Package 'officer'

November 11, 2019

Type Package

Title Manipulation of Microsoft Word and PowerPoint Documents

Version 0.3.6

Description Access and manipulate 'Microsoft Word' and 'Microsoft PowerPoint' documents from R. The package focuses 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 products to be able to write Microsoft files.

License GPL-3

LazyData TRUE

LinkingTo Rcpp

Imports Rcpp (>= 0.12.12), R6, grDevices, base64enc, zip (>= 2.0.3), digest,uuid,stats, magrittr,htmltools,rlang (>= 0.4.0), xml2 (>= 1.1.0)

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

Encoding UTF-8

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

RoxygenNote 6.1.1

Suggests testthat, devEMF,tibble,ggplot2, rmarkdown, knitr, rsvg

VignetteBuilder knitr

NeedsCompilation yes

Author David Gohel [aut, cre],
Frank Hangler [ctb] (function body_replace_all_text),
Liz Sander [ctb] (several documentation fixes),
Anton Victorson [ctb] (fixes xml structures),
Jon Calder [ctb] (update vignettes),
John Harrold [ctb] (fuction annotate_base),
John Muschelli [ctb] (google doc compatibility)

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

Repository CRAN

Date/Publication 2019-11-11 16:00:03 UTC

R topics documented:

add_sheet . . . 4
add_slide . . . 4
annotate_base . . . 5
block_list . . . 6
body_add_blocks . . . 6
body_add_break . . . 7
body_add_docx . . . 8
body_add_fpar . . . 9
body_add_gg . . . 10
body_add_img . . . 10
body_add_par . . . 11
body_add_table . . . 12
body_add_toc . . . 13
body_add_xml . . . 13
body_bookmark . . . 14
body_end_section . . . 14
body_remove . . . 16
body_replace_all_text . . . 17
body_replace_text_at_bkm . . . 18
break_column_before . . . 19
change_styles . . . 20
color_scheme . . . 21
cursor_begin . . . 21
docx_body_relationship . . . 23
docx_body_xml . . . 24
docx_bookmarks . . . 24
docx_dim . . . 25
docx_reference_img . . . 25
docx_show_chunk . . . 26
docx_summary . . . 26
doc_properties . . . 27
external_img . . . 27
fortify_location . . . 28
fpar . . . 29
fp_border . . . 30
fp_cell . . . 31
fp_par . . . 32
fp_sign . . . 33
fp_text . . . 33
ftext . . . 34
layout_properties . . . 35

layout_summary . . . 36
length.rpptx . . . 36
location_eval . . . 37
media_extract . . . 38
move_slide . . . 38
officer . . . 39
on_slide . . . 40
pack_folder . . . 41
ph_add_fpar . . . 41
ph_add_par . . . 42
ph_add_text . . . 43
ph_empty . . . 45
ph_from_xml . . . 46
ph_hyperlink . . . 46
ph_location . . . 47
ph_location_fullsize . . . 48
ph_location_label . . . 49
ph_location_left . . . 50
ph_location_right . . . 51
ph_location_template . . . 51
ph_location_type . . . 53
ph_remove . . . 54
ph_slidelink . . . 55
ph_with . . . 56
ph_with_fpars_at . . . 59
ph_with_gg . . . 60
ph_with_img . . . 61
ph_with_table . . . 61
ph_with_text . . . 62
ph_with_ul . . . 63
pptx_summary . . . 64
print.rpptx . . . 65
read_docx . . . 65
read_pptx . . . 66
read_xlsx . . . 67
remove_slide . . . 68
sanitize_images . . . 68
sections . . . 69
set_doc_properties . . . 70
sheet_select . . . 71
shortcuts . . . 71
slide_size . . . 72
slide_summary . . . 72
slip_in_footnote . . . 73
slip_in_img . . . 74
slip_in_seqfield . . . 75
slip_in_text . . . 76
slip_in_xml . . . 76

styles_info 77
unordered_list 77
unpack_folder 78
wml_link_images 78

Index **79**

`add_sheet` *add a sheet*

Description

add a sheet into an xlsx worksheet

Usage

```
add_sheet(x, label)
```

Arguments

`x`	rxlsx object
`label`	sheet label

Examples

```
my_ws <- read_xlsx()
my_pres <- add_sheet(my_ws, label = "new sheet")
```

`add_slide` *add a slide*

Description

add a slide into a pptx presentation

Usage

```
add_slide(x, layout = "Title and Content", master = "Office Theme")
```

Arguments

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

See Also

print.rpptx read_pptx layout_summary

Other functions slide manipulation: move_slide, on_slide, remove_slide

Examples

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

---

annotate_base *PowerPoint placeholder parameters annotation*

Description

generates a slide from each layout in the base document to identify the placeholder indexes, types, names, master names and layout names.

This is to be used when need to know what parameters should be used with ph_location* calls. The parameters are printed in their corresponding shapes.

Note that if there are duplicated ph_label, you should not use ph_location_label.

Usage

```
annotate_base(path = NULL, output_file = "annotated_layout.pptx")
```

Arguments

path	path to the pptx file to use as base document or NULL to use the officer default
output_file	filename to store the annotated powerpoint file or NULL to suppress generation

Value

rpptx object of the annotated PowerPoint file

See Also

Other functions for reading presentation informations: color_scheme, layout_properties, layout_summary, length.rpptx, slide_size, slide_summary

Examples

```
# To generate an anotation of the default base document with officer:
annotate_base(output_file = tempfile(fileext = ".pptx"))

# To generate an annotation of the base document 'mydoc.pptx' and place the
# annotated output in 'mydoc_annotate.pptx'
# annotate_base(path = 'mydoc.pptx', output_file='mydoc_annotate.pptx')
```

block_list *create paragraph blocks*

Description

a list of blocks can be used to gather several blocks (paragraphs or tables) into a single object. The function is to be used when adding footnotes or formatted paragraphs into a new slide.

Usage

```
block_list(...)
```

Arguments

...	a list of objects of class fpar or flextable.

Examples

```
img.file <- file.path( R.home("doc"), "html", "logo.jpg" )
bl <- block_list(
  fpar(ftext("hello world", shortcuts$fp_bold())),
  fpar(
    ftext("hello world", shortcuts$fp_bold()),
    external_img(src = img.file, height = 1.06, width = 1.39)
  )
)
```

body_add_blocks *add a list of blocks into a document*

Description

add a list of blocks produced by block_list into into an rdocx object

Usage

```
body_add_blocks(x, blocks, pos = "after")
```

Arguments

x	an rdocx object
blocks	set of blocks to be used as footnote content returned by function block_list.
pos	where to add the new element relative to the cursor, one of "after", "before", "on".

