Package: powerup (via r-universe)

October 29, 2026

Title Power Up Your PowerPoint Presentations

Version 0.3.1

Description Programmatically interact with new or existing 'Microsoft PowerPoint' documents, by adding new slides, updating existing slides, and inserting displays into specific locations in slides. Users can also add transition effects to slides, and add displays to specific locations in slides by pointing to specific placeholders or creating new ones. Preview the slides in a local viewer.

License Apache License (>= 2)

URL https://gsk-biostatistics.github.io/powerup/, https://github.com/GSK-Biostatistics/powerup/

BugReports https://github.com/GSK-Biostatistics/powerup/issues/

Depends R (>= 4.1.0), polish (>= 0.2.0)

Imports rlang, dplyr, officer (>= 0.6.7), purrr, tibble, xml2, cli, glue, ggplot2, htmltools, base64enc, shiny

Suggests testthat (>= 3.0.0), rmarkdown, webshot2, knitr, flextable, withr

Config/testthat/edition 3

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Ellis Hughes [aut, cre] (ORCID: <https://orcid.org/0000-0003-0637-4436>), Shannon Haughton [aut], Romain François [aut], GlaxoSmithKline Research & Development Limited [cph, fnd]

Maintainer Ellis Hughes <ellis.h.hughes@gsk.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-29 16:19:03 UTC

RemoteUrl https://github.com/cran/powerup

RemoteRef HEAD

RemoteSha 416baa7e2644fcb2a99145b02d76f859591f4d13

Contents

add_slide 2
add_table_style 4
content 5
extract_metadata_element 7
get_slide_file_name 8
get_table_styles 8
list_layouts 9
list_placeholders 10
list_slides 11
load_pptx 11
metadata 12
new_table_style 13
polish_content_pptx.preset_shape 14
preview_pptx 15
read_layout_placeholders 16
read_slide 17
read_slide_placeholders 18
ref_placeholder 19
remove_slide_ph 21
shape 22
slide_transition_generic 23
templated_placeholder 26
tint_color 27

Index **29**

`add_slide` *Working with Slides*

Description

`add_slide()` Adds a new slide at a specified position, and populate it with content from ....

`update_slide()` updates an existing slide from a specific index and populate it with content from ....

`move_slide()` moves the slide from one index to another.

`delete_slide()` deletes the slide based on the index.

Usage

```
add_slide(
  pptx,
  layout,
  ...,
  index,
  transition = NULL,
  polish_error_continue = TRUE,
  error_call = current_env()
)

delete_slide(pptx, index, error_call = current_env())

move_slide(pptx, index, to, error_call = current_env())

update_slide(
  pptx,
  index,
  ...,
  transition = NULL,
  polish_error_continue = TRUE,
  error_call = current_env(),
  verbose = getOption("powerup.verbose", default = TRUE)
)
```

Arguments

pptx	A PowerPoint wrapped in a <pptx_container> object. see load_pptx()
layout	slide layout to use when creating a slide in add_slide()
...	content objects created with content()
index	slide index. By default add_slide() creates a new slide at the end of the slide deck.
transition	Defines the type of transition to use when moving to this slide. See ?slide_transitions to see viable values.
polish_error_continue	Boolean indicating if when an error is throwing in either polishing or converting the placeholder if it should be propagated as an error (FALSE) or just inform the user and continue (TRUE). Defaults to TRUE.
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
to	new index to move the slide to
verbose	if TRUE, some information is cli::cli_inform()ed along the way

Value

a pptx_container object with the slide added, updated, moved, or deleted

Examples

```
library(powerup)

## Create a new presentation
ppt <- load_pptx()

## adding new slides
ppt <- ppt |>
  ## add first slide with content in title
  add_slide(layout = "Title and Content",
    content("hello world", ph = ph_title())
  ) |>
  ## add another slide with nothing on it yet
  add_slide(layout = "Title and Content") |>
  ## slide added at index 2 with content in title and body
  add_slide(layout = "Title and Content",
    index = 2,
    content("Hello World", ph = ph_title()),
    content("This is a body", ph = ph_body())
  )

## updating an existing slide wit new content
ppt <- ppt |>
   update_slide(index = 1,
     content("This placeholder now has a value", ph = ph_body())
   )

## moving a slide from index 2 to index 1
ppt <- ppt |>
   move_slide(index = 2, to = 1)

## deleting a slide at index 3
ppt <- ppt |>
   delete_slide(index = 3)
```

