Package 'sketch'

October 8, 2020

Type Package

Title Interactive Sketches

Version 1.0.3

Description Creates static / animated / interactive visualisations embeddable in R Markdown documents. It implements an R-to-JavaScript transpiler and enables users to write JavaScript applications using the syntax of R.

License Apache License (>= 2.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports magrittr, rlang, purrr, rstudioapi, glue, htmltools, base64enc, jsonlite, shiny

Suggests testthat, covr, knitr, rmarkdown,

BugReports https://github.com/kcf-jackson/sketch

VignetteBuilder knitr

NeedsCompilation no

Author Chun Fung Kwok [aut, cre], Kate Saunders [ctb]

Maintainer Chun Fung Kwok <kwokcf@unimelb.edu.au>

Repository CRAN

Date/Publication 2020-10-08 07:40:03 UTC

R topics documented:

`sketch-package` *Interactive visualisation*

Description

Creates interactive visualisation embeddable in R Markdown documents. It implements an R-to-JavaScript transpiler and enables users to write JavaScript applications using the syntax of R.

`assets` *Process assets in headers*

Description

Take a 'sketch' R file as input, extract and process the resources links as provided by the user with the '#!' header.

Usage

```
assets(file, ..., trace = FALSE)
```

Arguments

file	Character string; the file path.
...	(Optional) List of processors to pass to convert_src.
trace	TRUE or FALSE; if TRUE, assets are extracted, but not processed.

Examples

```
file <- system.file("test_files/test_RMD.R", package = "sketch")
assets(file, trace = TRUE)
assets(file, trace = FALSE)
```

---

basic_deparsers *A minimal list of deparsers for deparsing JavaScript*

Description

A minimal list of deparsers for deparsing JavaScript

Usage

```
basic_deparsers()
```

Note

This is used as input to deparse_js, compile_r and compile_exprs.

Examples

```
basic_deparsers()
```

---

compile_data *Compile a data file into a JavaScript file*

Description

Compile a data file into a JavaScript file

Usage

```
compile_data(input, output = tempfile(), ...)
```

Arguments

input	A character string; the path to the input file.
output	A character string; the path to the output file.
...	Extra arguments to be passed to to_json.

Examples

```
file <- system.file("test_files/test_csv.csv", package = "sketch")
readLines(compile_data(file))
```

---

compile_exprs *Compile R code into JavaScript code*

Description

Compile R code into JavaScript code

Usage

```
compile_exprs(x, rules = default_rules(), deparsers = default_deparsers())
```

Arguments

x	A character string; the expression to transpile to JS.
rules	A list of rewriting rules. See [make_rule] for more detail.
deparsers	A list of deparsers. See [make_deparser] for more detail.

Value

A character string.

Examples

```
compile_exprs("R + Cpp", list(make_rule('Cpp', 'JS')))
compile_exprs("math.add(a, b)", list(make_rule('math.add', '+')))
```

compile_r *Compile an R file into a JavaScript file*

Description

Compile an R file into a JavaScript file

Usage

```
compile_r(
  input,
  output = "",
  rules = default_rules(),
  deparsers = default_deparsers()
)
```

Arguments

input	A character string; the input file.
output	A character string; the output file. When the output is "", the result is printed to the standard output.
rules	A list of rewriting rules. See [make_rule] for more detail.
deparsers	A list of deparsers. See [make_deparser] for more detail.

Value

A character string; the output file path.

Examples

```
file <- system.file("test_files/test_source.R", package = "sketch")
readLines(file)
compile_r(input = file)
```

convert_src *Convert an asset link into a 'shiny.tag' object*

Description

Convert an asset link into a 'shiny.tag' object

Usage

```
convert_src(x, processors = default_processors())
```

Arguments

x	A character string; the header line (without the prefix #!).
processors	A list of handlers for processing the '#!' header.

Value

A 'shiny.tag' object.

---

default_deparsers *A list of default deparsers for deparsing JavaScript*

Description

A list of default deparsers for deparsing JavaScript

Usage

```
default_deparsers()
```

Note

This is used as input to compile_r and compile_exprs.

Examples

```
default_deparsers()
```

---

default_processors *List of handlers for processing the '#!' header*

Description

List of handlers for processing the '#!' header

Usage

```
default_processors()
```

Note

This is used as input to assets.

Examples

```
default_processors()
```

deparse_js *Expression Deparsing for JavaScript*

Description

Expression Deparsing for JavaScript

Usage

```
deparse_js(ast, deparsers)
```

Arguments

ast	language object.
deparsers	A list of "typed" deparsers.

Value

A character string.