Examples

```
library(magrittr)

img.file <- file.path( R.home("doc"), "html", "logo.jpg" )
bl <- block_list(
  fpar(ftext("hello", shortcuts$fp_bold())),
  fpar(
    ftext("hello world", shortcuts$fp_bold()),
    external_img(src = img.file, height = 1.06, width = 1.39)
  )
)

x <- read_docx() %>%
  body_add_blocks( blocks = bl ) %>%
  print(target = tempfile(fileext = ".docx"))
```

---

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 = tempfile(fileext = ".docx"))
```

body_add_docx *insert an external docx*

Description

add content of a docx into an rdocx object.

Usage

```
body_add_docx(x, src, pos = "after")
```

Arguments

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

Note

The function is using a 'Microsoft Word' feature: when the document will be edited, the content of the file will be inserted in the main document.

This feature is unlikely to work as expected if the resulting document is edited by another software.

Examples

```
library(magrittr)
file1 <- tempfile(fileext = ".docx")
file2 <- tempfile(fileext = ".docx")
file3 <- tempfile(fileext = ".docx")
read_docx() %>%
  body_add_par("hello world 1", style = "Normal") %>%
  print(target = file1)
read_docx() %>%
  body_add_par("hello world 2", style = "Normal") %>%
  print(target = file2)

read_docx(path = file1) %>%
  body_add_break() %>%
  body_add_docx(src = file2) %>%
  print(target = file3)
```

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 = tempfile(fileext = ".docx"))

# a way of using fpar to center an image in a Word doc ----
rlogo <- file.path( R.home("doc"), "html", "logo.jpg" )
img_in_par <- fpar(
  external_img(src = rlogo, height = 1.06/2, width = 1.39/2),
  fp_p = fp_par(text.align = "center") )

read_docx() %>% body_add_fpar(img_in_par) %>%
  print(target = tempfile(fileext = ".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

```
if( require("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 = tempfile(fileext = ".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, the basename of the file must not contain any blank.
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( 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 = tempfile(fileext = ".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 name
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 = tempfile(fileext = ".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 to add as a table
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	Specifies that the first column conditional formatting should be applied. Details for this and other conditional formatting options can be found at http://officeopenxml.com/WPtblLook.php
first_column	Specifies that the first column conditional formatting should be applied.
last_row	Specifies that the first column conditional formatting should be applied.
last_column	Specifies that the first column conditional formatting should be applied.
no_hband	Specifies that the first column conditional formatting should be applied.
no_vband	Specifies that the first column conditional formatting should be applied.

Examples

```
library(magrittr)

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

print(doc, target = tempfile(fileext = ".docx") )
```

body_add_toc *add table of content*

Description

add a table of content into an rdocx object. The TOC will be generated by Word, if the document is not edited with Word (i.e. Libre Office) the TOC will not be generated.

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 = tempfile(fileext = ".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. The bookmark is added on the first run of text in the current paragraph.

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 affects preceding paragraphs or tables.

Usage

```
body_end_section(x, landscape = FALSE, margins = c(top = NA, bottom =
  NA, left = NA, right = NA), colwidths = c(1), space = 0.05,
  sep = FALSE, continuous = FALSE)

body_default_section(x, landscape = FALSE, margins = c(top = NA, bottom
  = NA, left = NA, right = NA))
```

Arguments

x	an rdocx object
landscape	landscape orientation
margins	a named vector of margin settings in inches, margins not set remain at their default setting
colwidths	columns widths as percentages, summing to 1. If 3 values, 3 columns will be produced.
space	space in percent between columns.
sep	if TRUE a line is separating columns.
continuous	TRUE for a continuous section break.

Details

A section starts at the end of the previous section (or the beginning of the document if no preceding section exists), and 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.

Note

This function is deprecated, use body_end_section_continuous, body_end_section_landscape, body_end_section_portrait, body_end_section_columns or body_end_section_columns_landscape instead.

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
  slip_in_column_break(pos = "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, margins = c(top = 0.5, bottom = 0.5))

print(my_doc, target = tempfile(fileext = ".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")

new_doc_file <- print(my_doc,
  target = tempfile(fileext = ".docx"))

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

print(my_doc, target = tempfile(fileext = ".docx"))
```

`body_replace_all_text` *Replace text anywhere in the document, or at a cursor*

Description

Replace all occurrences of old_value with new_value. This method uses `grepl`/`gsub` for pattern matching; you may supply arguments as required (and therefore use `regex` features) using the optional `...` argument.

Note that by default, grepl/gsub will use `fixed=FALSE`, which means that `old_value` and `new_value` will be interepreted as regular expressions.

Chunking of text

Note that the behind-the-scenes representation of text in a Word document is frequently not what you might expect! Sometimes a paragraph of text is broken up (or "chunked") into several "runs," as a result of style changes, pauses in text entry, later revisions and edits, etc. If you have not styled the text, and have entered it in an "all-at-once" fashion, e.g. by pasting it or by outputing it programmatically into your Word document, then this will likely not be a problem. If you are working with a manually-edited document, however, this can lead to unexpected failures to find text.

You can use the officer function `docx_show_chunk` to show how the paragraph of text at the current cursor has been chunked into runs, and what text is in each chunk. This can help troubleshoot unexpected failures to find text.

Usage

```
body_replace_all_text(x, old_value, new_value, only_at_cursor = FALSE,
  warn = TRUE, ...)

headers_replace_all_text(x, old_value, new_value, only_at_cursor = FALSE,
  warn = TRUE, ...)

footers_replace_all_text(x, old_value, new_value, only_at_cursor = FALSE,
  warn = TRUE, ...)
```

Arguments

`x`	a docx device
`old_value`	the value to replace
`new_value`	the value to replace it with
`only_at_cursor`	if `TRUE`, only search-and-replace at the current cursor; if `FALSE` (default), search-and-replace in the entire document (this can be slow on large documents!)
`warn`	warn if `old_value` could not be found.
`...`	optional arguments to grepl/gsub (e.g. `fixed=TRUE`)

header_replace_all_text

Replacements will be performed in each header of all sections.

Replacements will be performed in each footer of all sections.

Author(s)

Frank Hangler, <frank@plotandscatter.com>

See Also

grep, regex, docx_show_chunk

Examples

```
library(magrittr)

doc <- read_docx() %>%
  body_add_par("Placeholder one") %>%
  body_add_par("Placeholder two")

# Show text chunk at cursor
docx_show_chunk(doc)  # Output is 'Placeholder two'

# Simple search-and-replace at current cursor, with regex turned off
doc <- body_replace_all_text(doc, old_value = "Placeholder",
  new_value = "new", only_at_cursor = TRUE, fixed = TRUE)
docx_show_chunk(doc)  # Output is 'new two'

# Do the same, but in the entire document and ignoring case
doc <- body_replace_all_text(doc, old_value = "placeholder",
  new_value = "new", only_at_cursor=FALSE, ignore.case = TRUE)
doc <- cursor_backward(doc)
docx_show_chunk(doc) # Output is 'new one'

# Use regex : replace all words starting with "n" with the word "example"
doc <- body_replace_all_text(doc, "\\bn.*?\\b", "example")
docx_show_chunk(doc) # Output is 'example one'
```

---

body_replace_text_at_bkm

replace text at a bookmark location

Description

replace text content enclosed in a bookmark with different 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, but bookmarking few words inside a paragraph is valid.

Usage

```
body_replace_text_at_bkm(x, bookmark, value)

body_replace_img_at_bkm(x, bookmark, value)

headers_replace_text_at_bkm(x, bookmark, value)

headers_replace_img_at_bkm(x, bookmark, value)

footers_replace_text_at_bkm(x, bookmark, value)

footers_replace_img_at_bkm(x, bookmark, value)
```

Arguments

x	a docx device
bookmark	bookmark id
value	the replacement string, of type 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_text_at_bkm("text_to_replace", "not left aligned")


# demo usage of bookmark and images ----
template <- system.file(package = "officer", "doc_examples/example.docx")

img.file <- file.path( R.home("doc"), "html", "logo.jpg" )

doc <- read_docx(path = template)
doc <- headers_replace_img_at_bkm(x = doc, bookmark = "bmk_header",
                          value = external_img(src = img.file, width = .53, height = .7))
doc <- footers_replace_img_at_bkm(x = doc, bookmark = "bmk_footer",
                          value = external_img(src = img.file, width = .53, height = .7))
print(doc, target = tempfile(fileext = ".docx"))
```

---

break_column_before *add a column break*

Description

add a column break into a Word document. A column break is used to add a break in a multi columns section in a Word Document.

