Package 'officer'

October 6, 2017

Type Package

Title Manipulation of Microsoft Word and PowerPoint Documents

Version 0.1.8

Description Access and manipulate 'Microsoft Word' and 'Microsoft PowerPoint' documents from R. The package focus on tabular and graphical reporting from R; it also provides two functions that let users get document content into data objects. A set of functions lets add and remove images, tables and paragraphs of text in new or existing documents. When working with 'PowerPoint' presentations, slides can be added or removed; shapes inside slides can also be added or removed. When working with 'Word' documents, a cursor can be used to help insert or delete content at a specific location in the document. The package does not require any installation of Microsoft product to be able to write Microsoft files.

License GPL-3

LazyData TRUE

LinkingTo Rcpp

Imports Rcpp (>= 0.12.12), purrr,dplyr,R6,tibble,ggplot2, R.utils,utils,grDevices, base64enc,zip, digest,uuid, magrittr,htmltools, xml2 (>= 1.1.0)

URL https://davidgohel.github.io/officer

BugReports https://github.com/davidgohel/officer/issues

RoxygenNote 6.0.1.9000

Suggests testthat, devEMF, knitr, rmarkdown, tidyr

VignetteBuilder knitr

NeedsCompilation yes

Author David Gohel [aut, cre]

Maintainer David Gohel <david.gohel@ardata.fr>

Repository CRAN

Date/Publication 2017-10-05 22:23:38 UTC

R topics documented:

`add_slide` *add a slide*

Description

add a slide into a pptx presentation

Usage

```
add_slide(x, layout, master)
```

Arguments

`x`	rpptx object
`layout`	slide layout name to use
`master`	master layout name where `layout` is located

Examples

```
my_pres <- read_pptx()
my_pres <- add_slide(my_pres,
  layout = "Two Content", master = "Office Theme")
```

body_add_break *add page break*

Description

add a page break into an rdocx object

Usage

```
body_add_break(x, pos = "after")
```

Arguments

x	an rdocx object
pos	where to add the new element relative to the cursor, one of "after", "before", "on".

Examples

```
library(magrittr)
doc <- read_docx() %>% body_add_break()
print(doc, target = "body_add_break.docx" )
```

body_add_fpar *add fpar*

Description

add an `fpar` (a formatted paragraph) into an rdocx object

Usage

```
body_add_fpar(x, value, style = NULL, pos = "after")
```

Arguments

x	a docx device
value	a character
style	paragraph style
pos	where to add the new element relative to the cursor, one of "after", "before", "on".

See Also

`fpar`

Examples

```
library(magrittr)
bold_face <- shortcuts$fp_bold(font.size = 30)
bold_redface <- update(bold_face, color = "red")
fpar_ <- fpar(ftext("Hello ", prop = bold_face),
              ftext("World", prop = bold_redface ),
              ftext(", how are you?", prop = bold_face ) )
doc <- read_docx() %>% body_add_fpar(fpar_)

print(doc, target = "body_add_fpar.docx" )
```

---

body_add_gg *add ggplot*

Description

add a ggplot as a png image into an rdocx object

Usage

```
body_add_gg(x, value, width = 6, height = 5, style = NULL, ...)
```

Arguments

x	an rdocx object
value	ggplot object
width	height in inches
height	height in inches
style	paragraph style
...	Arguments to be passed to png function.

Examples

```
library(ggplot2)

doc <- read_docx()

gg_plot <- ggplot(data = iris ) +
  geom_point(mapping = aes(Sepal.Length, Petal.Length))

if( capabilities(what = "png") )
  doc <- body_add_gg(doc, value = gg_plot, style = "centered" )

print(doc, target = "body_add_gg.docx" )
```

body_add_img *add image*

Description

add an image into an rdocx object

Usage

```
body_add_img(x, src, style = NULL, width, height, pos = "after")
```

Arguments

x	an rdocx object
src	image filename
style	paragraph style
width	height in inches
height	height in inches
pos	where to add the new element relative to the cursor, one of "after", "before", "on".

Examples

```
doc <- read_docx()

img.file <- file.path( Sys.getenv("R_HOME"), "doc", "html", "logo.jpg" )
if( file.exists(img.file) ){
  doc <- body_add_img(x = doc, src = img.file, height = 1.06, width = 1.39 )
}

print(doc, target = "body_add_img.docx" )
```

body_add_par *add paragraph of text*

Description

add a paragraph of text into an rdocx object

Usage

```
body_add_par(x, value, style = NULL, pos = "after")
```

Arguments

x	a docx device
value	a character
style	paragraph style
pos	where to add the new element relative to the cursor, one of "after", "before", "on".

