

Package: teal.reporter (via r-universe)

October 16, 2024

Title Reporting Tools for 'shiny' Modules

Version 0.3.1.9016

Date 2024-09-16

Description Prebuilt 'shiny' modules containing tools for the generation of 'rmarkdown' reports, supporting reproducible research and analysis.

License Apache License 2.0

URL <https://github.com/insightsengineering/teal.reporter>,
<https://insightsengineering.github.io/teal.reporter/>

BugReports <https://github.com/insightsengineering/teal.reporter/issues>

Imports bslib, checkmate ($\geq 2.1.0$), flextable ($\geq 0.9.2$), grid, htmltools ($\geq 0.5.4$), knitr (≥ 1.42), lifecycle ($\geq 0.2.0$), R6, rlistings ($\geq 0.2.8$), rmarkdown (≥ 2.23), rtables ($\geq 0.6.6$), shiny ($\geq 1.6.0$), shinybusy ($\geq 0.3.2$), shinyWidgets ($\geq 0.5.1$), yaml ($\geq 1.1.0$), zip ($\geq 1.1.0$)

Suggests DT (≥ 0.13), formatR (≥ 1.5), formatters, ggplot2 ($\geq 3.4.3$), lattice ($\geq 0.18-4$), png, testthat ($\geq 3.1.5$), tinytex, withr ($\geq 2.0.0$)

VignetteBuilder knitr, rmarkdown

RdMacros lifecycle

Config/Needs/verdepcheck rstudio/bslib, mllg/checkmate, davidgohel/flextable, rstudio/htmltools, yihui/knitr, r-lib/lifecycle, r-lib/R6, insightsengineering/rlistings, rstudio/rmarkdown, insightsengineering/rtables, rstudio/shiny, dreamRs/shinybusy, dreamRs/shinyWidgets, yaml=vubiostat/r-yaml, r-lib/zip, rstudio/DT, yihui/formatR, insightsengineering/formatters, tidyverse/ggplot2, deepayan/lattice, cran/png, r-lib/testthat, rstudio/tinytex, r-lib/withr

Config/Needs/website insightsengineering/nesttemplate

Encoding UTF-8

Language en-US
LazyData true
Roxygen list(markdown = TRUE)
RoxygenNote 7.3.2
Repository https://pharmaverse.r-universe.dev
RemoteUrl https://github.com/insightsengineering/teal.reporter
RemoteRef HEAD
RemoteSha 4fefa7ebac08caac2ca7b6588e497613ac4676fc

Contents

add_card_button	2
as_yaml_auto	3
download_report_button	5
print.rmd_yaml_header	6
ReportCard	7
Reporter	13
reporter_previewer	20
report_load_srv	22
report_load_ui	22
reset_report_button	23
rmd_outputs	23
rmd_output_arguments	24
simple_reporter	24

Index	26
--------------	-----------

add_card_button	<i>Add card button module</i>
-----------------	-------------------------------

Description

[Experimental]

Provides a button to add views/cards to a report.

For more details see the vignette: `vignette("simpleReporter", "teal.reporter")`.

Usage

```
add_card_button_ui(id)
```

```
add_card_button_srv(id, reporter, card_fun)
```

Arguments

id	(character(1)) this shiny module's id.
reporter	(Reporter) instance.
card_fun	(function) which returns a ReportCard instance. See Details.

Details

The card_fun function is designed to create a new ReportCard instance and optionally customize it:

- The card parameter allows for specifying a custom or default ReportCard instance.
- Use the comment parameter to add a comment to the card via card\$append_text() - if card_fun does not have the comment parameter, then comment from Add Card UI module will be added at the end of the content of the card.
- The label parameter enables customization of the card's name and its content through card\$append_text()- if card_fun does not have the label parameter, then card name will be set to the name passed in Add Card UI module, but no text will be added to the content of the card.

This module supports using a subclass of [ReportCard](#) for added flexibility. A subclass instance should be passed as the default value of the card argument in the card_fun function. See below:

```
CustomReportCard <- R6::R6Class(
  classname = "CustomReportCard",
  inherit = teal.reporter::ReportCard
)

custom_function <- function(card = CustomReportCard$new()) {
  card
}
```

Value

NULL.

as_yaml_auto

Parse a named list to yaml header for an Rmd file

Description**[Experimental]**

Converts a named list into a yaml header for Rmd, handling output types and arguments as defined in the rmarkdown package. This function simplifies the process of generating yaml headers.

