# Package 'teal.reporter'

February 17, 2024

**Title** Reporting Tools for 'shiny' Modules

**Version** 0.3.0

**Date** 2024-02-16

**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 (>= 2.1.0), flextable (>= 0.9.2), grid, htmltools (>= 0.5.4), knitr (>= 1.34), lifecycle (>= 0.2.0), R6, rmarkdown (>= 2.19), shiny (>= 1.6.0), shinyWidgets (>= 0.5.1), yaml (>= 1.1.0), zip (>= 1.1.0)

**Suggests** DT (>= 0.13), formatR (>= 1.5), ggplot2 (>= 3.4.0), lattice (>= 0.18-4), png, rtables (>= 0.5.1), testthat (>= 3.1.5), tinytex

**VignetteBuilder** knitr

**RdMacros** lifecycle

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

**Config/Needs/website** insightsengineering/nesttemplate

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.3.1

**NeedsCompilation** no

**Author** Dawid Kaledkowski [aut, cre] (<https://orcid.org/0000-0001-9533-457X>),
Kartikeya Kirar [aut],
Marcin Kosinski [aut],
Maciej Nasinski [aut],
Konrad Pagacz [aut],
Mahmoud Hallal [aut],
Chendi Liao [rev],
Dony Unardi [rev],
F. Hoffmann-La Roche AG [cph, fnd]

**Maintainer** Dawid Kaledkowski <dawid.kaledkowski@roche.com>

**Repository** CRAN

**Date/Publication** 2024-02-16 23:20:02 UTC

# R topics documented:



---

`add_card_button_srv` *Add Card Button Server*

---

## Description

**[Experimental]** server for adding views/cards to the Report.

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

## Usage

```
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. It can have optional card, comment and label parameters. If card parameter is added, then the ReportCard instance is created for the user. Use comment parameter to pass it's value whenever you prefer with card$append_text() - if card_fun does not have comment parameter, then comment from Add Card UI module will be added at the end of the content of the card. If label parameter is provided, you can use it to customize appearance of the card name and use if to specify card content with card$append_text() - if card_fun does not have 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. |

### Details

This module allows using a child of ReportCard instead of ReportCard. To properly support this, an instance of the child class must be passed as the default value of the card argument in the card_fun function. See below:

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

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

### Value

shiny::moduleServer

---

## add_card_button_ui *Add Card Button User Interface*

---

### Description

**[Experimental]** button for adding views/cards to the Report.

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

### Usage

```
add_card_button_ui(id)
```

### Arguments

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

### Value

shiny::tagList

---

## as_yaml_auto *Parse a Named List to the* Rmd yaml *Header*

---

### Description

**[Experimental]** parse a named list to the Rmd yaml header, so the developer gets automatically tabulated Rmd yaml header. Only a non nested (flat) list will be processed, where as a nested list is directly processed with the yaml::as.yaml function. All Rmd yaml header fields from the vector are supported, c("author", "date", "title", "subtitle", "abstract", "keywords", "subject", "description", "category", "lang"). Moreover all outputfield types in the rmarkdown package and their arguments are supported.

### 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 optionally wrap with result with the internal md_header(), default TRUE. |
| convert_logi | logical convert a character values to logical, if they are recognized as quoted yaml logical values , default TRUE. |
| multi_output | logical multi output slots in the input argument, default FALSE. |
| silent | logical suppress messages and warnings, default FALSE. |

### Value

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

## 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_srv`

*Download Button Server*

---

## Description

**[Experimental]** server for downloading the Report.

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

## Usage

```
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

| | |
|---|---|
| id | character(1) this shiny module's id. |
| reporter | Reporter instance. |
| global_knitr | list a of knitr parameters (passed to knitr::opts_chunk$set) for customizing the rendering process. |
| rmd_output | character vector with rmarkdown output types, by default all possible c("pdf_document", "html_document", "powerpoint_presentation", "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 User Interface 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

shiny::moduleServer

## download_report_button_ui *Download Button Reporter User Interface*

### Description

**[Experimental]** button for downloading the Report.

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

### Usage

```
download_report_button_ui(id)
```

### Arguments

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

### Value

`shiny::tagList`

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

### Description

**[Experimental]** Print method for the `yaml_header` class.

### Usage

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

### Arguments

| | |
|---|---|
| x | `rmd_yaml_header` class object. |
| ... | optional text. |

### Examples

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

ReportCard ReportCard

## Description

**[Experimental]** R6 class that supports creating a report card containing text, plot, table and meta data blocks that can be appended and rendered to form a report output from a shiny app.