Examples

```
library(magrittr)

doc <- read_docx() %>%
  body_add_par("A title", style = "heading 1") %>%
  body_add_par("Hello world!", style = "Normal") %>%
  body_add_par("centered text", style = "centered")

print(doc, target = "body_add_par.docx" )
```

---

body_add_table *add table*

Description

add a table into an rdocx object

Usage

```
body_add_table(x, value, style = NULL, pos = "after", header = TRUE,
  first_row = TRUE, first_column = FALSE, last_row = FALSE,
  last_column = FALSE, no_hband = FALSE, no_vband = TRUE)
```

Arguments

x	a docx device
value	a data.frame
style	table style
pos	where to add the new element relative to the cursor, one of "after", "before", "on".
header	display header if TRUE
first_row, last_row, first_column, last_column, no_hband, no_vband	logical for Word table options

Examples

```
library(magrittr)

doc <- read_docx() %>%
  body_add_table(iris, style = "table_template")

print(doc, target = "body_add_table.docx" )
```

---

body_add_toc *add table of content*

Description

add a table of content into an rdocx object

Usage

```
body_add_toc(x, level = 3, pos = "after", style = NULL, separator = ";")
```

Arguments

x	an rdocx object
level	max title level of the table
pos	where to add the new element relative to the cursor, one of "after", "before", "on".
style	optional. style in the document that will be used to build entries of the TOC.
separator	optional. Some configurations need "," (i.e. from Canada) separator instead of ";"

Examples

```
library(magrittr)
doc <- read_docx() %>% body_add_toc()

print(doc, target = "body_add_toc.docx" )
```

body_add_xml *add an xml string as document element*

Description

Add an xml string as document element in the document. This function is to be used to add custom openxml code.

Usage

```
body_add_xml(x, str, pos)
```

Arguments

x	an rdocx object
str	a wml string
pos	where to add the new element relative to the cursor, one of "after", "before", "on".

body_bookmark *add bookmark*

Description

Add a bookmark at the cursor location.

Usage

```
body_bookmark(x, id)
```

Arguments

x	an rdocx object
id	bookmark name

Examples

```
# cursor_bookmark ----
library(magrittr)

doc <- read_docx() %>%
  body_add_par("centered text", style = "centered") %>%
  body_bookmark("text_to_replace")
```

body_end_section *add section*

Description

add a section in a Word document. A section has effect on preceding paragraphs or tables.

Usage

```
body_end_section(x, landscape = FALSE, colwidths = c(1), space = 0.05,
  sep = FALSE, continuous = FALSE)

body_default_section(x, landscape = FALSE)

break_column_before(x)
```

Arguments

x	an rdocx object
landscape	landscape orientation
colwidths	columns widths in percent, if 3 values, 3 columns will be produced. Sum of this argument should be 1.
space	space in percent between columns.
sep	if TRUE a line is sperating columns.
continuous	TRUE for a continuous section break.

Details

A section start at the end of the previous section (or the beginning of the document if no preceding section exists), it stops where the section is declared. The function body_end_section() is reflecting that Word concept. The function body_default_section() is only modifying the default section of the document.

Examples

```
library(magrittr)

str1 <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit. " %>%
  rep(10) %>% paste(collapse = "")

my_doc <- read_docx() %>%
  # add a paragraph
  body_add_par(value = str1, style = "Normal") %>%
  # add a continuous section
  body_end_section(continuous = TRUE) %>%
  body_add_par(value = str1, style = "Normal") %>%
  body_add_par(value = str1, style = "Normal") %>%
```

```
  # preceding paragraph is on a new column
  break_column_before() %>%
  # add a two columns continous section
  body_end_section(colwidths = c(.6, .4),
                   space = .05, sep = FALSE, continuous = TRUE) %>%
  body_add_par(value = str1, style = "Normal") %>%
  # add a continuous section ... so far there is no break page
  body_end_section(continuous = TRUE) %>%
  body_add_par(value = str1, style = "Normal") %>%
  body_default_section(landscape = TRUE)

print(my_doc, target = "section.docx")
```

---

body_remove *remove an element*

Description

remove element pointed by cursor from a Word document

Usage

```
body_remove(x)
```

Arguments

x	an rdocx object

Examples

```
library(officer)
library(magrittr)

str1 <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit. " %>%
  rep(20) %>% paste(collapse = "")
str2 <- "Drop that text"
str3 <- "Aenean venenatis varius elit et fermentum vivamus vehicula. " %>%
  rep(20) %>% paste(collapse = "")

my_doc <- read_docx()  %>%
  body_add_par(value = str1, style = "Normal") %>%
  body_add_par(value = str2, style = "centered") %>%
  body_add_par(value = str3, style = "Normal")

print(my_doc, target = "init_doc.docx")

my_doc <- read_docx(path = "init_doc.docx")  %>%
  cursor_reach(keyword = "that text") %>%
  body_remove()

print(my_doc, target = "result_doc.docx")
```

body_replace_at *replace text at a bookmark location*

Description

replace text content enclosed in a bookmark by another text. A bookmark will be considered as valid if enclosing words within a paragraph, i.e. a bookmark along two or more paragraphs is invalid, a bookmark set on a whole paragraph is also invalid, bookmarking few words inside a paragraph is valid.

Usage

```
body_replace_at(x, bookmark, value)
```

Arguments

x	a docx device
bookmark	bookmark id
value	a character

Examples

```
library(magrittr)
doc <- read_docx() %>%
  body_add_par("centered text", style = "centered") %>%
  slip_in_text(". How are you", style = "strong") %>%
  body_bookmark("text_to_replace") %>%
  body_replace_at("text_to_replace", "not left aligned")
```

color_scheme *color scheme*

Description

get master layout color scheme into a data.frame.