`add_table_style` *Add Table Style*

Description

Given a provided table style, add it to the tableStyle file in the document. this makes it available to tables that get added to the PowerPoint as a table style option

Usage

```
add_table_style(
  pptx,
  table_style,
```

```
  verbose = getOption("powerup.verbose", default = FALSE),
  error_call = caller_env()
)
```

Arguments

pptx	A PowerPoint wrapped in a <pptx_container> object. see load_pptx()
table_style	an xml_node that defines a tblStyle. Created by new_table_style().
verbose	if TRUE, some information is cli::cli_inform()ed along the way
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

a pptx_container object with the new table style added

Examples

```
pptx <- example_pptx()
table_style <- new_table_style("My New Style")
pptx <- add_table_style(pptx, table_style) #'
```

---

content *Add or remove content from a slide*

Description

content() lets a user specify a value to add to a slide, and the location on the slide to add it through the ph argument. ph defines the placeholder, or height, width, x and y offset of the bounding box where the content should be added.

Usage

```
content(
  value,
  ph = ph_body(),
  metadata = NULL,
  ...,
  group_contents = TRUE,
  error_call = current_env()
)

remove_content(ph, error_call = current_env())
```

Arguments

value	value to be polished and then added at the placeholder defined
ph	Defining where values should be added to or removed from. Must be a placeholder object
metadata	named character vector of metadata to store in the p:extLst/p:ext/custom:meta/custom:metadata nodes
...	Additional arguments forwarded to the polish::polish_content_pptx() method for value's class. For example, ggplot objects accept height, width, res, device, and scale; character objects accept font_color, font_size, font_style, and font_typeface. See polish::polish_content_pptx() for the full list per content type.
group_contents	boolean indicating if the polished value should be shown in the PowerPoint as a grouped Sp. The default, and for most cases, TRUE.
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Details

remove_content() on the other hand, lets users specify a placeholder with content that should be removed from the slide using the same placeholder nomenclature. The placeholder must exist in the slide to be removed.

Any named arguments passed via ... are forwarded to the appropriate polish::polish_content_pptx() method when the content is rendered into the slide. The available arguments depend on the class of value.

Value

a powerup_content object for content() and a remove_powerup_content object for remove_content()

See Also

polish::polish_content_pptx() for the full set of polishing arguments available per content type.

Examples

```
content("My Title 2", ph = ph_title())
content(mtcars      , ph = ph_body())

# Pass polishing args for a ggplot (controls rendered dimensions)
content(
  ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) + ggplot2::geom_point(),
  ph = ph_body(),
  height = 3,
  width = 5
)
```

```
# Pass polishing args for character content (font styling)
content("My Title", ph = ph_title(), font_size = 24, font_color = "#C00000")

remove_content(ph = ph_title())
remove_content(ph = ref_placeholder(label = "An Existing Placeholder"))
```

`extract_metadata_element` *Extract metadata element*

Description

Extract metadata element

Usage

```
extract_metadata_element(metadata, name = "name", empty = "<display>")
```

Arguments

`metadata`	named character vector
`name`	name of element to extract
`empty`	synonyms for ""

Value

the value of the metadata element, or "" if not found

Examples

```
m <- c(foo = "abc", bar = "def")
extract_metadata_element(m, name = "foo")
extract_metadata_element(m, name = "bar", empty = "def") ## return ""
```

get_slide_file_name *Get the file name for a given slide*

Description

Get the file name for a given slide

Usage

```
get_slide_file_name(pptx, index, error_call = current_env())
```

Arguments

pptx	A PowerPoint wrapped in a <pptx_container> object. see load_pptx()
index	slide index
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

The file name for the slide, as a character string

Examples

```
pptx <- example_pptx()
get_slide_file_name(pptx, index = 1)
```

get_table_styles *List Table Styles*

Description

List the available table styles within the document

Usage

```
get_table_styles(pptx)
```

Arguments

pptx	A PowerPoint wrapped in a <pptx_container> object. see load_pptx()

Value

A data frame with columns `style_id` and `style_name` for each table style in the presentation.

