Package ‘sketch’

October 14, 2022

Type Package
Title Interactive Sketches
Version 1.1.17
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
RoxygenNote 7.1.1
Imports magrittr, rlang, purrr, rstudioapi, glue, htmltools, base64enc, jsonlite, shiny, methods
Suggests testthat, covr, knitr, rmarkdown,
BugReports https://github.com/kcf-jackson/sketch
NeedsCompilation no
Author Chun Fung Kwok [aut, cre] (<https://orcid.org/0000-0002-0716-3879>), Kate Saunders [ctb]
Maintainer Chun Fung Kwok <jkwok@svi.edu.au>
Repository CRAN
Date/Publication 2021-10-06 15:10:02 UTC

R topics documented:

  sketch-package .................................................. 2
  assets ................................................................. 3
  basic_deparers .................................................... 3
  bundle ............................................................... 4
  combine_rules ....................................................... 4
  compile_active ..................................................... 5
  compile_data ......................................................... 5
  compile_exprs ....................................................... 6
  compile_r ............................................................ 6
**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**

```r
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**

```r
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**

```r
basic_deparsers()
```

**Note**

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

**Examples**

```r
basic_deparsers()
```
**bundle**  
*Bundle a list of files into a single JavaScript file*

**Description**
Bundle a list of files into a single JavaScript file

**Usage**
bundle(fs)

**Arguments**
- **fs**  
  A character vector; a list of R or JavaScript files. The R files will be transpiled to JavaScript before bundling.

**Examples**
```r
library(sketch)
js <- bundle(c(src("dom"), src("io")))
cat(paste(readLines(js), collapse = "\n"))
```

**combine_rules**  
*Combine rules for fast transpilation*

**Description**
This function turns an n-pass transpilation into k-pass, where n is the number of rules and k is the number of precedence groups.

**Usage**
combine_rules(rs, group = rep(1, length(rs)))

**Arguments**
- **rs**  
  A list of rewriting rules (each of which is an output from **make_rule**).
- **group**  
  A numeric vector; the precedence group. Rules with a higher precedence come before the ones with lower precedence, and they are processed by the transpiler first. For rules with the same precedence, the natural order (in which they show up) determines which rules get processed first.

**Note**
The key insight about optimising the transpilation is that rewriting passes that do not interfere with each other can be combined, and it saves a full traversal of the parse tree.
**compile_active**  

Compile active file in 'RStudio'

**Description**

Compile active file in 'RStudio'

**Usage**

```r
compile_active(...)
```

**Arguments**

...  

Optional arguments to pass to `compile_r`.

**Examples**

```r
## Not run:
# At 'RStudio', opens a 'sketch' R file in the editor, then
# run the following:
compile_active()
## End(Not run)
```

---

**compile_data**  

Compile a data file into a JavaScript file

**Description**

Compile a data file into a JavaScript file

**Usage**

```r
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**

```r
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_deparse())
```

**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_deparse())
```
**convert_src**