Usage

```
break_column_before(x)

slip_in_column_break(x, pos = "before")
```

Arguments

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

---

change_styles *replace paragraphs styles*

Description

Replace styles with others in a Word document.

Usage

```
change_styles(x, mapstyles)
```

Arguments

x	an rdocx object
mapstyles	a named list, names are the replacement style, content (as a character vector) are the styles to be replaced.

Examples

```
library(magrittr)

mapstyles <- list( "centered" = c("Normal"),
    "heading 3" = c("heading 1", "heading 2") )
doc <- read_docx() %>%
  body_add_par("A title", style = "heading 1") %>%
  body_add_par("Another title", style = "heading 2") %>%
  body_add_par("Hello world!", style = "Normal") %>%
  change_styles( mapstyles = mapstyles )

print(doc, target = tempfile(fileext = ".docx"))
```

color_scheme *color scheme*

Description

get master layout color scheme into a data.frame.

Usage

```
color_scheme(x)
```

Arguments

x	an rpptx object

See Also

Other functions for reading presentation informations: annotate_base, layout_properties, layout_summary, length.rpptx, slide_size, slide_summary

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_bookmark

Set the cursor at a bookmark that has previously been set.

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`. The argument `keyword` is a regexpr pattern.

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 = tempfile(fileext = ".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 = tempfile(fileext = ".docx"))
```

`docx_body_relationship` *body xml document*

Description

Get the body document as xml. This function is not to be used by end users, it has been implemented to allow other packages to work with officer.

Usage

```
docx_body_relationship(x)
```

Arguments

x	an rdocx object

Examples

```
doc <- read_docx()
docx_body_relationship(doc)
```

docx_body_xml *body xml document*

Description

Get the body document as xml. This function is not to be used by end users, it has been implemented to allow other packages to work with officer.

Usage

```
docx_body_xml(x)
```

Arguments

x	an rdocx object

Examples

```
doc <- read_docx()
docx_body_xml(doc)
```

docx_bookmarks *List Word bookmarks*

Description

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

Usage

```
docx_bookmarks(x)
```

Arguments

x	an 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	an 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_show_chunk *Show underlying text tag structure*

Description

Show the structure of text tags at the current cursor. This is most useful when trying to troubleshoot search-and-replace functionality using body_replace_all_text.

Usage

```
docx_show_chunk(x)
```

Arguments

x	a docx device

See Also

body_replace_all_text

Examples

```
library(magrittr)

doc <- read_docx() %>%
  body_add_par("Placeholder one") %>%
  body_add_par("Placeholder two")

# Show text chunk at cursor
docx_show_chunk(doc)  # Output is 'Placeholder two'
```

docx_summary *get Word content in a data.frame*

Description

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

Usage

```
docx_summary(x)
```

Arguments

x	an rdocx object

Note

Documents included with `body_add_docx()` will not be accessible in the results.

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 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 in 'PowerPoint' slides.

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

See Also

ph_with

Examples

```
img.file <- file.path( R.home("doc"), "html", "logo.jpg" )

doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(x = doc,
  value = external_img(img.file, width = 1.39, height = 1.06),
  location = ph_location_type(type = "body"),
  use_loc_size = FALSE )
print(doc, target = tempfile(fileext = ".pptx"))
```

---

fortify_location *eval a location on the current slide*

Description

Eval a shape location against the current slide. This function is to be used to add custom openxml code. A list is returned, it contains informations width, height, left and top positions and other informations necessary to add a content on a slide.

Usage

```
fortify_location(x, doc, ...)
```

Arguments

x	a location for a placeholder.
doc	an rpptx object
...	unused arguments

See Also

ph_location, ph_with

Examples

```
doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content",
  master = "Office Theme")
fortify_location(ph_location_fullsize(), doc)
```

`fpar` *concatenate formatted text as a paragraph*

Description

Create a paragraph representation by concatenating formatted text or images.

`fpar` supports `ftext`, `external_img` and simple strings. All its arguments will be concatenated to create a paragraph where chunks of text and images are associated with formatting properties.

Default text and paragraph formatting properties can also be modified with update.

Usage

```
fpar(..., fp_p = fp_par(), fp_t = fp_text())

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

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

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

Arguments

`...`	unused
`fp_p`	paragraph formatting properties
`fp_t`	default text formatting properties. This is used as text formatting properties when simple text is provided as argument.
`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()))

# mix text and image -----
img.file <- file.path( R.home("doc"), "html", "logo.jpg" )

bold_face <- shortcuts$fp_bold(font.size = 12)
bold_redface <- update(bold_face, color = "red")
fpar_1 <- fpar(
  "Hello World, ",
  ftext("how ", prop = bold_redface ),
  external_img(src = img.file, height = 1.06/2, width = 1.39/2),
  ftext(" you?", prop = bold_face ) )
fpar_1

img_in_par <- fpar(
  external_img(src = img.file, height = 1.06/2, width = 1.39/2),
  fp_p = fp_par(text.align = "center") )
```

---

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, ...)
```

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
object	fp_border object
...	further arguments - not used

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`	`fp_cell` object
`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 set of formatting 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'
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 information about a particular slide layout into a data.frame.

Usage

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

Arguments

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

See Also

Other functions for reading presentation informations: annotate_base, color_scheme, layout_summary, length.rpptx, slide_size, slide_summary

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. This function returns a data.frame containing all layout and master names.

Usage

```
layout_summary(x)
```

Arguments

x	an rpptx object

See Also

Other functions for reading presentation informations: annotate_base, color_scheme, layout_properties, length.rpptx, slide_size, slide_summary

Examples

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

---

length.rpptx *number of slides*

Description

Function length will return the number of slides.

Usage

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

Arguments

x	an rpptx object

See Also

Other functions for reading presentation informations: annotate_base, color_scheme, layout_properties, layout_summary, slide_size, slide_summary

Examples

```
my_pres <- read_pptx()
my_pres <- add_slide(my_pres)
my_pres <- add_slide(my_pres)
length(my_pres)
```

---

location_eval *Utility to eval a location*

Description

Eval a shape location with fortify_location. This function will be removed in the next release; it was required when location was a quosure but this is no more necessary.