Examples

```
pptx <- example_pptx()
get_table_styles(pptx)
```

`list_layouts` *List layouts*

Description

List layouts

Usage

```
list_layouts(pptx, error_call = current_env())
```

Arguments

`pptx`	A PowerPoint wrapped in a <pptx_container> object. see `load_pptx()`
`error_call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

A tibble with information about the layouts

Examples

```
list_layouts(load_pptx())
```

`list_placeholders` *List the available placeholders in a slide or a given layout*

Description

List the available placeholders in a slide or a given layout

Usage

```
list_placeholders(
  pptx,
  index = NULL,
  layout = NULL,
  units = c("in", "cm"),
  keep_all = FALSE,
  error_call = current_env()
)
```

Arguments

`pptx`	A PowerPoint wrapped in a <pptx_container> object. see `load_pptx()`
`index`	Slide index
`layout`	the name of the layout in the PowerPoint. Overridden if index is provided
`units`	"in" or "cm", what units should be used for the values presented (offsets, dimensions)
`keep_all`	Keep all the placeholders (TRUE), or only the ons with a \<ph\> node (FALSE)
`error_call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

a `placeholder_tbl` object with the available placeholders in the slide or layout

Examples

```
pptx <- example_pptx()

# what are the available placeholders on the slides
list_placeholders(pptx, layout = "Two Content")
list_placeholders(pptx, index = 2)
```

list_slides *List slides*

Description

List slides

Usage

```
list_slides(pptx, keep_all = FALSE, error_call = current_env())
```

Arguments

pptx	A PowerPoint wrapped in a <pptx_container> object. see load_pptx()
keep_all	if FALSE, only keep non missing placeholders
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

A tibble with information about the slides

Examples

```
list_slides(load_pptx())
```

load_pptx *Load/Save a PowerPoint file*

Description

Load/Save a PowerPoint file

Usage

```
load_pptx(path = NULL, name)

slide_errors(pptx)

save_pptx(pptx, path, error_call = current_env())

example_pptx()
```

Arguments

path	output path of file to be saved
name	name of the file
pptx	A PowerPoint wrapped in a <pptx_container> object. see load_pptx()
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Details

slide_errors() Extract a tibble of errors that occurred when adding and updating slides

Value

load_pptx() returns a new pptx_container object that wraps objects created by officer::read_pptx() with custom functionality for powerup.

Examples

```
pptx <- load_pptx(system.file("ref_files/standard_powerpoint.pptx", package = "powerup"))

pptx2<- load_pptx()
```

---

metadata *extended properties metadata*

Description

extended properties metadata

Usage

```
metadata(data, uri = "r://package/powerup", error_call = current_env())
```

Arguments

data	Named character vector used as name= and value= for the custom:property nodes.
uri	uri, used in the p:ext node
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

a p:extLst tag

Examples

```
metadata(c(foo = "abc", bar = "def"))
```

`new_table_style` *Create a new table style node*

Description

Helper function to make it simple to define a table style node.

Usage

```
new_table_style(
  style_name,
  style_id = make_style_id(),
  text_color = "#000000",
  cell_color = "#f36633",
  border_color = "#959595",
  border_style = "sng",
  border_weight = 12700,
  row_header_text_color = text_color,
  row_total_text_color = text_color,
  col_first_text_color = text_color,
  col_last_text_color = text_color,
  row_header_cell_color = cell_color,
  row_total_cell_color = cell_color,
  col_first_cell_color = cell_color,
  col_last_cell_color = cell_color,
  cell_color_tint1 = tint_color(cell_color, tint = 0.4),
  cell_color_tint2 = tint_color(cell_color, tint = 0.2),
  major_border_color = border_color,
  major_border_style = border_style,
  major_border_weight = border_weight * 3,
  error_call = caller_env()
)
```

Arguments

`style_name`	The new name of the style. This is what will be referenced when defining the table style..
`style_id`	a unique string containing 8, 4, 4, 4, then 12 random values between A-F and 0-9, separated by dashes: `\\{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\\`

`text_color`, `cell_color`, `border_color`, `row_header_text_color`, `row_total_text_color`, `col_first_text_color`, `col_last_text_color`, `row_header_cell_color`, `row_total_cell_color`, `col_first_cell_color`, `col_last_cell_color`, `cell_color_tint1`, `cell_color_tint2`, `major_border_color`
: either a color name (as listed by colors()), a hexadecimal string

`border_style`, `major_border_style`
: Define how borders should be styled. Options include: "sng","dbl","thickThin","thinThick","tri"

`border_weight`, `major_border_weight`
: Define the thickness of the borders. Must be a value between 0 and 20116800.