**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**

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

**Description**

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

**Usage**

```r
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_2_deparsers**  
* A list of deparsers with additional features

**Description**
Support automatic variable declaration, automatic ‘return’ and shorthand notation for the DOM module.

**Usage**
default_2_deparsers()

**Note**
lifecycle: experimental  
This is used as input to compile_r and compile_exprs.

**Examples**
default_2_deparsers()  

---

**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**

A list of handlers for processing the '!' header

**Description**

A list of handlers for processing the '!' header

**Usage**

default_processors()

**Note**

This is used as input to assets.

**Examples**

default_processors()
Deparser for the "try" keyword
Deparser for the "tryCatch" keyword
Deparser for the "throw" keyword
Deparser for the "list" operator
Deparser for the "data.frame" operators
Deparser for the "summarise" operators
Deparser for the "mutate" operators
Deparser for the "R6Class" function
Deparser for the "new" operator
Deparser for the "typeof" operator
Deparser for the "export" operator
Deparser for the "async" and "await" operators
Deparser for the "let" operator
Deparser for the "const" operator
Deparser for the "var" operator
Deparser for the "dataURI" operator
Deparser for the "ifelse" operator
Deparser for the "lambda" operator
Deparser for the "pipe" operator
Deparser for the "assignment pipe" operator
Deparser for the raw string operator
Deparser for formula
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
Deparser for the HTML tags
Deparser for the d3.js 'attr' function
Deparser for the d3.js 'style' function
Deparser for '.macro'
Deparser for '.data'
Usage

deparse_sym(ast, ...)
deparse_NULL(ast, ...)
deparse_NA(ast, ...)
deparse_NaN(ast, ...)
deparse_call(ast, ...)
deparse_infix(ast, ...)
deparse_wrap(ast, ...)
deparse_for(ast, ...)
deparse_if(ast, ...)
deparse_while(ast, ...)
deparse_function(ast, ...)
deparse_function_with_return(ast, ...)
deparse_return(ast, ...)
deparse_assignment(ast, ...)
deparse_assignment_auto(ast, ...)
deparse_next(ast, ...)
deparse_try(ast, ...)
deparse_tryCatch(ast, ...)
deparse_throw(ast, ...)
deparse_list(ast, ...)
deparse_df(ast, ...)
deparse_df_summarise(ast, ...)
deparse_df_mutate(ast, ...)
deparse_R6Class(ast, ...)

deparse_new(ast, ...)  
deparse_typeof(ast, ...)  
deparse_export(ast, ...)  
deparse_async_await(ast, ...)  
deparse LET(ast, ...)  
deparse CONST(ast, ...)  
deparse_VAR(ast, ...)  
deparse_dataURI(ast, ...)  
deparse_ifelse(ast, ...)  
deparse_lambda(ast, ...)  
deparse_pipe(ast, ...)  
deparse_assignment_pipe(ast, ...)  
deparse_raw_string(ast, ...)  
deparse_formula(ast, ...)  
deparse_add(ast, ...)  
deparse_subtract(ast, ...)  
deparse_extract(ast, ...)  
deparse_extractAssign(ast, ...)  
deparse_extract2(ast, ...)  
deparse_extract2Assign(ast, ...)  
deparse_html_tags(ast, ...)  
deparse_d3_attr(ast, ...)  
deparse_d3_style(ast, ...)  
deparse_macro(ast, ...)
deparse_data(ast, ...)

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

Value
A character string.

Note
At the moment, the '.macro' / 'deparse_macro' function must be used with the 'compile_exprs' call. This is currently an experimental feature.

At the moment, the '.data' / 'deparse_data' function must be used with the 'compile_exprs' call. This is currently an experimental feature.

Expression deparsing for JavaScript

Description
This is the "master" deparser that dispatches the "worker" deparsers based on the type of the input.

Usage
deparse_js(ast, deparsers)

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

Value
A character string.
Examples

```r
eexpr_1 <- parse_expr("R.extract(x, 3, )")
deparse_js(expr_1, basic_deparsers())
deparse_js(expr_1, default_deparsers())

eexpr_2 <- parse_expr("R.data_frame(x = 1, y = 2)")
deparse_js(expr_2, basic_deparsers())
deparse_js(expr_2, default_deparsers())

eexpr_3 <- parse_expr("lambda(x, x + 1)")
deparse_js(expr_3, basic_deparsers())
```

---

dp

*Constructor function to combine low-level deparsers*

Description

Constructor function to combine low-level deparsers

Usage

dp(...)

Arguments

... character strings indicating the features needed of the deparsers. The supported features are "basic", "r", "auto" and "dom" corresponding to the basic deparsers, the R support, the automatic variable declaration and return, and the dom shorthand notation.

Note

lifecycle: experimental

---

eng_sketch

*A language engine for `sketch`*

Description

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

Usage

eeng_sketch(options)
**flatten_filelist**

Flatten a list of files and directories into a list of files

**Usage**