Examples

```
expr_1 <- parse_expr("R.extract(x, 3, )")
deparse_js(expr_1, basic_deparsers())
deparse_js(expr_1, default_deparsers())

expr_2 <- parse_expr("R.data_frame(x = 1, y = 2)")
deparse_js(expr_2, basic_deparsers())
deparse_js(expr_2, default_deparsers())

expr_3 <- parse_expr("lambda(x, x + 1)")
deparse_js(expr_3, basic_deparsers())
```

deparse_sym *Deparsers (specialised)*

Description

Deparsers (specialised)

Deparser for calls

Deparser for infix operators

Deparser for brackets

Deparser for the 'for' keyword

Deparser for the 'if' keyword

Deparser for the 'while' keyword

Deparser for the 'function' keyword

Deparser for the "list" operators

Deparser for the "data.frame" operators

Deparser for the "summarise" operators

Deparser for the "mutate" operators

Deparser for the "new" operator

Deparser for the "let" operator

Deparser for the "dataURI" operator

Deparser for the "ifelse" operator

Deparser for the "lambda" operator

Deparser for the "pipe" operator

Deparser for the "add" operator

Deparser for the "subtract" operator

Deparser for the "extract" operator

Deparser for the "extractAssign" operator

Deparser for the "extract2" operator

Deparser for the "extract2Assign" operator

Usage

```
deparse_sym(ast, ...)

deparse_call(ast, ...)

deparse_infix(ast, ...)

deparse_wrap(ast, ...)

deparse_for(ast, ...)

deparse_if(ast, ...)

deparse_while(ast, ...)

deparse_function(ast, ...)

deparse_list(ast, ...)

deparse_df(ast, ...)

deparse_df_summarise(ast, ...)
```

```
deparse_df_mutate(ast, ...)

deparse_new(ast, ...)

deparse_let(ast, ...)

deparse_dataURI(ast, ...)

deparse_ifelse(ast, ...)

deparse_lambda(ast, ...)

deparse_pipe(ast, ...)

deparse_add(ast, ...)

deparse_subtract(ast, ...)

deparse_extract(ast, ...)

deparse_extractAssign(ast, ...)

deparse_extract2(ast, ...)

deparse_extract2Assign(ast, ...)
```

Arguments

ast	A language object.
...	The contextual information to be passed on to the next call.

Value

A character string.

`eng_sketch` *A language engine for 'sketch'*

Description

This supports the use of 'sketch' code chunk in an R Markdown document.

Usage

```
eng_sketch(options)
```

Arguments

options	A list of chunk options.

Examples

```
# This line makes `sketch::eng_sketch` available to `knitr::knit_engines`.
# It is usually used in the 'setup' code chunk of an R Markdown document
knitr::knit_engines$set(sketch = sketch::eng_sketch)
```

---

html_tags *HTML templates*

Description

A list of 'shiny.tag' objects describing a HTML template. The list must have signature / structure of a named list: `[head = [shiny.tag], body = [shiny.tag]]`

Usage

```
default_tags()

basic_tags()
```

Examples

```
str(default_tags())

str(basic_tags())
```

---

insert_sketch *Insert a 'sketch' app into an R Markdown document*

Description

Insert a 'sketch' app into an R Markdown document

Usage

```
insert_sketch(file, id, output_dir = NULL, render = TRUE, ...)
```

Arguments

file	A character string; the path to the 'sketch' file.
id	A character string; an unique identifier for the 'sketch' file. Needed only when output_dir is not NULL.
output_dir	A character string; a separate directory to save the 'sketch' app. Default to be NULL, which embeds the app in the Rmd file.
render	TRUE or FALSE; if TRUE, call doRenderTags; if FALSE, return the 'shiny.tag' object.
...	(Optional) Other attributes to pass to iframes. Also supports the 'rules', 'deparsers' and 'debug' options to pass to 'source_r'.

Examples

```
# In an R code chunk of an R Markdown document
file <- system.file("test_files/test_RMD.R", package = "sketch")
insert_sketch(file, style = "width:500px; height:500px;", render = FALSE)
```

---

is_call *Predicate for calls*

Description

Predicate for calls

Usage

```
is_call(x, name = NULL, n = NULL, ns = NULL)
```

Arguments

x	An object to test. If a formula, the right-hand side is extracted.
name	An optional name that the call should match. It is passed to sym() before matching. This argument is vectorised and you can supply a vector of names to match. In this case, is_call() returns TRUE if at least one name matches.
n	An optional number of arguments that the call should match.
ns	The namespace of the call. If NULL, the namespace doesn't participate in the pattern-matching. If an empty string "" and x is a namespaced call, is_call() returns FALSE. If any other string, is_call() checks that x is namespaced within ns. Can be a character vector of namespaces, in which case the call has to match at least one of them, otherwise is_call() returns FALSE.