Usage

```
color_scheme(x)
```

Arguments

x	rpptx object

Examples

```
x <- read_pptx()
color_scheme ( x = x )
```

cursor_begin *set cursor in an rdocx object*

Description

a set of functions is available to manipulate the position of a virtual cursor. This cursor will be used when inserting, deleting or updating elements in the document.

Usage

```
cursor_begin(x)

cursor_bookmark(x, id)

cursor_end(x)

cursor_reach(x, keyword)

cursor_forward(x)

cursor_backward(x)
```

Arguments

x	a docx device
id	bookmark id
keyword	keyword to look for as a regular expression

cursor_begin

Set the cursor at the beginning of the document, on the first element of the document (usually a paragraph or a table).

cursor_end

Set the cursor at the end of the document, on the last element of the document.

cursor_reach

Set the cursor on the first element of the document that contains text specified in argument `keyword`.

cursor_forward

Move the cursor forward, it increments the cursor in the document.

cursor_backward

Move the cursor backward, it decrements the cursor in the document.

Examples

```
library(officer)
library(magrittr)

doc <- read_docx() %>%
  body_add_par("paragraph 1", style = "Normal") %>%
  body_add_par("paragraph 2", style = "Normal") %>%
  body_add_par("paragraph 3", style = "Normal") %>%
  body_add_par("paragraph 4", style = "Normal") %>%
  body_add_par("paragraph 5", style = "Normal") %>%
  body_add_par("paragraph 6", style = "Normal") %>%
  body_add_par("paragraph 7", style = "Normal") %>%

  # default template contains only an empty paragraph
  # Using cursor_begin and body_remove, we can delete it
  cursor_begin() %>% body_remove() %>%

  # Let add text at the beginning of the
  # paragraph containing text "paragraph 4"
  cursor_reach(keyword = "paragraph 4") %>%
  slip_in_text("This is ", pos = "before", style = "Default Paragraph Font") %>%

  # move the cursor forward and end a section
  cursor_forward() %>%
  body_add_par("The section stop here", style = "Normal") %>%
  body_end_section(landscape = TRUE) %>%

  # move the cursor at the end of the document
  cursor_end() %>%
  body_add_par("The document ends now", style = "Normal")

print(doc, target = "cursor.docx")


# cursor_bookmark ----
library(magrittr)

doc <- read_docx() %>%
  body_add_par("centered text", style = "centered") %>%
  body_bookmark("text_to_replace") %>%
  body_add_par("A title", style = "heading 1") %>%
  body_add_par("Hello world!", style = "Normal") %>%
  cursor_bookmark("text_to_replace") %>%
  body_add_table(value = iris, style = "table_template")

print(doc, target = "bookmark.docx")
```

`docx_bookmarks` *List Word bookmarks*

Description

List bookmarks id that can be found in an rdocx object.

Usage

```
docx_bookmarks(x)
```

Arguments

x	a rdocx object

Examples

```
library(magrittr)

doc <- read_docx() %>%
  body_add_par("centered text", style = "centered") %>%
  body_bookmark("text_to_replace") %>% body_add_par("centered text", style = "centered") %>%
  body_bookmark("text_to_replace2")

docx_bookmarks(doc)

docx_bookmarks(read_docx())
```

---

docx_dim *Word page layout*

Description

get page width, page height and margins (in inches). The return values are those corresponding to the section where the cursor is.

Usage

```
docx_dim(x)
```

Arguments

x	a rdocx object

Examples

```
docx_dim(read_docx())
```

docx_reference_img *add images into an rdocx object*

Description

reference images into a Word document. This function is to be used with `wml_link_images`.

Images need to be referenced into the Word document, this will generate unique identifiers that need to be known to link these images with their corresponding xml code (wml).

Usage

```
docx_reference_img(x, src)
```

Arguments

x	an rdocx object
src	a vector of character containing image filenames.

docx_summary *get Word content in a tidy format*

Description

read content of a Word document and return a tidy dataset representing the document.

Usage

```
docx_summary(x)
```

Arguments

x	an rdocx object

Examples

```
example_pptx <- system.file(package = "officer",
  "doc_examples/example.docx")
doc <- read_docx(example_pptx)
docx_summary(doc)
```

doc_properties *read document properties*

Description

read Word or PowerPoint document properties and get results in a tidy data.frame.

Usage

```
doc_properties(x)
```

Arguments

x	an rdocx or rpptx object

Examples

```
library(magrittr)
read_docx() %>% doc_properties()
```

external_img *external image*

Description

This function is used to insert images into flextable with function display

Usage

```
external_img(src, width = 0.5, height = 0.2)

## S3 method for class 'external_img'
dim(x)

## S3 method for class 'external_img'
as.data.frame(x, ...)

## S3 method for class 'external_img'
format(x, type = "console", ...)
```

Arguments

src	image file path
width	height in inches
height	height in inches
x	external_img object
...	unused
type	output format

Examples

```
# external_img("example.png")
```

---

fpar *concatenate formatted text*

Description

Create a paragraph representation by concatenating formatted text or images. Modify default text and paragraph formatting properties with update.