```r
flatten_filelist(fs, pattern = NULL, ...)
```

**Arguments**

- `fs` A character vector; a list of files.
- `pattern` An optional regular expression to pass to `list.files` for filtering files while expanding a directory into a list of files.
- `...` Additional parameters to pass to `list.files`.

**Examples**

```r
modules_dir <- system.file("modules", package = "sketch")
flatten_filelist(modules_dir)
```

---

**get_dependencies**

Extract the content of the `load_script` headers of a sketch R file

**Description**

Extract the content of the `load_script` headers of a sketch R file

**Usage**

```r
get_dependencies(app_script, local_only = TRUE)
```
Arguments

- `app_script` A character string; the path to the sketch R file.
- `local_only` TRUE / FALSE; if TRUE, exclude the ones that are web link.

Examples

```r
sample_file <- system.file("test_files/test_sketch.R", package = "sketch")
cat(readLines(sample_file), sep = "\n")  # Preview the file content
get_dependencies(sample_file)
```

---

### 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**

```r
default_tags(local = TRUE)
```

```
str(default_tags())
```

**Arguments**

- `local` TRUE / FALSE. If TRUE, the R base module is loaded from the local file stored in the package, otherwise, the module is served via a content delivery network (CDN).

**Examples**

```r
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**

```r
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', 'de-parsers' and 'debug' options to pass to 'source_r'.

**Value**

An HTML string if `render` is TRUE, or a 'shiny.tag' object if `render` is FALSE.

**Examples**

```r
# 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**

```r
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 \texttt{sym()} before matching. This argument is vectorised and you can supply a vector of names to match. In this case, \texttt{is\_call()} returns \texttt{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 \texttt{x} is a namespaced call, \texttt{is\_call()} returns \texttt{FALSE}. If any other string, \texttt{is\_call()} checks that \texttt{x} is namespaced within \texttt{ns}.

Can be a character vector of namespaces, in which case the call has to match at least one of them, otherwise \texttt{is\_call()} returns \texttt{FALSE}.

Note

This function is imported from ‘rlang’.

\begin{itemize}
\item \texttt{is\_Family}
\item \textit{Predicate for symbols, i.e. symbols or syntactic literals}
\end{itemize}

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 return
Predicate for assignments
Predicate for assignments
Predicate for the ”break” keyword
Predicate for the ”next” keyword
Predicate for the ”try” keyword
Predicate for the ”tryCatch” keyword
Predicate for the ”throw” keyword
Predicate for the ”list” operator
Predicate for the ”data.frame” operators
Predicate for the ”summarise” operators
Predicate for the "mutate" operators
Predicate for the "R6Class" function
Predicate for the "new" operator
Predicate for the "typeof" operator
Predicate for the "export" operator
Predicate for the "async" and "await" operators
Predicate for the "let" operator
Predicate for the "const" operator
Predicate for the "var" operator
Predicate for the "dataURI" operator
Predicate for the "ifelse" operator
Predicate for the "lambda" operator
Predicate for the "pipe" operator
Predicate for the "assignment pipe" operator
Predicate for the raw string operator
Predicate for formula
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
Predicate for the HTML tags
Predicate for the d3.js 'attr' function
Predicate for the d3.js 'style' function
Predicate for '.macro'
Predicate for '.data'

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_return(ast)
is_call_assignment(ast)
is_call_assignment_auto(ast)
is_call_break(ast)
is_call_next(ast)
is_call_try(ast)
is_call_tryCatch(ast)
is_call_throw(ast)
is_call_list(ast)
is_call_df(ast)
is_call_df_summarise(ast)
is_call_df_mutate(ast)
is_call_R6Class(ast)
is_call_new(ast)
is_call_typeof(ast)
is_call_export(ast)
is_call_async_await(ast)
is_call_let(ast)
is_call_const(ast)
is_call_var(ast)
is_call_dataURI(ast)
is_call_ifelse(ast)
is_syntactic_literal

is_call_lambda(ast)
is_call_pipe(ast)
is_call_assignment_pipe(ast)
is_call_raw_string(ast)
is_call_formula(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)
is_html_tags(ast)
is_d3_attr(ast)
is_d3_style(ast)
is_macro(ast)
is_data(ast)

Arguments

ast A language object.

Description

Predicate for syntactic literal

Usage

is_syntactic_literal(x)
Arguments
x An object to test.

Note
This function is imported from ‘rlang’.

let-declare-const Empty functions

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

Usage
let(...) declare(...) const(...)

Arguments
... Any arguments

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

license_info License information

Description
License information

Usage
license_info(x)

Arguments
x A character string; name of the library / assets.
Value

A named list containing the license information and the link from which the information is extracted.

Examples