Note

This function is imported from 'rlang'.

`is_sym` *Predicate for symbols, i.e. symbols or syntactic literals*

Description

Predicate for symbols, i.e. symbols or syntactic literals

Predicate for infix operators

Predicate for brackets

Predicate for the 'for' keyword

Predicate for the 'if' keyword

Predicate for the 'while' keyword

Predicate for the 'function' keyword

Predicate for the 'break' keyword

Predicate for the "list" operators

Predicate for the "data.frame" operators

Predicate for the "summarise" operators

Predicate for the "mutate" operators

Predicate for the "new" operator

Predicate for the "let" operator

Predicate for the "dataURI" operator

Predicate for the "ifelse" operator

Predicate for the "lambda" operator

Predicate for the "pipe" operator

Predicate for the "add" operator

Predicate for the "subtract" operator

Predicate for the "extract" operator

Predicate for the "extractAssign" operator

Predicate for the "extract2" operator

Predicate for the "extract2Assign" operator

Usage

```
is_sym(ast)

is_infix(ast)

is_wrap(ast)

is_call_for(ast)
```

```
is_call_if(ast)

is_call_while(ast)

is_call_function(ast)

is_call_break(ast)

is_call_list(ast)

is_call_df(ast)

is_call_df_summarise(ast)

is_call_df_mutate(ast)

is_call_new(ast)

is_call_let(ast)

is_call_dataURI(ast)

is_call_ifelse(ast)

is_call_lambda(ast)

is_call_pipe(ast)

is_call_add(ast)

is_call_subtract(ast)

is_call_extract(ast)

is_call_extractAssign(ast)

is_call_extract2(ast)

is_call_extract2Assign(ast)
```

Arguments

ast	A language object.

`let` *Empty functions*

Description

These functions do nothing. It is created to ensure the keywords ‘let‘ and ‘declare‘ are defined.

Usage

```
let(...)

declare(...)
```

Arguments

...	Any arguments

Examples

```
let (x)
let (x = 1, y = 2)
declare (x1, x2, x3)
```

---

load_library *Header functions*

Description

Header functions

Usage

```
load_library(package, ...)

load_script(src, ...)

load_data(x, cache = tempfile(), ...)
```

Arguments

package	A character string; name of a JavaScript library.
...	Additional arguments to pass to header processor.
src	A character string; the full web/local path to a JavaScript library.
x	A character string; the full path to the file containing the data.
cache	A character string; the full path to the cache file.

make_deparser *A constructor for a "typed" deparser*

Description

A constructor for a "typed" deparser

Usage

```
make_deparser(predicate_fun, deparse_fun)
```

Arguments

predicate_fun	A function that takes a "lang" object and return a logical.
deparse_fun	A function that takes a "lang" object and return a character string.

Value

A list; a deparser ready to be dispatched by "type".

Examples

```
str(make_deparser(predicate_fun = rlang::is_call, deparse_fun = deparse))
```

make_processor *Make a handle to process header*

Description

Make a handle to process header

Usage

```
make_processor(pred, fun)
```

Arguments

pred	A function, taking a string and returning a logical.
fun	A function, taking a string and returning a 'shiny.tag' object.

Value

A header processor / handler.

make_rule *Make a AST transformation rule*

Description

Make a AST transformation rule

Usage

```
make_rule(from, to)
```

Arguments

from	A character string.
to	A character string.

Value

A function that takes a language object and returns a language object.

Examples

```
library(sketch)

rule_1 <- make_rule("pi", "Math.PI")
expr <- rlang::parse_expr("2 * (3 + pi)")

rule_1(expr)  # this works but is not the preferred usage
rewrite(expr, list(rule_1))  # this is preferred

rule_2 <- make_rule("+", "Math.add")
rewrite(expr, list(rule_1, rule_2))
```

parse_expr *Parse R code*

Description

Parse R code

Usage

```
parse_expr(x)
```

Arguments

x	Text containing expressions to parse_expr for `parse_expr()` and `parse_exprs()`. Can also be an R connection, for instance to a file. If the supplied connection is not open, it will be automatically closed and destroyed.

Note

This function is imported from ‘rlang‘.

Examples

```
parse_expr("x <- 1 + 1")
```

---

print.rule *Print function for 'rule' objects*

Description

Print function for 'rule' objects

Usage

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

Arguments

x	A 'rule' object.
...	(Unused) Optional arguments.