## 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()`**:** Returns a `ReportCard` object.

*Usage:*

```
ReportCard$new()
```

*Returns:* a `ReportCard` object

*Examples:*

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

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

*Usage:*

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

*Arguments:*

`table` the appended table

*Returns:* invisibly self

*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` the appended plot

`dim integer vector` width and height in pixels.

*Returns:* invisibly self

*Examples:*

```
card <- ReportCard$new()$append_plot(
  ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)
```

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

*Usage:*

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

*Arguments:*

`text` (`character(0)` or `character(1)`) the text

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

*Returns:* invisibly self

*Examples:*

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

**Method** `append_rcode()`**:** Appends an `rmarkdown` R chunk to this `ReportCard`.

*Usage:*

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

*Arguments:*

`text` (`character(0)` or `character(1)`) the text

`...` any `rmarkdown` R chunk parameter and its value.

*Returns:* invisibly self

*Examples:*

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

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

*Usage:*

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

*Arguments:*

`content` (`ContentBlock`)

*Returns:* invisibly self

*Examples:*

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

**Method** `get_content()`**:** Returns the content of 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()`**:** Removes all objects added to this `ReportCard`.

*Usage:*

```
ReportCard$reset()
```

*Returns:* invisibly self

**Method** `get_metadata()`**:** Returns the metadata of this `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)`) name of meta data.
`value` value of meta data.

*Returns:* invisibly self

*Examples:*

```
card <- ReportCard$new()$append_text("Some text")$append_plot(
 ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)$append_text("Some text")$append_metadata(key = "lm",
                  value = lm(Ozone ~ Solar.R, airquality))
card$get_content()
card$get_metadata()
```

**Method** `get_name()`**:** get the Card name

*Usage:*

```
ReportCard$get_name()
```

*Returns:* `character` a Card name

*Examples:*

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

**Method** `set_name()`**:** set the Card name

*Usage:*

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

*Arguments:*

`name` `character` a Card name

*Returns:* invisibly self

*Examples:*

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

**Method** `to_list()`**:** Convert the `ReportCard` to a list.

*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.

*Examples:*

```
card <- ReportCard$new()$append_text("Some text")$append_plot(
 ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)$append_text("Some text")$append_metadata(key = "lm",
                  value = lm(Ozone ~ Solar.R, airquality))
card$get_content()

card$to_list(tempdir())
```

**Method** `from_list()`**:** Create the `ReportCard` from a list.

*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:* invisibly self

*Examples:*

```
card <- ReportCard$new()$append_text("Some text")$append_plot(
  ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)$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** `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

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

card <- ReportCard$new()


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

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


## ------------------------------------------------
## Method `ReportCard$append_plot`
## ------------------------------------------------

card <- ReportCard$new()$append_plot(
  ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)


## ------------------------------------------------
## 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$append_metadata`
## ------------------------------------------------

card <- ReportCard$new()$append_text("Some text")$append_plot(
  ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)$append_text("Some text")$append_metadata(key = "lm",
                 value = lm(Ozone ~ Solar.R, airquality))
card$get_content()
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()

## ------------------------------------------------
## Method `ReportCard$to_list`
## ------------------------------------------------

card <- ReportCard$new()$append_text("Some text")$append_plot(
  ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)$append_text("Some text")$append_metadata(key = "lm",
                  value = lm(Ozone ~ Solar.R, airquality))
card$get_content()

card$to_list(tempdir())


## ------------------------------------------------
## Method `ReportCard$from_list`
## ------------------------------------------------

card <- ReportCard$new()$append_text("Some text")$append_plot(
  ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)$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())
```