`error_call`
: The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

An xml_node that defines a tblStyle. Added to a PowerPoint via add_table_style().

Examples

```
example_table_style <- new_table_style("Example Style", text_color = "blue", border_color = "pink")
```

---

polish_content_pptx.preset_shape

Polish fit models

Description

Polish fit models

Usage

```
## S3 method for class 'preset_shape'
polish_content_pptx(x, ph = "<p:ph/>", pptx, ..., error_call = current_env())
```

Arguments

`x`
: preset_shape object to polish

`ph`
: placeholder

`pptx`
: The Powerpoint Presentation object

`...`
: These dots are for future extensions and must be empty.

`error_call`
: The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

a "polish_xml_nodeset" of viable PowerPoint 'OOXML' for addition to PowerPoint presentations representing a polished preset_shape object

Examples

```
pptx <- example_pptx() |>
 add_slide(
  layout = "Title and Content",
  content(shape("rightArrow", fill = "#000000", outline = "#ffffff"), ph = ph_title()),
  content(shape("heart", fill = "#ffffff", outline = "#000000"), ph = ph_body()),
  content(
    shape("funnel"),
    ph = new_placeholder(x_offset = "10%",y_offset="20%", height = "20%",width="22%")
   )
 )

file <- tempfile(fileext = ".pptx")
pptx |> save_pptx(file)
```

`preview_pptx` *Preview a powerup PPTX in the Viewer pane*

Description

Renders a `pptx_container` (or a path to a `.pptx` file) as an interactive slide viewer in the RStudio / Positron Viewer pane, or in the default browser. Uses PptxViewJS (bundled offline; no internet connection required).

Usage

```
preview_pptx(pptx, slide = 1L, view = rlang::is_interactive())
```

Arguments

`pptx`	A `pptx_container` object or a file path to a `.pptx` file.
`slide`	Integer. Which slide to show first (1-based). Default `1`.
`view`	Logical. If `TRUE` (default), opens the viewer. If `FALSE`, returns the html.

Value

An `htmltools::browsable` HTML object (invisibly). Called for its side-effect of opening the viewer.

Examples

```
## Not run:
pptx <- example_pptx()
preview_pptx(pptx)

## End(Not run)
```

`read_layout_placeholders` *Read layout placeholders*

Description

Read layout placeholders

Usage

```
read_layout_placeholders(
  pptx,
  layout,
  keep_all = FALSE,
  error_call = current_env()
)
```

Arguments

`pptx`	A PowerPoint wrapped in a <pptx_container> object. see `load_pptx()`
`layout`	layout
`keep_all`	if FALSE, only keep non missing placeholders
`error_call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

A tibble with columns describing the layout placeholders:

`layout_file`	character; basename of the layout file
`layout_name`	character; name of the layout
`id`	character; placeholder id
`offx`	numeric; x offset in inches
`offy`	numeric; y offset in inches
`cx`	numeric; width in inches
`cy`	numeric; height in inches

ph_type	character; placeholder type
ph	character; placeholder XML node as glue object
ph_id	character; placeholder idx attribute
layout_ph_name	character; placeholder name

The tibble has a dimensions attribute containing the slide size.

Examples

```
pptx <- example_pptx()
read_layout_placeholders(pptx, layout = "Title Slide")
```

---

read_slide *Read a single slide as xml*

Description

Read a single slide as xml

Usage

```
read_slide(pptx, index, error_call = current_env())
```

Arguments

pptx	A PowerPoint wrapped in a <pptx_container> object. see load_pptx()
index	slide index
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

The slide, as read by xml2::read_xml

Examples

```
ppt <- example_pptx()
read_slide(ppt, index = 1)
```

`read_slide_placeholders` *Read placeholders information from PowerPoint presentation*

Description

Read placeholders information from PowerPoint presentation

Usage

```
read_slide_placeholders(
  pptx,
  index,
  only_slide = FALSE,
  only_layout = FALSE,
  keep_all = FALSE,
  error_call = current_env()
)
```

Arguments

`pptx`	A PowerPoint wrapped in a <pptx_container> object. see `load_pptx()`
`index`	slide index
`only_slide`	if TRUE only extract placeholders from the slide, i.e. not from its associated layout
`only_layout`	if TRUE only extract placeholders from the layout, i.e. not occupied in the slide
`keep_all`	if FALSE, only keep non missing placeholders
`error_call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