Usage

```
as_yaml_auto(
  input_list,
  as_header = TRUE,
  convert_logi = TRUE,
  multi_output = FALSE,
  silent = FALSE
)
```

Arguments

input_list	(named list) non nested with slots names and their values compatible with Rmd yaml header.
as_header	(logical(1)) optionally wrap with result with the internal md_header(), default TRUE.
convert_logi	(logical(1)) convert a character values to logical, if they are recognized as quoted yaml logical values , default TRUE.
multi_output	(logical(1)) multi output slots in the input argument, default FALSE.
silent	(logical(1)) suppress messages and warnings, default FALSE.

Details

This function processes a non-nested (flat) named list into a yaml header for an Rmd document. It supports all standard Rmd yaml header fields, including author, date, title, subtitle, abstract, keywords, subject, description, category, and lang. Additionally, it handles output field types and arguments as defined in the rmarkdown package.

Value

character with rmd_yaml_header class, result of `yaml::as.yaml`, optionally wrapped with internal `md_header()`.

Note

Only non-nested lists are automatically parsed. Nested lists require direct processing with `yaml::as.yaml`.

Examples

```
# nested so using yaml::as.yaml directly
as_yaml_auto(
  list(author = "", output = list(pdf_document = list(toc = TRUE)))
)

# auto parsing for a flat list, like shiny input
input <- list(author = "", output = "pdf_document", toc = TRUE, keep_tex = TRUE)
as_yaml_auto(input)

as_yaml_auto(list(author = "", output = "pdf_document", toc = TRUE, keep_tex = "TRUE"))
```

```

as_yaml_auto(list(
  author = "", output = "pdf_document", toc = TRUE, keep_tex = TRUE,
  wrong = 2
))

as_yaml_auto(list(author = "", output = "pdf_document", toc = TRUE, keep_tex = 2),
  silent = TRUE
)

input <- list(author = "", output = "pdf_document", toc = TRUE, keep_tex = "True")
as_yaml_auto(input)
as_yaml_auto(input, convert_logi = TRUE, silent = TRUE)
as_yaml_auto(input, silent = TRUE)
as_yaml_auto(input, convert_logi = FALSE, silent = TRUE)

as_yaml_auto(
  list(
    author = "", output = "pdf_document",
    output = "html_document", toc = TRUE, keep_tex = TRUE
  ),
  multi_output = TRUE
)
as_yaml_auto(
  list(
    author = "", output = "pdf_document",
    output = "html_document", toc = "True", keep_tex = TRUE
  ),
  multi_output = TRUE
)

```

download_report_button

Download report button module

Description

[Experimental]

Provides a button that triggers downloading a report.

For more information, refer to the vignette: `vignette("simpleReporter", "teal.reporter")`.

Usage

```
download_report_button_ui(id)
```

```

download_report_button_srv(
  id,
  reporter,
  global_knitr = getOption("teal.reporter.global_knitr"),
  rmd_output = c(html = "html_document", pdf = "pdf_document", powerpoint =

```

```

    "powerpoint_presentation", word = "word_document"),
  rmd_yaml_args = list(author = "NEST", title = "Report", date =
    as.character(Sys.Date()), output = "html_document", toc = FALSE)
)

```

Arguments

<code>id</code>	(character(1)) this shiny module's id.
<code>reporter</code>	(Reporter) instance.
<code>global_knitr</code>	(list) of knitr parameters (passed to <code>knitr::opts_chunk\$set</code>) for customizing the rendering process.
<code>rmd_output</code>	(character) vector with rmarkdown output types, by default all possible <code>pdf_document</code> , <code>html_document</code> , <code>powerpoint_presentation</code> , and <code>word_document</code> . If vector is named then those names will appear in the UI.
<code>rmd_yaml_args</code>	(named list) with Rmd yaml header fields and their default values. This list will result in the custom subset of UI inputs for the download reporter functionality. Default <code>list(author = "NEST", title = "Report", date = Sys.Date(), output = "html_document", toc = FALSE)</code> . The list must include at least "output" field. The default value for "output" has to be in the <code>rmd_output</code> argument.

Details

To access the default values for the `global_knitr` parameter, use `getOption('teal.reporter.global_knitr')`. These defaults include:

- `echo = TRUE`
- `tidy.opts = list(width.cutoff = 60)`
- `tidy = TRUE` if `formatR` package is installed, `FALSE` otherwise

Value

NULL.