```r
license_info("mathjs")
license_info("p5")
```

| list-of-deparsers | Low-level lists of deparsers |

Description

Support of R functionalities

Usage

```r
dp_r_support()
dp_auto()
dp_dom()
dp_d3()
dp_macro()
```

| load_Family | Header functions |

Description

Header functions

Usage

```r
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.

---

**local**  
*A helper function to enable debugger option*

---

**Description**

A helper function to enable debugger option

**Usage**

```r
local(x, from_local = TRUE)
```

**Arguments**

- **x**: TRUE / FALSE; whether to attach a debugging console to the sketch application.
- **from_local**: TRUE / FALSE; whether to load the debugger console from the local package. If FALSE, the console will be loaded from a Content Delivery Network (CDN) link.

**Note**

Use ‘from_local=TRUE’ for self-contained applications, and ‘from_local=FALSE’ for reduced file size.

**Examples**

```r
# This function is designed to be used in the configuration header, e.g.
# config(debug = local(TRUE), rules = basic_rules(), deparsers = basic_deparsers())
local(TRUE)
```
**make_deparser**

A constructor for a "typed" deparser

**Description**

A constructor for a "typed" deparser

**Usage**

```r
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**

```r
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**

```r
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**

```r
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**

```r
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**

```r
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.sketch_rule  Print function for 'sketch_rule' objects

Description

Print function for 'sketch_rule' objects

Usage

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

Arguments

x  A 'sketch_rule' object.

...  (Unused) Optional arguments.

Examples

library(sketch)
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()

read_multilines  

Read one or more lines from the console for the first successful parse

Description

read_multilines reads one or more lines from the terminal (in interactive use).

Usage

read_multilines(prompt = "")

Arguments

prompt  

the string printed when prompting the user for input. Should usually end with a space " ".

Details

This function repeatedly calls `readline` until enough inputs are provided to reach a successful parse.

This can only be used in an interactive session.

Examples

```r
## Not run:
# In an interactive session
read_multilines()
1 + 2  # expect immediate success

read_multilines()
1 +
2 +
3  # expect success here

## End(Not run)
```

---

**rewrite**

*Interface for AST rewriting*

Description

Interface for AST rewriting

Usage

```r
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

```r
library(sketch)

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


```r
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**  
```r
## Not run:
runShinyApp()

## End(Not run)
```

---

**safeguard**  
*Perform pre-transpilation check*

**Description**  
Perform pre-transpilation check

**Usage**  
safeguard(ast, rules, deparsers)

**Arguments**
- `ast`: A language object.
- `rules`: A list of functions; the rewriting rules, each of which is the output from `make_rule`.
- `deparsers`: A list of functions; the deparsers, each of which is the output from `make_deparser`. 
Value

TRUE when the check is complete.

Examples

# Expect no warning
safeguard(parse_expr("a <- 3"),
    rules = default_rules(),
    deparsers = default_deparsers())

# Expect a warning (as 'max' is reserved to be a function by the transpiler)
safeguard(parse_expr("max <- 3"),
    rules = default_rules(),
    deparsers = default_deparsers())

source_active

Description

Source active file in 'RStudio'

Usage

source_active(...)  # This launches the default HTML viewer.

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)

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::browseURL` respectively; use NULL to suppress display.

Examples

```r
## 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")
```

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, asset_tags = default_tags(), ...)
split_rules

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::browseURL` 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