Usage

```
location_eval(location, x)
```

Arguments

location	a location for a placeholder.
x	an rpptx object

See Also

ph_location, ph_with

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
png_file <- tempfile(fileext = ".png")
media_extract(doc, path = media_file, target = png_file)
```

move_slide *move a slide*

Description

move a slide in a pptx presentation

Usage

```
move_slide(x, index, to)
```

Arguments

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

Note

cursor is set on the last slide.

See Also

Other functions slide manipulation: add_slide, on_slide, remove_slide

Examples

```
x <- read_pptx()
x <- add_slide(x)
x <- ph_with(x, "Hello world 1", location = ph_location_type())
x <- add_slide(x)
x <- ph_with(x, "Hello world 2", location = ph_location_type())
x <- move_slide(x, index = 1, to = 2)
```

---

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

Examples of manipulations are:

- 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>

Other contributors:

- Frank Hangler <frank@plotandscatter.com> (function body_replace_all_text) [contributor]
- Liz Sander <lsander@civisanalytics.com> (several documentation fixes) [contributor]
- Anton Victorson <anton@victorson.se> (fixes xml structures) [contributor]
- Jon Calder <jonmcalder@gmail.com> (update vignettes) [contributor]
- John Harrold <john.m.harrold@gmail.com> (fuction annotate_base) [contributor]
- John Muschelli <muschellij2@gmail.com> (google doc compatibility) [contributor]

See Also

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

---

on_slide *change current slide*

Description

change current slide index of an rpptx object.

Usage

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

Arguments

x	an rpptx object
index	slide index

See Also

Other functions slide manipulation: add_slide, move_slide, remove_slide

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(x = doc, "First title",
  location = ph_location_type(type="title"))
doc <- on_slide( doc, index = 3)
doc <- ph_with(x = doc, "Third title",
  location = ph_location_type(type="title"))

file <- tempfile(fileext = ".pptx")
print(doc, target = file )
```

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

ph_add_fpar *append fpar*

Description

append fpar (a formatted paragraph) in a placeholder The function let you add a new formatted paragraph (fpar) to an existing content in an exiisting shape, existing paragraphs will be preserved.

Usage

```
ph_add_fpar(x, value, type = "body", id = 1, id_chr = NULL,
  ph_label = NULL, level = 1, par_default = TRUE)
```

Arguments

x	an rpptx object
value	fpar object
type	placeholder type
id	placeholder index (integer) for a duplicated type. This is to be used when a placeholder type is not unique in the layout of the current slide, e.g. two placeholders with type 'body'. To add onto the first, use id = 1 and id = 2 for the second one. Values can be read from slide_summary.
id_chr	deprecated.
ph_label	label associated to the placeholder. Use column ph_label of result returned by slide_summary.
level	paragraph level
par_default	specify if the default paragraph formatting should be used.

Usage

If your goal is to add formatted text in a new shape, use ph_with with a block_list instead of this function.

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(location = ph_location(bg = "wheat", newlabel = "myph")) %>%
  ph_add_fpar(value = fpar_, ph_label = "myph", level = 2)

print(doc, target = tempfile(fileext = ".pptx"))
```

---

ph_add_par *append paragraph*

Description

append a new empty paragraph in a placeholder. The function let you add a new empty paragraph to an existing content in an exiisting shape, existing paragraphs will be preserved.

Usage

```
ph_add_par(x, type = "body", id = 1, id_chr = NULL, level = 1,
  ph_label = NULL)
```

Arguments

x	an rpptx object
type	placeholder type
id	placeholder index (integer) for a duplicated type. This is to be used when a placeholder type is not unique in the layout of the current slide, e.g. two placeholders with type 'body'. To add onto the first, use id = 1 and id = 2 for the second one. Values can be read from slide_summary.

id_chr	deprecated.
level	paragraph level
ph_label	label associated to the placeholder. Use column ph_label of result returned by slide_summary.

Usage

If your goal is to add formatted text in a new shape, use ph_with with a block_list instead of this function.

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("A text", location = ph_location_type(type = "body")) %>%
  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. The function let you add text to an existing content in an exiisting shape, existing text will be preserved.

Usage

```
ph_add_text(x, str, type = "body", id = 1, id_chr = NULL,
  ph_label = NULL, style = fp_text(font.size = 0), pos = "after",
  href = NULL, slide_index = NULL)
```

Arguments

x	an rpptx object
str	text to add
type	placeholder type

id	placeholder index (integer) for a duplicated type. This is to be used when a placeholder type is not unique in the layout of the current slide, e.g. two placeholders with type 'body'. To add onto the first, use `id = 1` and `id = 2` for the second one. Values can be read from `slide_summary`.
id_chr	deprecated.
ph_label	label associated to the placeholder. Use column `ph_label` of result returned by `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 to reach when clicking the text
slide_index	slide index to reach when clicking the text. It will be ignored if `href` is not NULL.

Usage

If your goal is to add formatted text in a new shape, use `ph_with` with a `block_list` instead of this function.

Examples

```
fileout <- tempfile(fileext = ".pptx")
my_pres <- read_pptx()
my_pres <- add_slide(my_pres)
my_pres <- ph_empty(my_pres,
  location = ph_location_type(type = "body"))

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

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

print(my_pres, target = fileout)

# another example ----
fileout <- tempfile(fileext = ".pptx")

doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Un titre 2",
  location = ph_location_type(type = "title"))
doc <- ph_empty(doc,
  location = ph_location(rotation = 90, bg = "red",
      newlabel = "myph"))
doc <- ph_add_par(doc, ph_label = "myph", level = 2)
doc <- ph_add_text(doc, str = "Jump here to slide 2!",
  ph_label = "myph")
```

```
print(doc, target = fileout)
```

---

ph_empty *add a new empty shape*

Description

add a new empty shape in the current slide. This function was implemented for development purpose and should not be used.

Usage

```
ph_empty(x, type = "body", index = 1, location = NULL)

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

Arguments

x	an pptx object
type	placeholder type (i.e. 'body', 'title')
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', the first one will be added with index 1 and the second one with index 2. It is recommanded to use argument location instead of type and index.
location	a placeholder location object. This is a convenient argument that can replace usage of arguments type and index. See ph_location_type, ph_location, ph_location_label, ph_location_left, ph_location_right, ph_location_fullsize.
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 = "body", index = 1)
doc <- ph_empty(x = doc, location = ph_location_right())

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	an pptx object
value	a character
type	placeholder type (i.e. 'body', 'title')
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', the first one will be added with index 1 and the second one with index 2. It is recommanded to use argument `location` instead of `type` and `index`.
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 = "body", id = 1, id_chr = NULL,
  ph_label = NULL, href)
```

Arguments

x	an rpptx object
type	placeholder type
id	placeholder index (integer) for a duplicated type. This is to be used when a placeholder type is not unique in the layout of the current slide, e.g. two placeholders with type 'body'. To add onto the first, use `id = 1` and `id = 2` for the second one. Values can be read from `slide_summary`.
id_chr	deprecated.
ph_label	label associated to the placeholder. Use column `ph_label` of result returned by `slide_summary`.
href	hyperlink (do not forget http or https prefix)

See Also

`ph_with`

Other functions for placeholders manipulation: `ph_remove`, `ph_slidelink`

Examples

```
fileout <- tempfile(fileext = ".pptx")
loc_manual <- ph_location(bg = "red", newlabel= "mytitle")
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(x = doc, "Un titre 1", location = loc_manual)
slide_summary(doc) # read column ph_label here
doc <- ph_hyperlink(x = doc, ph_label = "mytitle",
  href = "https://cran.r-project.org")

print(doc, target = fileout )
```

---

ph_location *create a location for a placeholder*

Description

The function will return a list that complies with expected format for argument `location` of functionph_with.

Usage