`print.rmd_yaml_header` *Print method for the `yaml_header` class*

Description

[Experimental]

Usage

```

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

```

Arguments

`x` (rmd_yaml_header) class object.
`...` optional text.

Value

NULL.

Examples

```
input <- list(author = "", output = "pdf_document", toc = TRUE, keep_tex = TRUE)
out <- as_yaml_auto(input)
out
print(out)
```

ReportCard

ReportCard: *An R6 class for building report elements*

Description**[Experimental]**

This R6 class that supports creating a report card containing text, plot, table and metadata blocks that can be appended and rendered to form a report output from a shiny app.

For more information about the various blocks, refer to the vignette: `vignette("teal-reporter-blocks-overview", "teal.reporter")`.

Methods**Public methods:**

- `ReportCard$new()`
- `ReportCard$append_table()`
- `ReportCard$append_plot()`
- `ReportCard$append_text()`
- `ReportCard$append_rcode()`
- `ReportCard$append_content()`
- `ReportCard$get_content()`
- `ReportCard$reset()`
- `ReportCard$get_metadata()`
- `ReportCard$append_metadata()`
- `ReportCard$get_name()`
- `ReportCard$set_name()`
- `ReportCard$to_list()`
- `ReportCard$from_list()`
- `ReportCard$clone()`

Method `new()`: Initialize a ReportCard object.

Usage:

```
ReportCard$new()
```

Returns: Object of class ReportCard, invisibly.

Examples:

```
card <- ReportCard$new()
```

Method `append_table()`: Appends a table to this ReportCard.

Usage:

```
ReportCard$append_table(table)
```

Arguments:

table A (data.frame or rtables or TableTree or ElementaryTable or listing_df) that can be coerced into a table.

Returns: self, invisibly.

Examples:

```
card <- ReportCard$new()$append_table(iris)
```

Method `append_plot()`: Appends a plot to this ReportCard.

Usage:

```
ReportCard$append_plot(plot, dim = NULL)
```

Arguments:

plot (ggplot or grob or trellis) plot object.

dim (numeric(2)) width and height in pixels.

Returns: self, invisibly.

Method `append_text()`: Appends a text paragraph to this ReportCard.

Usage:

```
ReportCard$append_text(text, style = TextBlock$new()$get_available_styles()[1])
```

Arguments:

text (character) The text content to add.

style (character(1)) the style of the paragraph. One of: default, header, verbatim

Returns: self, invisibly.

Examples:

```
card <- ReportCard$new()$append_text("A paragraph of default text")
```

Method `append_rcode()`: Appends an R code chunk to ReportCard.

Usage:

```
ReportCard$append_rcode(text, ...)
```


Arguments:

`text` (character) The R code to include.
... Additional rmarkdown parameters for formatting the R code chunk.

Returns: `self`, invisibly.

Examples:

```
card <- ReportCard$new()$append_rcode("2+2", echo = FALSE)
```

Method `append_content()`: Appends a generic ContentBlock to this ReportCard.

Usage:

```
ReportCard$append_content(content)
```

Arguments:

`content` (ContentBlock) object.

Returns: `self`, invisibly.

Examples:

```
NewpageBlock <- getFromNamespace("NewpageBlock", "teal.reporter")  
card <- ReportCard$new()$append_content(NewpageBlock$new())
```

Method `get_content()`: Get all content blocks from this ReportCard.

Usage:

```
ReportCard$get_content()
```

Returns: `list()` list of TableBlock, TextBlock and PictureBlock.

Examples:

```
card <- ReportCard$new()$append_text("Some text")$append_metadata("rc", "a <- 2 + 2")  
  
card$get_content()
```

Method `reset()`: Clears all content and metadata from ReportCard.

Usage:

```
ReportCard$reset()
```

Returns: `self`, invisibly.

Method `get_metadata()`: Get the metadata associated with ReportCard.

Usage:

```
ReportCard$get_metadata()
```

Returns: named `list` list of elements.

Examples:

```
card <- ReportCard$new()$append_text("Some text")$append_metadata("rc", "a <- 2 + 2")
card$get_metadata()
```

Method `append_metadata()`: Appends metadata to this ReportCard.

Usage:

```
ReportCard$append_metadata(key, value)
```

Arguments:

`key` (character(1)) string specifying the metadata key.

`value` value associated with the metadata key.

Returns: self, invisibly.