A tibble with columns describing the placeholders:

`slide_index`	numeric; slide index
`slide_file`	character; basename of the slide file
`layout_file`	character; basename of the layout file
`layout_name`	character; name of the layout
`id`	character; placeholder id
`offx`	numeric; x offset in inches
`offy`	numeric; y offset in inches
`cx`	numeric; width in inches

cy	numeric; height in inches
ph_type	character; placeholder type
ph	character; placeholder XML node as glue object
ph_id	character; placeholder idx attribute
slide_ph_name	character; placeholder name from slide
slide_ph_id	character; placeholder id from slide
layout_ph_name	character; placeholder name from layout
metadata	list; metadata extracted from placeholder

The tibble has a `dimensions` attribute containing the slide size.

Examples

```
ppt <- example_pptx()
read_slide_placeholders(ppt)
read_slide_placeholders(ppt, index = 1)
read_slide_placeholders(ppt, index = 1:2)

read_slide_placeholders(ppt, only_slide = TRUE)
read_slide_placeholders(ppt, index = 2, only_slide = TRUE)
```

---

ref_placeholder *Placeholder object*

Description

Object to define placeholders.

Usage

```
ref_placeholder(
  label = "",
  label_match = c("exact", "match"),
  label_from = c("slide", "layout"),
 type = c("body", "title", "ctrTitle", "ftr", "subTitle", "tbl", "chart", "img", "any"),
  tie_breaker = c("largest", "smallest", "left", "right", "top", "bottom"),
  replace = TRUE,
  error_call = current_env()
)

new_placeholder(
 type = c("body", "title", "ctrTitle", "ftr", "subTitle", "tbl", "chart", "img"),
  x_offset = "10%",
  y_offset = "20%",
  width = "80%",
```

```
  height = "60%",
  label = "",
  error_call = current_env()
)
```

Arguments

label	The placeholder label to select. ref_placeholder allows to pass a pattern to search with if label_match is set to "match".
label_match	Use "match" if label is a regex pattern, or "exact" for exact match. Defaults to "exact".
label_from	Which placeholder label should be used, the one listed in the "slide" or from the "layout". Generally new slides should use "layout", while updating a slide may use either. Listing both allows to check the first listed location then the other before declaring it cannot find it.
type	The type of placeholder. Options are "body", "title", "ftr", "subTitle", "tbl", "chart", "img". ref_placeholder() allows for "any" in this position to allow for referencing any type of placeholder, or any combination of to be types to be passed. new_placeholder() allows only one.
tie_breaker	Which pane to select when multiple are returned from label matching ("largest","left","right","top","bottor can pass multiple to clarify if there are ties remaining.
replace	Identify whether to replace the existing placeholder or not if it exists on the slide. defaults to TRUE
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
x_offset	x offset in centimeters (cm), inches (in), or percent(in) of slide width
y_offset	y offset in centimeters, inches, or percent of slide height
width	Width in centimeters, inches, or percent of slide width
height	Height in centimeters, inches, or percent of slide height

Details

ref_placeholder() defines what logic to apply when figuring out which a placeholder in a slide or layout to use to insert content.

new_placeholder() defines what logic to use when creating a new placeholder in a slide.

empty_placeholder() indicates that the content should not be added to the slide, and does not expect any content.

Value

A placeholder object

Examples

```
# reference placeholder, find the top-most placeholder in the slide or layout
# with the word "title" in the placeholder label where type is "title" and
# replace the contents when new values are inserted
ref_placeholder(
    label = "[Tt]itle",
    type = "title",
    tie_breaker = "top",
    label_match = "match",
    label_from = c("slide","layout"),
    replace = TRUE
    )

# reference placeholder, find the the exact placeholder with the label "Content Placeholder 5"
# based on the template layout
ref_placeholder(
    label = "Content Placeholder 5",
    label_match = "exact",
    label_from = "layout"
    )

# Create a new placeholder 20% down from the top of the slide, 10% off from the left side, and
# make it 10cm tall and 15cm wide
new_placeholder(
    x_offset = "10%",
    y_offset = "20%",
    height = "10cm",
    width = "15cm"
    )
```

---

remove_slide_ph *Remove placeholder from a slide*

Description

Remove placeholder from a slide

Usage

```
remove_slide_ph(
  pptx,
  ph_id,
  index,
  error_call = current_env(),
  verbose = getOption("powerup.verbose", default = FALSE)
)
```

Arguments

pptx	A PowerPoint wrapped in a <pptx_container> object. see load_pptx()
ph_id	placeholder id
index	index of slide
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
verbose	inform about the removed placeholders

Value

Returns the modified pptx object (of class pptx_container) with the placeholder removed from the slide.