Usage

```
fpar(...)

## S3 method for class 'fpar'
update(object, fp_p = NULL, fp_t = NULL, ...)

fortify_fpar(x)

## S3 method for class 'fpar'
as.data.frame(x, ...)

## S3 method for class 'fpar'
format(x, type = "pml", ...)

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

Arguments

...	unused
fp_p	paragraph formatting properties
fp_t	default text formatting properties
x, object	fpar object
type	a string value ("pml", "wml" or "html").

Details

`fortify_fpar`, `as.data.frame` are used internally and are not supposed to be used by end user.

Examples

```
fpar(ftext("hello", shortcuts$fp_bold()))
```

---

fp_border *border properties object*

Description

create a border properties object.

Usage

```
fp_border(color = "black", style = "solid", width = 1)

## S3 method for class 'fp_border'
update(object, color, style, width, ...)

## S3 method for class 'fp_border'
format(x, type = "pml", ...)
```

Arguments

`color`	border color - single character value (e.g. "#000000" or "black")
`style`	border style - single character value : "none" or "solid" or "dotted" or "dashed"
`width`	border width - an integer value : 0>= value
`...`	further arguments - not used
`x, object`	object `fp_border`
`type`	output type - one of 'pml'.

Examples

```
fp_border()
fp_border(color="orange", style="solid", width=1)
fp_border(color="gray", style="dotted", width=1)

# modify object ------
border <- fp_border()
update(border, style="dotted", width=3)
```

fp_cell *Cell formatting properties*

Description

Create a fp_cell object that describes cell formatting properties.

Usage

```
fp_cell(border = fp_border(width = 0), border.bottom, border.left, border.top,
  border.right, vertical.align = "center", margin = 0, margin.bottom,
  margin.top, margin.left, margin.right, background.color = "transparent",
  text.direction = "lrtb")

## S3 method for class 'fp_cell'
format(x, type = "wml", ...)

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

## S3 method for class 'fp_cell'
update(object, border, border.bottom, border.left, border.top,
  border.right, vertical.align, margin = 0, margin.bottom, margin.top,
  margin.left, margin.right, background.color, text.direction, ...)
```

Arguments

border	shortcut for all borders.
border.bottom, border.left, border.top, border.right	fp_border for borders.
vertical.align	cell content vertical alignment - a single character value , expected value is one of "center" or "top" or "bottom"
margin	shortcut for all margins.
margin.bottom, margin.top, margin.left, margin.right	cell margins - 0 or positive integer value.
background.color	cell background color - a single character value specifying a valid color (e.g. "#000000" or "black").
text.direction	cell text rotation - a single character value, expected value is one of "lrtb", "tbrl", "btlr".
x, object	object fp_cell
type	output type - one of 'wml', 'pml', 'html'.
...	further arguments - not used

Examples

```
obj <- fp_cell(margin = 1)
update( obj, margin.bottom = 5 )
```

`fp_par` *Paragraph formatting properties*

Description

Create a `fp_par` object that describes paragraph formatting properties.

Usage

```
fp_par(text.align = "left", padding = 0, border = fp_border(width = 0),
  padding.bottom, padding.top, padding.left, padding.right, border.bottom,
  border.left, border.top, border.right, shading.color = "transparent")

## S3 method for class 'fp_par'
dim(x)

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

## S3 method for class 'fp_par'
update(object, text.align, padding, border, padding.bottom,
  padding.top, padding.left, padding.right, border.bottom, border.left,
  border.top, border.right, shading.color, ...)
```

Arguments

`text.align`	text alignment - a single character value, expected value is one of 'left', 'right', 'center', 'justify'.
`padding`	paragraph paddings - 0 or positive integer value. Argument `padding` overwrites arguments `padding.bottom`, `padding.top`, `padding.left`, `padding.right`.
`border`	shortcut for all borders.
`padding.bottom, padding.top, padding.left, padding.right`	paragraph paddings - 0 or positive integer value.
`border.bottom, border.left, border.top, border.right`	`fp_border` for borders. overwrite other border properties.
`shading.color`	shading color - a single character value specifying a valid color (e.g. "#000000" or "black").
`x, object`	`fp_par` object
`...`	further arguments - not used

Value

a fp_par object

Examples

```
fp_par(text.align = "center", padding = 5)
obj <- fp_par(text.align = "center", padding = 1)
update( obj, padding.bottom = 5 )
```

---

fp_sign *object unique signature*

Description

Get unique signature for a formatting properties object.

Usage

```
fp_sign(x)
```

Arguments

x	a formatting set of properties

Examples

```
fp_sign( fp_text(color="orange") )
```

---

fp_text *Text formatting properties*

Description

Create a fp_text object that describes text formatting properties.

Usage

```
fp_text(color = "black", font.size = 10, bold = FALSE, italic = FALSE,
  underlined = FALSE, font.family = "Arial", vertical.align = "baseline",
  shading.color = "transparent")

## S3 method for class 'fp_text'
format(x, type = "wml", ...)

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

```
## S3 method for class 'fp_text'
as.data.frame(x, ...)

## S3 method for class 'fp_text'
update(object, color, font.size, bold = FALSE,
  italic = FALSE, underlined = FALSE, font.family, vertical.align,
  shading.color, ...)
```

Arguments

color	font color - a single character value specifying a valid color (e.g. "#000000" or "black").
font.size	font size (in point) - 0 or positive integer value.
bold	is bold
italic	is italic
underlined	is underlined
font.family	single character value specifying font name.
vertical.align	single character value specifying font vertical alignments. Expected value is one of the following : default 'baseline' or 'subscript' or 'superscript'
shading.color	shading color - a single character value specifying a valid color (e.g. "#000000" or "black").
x	fp_text object
type	output type - one of 'wml', 'pml', 'html'.
...	further arguments - not used
object	fp_text object to modify
format	format type, wml for MS word, pml for MS PowerPoint and html.