Method `get_name()`: Get the name of the ReportCard.

Usage:

```
ReportCard$get_name()
```

Returns: character a card name.

Examples:

```
ReportCard$new()$set_name("NAME")$get_name()
```

Method `set_name()`: Set the name of the ReportCard.

Usage:

```
ReportCard$set_name(name)
```

Arguments:

`name` (character(1)) a card name.

Returns: self, invisibly.

Examples:

```
ReportCard$new()$set_name("NAME")$get_name()
```

Method `to_list()`: Convert the ReportCard to a list, including content and metadata.

Usage:

```
ReportCard$to_list(output_dir)
```

Arguments:

`output_dir` (character) with a path to the directory where files will be copied.

Returns: (named list) a ReportCard representation.

Method `from_list()`: Reconstructs the ReportCard from a list representation.

Usage:

```
ReportCard$from_list(card, output_dir)
```

Arguments:

`card` (named list) a ReportCard representation.

`output_dir` (character) with a path to the directory where a file will be copied.

Returns: self, invisibly.

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

Usage:

```
ReportCard$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
library(ggplot2)

card <- ReportCard$new()$append_plot(
  ggplot(iris, aes(x = Petal.Length)) + geom_histogram()
)

library(ggplot2)

card <- ReportCard$new()$append_text("Some text")$append_plot(
  ggplot(iris, aes(x = Petal.Length)) + geom_histogram()
)$append_text("Some text")$append_metadata(key = "lm",
  value = lm(Ozone ~ Solar.R, airquality))
card$get_content()
card$get_metadata()

library(ggplot2)

card <- ReportCard$new()$append_text("Some text")$append_plot(
  ggplot(iris, aes(x = Petal.Length)) + geom_histogram()
)$append_text("Some text")$append_metadata(key = "lm",
  value = lm(Ozone ~ Solar.R, airquality))
card$get_content()

card$to_list(tempdir())

library(ggplot2)

card <- ReportCard$new()$append_text("Some text")$append_plot(
  ggplot(iris, aes(x = Petal.Length)) + geom_histogram()
)$append_text("Some text")$append_metadata(key = "lm",
  value = lm(Ozone ~ Solar.R, airquality))
card$get_content()

ReportCard$new()$from_list(card$to_list(tempdir()), tempdir())

## -----
## Method `ReportCard$new`
```

```

## -----
card <- ReportCard$new()

## -----
## Method `ReportCard$append_table`
## -----

card <- ReportCard$new()$append_table(iris)

## -----
## Method `ReportCard$append_text`
## -----

card <- ReportCard$new()$append_text("A paragraph of default text")

## -----
## Method `ReportCard$append_rcode`
## -----

card <- ReportCard$new()$append_rcode("2+2", echo = FALSE)

## -----
## Method `ReportCard$append_content`
## -----

NewpageBlock <- getFromNamespace("NewpageBlock", "teal.reporter")
card <- ReportCard$new()$append_content(NewpageBlock$new())

## -----
## Method `ReportCard$get_content`
## -----

card <- ReportCard$new()$append_text("Some text")$append_metadata("rc", "a <- 2 + 2")

card$get_content()

## -----
## Method `ReportCard$get_metadata`
## -----

card <- ReportCard$new()$append_text("Some text")$append_metadata("rc", "a <- 2 + 2")

card$get_metadata()

```

```

## -----
## Method `ReportCard$get_name`
## -----

ReportCard$new()$set_name("NAME")$get_name()

## -----
## Method `ReportCard$set_name`
## -----

ReportCard$new()$set_name("NAME")$get_name()

```

Reporter

Reporter: *An R6 class for managing report cards*

Description

[Experimental]

This R6 class is designed to store and manage report cards, facilitating the creation, manipulation, and serialization of report-related data.

Methods

Public methods:

- [Reporter\\$new\(\)](#)
- [Reporter\\$append_cards\(\)](#)
- [Reporter\\$get_cards\(\)](#)
- [Reporter\\$get_blocks\(\)](#)
- [Reporter\\$reset\(\)](#)
- [Reporter\\$remove_cards\(\)](#)
- [Reporter\\$swap_cards\(\)](#)
- [Reporter\\$get_reactive_add_card\(\)](#)
- [Reporter\\$get_metadata\(\)](#)
- [Reporter\\$append_metadata\(\)](#)
- [Reporter\\$from_reporter\(\)](#)
- [Reporter\\$to_list\(\)](#)
- [Reporter\\$from_list\(\)](#)
- [Reporter\\$to_jsondir\(\)](#)
- [Reporter\\$from_jsondir\(\)](#)
- [Reporter\\$set_id\(\)](#)
- [Reporter\\$get_id\(\)](#)
- [Reporter\\$clone\(\)](#)

Method `new()`: Initialize a Reporter object.