---

Reporter Reporter

---

## Description

**[Experimental]** R6 class that stores and manages report cards.

## 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$from_jsondir()
- Reporter$clone()

**Method** `new()`**:** Returns a `Reporter` object.

*Usage:*

```
Reporter$new()
```

*Returns:* a `Reporter` object

*Examples:*

```
reporter <- teal.reporter::Reporter$new()
```

**Method** `append_cards()`**:** Appends a table to this `Reporter`.

*Usage:*

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

*Arguments:*

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

*Returns:* invisibly self

*Examples:*

```
card1 <- teal.reporter::ReportCard$new()

card1$append_text("Header 2 text", "header2")
card1$append_text("A paragraph of default text", "header2")
card1$append_plot(
 ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)

card2 <- teal.reporter::ReportCard$new()

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

```
reporter <- teal.reporter::Reporter$new()
reporter$append_cards(list(card1, card2))
```

**Method** `get_cards()`**:** Returns cards of this `Reporter`.

*Usage:*

```
Reporter$get_cards()
```

*Returns:* `list()` list of `ReportCard`

*Examples:*

```
card1 <- teal.reporter::ReportCard$new()

card1$append_text("Header 2 text", "header2")
card1$append_text("A paragraph of default text", "header2")
card1$append_plot(
 ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)

card2 <- teal.reporter::ReportCard$new()

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

reporter <- teal.reporter::Reporter$new()
reporter$append_cards(list(card1, card2))
reporter$get_cards()
```

**Method** `get_blocks()`**:** Returns blocks of all `ReportCard` of this `Reporter`.

*Usage:*

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

*Arguments:*

`sep` the element inserted between each content element in this `Reporter`. Pass `NULL` to return content without any additional elements. Default: `NewpageBlock$new()`

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

*Examples:*

```
card1 <- teal.reporter::ReportCard$new()

card1$append_text("Header 2 text", "header2")
card1$append_text("A paragraph of default text", "header2")
card1$append_plot(
 ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
```

```
)

card2 <- teal.reporter::ReportCard$new()

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

reporter <- teal.reporter::Reporter$new()
reporter$append_cards(list(card1, card2))
reporter$get_blocks()
```

**Method** `reset()`**:** Removes all `ReportCard` objects added to this `Reporter`. Additionally all metadata are removed.

*Usage:*

```
Reporter$reset()
```

*Returns:* invisibly self

**Method** `remove_cards()`**:** remove a specific Card in the Reporter

*Usage:*

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

*Arguments:*

`ids` `integer` the indexes of cards

*Returns:* invisibly self

**Method** `swap_cards()`**:** swap two cards in 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:* invisibly self

**Method** `get_reactive_add_card()`**:** get a value for the reactive value for the add card

*Usage:*

```
Reporter$get_reactive_add_card()
```

*Returns:* `reactive_add_card` field value

*Examples:*

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

**Method** `get_metadata()`**:** get metadata of this `Reporter`.

*Usage:*

```
Reporter$get_metadata()
```

*Returns:* metadata

*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` (`list`) of metadata.

*Returns:* invisibly self

*Examples:*

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

**Method** `from_reporter()`**:** Create/Recreate a Reporter from another Reporter

*Usage:*

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

*Arguments:*

`reporter` `Reporter` instance.

*Returns:* invisibly self

*Examples:*

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

**Method** `to_list()`**:** Convert a Reporter to a list and transfer files

*Usage:*

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

*Arguments:*

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

*Returns:* `named list` `Reporter` representation

*Examples:*

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

**Method** `from_list()`**:** Create/Recreate a Reporter from a list and directory with files

*Usage:*

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

*Arguments:*

`rlist named list` `Reporter` representation.

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

*Returns:* invisibly self

*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()`**:** Create/Recreate a Reporter to a directory with `JSON` file and static files

*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:* invisibly self

*Examples:*

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

**Method** `from_jsondir()`**:** Create/Recreate a Reporter from a directory with `JSON` file and static files

*Usage:*

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

*Arguments:*

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

*Returns:* invisibly self

*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** `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.

## Examples

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

reporter <- teal.reporter::Reporter$new()


## ------------------------------------------------
## Method `Reporter$append_cards`
## ------------------------------------------------

card1 <- teal.reporter::ReportCard$new()

card1$append_text("Header 2 text", "header2")
card1$append_text("A paragraph of default text", "header2")
card1$append_plot(
 ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)

card2 <- teal.reporter::ReportCard$new()

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

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


## ------------------------------------------------
## Method `Reporter$get_cards`
## ------------------------------------------------

card1 <- teal.reporter::ReportCard$new()

card1$append_text("Header 2 text", "header2")
card1$append_text("A paragraph of default text", "header2")
card1$append_plot(
 ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)

card2 <- teal.reporter::ReportCard$new()
```