```r
## 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)
```

split_rules

*Split rules for customisation*

Description

This function is the left-inverse of `combine_rules`, i.e. `split_rules(combine_rules(rs, group)) = rs` for any variable ‘group’. It is created to facilitate the addition or removal of rewriting rules.

Usage

`split_rules(rs)`

Arguments

- **rs**: A list of (grouped) rewriting rules. Note that a list of n rules without grouping is a list of n groups of single rule.
### 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")
```

---

### test_sketch

*Test a sketch application*

**Description**

Test a sketch application

**Usage**

```
test_sketch(app_script, test_script, port = 9454, ...)
```

**Arguments**

- `app_script` A character string; the file path to the app.  
- `test_script` A character string; the file path to the tests.  
- `port` An integer to pass to `websocket$new()`.  
- `...` Additional arguments to pass to `source_r`.

**Value**

A "websocket" object.
### Examples

```r
## Not run:
app_file <- system.file("test_files/test_testthat_app.R", package = "sketch")
test_file <- system.file("test_files/test_testthat_test.R", package = "sketch")
# This following command will launch the default browser
res <- test_sketch(app_file, test_file)
## End(Not run)
```

---

#### 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

```r
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`.

---

#### websocket

*Websocket for 'sketch' applications*

#### Description

This combines the *-Server family of functions in 'httpuv' with the transpilation functionality provided by 'sketch'.

**Public fields**

- **app** A list of functions that define the application.
- **server** A server handle to be used by `stopServer`.
- **log** A character vector that keep tracks of all the commands sent to the browser session.
- **ws** A WebSocket channel to handle the communication between the R session and the browser session.
- **in_handler** A function to handle instructions sent by the browser session.
- **out_handler** A function to handle instructions sent to the browser session.
- **env** An environment to store variables temporarily.
- **port** An integer; the TCP port number.
- **message** TRUE or FALSE; whether to display a prompt when a server is started and when it is stopped.
- **connected** TRUE or FALSE; whether a connection has been established. One should always start the WebSocket server before visiting the web page that connects to the server.
- **started** TRUE or FALSE; whether a server has been started. Use the `startServer` method to start a server.

**Methods**

**Public methods:**

- `websocket$startServer()`
- `websocket$stopServer()`
- `websocket$listServers()`
- `websocket$stopAllServers()`
- `websocket$sketch_mode()`
- `websocket$new_app()`
- `websocket$new()`
- `websocket$clone()`

**Method** `startServer()`: Start a WebSocket server

*Usage:*

```
websocket$startServer()
```

**Method** `stopServer()`: Stop a WebSocket server

*Usage:*

```
websocket$stopServer()
```

**Method** `listServers()`: List all running WebSocket servers

*Usage:*

```
websocket$listServers()
```

**Method** `stopAllServers()`: Stop all running WebSocket servers

*Usage:*