Examples

```
pptx <- load_pptx() |>
  add_slide(layout = "Title and Content", index = 1, content("hello", ph = ph_title()))

## remove the placeholder from the slide
pptx <- remove_slide_ph(pptx,
          index = 1,
          ph_id = read_slide_placeholders(pptx, index = 1)$slide_ph_id[1]
        )
```

---

shape *Define a shape to add*

Description

Programmatically define a shape by its type, fill, effect. Will fill the ph it is dropped into.

Usage

```
shape(x, fill = "default", outline = "default", ...)
```

Arguments

x	type of preset shape. See details for list of optional shapes
fill, outline	hex code defining shape fill and outline colors. To use the default colors, put "default". To make the fill or outline see through, use NULL.
...	unused, for future expansion of options.

Details

List of available shapes:

accentBorderCallout1, accentBorderCallout2, accentBorderCallout3, accentCallout1, accentCallout2, accentCallout3, actionButtonBackPrevious, actionButtonBeginning, actionButtonBlank, actionButtonDocument, actionButtonEnd, actionButtonForwardNext, actionButtonHelp, actionButtonHome, actionButtonInformation, actionButtonMovie, actionButtonReturn, actionButtonSound, arc, bentArrow, bentConnector2, bentConnector3, bentConnector4, bentConnector5, bentUpArrow, bevel, blockArc, borderCallout1, borderCallout2, borderCallout3, bracePair, bracketPair, callout1, callout2, callout3, can, chartPlus, chartStar, chartX, chevron, chord, circularArrow, cloud, cloudCallout, corner, cornerTabs, cube, curvedConnector2, curvedConnector3, curvedConnector4, curvedConnector5, curvedDownArrow, curvedLeftArrow, curvedRightArrow, curvedUpArrow, decagon, diagStripe, diamond, dodecagon, donut, doubleWave, downArrow, downArrowCallout, ellipse, ellipseRibbon, ellipseRibbon2, flowChartAlternateProcess, flowChartCollate, flowChartConnector, flowChartDecision, flowChartDelay, flowChartDisplay, flowChartDocument, flowChartExtract, flowChartInputOutput, flowChartInternalStorage, flowChartMagneticDisk, flowChartMagneticDrum, flowChartMagneticTape, flowChartManualInput, flowChartManualOperation, flowChartMerge, flowChartMultidocument, flowChartOfflineStorage, flowChartOffpageConnector, flowChartOnlineStorage, flowChartOr, flowChartPredefinedProcess, flowChartPreparation, flowChartProcess, flowChartPunchedCard, flowChartPunchedTape, flowChartSort, flowChartSummingJunction, flowChartTerminator, folderCorner, frame, funnel, gear6, gear9, halfFrame, heart, heptagon, hexagon, homePlate, horizontalScroll, irregularSeal1, irregularSeal2, leftArrow, leftArrowCallout, leftBrace, leftBracket, leftCircularArrow, leftRightArrow, leftRightArrowCallout, leftRightCircularArrow, leftRightRibbon, irregularSeal1, leftRightUpArrow, leftUpArrow, lightningBolt, line, lineInv, mathDivide, mathEqual, mathMinus, mathMultiply, mathNotEqual, mathPlus, moon, nonIsoscelesTrapezoid, noSmoking, notchedRightArrow, octagon, parallelogram, pentagon, pie, pieWedge, plaque, plaqueTabs, plus, quadArrow, quadArrowCallout, rect, ribbon, ribbon2, rightArrow, rightArrowCallout, rightBrace, rightBracket, round1Rect, round2DiagRect, round2SameRect, roundRect, rtTriangle, smileyFace, snip1Rect, snip2DiagRect, snip2SameRect, snipRoundRect, squareTabs, star10, star12, star16, star24, star32, star4, star5, star6, star7, star8, straightConnector1, stripedRightArrow, sun, swooshArrow, teardrop, trapezoid, triangle, upArrow, upArrowCallout, upDownArrow, upDownArrowCallout, uturnArrow, verticalScroll, wave, wedgeEllipseCallout, wedgeRectCallout, wedgeRoundRectCallout

Value

a `preset_shape` object that can be added to a slide via `content()`

Examples

```
shape("rightArrow", fill = "#000000", outline = "#ffffff")
shape("heart", fill = "orange", outline = "grey")
```

`slide_transition_generic`

Defined Slide Transitions

Description

Slide transitions define the transition to show when moving to the slide. `slide_transition_generic()` allows the user to create and pass new types of transitions. Otherwise, most standard transitions are already defined as `slide_transition_*()` to allow the user to easily define a transition and any other assets the transition may need.