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

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

## ------------------------------------------------
## Method `Reporter$get_blocks`
## ------------------------------------------------

card1 <- teal.reporter::ReportCard$new()

card1$append_text("Header 2 text", "header2")
card1$append_text("A paragraph of default text", "header2")
card1$append_plot(
 ggplot2::ggplot(iris, ggplot2::aes(x = Petal.Length)) + ggplot2::geom_histogram()
)

card2 <- teal.reporter::ReportCard$new()

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

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


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

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)
```

reporter_previewer_srv

*Reporter Previewer Server*

## Description

**[Experimental]** server supporting the functionalities of the reporter previewer For more details see the vignette: `vignette("previewerReporter", "teal.reporter")`.

## Usage

```
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)
)
```

## Arguments

| | |
|---|---|
| `id` | `character(1)` this `shiny` module's id. |
| `reporter` | `Reporter` instance |
| `global_knitr` | `list` a of `knitr` parameters (passed to `knitr::opts_chunk$set`) for customizing the rendering process. |
| `rmd_output` | `character` vector with `rmarkdown` output types, by default all possible `c("pdf_document", "html_document", "powerpoint_presentation", "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 User Interface 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

## reporter_previewer_ui *Reporter Previewer User Interface*

### Description

**[Experimental]** reporter previewer user interface to visualize and manipulate the already added report Cards

### Usage

```
reporter_previewer_ui(id)
```

### Arguments

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

## reset_report_button_srv *Reset Button Server*

### Description

**[Experimental]** server for resetting the Report content.

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

### Usage

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

### Arguments

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

### Value

shiny::moduleServer

## reset_report_button_ui *Reset Button Reporter User Interface*

### Description

**[Experimental]** button for resetting the report content.

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

### Usage

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

### Arguments

| | |
|---|---|
| `id` | `character(1)` this `shiny` module's id. |
| `label` | `character(1)` label before the icon. By default `NULL`. |

### Value

`shiny::tagList`

## rmd_outputs *Get document output types from the* `rmarkdown` *package*

### Description

**[Experimental]** get document output types from the `rmarkdown` package.

### Usage

```
rmd_outputs()
```

### Value

`character` vector.

### Examples

```
rmd_outputs()
```

## rmd_output_arguments *Get document output arguments from the* rmarkdown *package*

### Description

**[Experimental]** get document output arguments from the rmarkdown package

### Usage

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

### Arguments

| | |
|---|---|
| output_name | character rmarkdown output name. |
| default_values | logical if to return a default values for each argument. |

### Examples

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

## simple_reporter_srv *Simple Reporter Server*

### Description

**[Experimental]** three buttons for adding cards, downloading and resetting the Report. The add module has add_report_card_simple id, the download module the download_button_simple id and the reset module the reset_button_simple id.

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

### Usage

```
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) this shiny module's 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 c("pdf_document", "html_document", "powerpoint_presentation", "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 User Interface 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

shiny::moduleServer

---

## simple_reporter_ui *Simple Reporter User Interface*

---

### Description

**[Experimental]** three buttons for adding cards, downloading and resetting the Report.

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

### Usage

```
simple_reporter_ui(id)
```

### Arguments

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

## Value

shiny.tag

## Examples

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

# Index