Value

a fp_text object

Examples

```
print( fp_text (color="red", font.size = 12) )
```

ftext *formatted text*

Description

Format a chunk of text with text formatting properties.

Usage

```
ftext(text, prop)

## S3 method for class 'ftext'
format(x, type = "console", ...)

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

Arguments

text	text value
prop	formatting text properties
x	ftext object
type	output format, one of wml, pml, html, console, text.
...	unused

Examples

```
ftext("hello", fp_text())
```

layout_properties *slide layout properties*

Description

get informations about a particular slide layout into a data.frame.

Usage

```
layout_properties(x, layout = NULL, master = NULL)
```

Arguments

x	rpptx object
layout	slide layout name to use
master	master layout name where layout is located

Examples

```
x <- read_pptx()
layout_properties ( x = x, layout = "Title Slide", master = "Office Theme" )
layout_properties ( x = x, master = "Office Theme" )
layout_properties ( x = x, layout = "Two Content" )
layout_properties ( x = x )
```

---

layout_summary *presentation layouts summary*

Description

get informations about slide layouts and master layouts into a data.frame.

Usage

```
layout_summary(x)
```

Arguments

x	rpptx object

Examples

```
my_pres <- read_pptx()
layout_summary ( x = my_pres )
```

---

media_extract *Extract media from a document object*

Description

Extract files from an rdocx or rpptx object.

Usage

```
media_extract(x, path, target)
```

Arguments

x	an rpptx object or an rdocx object
path	media path, should be a relative path
target	target file

Examples

```
example_pptx <- system.file(package = "officer",
  "doc_examples/example.pptx")
doc <- read_pptx(example_pptx)
content <- pptx_summary(doc)
image_row <- content[content$content_type %in% "image", ]
media_file <- image_row$media_file
media_extract(doc, path = media_file, target = "extract.png")
```

`officer` *officer: Manipulate Microsoft Word and PowerPoint Documents*

Description

The officer package facilitates access to and manipulation of 'Microsoft Word' and 'Microsoft PowerPoint' documents from R.

Details

- read Word and PowerPoint files into data objects
- add/edit/remove image, table and text content from documents and slides
- write updated content back to Word and PowerPoint files

To learn more about officer, start with the vignettes: 'browseVignettes(package = "officer")'

Author(s)

Maintainer: David Gohel <david.gohel@ardata.fr>

See Also

Useful links:

- https://davidgohel.github.io/officer
- Report bugs at https://github.com/davidgohel/officer/issues

on_slide *change current slide*

Description

change current slide index of an rpptx object.

Usage

```
on_slide(x, index)
```

Arguments

x	rpptx object
index	slide index

Examples

```
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- on_slide( doc, index = 1)
doc <- ph_with_text(x = doc, type = "title", str = "First title")
doc <- on_slide( doc, index = 3)
doc <- ph_with_text(x = doc, type = "title", str = "Third title")

print(doc, target = "on_slide.pptx" )
```

pack_folder *compress a folder*

Description

compress a folder to a target file. The function returns the complete path to target file.

Usage

```
pack_folder(folder, target)
```

Arguments

folder	folder to compress
target	path of the archive to create

Details

The function is using zip, it needs a zip program.

ph_add_fpar *append fpar*

Description

append `fpar` (a formatted paragraph) in a placeholder

Usage

```
ph_add_fpar(x, value, type = "body", id_chr = NULL, level = 1)
```

Arguments

`x`	a pptx device
`value`	fpar object
`type`	placeholder type
`id_chr`	placeholder id (a string). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'. Values can be read from `slide_summary`.
`level`	paragraph level

See Also

`fpar`

Examples

```
library(magrittr)

bold_face <- shortcuts$fp_bold(font.size = 30)
bold_redface <- update(bold_face, color = "red")

fpar_ <- fpar(ftext("Hello ", prop = bold_face),
  ftext("World", prop = bold_redface ),
  ftext(", how are you?", prop = bold_face ) )

doc <- read_pptx() %>%
  add_slide(layout = "Title and Content", master = "Office Theme") %>%
  ph_empty(type = "body") %>%
  ph_add_fpar(value = fpar_, type = "body", level = 2)

print(doc, target = "ph_add_fpar.pptx")
```

ph_add_par *append paragraph*

Description

append a new empty paragraph in a placeholder

Usage

```
ph_add_par(x, type = NULL, id_chr = NULL, level = 1)
```

Arguments

x	a pptx device
type	placeholder type
id_chr	placeholder id (a string). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'. Values can be read from slide_summary.
level	paragraph level

Examples