```
ph_location(left = 1, top = 1, width = 4, height = 3,
  newlabel = "", bg = NULL, rotation = NULL, ...)
```

Arguments

left, top, width, height	place holder coordinates in inches.
newlabel	a label for the placeholder. See section details.
bg	background color
rotation	rotation angle
...	unused arguments

Details

The location of the bounding box associated to a placeholder within a slide is specified with the left top coordinate, the width and the height. These are defined in inches:

left left coordinate of the bounding box

top top coordinate of the bounding box

width width of the bounding box

height height of the bounding box

In addition to these attributes, a label can be associated with the shape. Shapes, text boxes, images and other objects will be identified with that label in the Selection Pane of PowerPoint. This label can then be reused by other functions such as `ph_location_label()`. It can be set with argument `newlabel`.

See Also

Other functions for placeholder location: `ph_location_fullsize`, `ph_location_label`, `ph_location_left`, `ph_location_right`, `ph_location_template`, `ph_location_type`

Examples

```
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Hello world",
  location = ph_location(width = 4, height = 3, newlabel = "hello") )
print(doc, target = tempfile(fileext = ".pptx") )
```

---

ph_location_fullsize *location of a full size element*

Description

The function will return the location corresponding to a full size display.

Usage

```
ph_location_fullsize(newlabel = "", ...)
```

Arguments

newlabel	a label to associate with the placeholder.
...	unused arguments

See Also

Other functions for placeholder location: ph_location_label, ph_location_left, ph_location_right, ph_location_template, ph_location_type, ph_location

Examples

```
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Hello world", location = ph_location_fullsize() )
print(doc, target = tempfile(fileext = ".pptx") )
```

---

ph_location_label *location of a named placeholder*

Description

The function will use the label of a placeholder to find the corresponding location.

Usage

```
ph_location_label(ph_label, newlabel = NULL, ...)
```

Arguments

ph_label	placeholder label of the used layout. It can be read in PowerPoint or with function layout_properties() in column ph_label.
newlabel	a label to associate with the placeholder.
...	unused arguments

Details

The location of the bounding box associated to a placeholder within a slide is specified with the left top coordinate, the width and the height. These are defined in inches:

left left coordinate of the bounding box

top top coordinate of the bounding box

width width of the bounding box

height height of the bounding box

In addition to these attributes, a label can be associated with the shape. Shapes, text boxes, images and other objects will be identified with that label in the Selection Pane of PowerPoint. This label can then be reused by other functions such as ph_location_label(). It can be set with argument newlabel.

See Also

Other functions for placeholder location: ph_location_fullsize, ph_location_left, ph_location_right, ph_location_template, ph_location_type, ph_location

Examples

```
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Hello world",
  location = ph_location_label(ph_label = "Content Placeholder 2") )
print(doc, target = tempfile(fileext = ".pptx") )
```

---

ph_location_left *location of a left body element*

Description

The function will return the location corresponding to a left bounding box. The function assume the layout 'Two Content' is existing.

Usage

```
ph_location_left(newlabel = NULL, ...)
```

Arguments

newlabel	a label to associate with the placeholder.
...	unused arguments

See Also

Other functions for placeholder location: ph_location_fullsize, ph_location_label, ph_location_right, ph_location_template, ph_location_type, ph_location

Examples

```
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Hello left", location = ph_location_left() )
doc <- ph_with(doc, "Hello right", location = ph_location_right() )
print(doc, target = tempfile(fileext = ".pptx") )
```

ph_location_right *location of a right body element*

Description

The function will return the location corresponding to a right bounding box. The function assume the layout 'Two Content' is existing.

Usage

```
ph_location_right(newlabel = NULL, ...)
```

Arguments

newlabel	a label to associate with the placeholder.
...	unused arguments

See Also

Other functions for placeholder location: ph_location_fullsize, ph_location_label, ph_location_left, ph_location_template, ph_location_type, ph_location

Examples

```
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Hello left", location = ph_location_left() )
doc <- ph_with(doc, "Hello right", location = ph_location_right() )
print(doc, target = tempfile(fileext = ".pptx") )
```

ph_location_template *create a location for a placeholder based on a template*

Description

The function will return a list that complies with expected format for argument location of function ph_with. A placeholder will be used as template and its positions will be updated with values left, top, width, height.

Usage

```
ph_location_template(left = 1, top = 1, width = 4, height = 3,
  newlabel = "", type = NULL, id = 1, ...)
```

Arguments

left, top, width, height	place holder coordinates in inches.
newlabel	a label for the placeholder. See section details.
type	placeholder type to look for in the slide layout, one of 'body', 'title', 'ctrTitle', 'subTitle', 'dt', 'ftr', 'sldNum'. It will be used as a template placeholder.
id	index of the placeholder template. If two body placeholder, there can be two different index: 1 and 2 for the first and second body placeholders defined in the layout.
...	unused arguments

Details

The location of the bounding box associated to a placeholder within a slide is specified with the left top coordinate, the width and the height. These are defined in inches:

left left coordinate of the bounding box

top top coordinate of the bounding box

width width of the bounding box

height height of the bounding box

In addition to these attributes, a label can be associated with the shape. Shapes, text boxes, images and other objects will be identified with that label in the Selection Pane of PowerPoint. This label can then be reused by other functions such as `ph_location_label()`. It can be set with argument `newlabel`.

See Also

Other functions for placeholder location: `ph_location_fullsize`, `ph_location_label`, `ph_location_left`, `ph_location_right`, `ph_location_type`, `ph_location`

Examples

```
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Title",
  location = ph_location_type(type = "title") )
doc <- ph_with(doc, "Hello world",
    location = ph_location_template(top = 4, type = "title") )
print(doc, target = tempfile(fileext = ".pptx") )
```

ph_location_type *location of a placeholder based on a type*

Description

The function will use the type name of the placeholder (e.g. body, title), the layout name and few other criterias to find the corresponding location.

Usage

```
ph_location_type(type = "body", position_right = TRUE,
  position_top = TRUE, newlabel = NULL, id = NULL, ...)
```

Arguments

type	placeholder type to look for in the slide layout, one of 'body', 'title', 'ctrTitle', 'subTitle', 'dt', 'ftr', 'sldNum'.
position_right	the parameter is used when a selection with above parameters does not provide a unique position (for example layout 'Two Content' contains two element of type 'body'). If TRUE, the element the most on the right side will be selected, otherwise the element the most on the left side will be selected.
position_top	same than position_right but applied to top versus bottom.
newlabel	a label to associate with the placeholder.
id	index of the placeholder. If two body placeholder, there can be two different index: 1 and 2 for the first and second body placeholders defined in the layout. If this argument is used, position_right and position_top will be ignored.
...	unused arguments

Details

The location of the bounding box associated to a placeholder within a slide is specified with the left top coordinate, the width and the height. These are defined in inches:

left left coordinate of the bounding box

top top coordinate of the bounding box

width width of the bounding box

height height of the bounding box

In addition to these attributes, a label can be associated with the shape. Shapes, text boxes, images and other objects will be identified with that label in the Selection Pane of PowerPoint. This label can then be reused by other functions such as ph_location_label(). It can be set with argument newlabel.

See Also

Other functions for placeholder location: ph_location_fullsize, ph_location_label, ph_location_left, ph_location_right, ph_location_template, ph_location

Examples

```
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Hello world",
  location = ph_location_type(type = "body") )
print(doc, target = tempfile(fileext = ".pptx") )
```

---

ph_remove *remove a shape*

Description

remove a shape in a slide

Usage