```
websocket$stopAllServers()
```
websocket$stopAllServers()

**Method** sketch_mode(): Enter sketch mode, in which all commands go through the transpiler before reaching the browser session.

*Usage:*

websocket$sketch_mode()

**Method** new_app(): Create a blank HTML page with interactive access. This function is designed for newcomers.

*Usage:*

websocket$new_app(
    preamble = list(library = c(), script = c(), data = c()),
    ...
)

*Arguments:*
preamble (Optional) A named list; the preamble to include. Use the name 'lib' for arguments to load_library, 'script' for arguments to load_script and 'data' for arguments to load_data. Note that the "dom" and "websocket" modules are required and loaded by default.

... Extra parameters to pass to source_r.

*Returns:* The (invisible) temporary file path to the app.

**Method** new(): Initialise a WebSocket connection

*Usage:*

websocket$new(in_handler, out_handler, message = TRUE, port = 9454)

*Arguments:*
in_handler A function to handle incoming message, default to be print which only displays the message without any processing.

out_handler A function to handle outgoing message, default to be compile_exprs which transpiles R commands into JavaScript commands.

message TRUE or FALSE; whether to display a prompt when a server is started and when it is stopped.

port An integer; the TCP port number.

*Returns:* A 'websocket' object.

*Examples:*

\dontrun{
    # Launch a WebSocket server
    ws <- websocket$new()
    ws$startServer()
    ws$listServers() # Check that a server is running

    # Launch a 'sketch' application with WebSocket functionality
    file <- system.file("test_files/test_websocket.R", package = "sketch")
    source_r(file, debug = TRUE) # Launch the default browser
# Enter sketch mode to send commands to the application
ws$sketch_mode()
# Within sketch mode
print("1234")
  x <- 10
  print(x + 1)
  q()

# Back to normal mode, inspect the log and stop the server
ws$log
ws$stopServer()
ws$listServers()  # Confirm no server is running

Method clone(): The objects of this class are cloneable with this method.

Usage:
websocket$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

Examples

## Method `websocket$new`

## Not run:
# Launch a WebSocket server
ws <- websocket$new()
ws$startServer()
ws$listServers()  # Check that a server is running

# Launch a 'sketch' application with WebSocket functionality
file <- system.file("test_files/test_websocket.R", package = "sketch")
source_r(file, debug = TRUE)  # Launch the default browser

# Enter sketch mode to send commands to the application
ws$sketch_mode()
# Within sketch mode
print("1234")
  x <- 10
  print(x + 1)
  q()

# Back to normal mode, inspect the log and stop the server
ws$log
ws$stopServer()
ws$listServers()  # Confirm no server is running
## End (Not run)
Index

assets, 3, 9
basic_deparsers, 3
basic_rules (r-to-js-rules), 28
basic_tags (html_tags), 16
bundle, 4

combine_rules, 4
compile_active, 5
compile_data, 5
compile_exprs, 3, 6, 8, 28, 37
compile_r, 3, 6, 8, 28
const (let-declare-const), 22
convert_src, 3, 7

declare (let-declare-const), 22
default_2_deparsers, 8
default_deparsers, 8
default_processors, 9
default_rules (r-to-js-rules), 28
default_tags (html_tags), 16
deparse_add (deparse_Family), 9
deparse_assignment (deparse_Family), 9
deparse_assignment_auto (deparse_Family), 9
deparse_assignment_pipe (deparse_Family), 9
deparse_asyncAwait (deparse_Family), 9
deparse_call (deparse_Family), 9
deparse_const (deparse_Family), 9
deparse_d3_attr (deparse_Family), 9
deparse_d3_style (deparse_Family), 9
deparse_data (deparse_Family), 9
deparse_dataURI (deparse_Family), 9
deparse_df (deparse_Family), 9
deparse_df_mutate (deparse_Family), 9