Usage:

```
Reporter$new()
```

Returns: Object of class Reporter, invisibly.

Examples:

```
reporter <- Reporter$new()
```

Method `append_cards()`: Append one or more ReportCard objects to the Reporter.

Usage:

```
Reporter$append_cards(cards)
```

Arguments:

`cards` (ReportCard) or a list of such objects

Returns: self, invisibly.

Method `get_cards()`: Retrieves all ReportCard objects contained in the Reporter.

Usage:

```
Reporter$get_cards()
```

Returns: A (list) of ReportCard objects.

Method `get_blocks()`: Compiles and returns all content blocks from the ReportCard in the Reporter.

Usage:

```
Reporter$get_blocks(sep = NewpageBlock$new())
```

Arguments:

`sep` An optional separator to insert between each content block. Default is a NewpageBlock\$new() object.

Returns: list() list of TableBlock, TextBlock, PictureBlock and NewpageBlock.

Method `reset()`: Resets the Reporter, removing all ReportCard objects and metadata.

Usage:

```
Reporter$reset()
```

Returns: self, invisibly.

Method `remove_cards()`: Removes specific ReportCard objects from the Reporter by their indices.

Usage:

```
Reporter$remove_cards(ids = NULL)
```

Arguments:

`ids` (integer(id)) the indexes of cards

Returns: self, invisibly.

Method `swap_cards()`: Swaps the positions of two ReportCard objects within the Reporter.

Usage:

```
Reporter$swap_cards(start, end)
```

Arguments:

start (integer) the index of the first card

end (integer) the index of the second card

Returns: self, invisibly.

Method `get_reactive_add_card()`: Gets the current value of the reactive variable for adding cards.

Usage:

```
Reporter$get_reactive_add_card()
```

Returns: reactive_add_card current numeric value of the reactive variable.

Examples:

```
library(shiny)
```

```
isolate(Reporter$new())$get_reactive_add_card())
```

Method `get_metadata()`: Get the metadata associated with this Reporter.

Usage:

```
Reporter$get_metadata()
```

Returns: named list of metadata to be appended.

Examples:

```
reporter <- Reporter$new()$append_metadata(list(sth = "sth"))
```

```
reporter$get_metadata()
```

Method `append_metadata()`: Appends metadata to this Reporter.

Usage:

```
Reporter$append_metadata(meta)
```

Arguments:

meta (named list) of metadata to be appended.

Returns: self, invisibly.

Examples:

```
reporter <- Reporter$new()$append_metadata(list(sth = "sth"))
```

```
reporter$get_metadata()
```

Method `from_reporter()`: Reinitializes a Reporter instance by copying the report cards and metadata from another Reporter.

Usage:

```
Reporter$from_reporter(reporter)
```

Arguments:

reporter (Reporter) instance to copy from.

Returns: invisibly self

Examples:

```
reporter <- Reporter$new()
reporter$from_reporter(reporter)
```

Method `to_list()`: Convert a Reporter to a list and transfer any associated files to specified directory.

Usage:

```
Reporter$to_list(output_dir)
```

Arguments:

`output_dir` (character(1)) a path to the directory where files will be copied.

Returns: named list representing the Reporter instance, including version information, metadata, and report cards.

Examples:

```
reporter <- Reporter$new()
tmp_dir <- file.path(tempdir(), "testdir")
dir.create(tmp_dir)
reporter$to_list(tmp_dir)
```

Method `from_list()`: Reinitializes a Reporter from a list representation and associated files in a specified directory.

Usage:

```
Reporter$from_list(rlist, output_dir)
```

Arguments:

`rlist` (named list) representing a Reporter instance.

`output_dir` (character(1)) a path to the directory from which files will be copied.

Returns: self, invisibly.

Examples:

```
reporter <- Reporter$new()
tmp_dir <- file.path(tempdir(), "testdir")
unlink(tmp_dir, recursive = TRUE)
dir.create(tmp_dir)
reporter$from_list(reporter$to_list(tmp_dir), tmp_dir)
```

Method `to_jsondir()`: Serializes the Reporter to a JSON file and copies any associated files to a specified directory.