```
ph_remove(x, type = "body", id = 1, ph_label = NULL, id_chr = NULL)
```

Arguments

x	an rpptx object
type	placeholder type
id	placeholder index (integer) for a duplicated type. This is to be used when a placeholder type is not unique in the layout of the current slide, e.g. two placeholders with type 'body'. To add onto the first, use `id = 1` and `id = 2` for the second one. Values can be read from slide_summary.
ph_label	label associated to the placeholder. Use column `ph_label` of result returned by slide_summary.
id_chr	deprecated.

See Also

ph_with

Other functions for placeholders manipulation: ph_hyperlink, ph_slidelink

Examples

```
fileout <- tempfile(fileext = ".pptx")
dummy_fun <- function(doc){
  doc <- add_slide(doc, layout = "Two Content",
    master = "Office Theme")
  doc <- ph_with(x = doc, value = "Un titre",
    location = ph_location_type(type = "title"))
  doc <- ph_with(x = doc, value = "Un corps 1",
    location = ph_location_type(type = "body", id = 1))
  doc <- ph_with(x = doc, value = "Un corps 2",
    location = ph_location_type(type = "body", id = 2))
  doc
}
doc <- read_pptx()
for(i in 1:3)
  doc <- dummy_fun(doc)

doc <- on_slide(doc, index = 1)
doc <- ph_remove(x = doc, type = "title")

doc <- on_slide(doc, index = 2)
doc <- ph_remove(x = doc, type = "body", id = 2)

doc <- on_slide(doc, index = 3)
doc <- ph_remove(x = doc, type = "body", id = 1)

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 = "body", id = 1, id_chr = NULL,
  ph_label = NULL, slide_index)
```

Arguments

x	an rpptx object
type	placeholder type
id	placeholder index (integer) for a duplicated type. This is to be used when a placeholder type is not unique in the layout of the current slide, e.g. two placeholders with type 'body'. To add onto the first, use `id = 1` and `id = 2` for the second one. Values can be read from `slide_summary`.

id_chr	deprecated.
ph_label	label associated to the placeholder. Use column ph_label of result returned by slide_summary.
slide_index	slide index to reach

See Also

ph_with

Other functions for placeholders manipulation: ph_hyperlink, ph_remove

Examples

```
fileout <- tempfile(fileext = ".pptx")
loc_title <- ph_location_type(type = "title")
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(x = doc, "Un titre 1", location = loc_title)
doc <- add_slide(doc)
doc <- ph_with(x = doc, "Un titre 2", location = loc_title)
doc <- on_slide(doc, 1)
slide_summary(doc) # read column ph_label here
doc <- ph_slidelink(x = doc, ph_label = "Title 1", slide_index = 2)

print(doc, target = fileout )
```

---

ph_with *add objects into a new shape on the current slide*

Description

add object into a new shape in the current slide. This function is able to add all supported outputs to a presentation and should replace calls to older functions starting with ph_with_*.

Usage

```
ph_with(x, value, ...)

## S3 method for class 'character'
ph_with(x, value, location, ...)

## S3 method for class 'numeric'
ph_with(x, value, location, format_fun = format, ...)

## S3 method for class 'factor'
ph_with(x, value, location, ...)

## S3 method for class 'logical'
```

```
ph_with(x, value, location, format_fun = format, ...)

## S3 method for class 'block_list'
ph_with(x, value, location, ...)

## S3 method for class 'unordered_list'
ph_with(x, value, location, ...)

## S3 method for class 'data.frame'
ph_with(x, value, location, header = TRUE,
  first_row = TRUE, first_column = FALSE, last_row = FALSE,
  last_column = FALSE, ...)

## S3 method for class 'gg'
ph_with(x, value, location, ...)

## S3 method for class 'external_img'
ph_with(x, value, location, use_loc_size = TRUE,
  ...)

## S3 method for class 'fpar'
ph_with(x, value, location, ...)

## S3 method for class 'xml_document'
ph_with(x, value, location, ...)
```

Arguments

x	an rpptx object
value	object to add as a new shape. Supported objects are vectors, data.frame, graphics, block of formatted paragraphs, unordered list of formatted paragraphs, pretty tables with package flextable, editable graphics with package rvg, 'Microsoft' charts with package mschart.
...	Arguments to be passed to methods
location	a placeholder location object. It will be used to specify the location of the new shape. This location can be defined with a call to one of the ph_location functions. See section see also.
format_fun	format function for non character vectors
header	display header if TRUE
first_row, last_row, first_column, last_column	logical for PowerPoint table options
use_loc_size	if set to FALSE, external_img width and height will be used.

Methods (by class)

- character: add a character vector to a new shape on the current slide, values will be added as paragraphs.

- `numeric`: add a numeric vector to a new shape on the current slide, values will be be first formatted then added as paragraphs.
- `factor`: add a factor vector to a new shape on the current slide, values will be be converted as character and then added as paragraphs.
- `block_list`: add a `block_list` made of `fpar` to a new shape on the current slide.
- `unordered_list`: add a `unordered_list` made of `fpar` to a new shape on the current slide.
- `data.frame`: add a data.frame to a new shape on the current slide. Use package `flextable` instead for more advanced formattings.
- `gg`: add a ggplot object to a new shape on the current slide. Use package `rvg` for more advanced graphical features.
- `external_img`: add a `external_img` to a new shape on the current slide.

  When value is a external_img object, image will be copied into the PowerPoint presentation. The width and height specified in call to `external_img` will be ignored, their values will be those of the location, unless use_loc_size is set to FALSE.
- `fpar`: add an `fpar` to a new shape on the current slide as a single paragraph in a `block_list`.
- `xml_document`: add an xml_document object to a new shape on the current slide. This function is to be used to add custom openxml code.

See Also

ph_location_type, ph_location, ph_location_label, ph_location_left, ph_location_right, ph_location_fullsize, ph_location_template

Examples

```
fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Two Content",
  master = "Office Theme")
doc <- ph_with(x = doc, value = c("Un titre", "Deux titre"),
               location = ph_location_left() )
doc <- ph_with(x = doc, value = iris[1:4, 3:5],
               location = ph_location_right() )

if( require("ggplot2") ){
  doc <- add_slide(doc)
  gg_plot <- ggplot(data = iris ) +
    geom_point(mapping = aes(Sepal.Length, Petal.Length),
      size = 3) +
    theme_minimal()
  doc <- ph_with(x = doc, value = gg_plot,
                 location = ph_location_fullsize() )
  doc <- ph_with(x = doc, value = "graphic title",
               location = ph_location_type(type="title") )
}

doc <- add_slide(doc, layout = "Title and Content",
  master = "Office Theme")
```

```
img.file <- file.path( R.home("doc"), "html", "logo.jpg" )
doc <- ph_with(x = doc, external_img(img.file, 100/72, 76/72),
               location = ph_location_right(), use_loc_size = FALSE )

svg_file <- file.path(R.home(component = "doc"), "html/Rlogo.svg")
if( require("rsvg") ){
  doc <- ph_with(x = doc, external_img(svg_file),
    location = ph_location_left(),
    use_loc_size = TRUE )
}
# block list ------
bl <- block_list(
  fpar(ftext("hello world", shortcuts$fp_bold(color = "pink"))),
  fpar(
    ftext("hello", shortcuts$fp_bold()),
    ftext("hello", shortcuts$fp_italic(color="red"))
  ))
doc <- add_slide(doc)
doc <- ph_with(x = doc, value = bl,
               location = ph_location_type(type="body") )

