Package 'flair'

October 13, 2022

Title Highlight, Annotate, and Format your R Source Code

Version 0.0.2

Description Facilitates easier formatting and highlighting of R source code in a R Markdown-based presentation. The main goal of the package is to allow users to preserve their code creation process within code chunks, then to specify formatting details for the source code, such as highlighting of particular syntactical elements.

License MIT + file LICENSE

URL https://github.com/kbodwin/flair, https://kbodwin.github.io/flair/index.html

BugReports https://github.com/kbodwin/flair/issues

Imports dplyr, evaluate, ggplot2, glue, knitr, magrittr, purrr, rmarkdown, stringr

Suggests learnr, rstudioapi, shiny, testthat (>= 2.1.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Kelly Bodwin [aut, cre], Hunter Glanz [aut]

Maintainer Kelly Bodwin <kelly@bodwin.us>

Repository CRAN

Date/Publication 2020-04-23 18:00:05 UTC

R topics documented:

`flair-package` *A package for adding flair to your displayed source code.*

Description

This package provides ...

Flair functions

The `flair_*` functions add decorative formatting to source code.

Decorate functions

The `decorate` functions execute source code and simultaneously prepare it to have flair added, to display when knitted.

code_from_editor *Converts raw editor text to a string of code*

Description

Raw editor text has been taken from an active RStudio session via `rstudioapi::getSourceEditorContext()`. Chunk delimiters and html is removed, all formatting is otherwise preserved.

Usage

```
code_from_editor(.contents, chunk_name)
```

Arguments

`.contents`	chunk contents passed from editor context
`chunk_name`	label of chunk

Value

chunk text

decorate *Builds a `with_flair` object from either a code chunk or a string object containing source code.*

Description

`decorate` does its best to guess if it has been given a code string or a chunk name, based on the presence of special characters.

Usage

```
decorate(x, ...)
```

Arguments

`x`	A string, containing either a chunk label or R code.
`...`	Chunk options to pass along

Value

A `with_flair` object.

See Also

`decorate_chunk`, `decorate_code`

decorate_chunk *Builds a* with_flair *object from a code chunk*

Description

This function reads the source code from a given named code chunk; i.e., {r chunk_name, echo = FALSE}.

Usage

```
decorate_chunk(chunk_name, eval = TRUE, echo = TRUE, include = TRUE, ...)
```

Arguments

chunk_name	The label name of the chunk we plan to add flair to.
eval	Evaluation options for chunk; behaves identically to ordinary knitr code chunk option eval
echo	Evaluation options for chunk; behaves identically to ordinary knitr code chunk option echo
include	Evaluation options for chunk; behaves identically to ordinary knitr code chunk option include
...	Any number of other chunk options to override.

Details

When run directly in a source file, decorate_chunk() reads the text of the active file and extracts the relevant string of source code from the chosen chunk. (Important: this only works in RStudio.)

When run during the knitr::knit() process, decorate_chunk() pulls the relevant chunk source during knitr::knit_hooks$set("source").

Value

An object of class with_flair

decorate_code *Creates an object of the class* with_flair

Description

Creates an object of the class with_flair

Usage

```
decorate_code(text, ...)
```

Arguments

`text`	A string, presumably representing R code.
`...`	Any number of default chunk options to override.

Value

A `with_flair` object.

See Also

`flair`

Examples

```
# When run in console, this will print the results of mean(1:10)
my_code <- decorate_code(text = 'mean(1:10)') %>% flair_funs()

# The object itself, when printed, previews your code with flair

my_code


# Objects defined by decorate_code are created in the current environment for later use.

my_code <- decorate_code('foo <- mean(1:10)')

foo + 5
```

`flair` *Formats source code*

Description

Adds decorative formatting to parts of a string or source code.

Usage

```
flair(x, pattern, before = NULL, after = NULL, ...)

flair_rx(x, pattern, before = NULL, after = NULL, ...)

## S3 method for class 'with_flair'
flair_rx(x, pattern, before = NULL, after = NULL, ...)

## Default S3 method:
flair_rx(x, pattern, before = NULL, after = NULL, ...)
```

```
flair_quick(x, pattern, before = NULL, after = NULL, ...)

flair_all(x, ...)

## Default S3 method:
flair_all(x, ...)

## S3 method for class 'with_flair'
flair_all(x, ...)

flair_args(x, ...)

flair_funs(x, ...)

flair_input_vals(x, ...)
```

Arguments

x	A string or with_flair object
pattern	A pattern to match. By default, this is a fixed pattern; use flair_rx for regular expressions.
before	String giving specific html tags to insert before matched text.
after	String giving specific html tags to insert after matched text.
...	Formatting style options, passed to txt_style

Details

If input is a string object, flair returns a formatted string.