deparse_df_summarise (deparse_Family), 9
deparse_export (deparse_Family), 9
deparse_extract (deparse_Family), 9
deparse_extract2 (deparse_Family), 9
deparse_extract2Assign (deparse_Family), 9
deparse_extractAssign (deparse_Family), 9
deparse_Family, 9
deparse_for (deparse_Family), 9
deparse_formula (deparse_Family), 9
deparse_function (deparse_Family), 9
deparse_function_with_return (deparse_Family), 9
deparse_html_tags (deparse_Family), 9
deparse_if (deparse_Family), 9
deparse_ifelse (deparse_Family), 9
deparse_infix (deparse_Family), 9
deparse_js, 3, 13
deparse_lambda (deparse_Family), 9
deparse_list (deparse_Family), 9
deparse_macro (deparse_Family), 9
deparse_NA (deparse_Family), 9
deparse_NaN (deparse_Family), 9
deparse_new (deparse_Family), 9
deparse_next (deparse_Family), 9
deparse_NULL (deparse_Family), 9
deparse_pipe (deparse_Family), 9
deparse_R6Class (deparse_Family), 9
deparse_raw_string (deparse_Family), 9
deparse_return (deparse_Family), 9
deparse_subtract (deparse_Family), 9
deparse_sym (deparse_Family), 9
deparse_throw (deparse_Family), 9
deparse_try (deparse_Family), 9
deparse_tryCatch (deparse_Family), 9
deparse_typeof (deparse_Family), 9
deparse_var (deparse_Family), 9
deparse_while (deparse_Family), 9
deparse_wrap (deparse_Family), 9
doRenderTags, 17
dp, 14
富豪（list-of-deparsers), 23
富豪_d3 (list-of-deparsers), 23
富豪_dom (list-of-deparsers), 23
富豪_macro (list-of-deparsers), 23
富豪_r_support (list-of-deparsers), 23
富豪_sketch, 14
富豪_flatten_filelist, 15
富豪_get_dependencies, 15
富豪_html_tags, 16
富豪_insert_sketch, 17
富豪_is_call (is_Family), 18
富豪_is_call_add (is_Family), 18
富豪_is_call_assignment (is_Family), 18
富豪_is_call_assignment_auto (is_Family), 18
富豪_is_call_assignment_pipe (is_Family), 18
富豪_is_call_async_await (is_Family), 18
富豪_is_call_break (is_Family), 18
富豪_is_call_const (is_Family), 18
富豪_is_call_dataURI (is_Family), 18
富豪_is_call_df (is_Family), 18
富豪_is_call_df_mutate (is_Family), 18
富豪_is_call_df_summarise (is_Family), 18
富豪_is_call_export (is_Family), 18
富豪_is_call_extract (is_Family), 18
富豪_is_call_extract2 (is_Family), 18
富豪_is_call_extract2Assign (is_Family), 18
富豪_is_call_extractAssign (is_Family), 18
富豪_is_call_extractAssign_pipe (is_Family), 18
富豪_is_call_for (is_Family), 18
富豪_is_call_formula (is_Family), 18
富豪_is_call_function (is_Family), 18
富豪_is_call_if (is_Family), 18
富豪_is_call_ifelse (is_Family), 18
富豪_is_call_lambda (is_Family), 18
富豪_is_call_let (is_Family), 18
富豪_is_call_list (is_Family), 18
富豪_is_call_new (is_Family), 18
富豪_is_call_next (is_Family), 18
富豪_is_call_pipe (is_Family), 18
富豪_is_call_R6Class (is_Family), 18
富豪_is_call_raw_string (is_Family), 18
富豪_is_call_return (is_Family), 18
富豪_is_call_subtract (is_Family), 18
富豪_is_call_throw (is_Family), 18
富豪_is_call_try (is_Family), 18
富豪_is_call_tryCatch (is_Family), 18
富豪_is_call_typeof (is_Family), 18
富豪_is_call_var (is_Family), 18
富豪_is_call_while (is_Family), 18
富豪_is_d3_attr (is_Family), 18
富豪_is_d3_style (is_Family), 18
富豪_is_data (is_Family), 18
富豪_is_Family, 18
富豪_is_html_tags (is_Family), 18
富豪_is_infix (is_Family), 18
富豪_is_macro (is_Family), 18
富豪_is_sym (is_Family), 18
富豪_is_syntactic_literal, 21
富豪_is_wrap (is_Family), 18
富豪_load_data (load_Family), 23
富豪_load_Family, 23
富豪_load_library (load_Family), 23
富豪_load_script (load_Family), 23
富豪_local, 24
富豪_make_deparser, 6, 7, 25, 30
富豪_make_processor, 25
富豪_make_rule, 4, 6, 7, 26, 30
富豪_parse_expr, 26
富豪_print, 37
富豪_print.sketch_rule, 27
富豪_r-to-js-rules, 28
富豪_read_multilines, 28
富豪_rewrite, 29
富豪_runShinyApp, 30
富豪_safeguard, 30
富豪_sketch (sketch-package), 2
富豪_sketch-package, 2
富豪_source_active, 31
富豪_source_js, 32
富豪_source_r, 32, 34, 37
富豪_split_rules, 33
富豪_src, 34
富豪_sym(), 18
富豪_test_sketch, 34
富豪_to_json, 34
websocket, 35