# fpar ------
hw <- fpar(ftext("hello world", shortcuts$fp_bold(color = "pink")))
doc <- add_slide(doc)
doc <- ph_with(x = doc, value = hw,
               location = ph_location_type(type="body") )
# unordered_list ----
ul <- unordered_list(
  level_list = c(1, 2, 2, 3, 3, 1),
  str_list = c("Level1", "Level2", "Level2", "Level3", "Level3", "Level1"),
  style = fp_text(color = "red", font.size = 0) )
doc <- add_slide(doc)
doc <- ph_with(x = doc, value = ul,
               location = ph_location_fullsize() )

print(doc, target = fileout )
```

---

ph_with_fpars_at *add multiple formated paragraphs*

Description

add several formated paragraphs in a new shape in the current slide.

Usage

```
ph_with_fpars_at(x, fpars = list(), fp_pars = list(), left, top, width,
  height, bg = "transparent", rot = 0, template_type = NULL,
  template_index = 1)
```

Arguments

x	rpptx object
fpars	list of fpar objects
fp_pars	list of fp_par objects. The list can contain NULL to keep defaults.
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'.

---

ph_with_gg *add ggplot to a pptx presentation*

Description

add a ggplot as a png image into an rpptx object This function will be deprecated in favor of ph_with in the next release.

Usage

```
ph_with_gg(x, value, type = "body", index = 1, width = NULL,
  height = NULL, location = NULL, ...)

ph_with_gg_at(x, value, width, height, left, top, ...)
```

Arguments

x	an pptx object
value	ggplot object
type	placeholder type (i.e. 'body', 'title')
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', the first one will be added with index 1 and the second one with index 2. It is recommanded to use argument location instead of type and index.
width, height	image size in inches
location	a placeholder location object. This is a convenient argument that can replace usage of arguments type and index. See ph_location_type, ph_location, ph_location_label, ph_location_left, ph_location_right, ph_location_fullsize.
...	Arguments to be passed to png function.
left, top	location of the new shape on the slide

ph_with_img *add image*

Description

add an image as a new shape in the current slide. This function will be deprecated in favor of ph_with in the next release.

Usage

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

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

Arguments

x	an pptx object
src	image filename, the basename of the file must not contain any blank.
type	placeholder type (i.e. 'body', 'title')
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', the first one will be added with index 1 and the second one with index 2. It is recommanded to use argument location instead of type and index.
width, height	image size in inches
location	a placeholder location object. This is a convenient argument that can replace usage of arguments type and index. See ph_location_type, ph_location, ph_location_label, ph_location_left, ph_location_right, ph_location_fullsize.
left, top	location of the new shape on the slide
rot	rotation angle

ph_with_table *add table*

Description

add a table as a new shape in the current slide. This function will be deprecated in favor of ph_with in the next release.

Usage

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

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	an pptx object
value	data.frame
type	placeholder type (i.e. 'body', 'title')
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', the first one will be added with index 1 and the second one with index 2. It is recommanded to use argument location instead of type and index.
header	display header if TRUE
first_row, last_row, first_column, last_column	logical for PowerPoint table options
location	a placeholder location object. This is a convenient argument that can replace usage of arguments type and index. See ph_location_type, ph_location, ph_location_label, ph_location_left, ph_location_right, ph_location_fullsize.
left, top	location of the new shape on the slide
width, height	shape size in inches

---

ph_with_text *add text into a new shape*

Description

add text into a new shape in a slide. This function will be deprecated in favor of ph_with in the next release.

Usage

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

Arguments

x	an pptx object
str	text to add
type	placeholder type (i.e. 'body', 'title')
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', the first one will be added with index 1 and the second one with index 2. It is recommanded to use argument `location` instead of `type` and `index`.
location	a placeholder location object. This is a convenient argument that can replace usage of arguments `type` and `index`. See ph_location_type, ph_location, ph_location_label, ph_location_left, ph_location_right, ph_location_fullsize.

Examples

```
# define locations for placeholders ----
loc_title <- ph_location_type(type = "title")
loc_footer <- ph_location_type(type = "ftr")
loc_dt <- ph_location_type(type = "dt")
loc_slidenum <- ph_location_type(type = "sldNum")
loc_body <- ph_location_type(type = "body")


doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(x = doc, "Un titre", location = loc_title)
doc <- ph_with(x = doc, "pied de page", location = loc_footer)
doc <- ph_with(x = doc, format(Sys.Date()), location = loc_dt)
doc <- ph_with(x = doc, "slide 1", location = loc_slidenum)
doc <- ph_with(x = doc, letters[1:10], location = loc_body)

loc_subtitle <- ph_location_type(type = "subTitle")
loc_ctrtitle <- ph_location_type(type = "ctrTitle")
doc <- add_slide(doc, layout = "Title Slide", master = "Office Theme")
doc <- ph_with(x = doc, "Un sous titre", location = loc_subtitle)
doc <- ph_with(x = doc, "Un titre", location = loc_ctrtitle)

fileout <- tempfile(fileext = ".pptx")
print(doc, target = fileout )
```

---

ph_with_ul *add unordered list to a pptx presentation*

Description

add an unordered list of text into an rpptx object. Each text is associated with a hierarchy level. This function will be deprecated in favor of `ph_with` in the next release.

Usage

```
ph_with_ul(x, type = "body", index = 1, str_list = character(0),
  level_list = integer(0), style = NULL, location = NULL)
```

Arguments

x	an pptx object
type	placeholder type (i.e. 'body', 'title')
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', the first one will be added with index 1 and the second one with index 2. It is recommanded to use argument `location` instead of `type` and `index`.
str_list	list of strings to be included in the object
level_list	list of levels for hierarchy structure
style	text style, a `fp_text` object list or a single `fp_text` objects. Use `fp_text(font.size = 0, ...)` to inherit from default sizes of the presentation.
location	a placeholder location object. This is a convenient argument that can replace usage of arguments `type` and `index`. See ph_location_type, ph_location, ph_location_label, ph_location_left, ph_location_right, ph_location_fullsize.

---

pptx_summary *get PowerPoint content in a data.frame*

Description

read content of a PowerPoint document and return a 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)
pptx_summary(example_pptx)
```

print.rpptx *write a 'PowerPoint' file.*

Description

write a 'PowerPoint' file.