If input is a with_flair object, flair returns a with_flair object with the source elements formatted.

Currently, flair is only built for html formatting.

Value

A string with formatting wrappers.

Examples

```
code_string <- "foo <- mean(1:10, na.rm = TRUE)"

code_string %>% flair("foo")

code_string %>% flair_args()

code_string %>% flair_funs(color = "red")
```

`flair_lines` *Adds decorative formatting (flair) to parts of a string or source code, specified by line(s).*

Description

`flair_lines` returns a string with formatting wrappers(currently only html), or applies the formatting to the source elements of a `with_flair` object.

Usage

```
flair_lines(x, lines)

## Default S3 method:
flair_lines(x, lines)
```

Arguments

`x`	A string or `with_flair` object
`lines`	Integer vector indicating which lines to apply the flair styling to.

Value

A string with formatting wrappers.

Examples

```
code_string <- "x <- mean(1:10, na.rm = TRUE)
sqrt(x)" %>% flair_lines(2)
```

`flair_lines.with_flair` *S3 method for `with_flair` objects*

Description

Applies flair to the appropriate line(s) of source code.

Usage

```
## S3 method for class 'with_flair'
flair_lines(x, lines)
```

Arguments

x	An object of class with_flair.
lines	An integer vector specifying code lines to highlight.

Value

An object of class with_flair.

---

flair_sublines *Helper for* flair_lines

Description

Helper for flair_lines

Usage

```
flair_sublines(text, nums, lines)
```

Arguments

text	Text from knitted source code
nums	List of overall lines contained
lines	Which lines to highlight

Value

Text with formatting wrappers

---

knit_print.with_flair *S3 method for knitting a* with_flair *object*

Description

S3 method for knitting a with_flair object

Usage

```
knit_print.with_flair(x, ...)
```

Arguments

x	A with_flair object.
...	Other knit_print options

Value

"as-is" html output, to be rendered when knitted

make_sandwiches *Recombines "sandwich" elements in proper order*

Description

Helper for `split_sandwiches`. Once sections have been identified and extracted, recombines them in the correct order.

Usage

```
make_sandwiches(meat, top_buns, bottom_buns = NULL)
```

Arguments

`meat`	Character vector containing string sections between delimiters
`top_buns`	Character vector containing starting delimiters
`bottom_buns`	Character vector containing ending delimiters

Value

A character vector.

mask *Blanks out part of the string*

Description

Blanks out part of the string

S3 method for `with_flair` objects

Default S3 method for `flair_rx`.

Usage

```
mask(x, pattern, before = NULL, after = NULL, ...)

mask_rx(x, pattern, before = NULL, after = NULL, ...)

## S3 method for class 'with_flair'
mask_rx(x, pattern, before = NULL, after = NULL, ...)

## Default S3 method:
mask_rx(x, pattern, before = NULL, after = NULL, ...)

mask_quick(x, pattern, before = NULL, after = NULL, ...)
```

Arguments

x	A string object or decorate_code object.
pattern	A pattern to match
before	Custom preceding html tag
after	Custom ending html tag
...	Further formatting options, passed to txt_style

---

prep_source *Helper for* knit_print.with_flair

Description

Helper for knit_print.with_flair

Usage

```
prep_source(x, doc_type)
```

Arguments

x	Text of source code.
doc_type	Document type to knit to.

Value

Properly wrapped text.

---

print.with_flair *When run interactively, a* with_flair *object should preview the flaired source code in the viewer pane. (Only if in RStudio.)*

Description

When run interactively, a with_flair object should preview the flaired source code in the viewer pane. (Only if in RStudio.)

Usage

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

Arguments

x	A with_flair object.
...	Other print options

Value

None

`scope_and_run` *Runs code from string, in parent environment*

Description

Shortcut function to rescope a code string and then run (but not print). Looks for object assignments of the form `foo <-` and rescopes to `foo <<-`, then evaluates code string.