Usage

```
slide_transition_generic(
  type,
  transition_type_xml,
  speed = c("med", "fast", "slow")
)

slide_transition_blinds(orientation = c("horz", "vert"), speed = "med")

slide_transition_checker(orientation = c("horz", "vert"), speed = "med")

slide_transition_circle(orientation = c("horz", "vert"), speed = "med")

slide_transition_comb(orientation = c("horz", "vert"), speed = "med")

slide_transition_cover(
  direction = c("d", "l", "r", "u", "ld", "lu", "rd", "ru"),
  speed = "med"
)

slide_transition_cut(black_screen = FALSE, speed = "med")

slide_transition_diamond(speed = "med")

slide_transition_dissolve(speed = "med")

slide_transition_fade(black_screen = FALSE, speed = "med")

slide_transition_newsflash(speed = "med")

slide_transition_plus(speed = "med")

slide_transition_pull(direction = c("ld", "lu", "rd", "ru"), speed = "med")

slide_transition_push(direction = c("d", "l", "r", "u"), speed = "med")

slide_transition_random(speed = "med")

slide_transition_randomBar(orientation = c("horz", "vert"), speed = "med")

slide_transition_split(
```

```
  split = c("in", "out"),
  orientation = c("horz", "vert"),
  speed = "med"
)

slide_transition_pull(direction = c("ld", "lu", "rd", "ru"), speed = "med")

slide_transition_wedge(speed = "med")

slide_transition_wheel(spokes = 4L, speed = "med")

slide_transition_wipe(direction = c("d", "l", "r", "u"), speed = "med")

slide_transition_zoom(zoom = c("in", "out"), speed = "med")

slide_transition_null()
```

Arguments

`type`	string defining the slide transition type
`transition_type_xml`	string defining the 'OOXML' of the transition type
`speed`	speed of the slide transition - med, fast, or slow
`orientation`	specify "horz" or "vert", defining the orientation of a transition
`direction`	specify the direction of transition, with possible values of d (down), l (left), r (right), u (up), ld (left down), lu (left up), rd (right down), ru (right up). Not every transition allows every direction. Check the available default values to see what values can be used for every transition
`black_screen`	boolean identifying if the transition starts from a black screen and then transitions to new slide over black
`spokes`	specify the number of spokes in the wheel; values are integers.
`zoom`, `split`	specify the direction of zoom, "in" or "out"

Value

a `slide_transition` object for the defined transition, or a `slide_transition_null` object for no transition or removing a transition. The `slide_transition` object can be added to a slide via `add_slide()` or `update_slide()`.

Examples

```
pptx <- example_pptx()

checker_transition <- slide_transition_checker()
dissolve_transition <- slide_transition_dissolve()

## add a transition to a new slide
pptx <- pptx |>
```

```
  add_slide(
    layout = "Title and Content",
    transition = checker_transition
  )

## Update/add transition to n existing slide
pptx <- pptx |>
  update_slide(
    index = 1,
    transition = dissolve_transition
  )
```

`templated_placeholder` *Placeholder Templates*

Description

Included standard template locations to put standard contents when constructing a slide.

These are all "templated_placeholder" objects, which allows the user to specify two potential placeholders, allowing passing a `existing_placeholder` object(`ref_placeholder()`), which tries to use an existing placeholder either in the slide or defined in the layout, and the fallback must be a `new_placeholder` object (`new_placeholder()`), which defines a placeholder into the slide without needing there to be a pre-defined placeholder in the slide layout.