Usage:

```
Reporter$to_jsondir(output_dir)
```

Arguments:

`output_dir` (character(1)) a path to the directory where files will be copied, JSON and statics.

Returns: `output_dir` argument.

Examples:

```
reporter <- Reporter$new()
tmp_dir <- file.path(tempdir(), "jsondir")
dir.create(tmp_dir)
reporter$to_jsondir(tmp_dir)
```

Method `from_jsondir()`: Reinitializes a Reporter from a JSON file and files in a specified directory.

Usage:

```
Reporter$from_jsondir(output_dir)
```

Arguments:

`output_dir` (`character(1)`) a path to the directory with files, JSON and statics.

Returns: `self`, invisibly.

Examples:

```
reporter <- Reporter$new()
tmp_dir <- file.path(tempdir(), "jsondir")
dir.create(tmp_dir)
unlink(list.files(tmp_dir, recursive = TRUE))
reporter$to_jsondir(tmp_dir)
reporter$from_jsondir(tmp_dir)
```

Method `set_id()`: Set the Reporter id Optionally add id to a Reporter which will be compared when it is rebuilt from a list. The id is added to the downloaded file name.

Usage:

```
Reporter$set_id(id)
```

Arguments:

`id` (`character(1)`) a Report id.

Returns: `self`, invisibly.

Method `get_id()`: Get the Reporter id

Usage:

```
Reporter$get_id()
```

Returns: `character(1)` the Reporter id.

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

Usage:

```
Reporter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

The function has to be used in the shiny reactive context.

if Report has an id when converting to JSON then It will be compared to the currently available one.

if Report has an id when converting to JSON then It will be compared to the currently available one.

Examples

```

library(ggplot2)
library(rtables)

card1 <- ReportCard$new()

card1$append_text("Header 2 text", "header2")
card1$append_text("A paragraph of default text", "header2")
card1$append_plot(
  ggplot(iris, aes(x = Petal.Length)) + geom_histogram()
)

card2 <- ReportCard$new()

card2$append_text("Header 2 text", "header2")
card2$append_text("A paragraph of default text", "header2")
lyt <- analyze(split_rows_by(basic_table(), "Day"), "Ozone", afun = mean)
table_res2 <- build_table(lyt, airquality)
card2$append_table(table_res2)
card2$append_table(iris)

reporter <- Reporter$new()
reporter$append_cards(list(card1, card2))

library(ggplot2)
library(rtables)

card1 <- ReportCard$new()

card1$append_text("Header 2 text", "header2")
card1$append_text("A paragraph of default text", "header2")
card1$append_plot(
  ggplot(iris, aes(x = Petal.Length)) + geom_histogram()
)

card2 <- ReportCard$new()

card2$append_text("Header 2 text", "header2")
card2$append_text("A paragraph of default text", "header2")
lyt <- analyze(split_rows_by(basic_table(), "Day"), "Ozone", afun = mean)
table_res2 <- build_table(lyt, airquality)
card2$append_table(table_res2)
card2$append_table(iris)

reporter <- Reporter$new()
reporter$append_cards(list(card1, card2))
reporter$get_cards()

library(ggplot2)
library(rtables)

```

```

card1 <- ReportCard$new()

card1$append_text("Header 2 text", "header2")
card1$append_text("A paragraph of default text", "header2")
card1$append_plot(
  ggplot(iris, aes(x = Petal.Length)) + geom_histogram()
)

card2 <- ReportCard$new()

card2$append_text("Header 2 text", "header2")
card2$append_text("A paragraph of default text", "header2")
lyt <- analyze(split_rows_by(basic_table(), "Day"), "Ozone", afun = mean)
table_res2 <- build_table(lyt, airquality)
card2$append_table(table_res2)
card2$append_table(iris)

reporter <- Reporter$new()
reporter$append_cards(list(card1, card2))
reporter$get_blocks()

## -----
## Method `Reporter$new`
## -----

reporter <- Reporter$new()

## -----
## Method `Reporter$get_reactive_add_card`
## -----

library(shiny)

isolate(Reporter$new()$get_reactive_add_card())

## -----
## Method `Reporter$get_metadata`
## -----

reporter <- Reporter$new()$append_metadata(list(sth = "sth"))
reporter$get_metadata()

## -----
## Method `Reporter$append_metadata`
## -----

reporter <- Reporter$new()$append_metadata(list(sth = "sth"))
reporter$get_metadata()

```