```
library(magrittr)

fileout <- tempfile(fileext = ".pptx")
default_text <- fp_text(font.size = 0, bold = TRUE, color = "red")

doc <- read_pptx() %>%
  add_slide(layout = "Title and Content", master = "Office Theme") %>%
  ph_with_text(type = "body", str = "A text") %>%
  ph_add_par(level = 2) %>%
  ph_add_text(str = "and another, ", style = default_text ) %>%
  ph_add_par(level = 3) %>%
  ph_add_text(str = "and another!",
    style = update(default_text, color = "blue"))

print(doc, target = fileout)
```

ph_add_text *append text*

Description

append text in a placeholder

Usage

```
ph_add_text(x, str, type = NULL, id_chr = NULL, style = fp_text(font.size
  = 0), pos = "after", href = NULL)
```

Arguments

x	a pptx device
str	text to add
type	placeholder type
id_chr	placeholder id (a string). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'. Values can be read from slide_summary.
style	text style, a fp_text object
pos	where to add the new element relative to the cursor, "after" or "before".
href	hyperlink

Examples

```
library(magrittr)
fileout <- tempfile(fileext = ".pptx")
my_pres <- read_pptx() %>%
  add_slide(layout = "Title and Content", master = "Office Theme") %>%
  ph_empty(type = "body")

small_red <- fp_text(color = "red", font.size = 14)

my_pres <- my_pres %>%
  ph_add_par(level = 3) %>%
  ph_add_text(str = "A small red text.", style = small_red) %>%
  ph_add_par(level = 2) %>%
  ph_add_text(str = "Level 2")

print(my_pres, target = fileout)
```

---

ph_empty *add a new empty shape*

Description

add a new empty shape in the current slide.

Usage

```
ph_empty(x, type = "title", index = 1)

ph_empty_at(x, left, top, width, height, bg = "transparent", rot = 0,
  template_type = NULL, template_index = 1)
```

Arguments

x	a pptx device
type	placeholder type
index	placeholder index (integer). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'.
left, top	location of the new shape on the slide
width, height	shape size in inches
bg	background color
rot	rotation angle
template_type	placeholder template type. If used, the new shape will inherit the style from the placeholder template. If not used, no text property is defined and for example text lists will not be indented.
template_index	placeholder template index (integer). To be used when a placeholder template type is not unique in the current slide, e.g. two placeholders with type 'body'.

Examples

```
fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_empty(x = doc, type = "title")

print(doc, target = fileout )

# demo ph_empty_at ------
fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_empty_at(x = doc, left = 1, top = 2, width = 5, height = 4)

print(doc, target = fileout )
```

---

ph_from_xml *add an xml string as new shape*

Description

Add an xml string as new shape in the current slide. This function is to be used to add custom openxml code.

Usage

```
ph_from_xml(x, value, type = "body", index = 1)

ph_from_xml_at(x, value, left, top, width, height)
```

Arguments

x	a pptx device
value	a character
type	placeholder type
index	placeholder index (integer). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'.
left, top	location of the new shape on the slide
width, height	shape size in inches

---

ph_hyperlink *hyperlink a placeholder*

Description

add hyperlink to a placeholder in the current slide.

Usage

```
ph_hyperlink(x, type = NULL, id_chr = NULL, href)
```

Arguments

x	a pptx device
type	placeholder type
id_chr	placeholder id (a string). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'. Values can be read from slide_summary.
href	hyperlink (do not forget prefix http or https)

Examples

```
fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_with_text(x = doc, type = "title", str = "Un titre 1")
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_with_text(x = doc, type = "title", str = "Un titre 2")
doc <- on_slide(doc, 1)
slide_summary(doc) # read column id here
doc <- ph_hyperlink(x = doc, id_chr = "2",
  href = "https://cran.r-project.org")

print(doc, target = fileout )
```

ph_remove *remove shape*

Description

remove a shape in a slide

Usage

```
ph_remove(x, type = NULL, id_chr = NULL)
```

Arguments

x	a pptx device
type	placeholder type
id_chr	placeholder id (a string). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'. Values can be read from slide_summary.

Examples

```
fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_with_text(x = doc, type = "title", str = "Un titre")
slide_summary(doc) # read column id here
doc <- ph_remove(x = doc, type = "title", id_chr = "2")

print(doc, target = fileout )
```

ph_slidelink *slide link to a placeholder*

Description

add slide link to a placeholder in the current slide.

Usage

```
ph_slidelink(x, type = NULL, id_chr = NULL, slide_index)
```

Arguments

x	a pptx device
type	placeholder type
id_chr	placeholder id (a string). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'. Values can be read from slide_summary.
slide_index	slide index to reach

Examples

```
fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_with_text(x = doc, type = "title", str = "Un titre 1")
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_with_text(x = doc, type = "title", str = "Un titre 2")
doc <- on_slide(doc, 1)
slide_summary(doc) # read column id here
doc <- ph_slidelink(x = doc, id_chr = "2", slide_index = 2)

print(doc, target = fileout )
```

---

ph_with_img *add image*

Description

add an image as a new shape in the current slide.

Usage

```
ph_with_img(x, src, type = "body", index = 1, width = NULL,
  height = NULL)

ph_with_img_at(x, src, left, top, width, height, rot = 0)
```

Arguments

x	a pptx device
src	image path
type	placeholder type
index	placeholder index (integer). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'.
width, height	image size in inches
left, top	location of the new shape on the slide
rot	rotation angle

Examples