Usage

```
templated_placeholder(preferred, fallback, error_call = current_env())

ph_title(replace = TRUE)

ph_subtitle(replace = TRUE)

ph_body(replace = TRUE)

ph_body_left(replace = TRUE)

ph_body_right(replace = TRUE)

ph_footer(replace = TRUE)

ph_footer_left(replace = TRUE)

ph_footer_right(replace = TRUE)
```

Arguments

preferred	placeholder object of class existing placeholder
fallback	placeholder object of class new_placeholder
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
replace	Identify whether to replace the existing placeholder if it exists on the slide. Defaults to TRUE.

Value

a templated_placeholder object

Examples

```
## create a new placeholder with a label of "my_placeholder"
templated_placeholder(
  preferred = ref_placeholder(label = "title", type = "title"),
  fallback = new_placeholder(label = "my_placeholder",
                             type = "body",
                             x_offset = "5%",
                             width = "90%",
                             y_offset = "5%",
                             height = "12.5%"
  )
)
```

---

tint_color *Apply tint to a provided color*

Description

Apply tint to a provided color

Usage

```
tint_color(x, tint)
```

Arguments

x	either a color name (as listed by colors()), a hexadecimal string
tint	the percentage to tint the color. Must be a value between 0 and 1

Value

A hexadecimal color string representing the tinted color.

Examples

```
tint_color("blue", tint = .5)
tint_color("#0000FF", tint = .5)
```

Index

abort(), *3*, *5*, *6*, *8–12*, *14*, *16–18*, *20*, *22*, *27*
add_slide, 2
add_slide(), *25*
add_table_style, 4

cli::cli_inform(), *3*, *5*
content, 5
content(), *3*, *5*, *6*, *23*

delete_slide *(*add_slide*)*, 2

example_pptx *(*load_pptx*)*, 11
extract_metadata_element, 7

get_slide_file_name, 8
get_table_styles, 8

list_layouts, 9
list_placeholders, 10
list_slides, 11
load_pptx, 11
load_pptx(), *3*, *5*, *8–12*, *16–18*, *22*

metadata, 12
move_slide *(*add_slide*)*, 2

new_placeholder *(*ref_placeholder*)*, 19
new_table_style, 13
new_table_style(), *5*

officer::read_pptx(), *12*

ph_body *(*templated_placeholder*)*, 26
ph_body_left *(*templated_placeholder*)*, 26
ph_body_right *(*templated_placeholder*)*, 26
ph_footer *(*templated_placeholder*)*, 26
ph_footer_left *(*templated_placeholder*)*, 26
ph_footer_right *(*templated_placeholder*)*, 26
ph_subtitle *(*templated_placeholder*)*, 26
ph_title *(*templated_placeholder*)*, 26
polish::polish_content_pptx(), *6*
polish_content_pptx.preset_shape, 14
preview_pptx, 15

read_layout_placeholders, 16
read_slide, 17
read_slide_placeholders, 18
ref_placeholder, 19
remove_content *(*content*)*, 5
remove_content(), *6*
remove_slide_ph, 21

save_pptx *(*load_pptx*)*, 11
shape, 22
slide_errors *(*load_pptx*)*, 11
slide_transition_blinds *(*slide_transition_generic*)*, 23
slide_transition_checker *(*slide_transition_generic*)*, 23
slide_transition_circle *(*slide_transition_generic*)*, 23
slide_transition_comb *(*slide_transition_generic*)*, 23
slide_transition_cover *(*slide_transition_generic*)*, 23
slide_transition_cut *(*slide_transition_generic*)*, 23
slide_transition_diamond *(*slide_transition_generic*)*, 23
slide_transition_dissolve *(*slide_transition_generic*)*, 23
slide_transition_fade *(*slide_transition_generic*)*, 23
slide_transition_generic, 23
slide_transition_generic(), *24*
slide_transition_newsflash *(*slide_transition_generic*)*, 23

slide_transition_null
(slide_transition_generic), 23
slide_transition_plus
(slide_transition_generic), 23
slide_transition_pull
(slide_transition_generic), 23
slide_transition_push
(slide_transition_generic), 23
slide_transition_random
(slide_transition_generic), 23
slide_transition_randomBar
(slide_transition_generic), 23
slide_transition_split
(slide_transition_generic), 23
slide_transition_wedge
(slide_transition_generic), 23
slide_transition_wheel
(slide_transition_generic), 23
slide_transition_wipe
(slide_transition_generic), 23
slide_transition_zoom
(slide_transition_generic), 23

templated_placeholder, 26
tint_color, 27

update_slide (add_slide), 2
update_slide(), *25*

xml2::read_xml, *17*