```

## -----
## Method `Reporter$from_reporter`
## -----

reporter <- Reporter$new()
reporter$from_reporter(reporter)

## -----
## Method `Reporter$to_list`
## -----

reporter <- Reporter$new()
tmp_dir <- file.path(tempdir(), "testdir")
dir.create(tmp_dir)
reporter$to_list(tmp_dir)

## -----
## Method `Reporter$from_list`
## -----

reporter <- Reporter$new()
tmp_dir <- file.path(tempdir(), "testdir")
unlink(tmp_dir, recursive = TRUE)
dir.create(tmp_dir)
reporter$from_list(reporter$to_list(tmp_dir), tmp_dir)

## -----
## Method `Reporter$to_jsondir`
## -----

reporter <- Reporter$new()
tmp_dir <- file.path(tempdir(), "jsondir")
dir.create(tmp_dir)
reporter$to_jsondir(tmp_dir)

## -----
## Method `Reporter$from_jsondir`
## -----

reporter <- Reporter$new()
tmp_dir <- file.path(tempdir(), "jsondir")
dir.create(tmp_dir)
unlink(list.files(tmp_dir, recursive = TRUE))
reporter$to_jsondir(tmp_dir)
reporter$from_jsondir(tmp_dir)

```

Description

[Experimental]

Module offers functionalities to visualize, manipulate, and interact with report cards that have been added to a report. It includes a previewer interface to see the cards and options to modify the report before downloading.

Cards are saved by the shiny bookmarking mechanism.

For more details see the vignette: `vignette("previewerReporter", "teal.reporter")`.

Usage

```
reporter_previewer_ui(id)

reporter_previewer_srv(
  id,
  reporter,
  global_knitr = getOption("teal.reporter.global_knitr"),
  rmd_output = c(html = "html_document", pdf = "pdf_document", powerpoint =
    "powerpoint_presentation", word = "word_document"),
  rmd_yaml_args = list(author = "NEST", title = "Report", date =
    as.character(Sys.Date()), output = "html_document", toc = FALSE),
  previewer_buttons = c("download", "load", "reset")
)
```

Arguments

<code>id</code>	(character(1)) shiny module instance id.
<code>reporter</code>	(Reporter) instance.
<code>global_knitr</code>	(list) of knitr parameters (passed to <code>knitr::opts_chunk\$set</code>) for customizing the rendering process.
<code>rmd_output</code>	(character) vector with rmarkdown output types, by default all possible <code>pdf_document</code> , <code>html_document</code> , <code>powerpoint_presentation</code> , and <code>word_document</code> . If vector is named then those names will appear in the UI.
<code>rmd_yaml_args</code>	(named list) with Rmd yaml header fields and their default values. This list will result in the custom subset of UI inputs for the download reporter functionality. Default <code>list(author = "NEST", title = "Report", date = Sys.Date(), output = "html_document", toc = FALSE)</code> . The list must include at least "output" field. The default value for "output" has to be in the <code>rmd_output</code> argument.
<code>previewer_buttons</code>	(character) set of modules to include with <code>c("download", "load", "reset")</code> possible values and "download" is required. Default <code>c("download", "load", "reset")</code>

Details

To access the default values for the `global_knitr` parameter, use `getOption('teal.reporter.global_knitr')`. These defaults include:

- echo = TRUE
- tidy.opts = list(width.cutoff = 60)
- tidy = TRUE if formatR package is installed, FALSE otherwise

Value

NULL.

report_load_srv	<i>Server to Load Reporter</i>
-----------------	--------------------------------

Description

[Experimental] Server to load ReporterCard(s) to the Reporter
 For more details see the vignette: vignette("simpleReporter", "teal.reporter").

Usage

```
report_load_srv(id, reporter)
```

Arguments

id	character(1) this shiny module's id.
reporter	Reporter instance.

Value

shiny::moduleServer

report_load_ui	<i>User Interface to Load Reporter</i>
----------------	--

Description

[Experimental] Button to upload ReporterCard(s) to the Reporter.
 For more details see the vignette: vignette("simpleReporter", "teal.reporter").

Usage

```
report_load_ui(id)
```

Arguments

id	character(1) this shiny module's id.
----	--------------------------------------

Value

shiny::tagList

reset_report_button *Reset report button module*

Description

[Experimental]

Provides a button that triggers resetting the report content.

For more information, refer to the vignette: `vignette("simpleReporter", "teal.reporter")`.

Usage

