Package: breakerofchains (via r-universe)

September 3, 2024

Title Break Chained Expressions and Run Them with Printed Output

Version 0.3.2

Description Run an infix operator expression chain up to the line your cursor is on, printing the output, and ignoring any result assignment step. This facilitates easier interactive debugging of chained code. Common examples of code amenable to breaking with this tool are {dplyr} wrangling chained with `%>%`, and {ggplot2} plotting chained with `+`.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports rstudioapi, sourcetools, dplyr, magrittr, utils

Repository https://milesmcbain.r-universe.dev

RemoteUrl https://github.com/milesmcbain/breakerofchains

RemoteRef main

RemoteSha 5c8d0e176ed2188a14f930a31640bc5465bfb890

Contents

`break_chain` *break an infix (like %>%) chain and run.*

Description

Run a chain of piped or otherwise infixed commands up to and including the cursor line. The chain is assumed to end each line with the chaining operator, as is common in ' the tidyverse style guide.

When a chain begins with an assignment via = or <- the assignment is not performed. Results of running the chain section are printed to the console, and by default stored in a global variable called `.chain`.

Usage

```
break_chain(
  print_result = TRUE,
  assign_result = getOption("breakerofchains_store_result", TRUE)
)
```

Arguments

`print_result`	Enable/disable printing of chain evaluation result in console. Useful when wrapping this function to display results in a custom way.
`assign_result`	assign the result of chain evaluation to `.chain` in Global environment?

Details

Storing results in `.chain` can be disabled by setting `options(breakerofchains_store_result = FALSE)`.

Your code is read via the rstudioapi in RStudio or rstudioapi emulation in VSCode. Code is parsed up to the cursor line before an algorithm works backwards to find the chain start. Unfortunately this means all code above the cursor line must be valid parsable R code.

It is unlikely you want to run this function directly. You probably want to bind it to a keyboard shortcut. See README for more information.

Developers: You can create addins / shortcuts that treat the result of chain evaluation differently by wrapping this function. e.g. `view(break_chain())` The parameters of this function are inteded to be useful for this e.g. `view(break_chain(print_result = FALSE))`

Value

the result of chain execution invisibly

find_chain_start *find the start of an infix chain*

Description

Working upward from the last line, find the start of the chain.

Usage

```
find_chain_start(doc_lines)
```

Arguments

doc_lines	lines of code to examine.

Value

the index into doc_lines that contains the start of the chain

get_broken_chain *get a broken chain as text*

Description

This interface is intended for developers who want to hook into the chain breaking algorithm to create bindings in other text editors.

Usage

```
get_broken_chain(doc_lines, doc_cursor_line)
```

Arguments

doc_lines	a character vector of R code, one element per line.
doc_cursor_line	a number representing the line the cursor is on.

Details

Given a character vector of R code lines, and the line number of the cursor, it returns a character vector of R code lines which is the start of the chained expression the cursor is on, up to the cursor line.

Any assignment with <- or = at the head of the chain is removed.

Value

a character vector of R code representing the broken chain.

Examples

```
get_broken_chain(
    c(
     "species_scatter <- starwars %>%",
     "group_by(species, sex) %>%",
     "select(height, mass)",
     "    .99s.scatter <- starwars %>%",
     "group_by(species, sex) %>%",
     "select(height, mass)"
    ),
    3
)
```

Index