Examples

```
rule_1 <- make_rule("+", "Math.add")
print(rule_1)
```

r-to-js-rules *Mapping R operators into JavaScript operators*

Description

Mapping R operators into JavaScript operators

Usage

```
basic_rules()

default_rules()
```

Note

These functions are used as inputs to compile_r and compile_exprs.

References

R operators: https://cran.r-project.org/doc/manuals/r-release/R-lang.html#Operators

R infix and prefix operators: https://cran.r-project.org/doc/manuals/r-release/R-lang.html#Infix-and-prefix-operators

JavaScript operators: https://www.w3schools.com/js/js_operators.asp

Examples

```
basic_rules()

default_rules()
```

rewrite *Interface for AST rewriting*

Description

Interface for AST rewriting

Usage

```
rewrite(ast, rules)
```

Arguments

ast	A language object.
rules	A list of functions, each of which is the output from ‘make_rule‘.

Value

A language object.

Examples

```
library(sketch)

rewrite(
  ast = rlang::parse_expr("2 * (3 + pi)"),
  rules = list(make_rule("pi", "Math.PI"))
)

rewrite(
  ast = rlang::parse_expr("2 + pi"),
  rules = list(
    make_rule("pi", "Math.PI"),
    make_rule("+", "Math.add")
  )
)
```

---

runShinyApp *Run 'Shiny' Application*

Description

Run 'Shiny' Application

Usage

```
runShinyApp()
```

Examples

```
## Not run:
runShinyApp()

## End(Not run)
```

source_active *Source active file in 'RStudio'*

Description

Source active file in 'RStudio'

Usage

```
source_active(...)
```

Arguments

...	Optional arguments to pass to source_r.

Examples

```
## Not run:
# At 'RStudio', opens a 'sketch' R file in the editor, then
# run the following:
source_active()  # This launches the default HTML viewer.

## End(Not run)
```

source_js *Serve a compiled 'sketch' JavaScript file*

Description

Serve a compiled 'sketch' JavaScript file

Usage

```
source_js(
  file,
  debug = FALSE,
  asset_tags = default_tags(),
  launch_browser = "viewer"
)
```

Arguments

file	A character string; path to the compiled JS file.
debug	TRUE or FALSE; if TRUE, a console for debugging is attached to your app.
asset_tags	An optional list of 'shiny.tag' objects to be added to the html template. The list must have signature / structure of a named list: [head = [shiny.tag],body = [shiny.tag]], containing the head and body elements, each of which is a list of shiny.tag object.
launch_browser	A character string; "viewer" or "browser", which calls 'rstudioapi::viewer' and 'utils::browserURL' respectively; use NULL to suppress display.

Examples

```
## Not run:
file <- system.file("test_files/test_source.js", package = "sketch")
# The next line launches the default HTML browser
source_js(file, debug = TRUE, launch_browser = "browser")

## End(Not run)
```

---

source_r *Source a 'sketch' R file*

Description

This function compiles a 'sketch' R file, resolves the dependencies and serves it in the viewer.

Usage

```
source_r(
  file,
  debug = FALSE,
  launch_browser = "viewer",
  asset_tags = default_tags(),
  ...
)
```

Arguments

file	A character string; path to the R file.
debug	TRUE or FALSE; if TRUE, a console for debugging is attached to your app.
launch_browser	A character string; "viewer" or "browser", which calls 'rstudioapi::viewer' and 'utils::browserURL' respectively; use NULL to suppress display.
asset_tags	An optional list of 'shiny.tag' objects to be added to the html template. The list must have signature / structure of a named list: [head = [shiny.tag],body = [shiny.tag]],
...	Additional arguments to pass to 'compile_r'.

Examples

```
## Not run:
file <- system.file("test_files/test_source.R", package = "sketch")
# The next line launches the default HTML browser
source_r(file, debug = TRUE, launch_browser = "browser")

## End(Not run)
```

`src` *Get the source link of a JavaScript library*

Description

Get the source link of a JavaScript library

Usage

```
src(x)
```

Arguments

`x`	A character string; name of the JavaScript library

Value

A character string; the path to the library.

Examples

```
src("mathjs")
src("p5")
```

`to_json` *Convert a file into a JavaScript expression*

Description

It supports csv and json by default and lets users provide custom handlers if other file formats are used.

Usage

```
to_json(input, as_data_frame, read_fun, ...)
```

Arguments

input	A character string; the path to the input file.
as_data_frame	TRUE or FALSE; whether the data are loaded as a data-frame.
read_fun	A function to load the input file. Default settings are provided for CSV files and JSON files. The function has to load a data file into an object that can be handled by 'jsonlite::toJSON'. Possible choices include 'utils::read_delim', 'readr::read_csv2', etc.
...	Extra arguments to be passed to 'read_fun'.

Index