Usage

```
scope_and_run(.code_string)
```

Arguments

`.code_string`	A string containing runnable R code.

Value

Nothing; side effects in environment only.

`scope_run_print` *Runs and prints from string, in parent environment*

Description

Shortcut function to rescope a code string and then run and print output. Looks for object assignments of the form `foo <-` and rescopes to `foo <<-`, then evaluates code string.

Usage

```
scope_run_print(.code_string)
```

Arguments

`.code_string`	A string containing runnable R code.

Value

Nothing; side effects from `print()` only.

`split_sandwiches` *Separates a string into sections by delimiter*

Description

This function takes delimiters for the beginning and (optionally, if different) end of sections of a string, and returns a vector with split string elements.

Usage

```
split_sandwiches(.string, start_rx, end_rx = NULL)
```

Arguments

`.string`	A string
`start_rx`	A regular expression denoting the beginning of a section. Use `fixed()` for literals.
`end_rx`	A regular expression denoting the end of a section. If none supplied, sections end when the next `start_rx` is encountered.

Details

The main use case for `split_sandwiches()` is for html editing: You might want to separate the original text from the html tags, make certain edits to the text only, and then re-wrap the tags.

This is different from `str_split()` or similar, because the delimiters are preserved and remain attached to a section.

Note that `split_sandwiches()` is not vectorized (sorry). It only takes a single character object.

Value

A vector of strings

Examples

```
my_string <- "<span style='text-color:blue'> I am blue and <b>bold</b>, yay! </span>"

split_sandwiches(my_string, "\\<[^\\>\\<]*\\>")
```

src_to_list — *Takes plain text of knitted code and converts it to a list, in which code sources have the class* source.

Description

Takes plain text of knitted code and converts it to a list, in which code sources have the class source.

Usage

```
src_to_list(knitted)
```

Arguments

knitted	Text of knitted code

Value

A list with code sources and knitted output.

txt_style — *Wraps text in html or latex code for formatting*

Description

txt_style adds appropriate html style wrappers to a string. Any number of options can be specified, as long as they match html CSS tags names.

Usage

```
txt_style(
  x,
  type = "html",
  bold = FALSE,
  underline = FALSE,
  italics = FALSE,
  ...
)

txt_color(x, color = "red")

txt_colour(x, colour = "red")

txt_size(x, size = "large")

txt_background(x, bg_color = "#ffff7f")
```

```
txt_font(x, font)

txt_bold(x)

txt_emph(x)

txt_ul(x)

txt_tocode(x)

txt_tag(x, before, after)
```

Arguments

x	The string to be wrapped
type	The style of display, defaults to "html" (currently nothing else is supported, sorry)
bold	Should the text be bolded?
underline	Should the text be underlined?
italics	Should the text be italicized?
...	various display options: any html CSS `style` options, or one of `font`, `size`, `color`, `background`, `style`.
color	Named or html hex color for font.
colour	Named or html hex color for font.
size	Font size
bg_color	Named or html hex color for text background.
font	A valid font family.
before	String giving specific html tags to insert before text.
after	String giving specific html tags to insert after text.

Details

`txt_*` are shortcuts for specific individual style options

Warning: These are simple direct wrappers for strings only. If you are using `with_flair` objects, you should instead use the `flair` functions.

Value

A string containing `x` with html wrappers.

See Also

`flair`

Examples

```
# General use
txt_style("I am highlighted!")
txt_style("I am blue and bold.", color = "blue", bold = TRUE)

# Shortcuts
txt_color("I am red.")
txt_color("I am blue.", color = "blue")

# Code styling wrapper
txt_tocode("I am code.")
```

---

with_flair *Create a* with_flair *object*

Description

(The preferred function is decorate)

Usage

```
with_flair(x)
```

Arguments

x	A text object or code chunk label

Value

An object of class with_flair

---

word_to_blanks *helper for mask*

Description

helper for mask

Usage

```
word_to_blanks(word)
```

Arguments

word	A word to replace with blank spaces of the same length

wrap_html *Wraps text in html for formatting*

Description

wrap_html returns a string with styling wrappers from a list of style tag names and values.

Usage

```
wrap_html(x, opts_list, type = "html")
```

Arguments

x	The string to be wrapped
opts_list	The html options to include, in named vector form
type	The style of display, defaults to "html" (currently nothing else is supported, sorry)

Index