```
fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")

img.file <- file.path( Sys.getenv("R_HOME"), "doc", "html", "logo.jpg" )
if( file.exists(img.file) ){
  doc <- ph_with_img(x = doc, type = "body", src = img.file, height = 1.06, width = 1.39 )
}

print(doc, target = fileout )

fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")

img.file <- file.path( Sys.getenv("R_HOME"), "doc", "html", "logo.jpg" )
if( file.exists(img.file) ){
  doc <- ph_with_img_at(x = doc, src = img.file, height = 1.06, width = 1.39,
    left = 4, top = 4, rot = 45 )
}

print(doc, target = fileout )
```

---

ph_with_table *add table*

Description

add a table as a new shape in the current slide.

Usage

```
ph_with_table(x, value, type = "title", index = 1, header = TRUE,
  first_row = TRUE, first_column = FALSE, last_row = FALSE,
  last_column = FALSE)

ph_with_table_at(x, value, left, top, width, height, header = TRUE,
  first_row = TRUE, first_column = FALSE, last_row = FALSE,
  last_column = FALSE)
```

Arguments

x	a pptx device
value	data.frame
type	placeholder type
index	placeholder index (integer). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'.

header	display header if TRUE
first_row, last_row, first_column, last_column	logical for PowerPoint table options
left, top	location of the new shape on the slide
width, height	shape size in inches

Examples

```
library(magrittr)

doc <- read_pptx() %>%
  add_slide(layout = "Title and Content", master = "Office Theme") %>%
  ph_with_table(value = mtcars[1:6,], type = "body",
    last_row = FALSE, last_column = FALSE, first_row = TRUE)

print(doc, target = "ph_with_table.pptx")

library(magrittr)

doc <- read_pptx() %>%
  add_slide(layout = "Title and Content", master = "Office Theme") %>%
  ph_with_table_at(value = mtcars[1:6,],
    height = 4, width = 8, left = 4, top = 4,
    last_row = FALSE, last_column = FALSE, first_row = TRUE)

print(doc, target = "ph_with_table2.pptx")
```

---

ph_with_text *add text into a new shape*

Description

add text into a new shape in a slide.

Usage

```
ph_with_text(x, str, type = "title", index = 1)
```

Arguments

x	a pptx device
str	text to add
type	placeholder type
index	placeholder index (integer). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'.

Examples

```
fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_with_text(x = doc, type = "title", str = "Un titre")
doc <- ph_with_text(x = doc, type = "ftr", str = "pied de page")
doc <- ph_with_text(x = doc, type = "dt", str = format(Sys.Date()))
doc <- ph_with_text(x = doc, type = "sldNum", str = "slide 1")

doc <- add_slide(doc, layout = "Title Slide", master = "Office Theme")
doc <- ph_with_text(x = doc, type = "subTitle", str = "Un sous titre")
doc <- ph_with_text(x = doc, type = "ctrTitle", str = "Un titre")

print(doc, target = fileout )
```

---

pptx_summary *get PowerPoint content in a tidy format*

Description

read content of a PowerPoint document and return a tidy dataset representing the document.

Usage

```
pptx_summary(x)
```

Arguments

x	an rpptx object

Examples

```
example_pptx <- system.file(package = "officer",
  "doc_examples/example.pptx")
doc <- read_pptx(example_pptx)
pptx_summary(doc)
```

---

read_docx *open a connexion to a 'Word' file*

Description

read and import a docx file as an R object representing the document.

Usage

```
read_docx(path = NULL)

## S3 method for class 'rdocx'
print(x, target = NULL, ...)

## S3 method for class 'rdocx'
length(x)
```

Arguments

path	path to the docx file to use a base document.
x	a rdocx object
target	path to the docx file to write
...	unused

Examples

```
# create a rdocx object with default template ---
read_docx()

print(read_docx())
# write a rdocx object in a docx file ----
if( require(magrittr) ){
  read_docx() %>% print(target = "out.docx")
  # full path of produced file is returned
  print(.Last.value)
}

# how many element are there in the document ----
length( read_docx() )
```

---

read_pptx *open a connexion to a 'PowerPoint' file*

Description

read and import a pptx file as an R object representing the document.

Usage

```
read_pptx(path = NULL)

## S3 method for class 'rpptx'
print(x, target = NULL, ...)

## S3 method for class 'rpptx'
length(x)
```

Arguments

path	path to the pptx file to use a base document.
x	an rpptx object
target	path to the pptx file to write
...	unused

number of slides

Function length will return the number of slides.

Examples

```
read_pptx()
# write a rdocx object in a docx file ----
if( require(magrittr) ){
  read_pptx() %>% print(target = "out.pptx")
  # full path of produced file is returned
  print(.Last.value)
}
```

---

remove_slide *remove a slide*

Description

remove a slide from a pptx presentation

Usage

```
remove_slide(x, index = NULL)
```

Arguments

x	rpptx object
index	slide index, default to current slide position.

Note

cursor is set on the last slide.

Examples

```
my_pres <- read_pptx()
my_pres <- add_slide(my_pres,
  layout = "Two Content", master = "Office Theme")

my_pres <- remove_slide(my_pres)
```

set_doc_properties *set document properties*

Description

set Word or PowerPoint document properties. These are not visible in the document but are available as metadata of the document.

Usage

