expr` into `if(cond) var <-expr` so the diagram displays better
`out`	path to output (`.html` or `.md`), if left `NULL` a temp *html* file will be created and opened.
`failed_only`	whether to restrict the report to failing tests only

Value

Returns `NULL` invisibly (called for side effects)

`flow_view` *View function as flow chart*

Description

`flow_view()` shows the code of a function as a flow diagram, `flow_run()` runs a call and draws the logical path taken by the code.

Usage

```
flow_view(
  x,
  prefix = NULL,
  code = TRUE,
  narrow = FALSE,
  truncate = NULL,
  nested_fun = NULL,
  swap = TRUE,
  out = NULL,
  engine = c("nomnoml", "plantuml")
)

flow_run(
  x,
  prefix = NULL,
  code = TRUE,
  narrow = FALSE,
  truncate = NULL,
  swap = TRUE,
  out = NULL,
  browse = FALSE
)
```

Arguments

`x`	a call, a function, or a path to a script
`prefix`	prefix to use for special comments in our code used as block headers, must start with `"#"`, several prefixes can be provided
`code`	Whether to display the code in code blocks or only the header, to be more compact, if `NA`, the code will be displayed only if no header is defined by special comments
`narrow`	`TRUE` makes sure the diagram stays centered on one column (they'll be longer but won't shift to the right)
`truncate`	maximum number of characters to be printed per line
`nested_fun`	if not `NULL`, the index or name of the function definition found in x that we wish to inspect

swap	whether to change `var <-if(cond) expr` into `if(cond) var <-expr` so the diagram displays better
out	a path to save the diagram to. Special values "html", "htm", "png", "pdf", "jpg" and "jpeg" can be used to export the objec to a temp file of the relevant format and open it, if a regular path is used the format will be guessed from the extension.
engine	either `"nomnoml"` (default) or `"plantuml"` (experimental), if the latter, arguments `prefix`, `narrow`, and `code`
browse	whether to debug step by step (block by block), can also be a vector of block ids, in this case `browser()` calls will be inserted at the start of these blocks

Details

On some system the output might sometimes display the box character when using the nomnoml engine, this is due to the system not recognizing the Braille character \u2800. This character is used to circumvent a nomnoml shortcoming: lines can't start with a standard space and multiple subsequent spaces might be collapsed. To choose another character, set the option `flow.indenter`, for instance : `options(flow.indenter = "\u00b7")`.

Value

`flow_view()` returns `NULL` invisibly, or the output path invisibly if `out` is not `NULL` (called for side effects). `flow_run()` returns the output of the wrapped call.

Examples

```
flow_view(rle)
flow_run(rle(c(1, 2, 2, 3)))
```

Index