```
reset_report_button_ui(id, label = NULL)
```

```
reset_report_button_srv(id, reporter)
```

Arguments

`id` (character(1)) shiny module instance id.
`label` (character(1)) label before the icon. By default NULL.
`reporter` (Reporter) instance.

Value

NULL.

rmd_outputs *Get document output types from the rmarkdown package*

Description

[Experimental]

Retrieves vector of available document output types from the rmarkdown package, such as `pdf_document`, `html_document`, etc.

Usage

```
rmd_outputs()
```

Value

character vector.

Examples

```
rmd_outputs()
```

rmd_output_arguments *Get document output arguments from the rmarkdown package*

Description

[Experimental]

Retrieves the arguments for a specified document output type from the rmarkdown package.

Usage

```
rmd_output_arguments(output_name, default_values = FALSE)
```

Arguments

output_name (character) rmarkdown output name.
 default_values (logical(1)) if to return a default values for each argument.

Examples

```
rmd_output_arguments("pdf_document")
rmd_output_arguments("pdf_document", TRUE)
```

simple_reporter *Simple reporter module*

Description

[Experimental]

Module provides compact UI and server functions for managing a report in a shiny app. This module combines functionalities for [adding cards to a report](#), [downloading the report](#), and [resetting report content](#).

For more details see the vignette: `vignette("simpleReporter", "teal.reporter")`.

Usage

```
simple_reporter_ui(id)

simple_reporter_srv(
  id,
  reporter,
  card_fun,
  global_knitr = getOption("teal.reporter.global_knitr"),
  rmd_output = c(html = "html_document", pdf = "pdf_document", powerpoint =
    "powerpoint_presentation", word = "word_document"),
  rmd_yaml_args = list(author = "NEST", title = "Report", date =
    as.character(Sys.Date()), output = "html_document", toc = FALSE)
)
```


Arguments

id	(character(1)) shiny module instance id.
reporter	(Reporter) instance.
card_fun	(function) which returns a ReportCard instance, the function has a card argument and an optional comment argument.
global_knitr	(list) a global knitr parameters for customizing the rendering process.
rmd_output	(character) vector with rmarkdown output types, by default all possible pdf_document, html_document, powerpoint_presentation, and word_document. If vector is named then those names will appear in the UI.
rmd_yaml_args	(named list) with Rmd yaml header fields and their default values. This list will result in the custom subset of UI inputs for the download reporter functionality. Default list(author = "NEST", title = "Report", date = Sys.Date(), output = "html_document", toc = FALSE). The list must include at least "output" field. The default value for "output" has to be in the rmd_output argument.

Details

To access the default values for the global_knitr parameter, use `getOption('teal.reporter.global_knitr')`. These defaults include:

- echo = TRUE
- tidy.opts = list(width.cutoff = 60)
- tidy = TRUE if formatR package is installed, FALSE otherwise

Value

NULL.

Examples

```
if (interactive()) {
  library(shiny)

  shinyApp(
    ui = fluidPage(simple_reporter_ui("simple")),
    server = function(input, output, session) {
      simple_reporter_srv("simple", Reporter$new(), function(card) card)
    }
  )
}
```

Index

`add_card_button`, 2
`add_card_button_srv` (`add_card_button`), 2
`add_card_button_ui` (`add_card_button`), 2
adding cards to a report, 24
`as_yaml_auto`, 3

`download_report_button`, 5
`download_report_button_srv`
 (`download_report_button`), 5
`download_report_button_ui`
 (`download_report_button`), 5
downloading the report, 24

`print.rmd_yaml_header`, 6

`report_load_srv`, 22
`report_load_ui`, 22
`ReportCard`, 3, 7, 14, 25
`Reporter`, 13, 22
`reporter_previewer`, 20
`reporter_previewer_srv`
 (`reporter_previewer`), 20
`reporter_previewer_ui`
 (`reporter_previewer`), 20
`reset_report_button`, 23
`reset_report_button_srv`
 (`reset_report_button`), 23
`reset_report_button_ui`
 (`reset_report_button`), 23
resetting report content, 24
`rmd_output_arguments`, 24
`rmd_outputs`, 23

`simple_reporter`, 24
`simple_reporter_srv` (`simple_reporter`),
 24
`simple_reporter_ui` (`simple_reporter`), 24

`yaml::as_yaml`, 4