```
set_doc_properties(x, title = NULL, subject = NULL, creator = NULL,
  description = NULL, created = NULL)
```

Arguments

x	a rdocx or rpptx object
title, subject, creator, description	text fields
created	a date object

Note

Fields "last modified" and "last modified by" will be automatically be updated when file will be written.

Examples

```
library(magrittr)
read_docx() %>% set_doc_properties(title = "title",
  subject = "document subject", creator = "Me me me",
  description = "this document is empty",
  created = Sys.time()) %>% doc_properties()
```

shortcuts *shortcuts for formatting properties*

Description

Shortcuts for fp_text, fp_par, fp_cell and fp_border.

Usage

```
shortcuts
```

Examples

```
shortcuts$fp_bold()
shortcuts$fp_italic()
shortcuts$b_null()
```

`slide_summary` *get PowerPoint slide content in a tidy format*

Description

get content and positions of current slide into a data.frame. If any table, image or paragraph, data is imported into the resulting data.frame.

Usage

```
slide_summary(x, index = NULL)
```

Arguments

x	rpptx object
index	slide index

Examples

```
library(magrittr)

my_pres <- read_pptx() %>%
  add_slide(layout = "Two Content", master = "Office Theme") %>%
  ph_with_text(type = "dt", str = format(Sys.Date())) %>%
  add_slide(layout = "Title and Content", master = "Office Theme")

slide_summary(my_pres)
slide_summary(my_pres, index = 1)
```

`slip_in_img` *append an image*

Description

append an image into a paragraph of an rdocx object

Usage

```
slip_in_img(x, src, style = NULL, width, height, pos = "after")
```

Arguments

x	an rdocx object
src	image filename
style	text style
width	height in inches
height	height in inches
pos	where to add the new element relative to the cursor, "after" or "before".

Examples

```
library(magrittr)
img.file <- file.path( Sys.getenv("R_HOME"), "doc", "html", "logo.jpg" )
x <- read_docx() %>%
  body_add_par("R logo: ", style = "Normal") %>%
  slip_in_img(src = img.file, style = "strong", width = .3, height = .3)

print(x, target = "append_img.docx")
```

---

slip_in_seqfield *append seq field*

Description

append seq field into a paragraph of an rdocx object

Usage

```
slip_in_seqfield(x, str, style = NULL, pos = "after")
```

Arguments

x	an rdocx object
str	seq field value
style	text style
pos	where to add the new element relative to the cursor, "after" or "before".

Examples

```
library(magrittr)
x <- read_docx() %>%
  body_add_par("Time is: ", style = "Normal") %>%
  slip_in_seqfield(
    str = "TIME \u005C@ \"HH:mm:ss\" \u005C* MERGEFORMAT",
    style = 'strong') %>%

  body_add_par(" - This is a figure title", style = "centered") %>%
```

```
  slip_in_seqfield(str = "SEQ Figure \u005C* roman",
    style = 'Default Paragraph Font', pos = "before") %>%
  slip_in_text("Figure: ", style = "strong", pos = "before") %>%

  body_add_par(" - This is another figure title", style = "centered") %>%
  slip_in_seqfield(str = "SEQ Figure \u005C* roman",
    style = 'strong', pos = "before")  %>%
  slip_in_text("Figure: ", style = "strong", pos = "before") %>%
  body_add_par("This is a symbol: ", style = "Normal") %>%
  slip_in_seqfield(str = "SYMBOL 100 \u005Cf Wingdings",
    style = 'strong')

print(x, target = "seqfield.docx")
```

`slip_in_text` *append text*

Description

append text into a paragraph of an rdocx object

Usage

```
slip_in_text(x, str, style = NULL, pos = "after")
```

Arguments

`x`	an rdocx object
`str`	text
`style`	text style
`pos`	where to add the new element relative to the cursor, "after" or "before".

Examples

```
library(magrittr)
x <- read_docx() %>%
  body_add_par("Hello ", style = "Normal") %>%
  slip_in_text("world", style = "strong") %>%
  slip_in_text("Message is", style = "strong", pos = "before")

print(x, target = "append_run.docx")
```

slip_in_xml *add a wml string into a Word document*

Description

The function add a wml string into the document after, before or on a cursor location.

Usage

```
slip_in_xml(x, str, pos)
```

Arguments

x	an rdocx object
str	a wml string
pos	where to add the new element relative to the cursor, "after" or "before".

styles_info *read Word styles*

Description

read Word styles and get results in a tidy data.frame.

Usage

```
styles_info(x)
```

Arguments

x	a rdocx object

Examples

```
library(magrittr)
read_docx() %>% styles_info()
```

unpack_folder *Extract files from a zip file*

Description

Extract files from a zip file to a folder. The function returns the complete path to destination folder.

Usage

```
unpack_folder(file, folder)
```

Arguments

file	path of the archive to unzip
folder	folder to create

Details

The function is using unzip, it needs an unzip program.

wml_link_images *transform an xml string with images references*

Description

The function replace images filenames in an xml string with their id. The wml code cannot be valid without this operation.

Usage

```
wml_link_images(x, str)
```

Arguments

x	an rdocx object
str	wml string

Details

The function is available to let the creation of valid wml code containing references to images.

Index