Usage

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

Arguments

x	an rpptx object
target	path to the pptx file to write
...	unused

See Also

read_pptx

Examples

```
# write a rdocx object in a docx file ----
file <- tempfile(fileext = ".pptx")
doc <- read_pptx()
print(doc, target = file)
```

read_docx *open a connection 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 as base document.
x	an rdocx object
target	path to the docx file to write
...	unused

Examples

```
# create an 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 = tempfile(fileext = ".docx"))
}

# how many elements 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. The function is called `read_pptx` because it allows you to initialize an object of class `rpptx` from an existing PowerPoint file. Content will be added to the existing presentation. By default, an empty document is used.

Usage

```
read_pptx(path = NULL)
```

Arguments

path	path to the pptx file to use as base document.

master layouts and slide layouts

`read_pptx()` uses a PowerPoint file as the initial document. This is the original PowerPoint document where all slide layouts, placeholders for shapes and styles come from. Major points to be aware of are:

- Slide layouts are relative to a master layout. A document can contain one or more master layouts; a master layout can contain one or more slide layouts.

- A slide layout inherits design properties from its master layout but some properties can be overwritten.
- Designs and formatting properties of layouts and shapes (placeholders in a layout) are defined within the initial document. There is no R function to modify these values - they must be defined in the initial document.

See Also

print.rpptx add_slide

Examples

```
read_pptx()
```

---

read_xlsx *open a connexion to an 'Excel' file*

Description

read and import an xlsx file as an R object representing the document. This function is experimental.

Usage

```
read_xlsx(path = NULL)

## S3 method for class 'rxlsx'
length(x)

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

Arguments

path	path to the xlsx file to use as base document.
x	an rxlsx object
target	path to the xlsx file to write
...	unused

Examples

```
read_xlsx()
# write a rdocx object in a docx file ----
if( require(magrittr) ){
  read_xlsx() %>% print(target = tempfile(fileext = ".xlsx"))
  # 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	an rpptx object
index	slide index, default to current slide position.

Note

cursor is set on the last slide.

See Also

Other functions slide manipulation: add_slide, move_slide, on_slide

Examples

```
my_pres <- read_pptx()
my_pres <- add_slide(my_pres)
my_pres <- remove_slide(my_pres)
```

sanitize_images *remove unused media from a document*

Description

the function will scan the media directory and delete images that are not used anymore. This function is to be used when images have been replaced many times.

Usage

```
sanitize_images(x)
```

Arguments

x	rdocx or rpptx object

sections *sections*

Description

Add sections in a Word document.

Usage

```
body_end_section_continuous(x)

body_end_section_landscape(x, w = 21/2.54, h = 29.7/2.54)

body_end_section_portrait(x, w = 21/2.54, h = 29.7/2.54)

body_end_section_columns(x, widths = c(2.5, 2.5), space = 0.25,
  sep = FALSE)

body_end_section_columns_landscape(x, widths = c(2.5, 2.5),
  space = 0.25, sep = FALSE, w = 21/2.54, h = 29.7/2.54)
```

Arguments

x	an rdocx object
w, h	width and height in inches of the section page. This will be ignored if the default section (of the reference_docx file) already has a width and a height.
widths	columns widths in inches. If 3 values, 3 columns will be produced.
space	space in inches between columns.
sep	if TRUE a line is separating columns.

Details

A section starts at the end of the previous section (or the beginning of the document if no preceding section exists), and stops where the section is declared.

Examples

```
library(magrittr)

str1 <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit. " %>%
  rep(5) %>% paste(collapse = "")
str2 <- "Aenean venenatis varius elit et fermentum vivamus vehicula. " %>%
  rep(5) %>% paste(collapse = "")

my_doc <- read_docx()  %>%
  body_add_par(value = "Default section", style = "heading 1") %>%
  body_add_par(value = str1, style = "centered") %>%
```

```
  body_add_par(value = str2, style = "centered") %>%

  body_end_section_continuous() %>%
  body_add_par(value = "Landscape section", style = "heading 1") %>%
  body_add_par(value = str1, style = "centered") %>%
  body_add_par(value = str2, style = "centered") %>%
  body_end_section_landscape() %>%

  body_add_par(value = "Columns", style = "heading 1") %>%
  body_end_section_continuous() %>%
  body_add_par(value = str1, style = "centered") %>%
  body_add_par(value = str2, style = "centered") %>%
  slip_in_column_break() %>%
  body_add_par(value = str1, style = "centered") %>%
  body_end_section_columns(widths = c(2,2), sep = TRUE, space = 1) %>%

  body_add_par(value = str1, style = "Normal") %>%
  body_add_par(value = str2, style = "Normal") %>%
  slip_in_column_break() %>%
  body_end_section_columns_landscape(widths = c(3,3), sep = TRUE, space = 1)

print(my_doc, target = tempfile(fileext = ".docx"))
```

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

Note

The "last modified" and "last modified by" fields will be automatically be updated when the file is 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()
```

---

sheet_select *select sheet*

Description

set a particular sheet selected when workbook will be edited.

Usage

```
sheet_select(x, sheet)
```

Arguments

x	rxlsx object
sheet	sheet name

Examples

```
my_ws <- read_xlsx()
my_pres <- add_sheet(my_ws, label = "new sheet")
my_pres <- sheet_select(my_ws, sheet = "new sheet")
print(my_ws, target = tempfile(fileext = ".xlsx") )
```

---

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_size *slides width and height*

Description

get the width and height of slides in inches as a named vector.

Usage

```
slide_size(x)
```

Arguments

x	an rpptx object

See Also

Other functions for reading presentation informations: annotate_base, color_scheme, layout_properties, layout_summary, length.rpptx, slide_summary

Examples

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

slide_summary *get PowerPoint slide content in a data.frame*

Description

get content and positions of current slide into a data.frame. Data for any tables, images, or paragraphs are imported into the resulting data.frame.

Usage

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

Arguments

x	an rpptx object
index	slide index

Note

The column id of the result is not to be used by users. This is a technical string id whose value will be used by office when the document will be rendered. This is not related to argument index required by functions ph_with.

See Also

Other functions for reading presentation informations: annotate_base, color_scheme, layout_properties, layout_summary, length.rpptx, slide_size

Examples

```
my_pres <- read_pptx()
my_pres <- add_slide(my_pres)
my_pres <- ph_with(my_pres, format(Sys.Date()),
  location = ph_location_type(type="dt"))
my_pres <- add_slide(my_pres)
my_pres <- ph_with(my_pres, iris[1:2,],
  location = ph_location_type(type="body"))
slide_summary(my_pres)
slide_summary(my_pres, index = 1)
```

---

slip_in_footnote *append a footnote*

Description

append a new footnote into a paragraph of an rdocx object

Usage

```
slip_in_footnote(x, style = NULL, blocks, pos = "after")
```

Arguments

x	an rdocx object
style	text style to be used for the reference note
blocks	set of blocks to be used as footnote content returned by function block_list.
pos	where to add the new element relative to the cursor, "after" or "before".

Examples

```
library(magrittr)

img.file <- file.path( R.home("doc"), "html", "logo.jpg" )
bl <- block_list(
  fpar(ftext("hello", shortcuts$fp_bold())),
  fpar(
```

```
    ftext("hello world", shortcuts$fp_bold()),
    external_img(src = img.file, height = 1.06, width = 1.39)
  )
)

x <- read_docx() %>%
  body_add_par("Hello ", style = "Normal") %>%
  slip_in_text("world", style = "strong") %>%
  slip_in_footnote(style = "reference_id", blocks = bl)

print(x, target = tempfile(fileext = ".docx"))
```

`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, the basename of the file must not contain any blank.
`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( 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 = tempfile(fileext = ".docx"))
```

slip_in_seqfield *append seq field*

Description

append seq field into a paragraph of an rdocx object. This feature is only available when document are edited with Word, when edited with Libre Office or another program, seq field will not be calculated and not displayed.

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 = tempfile(fileext = ".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", hyperlink = NULL)
```

Arguments

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

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") %>%
  slip_in_text("with a link", style = "strong",
    pos = "after", hyperlink = "https://davidgohel.github.io/officer/")

print(x, target = tempfile(fileext = ".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	an rdocx object

Examples

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

unordered_list *unordered list*

Description

unordered list of text for PowerPoint presentations. Each text is associated with a hierarchy level.

Usage

```
unordered_list(str_list = character(0), level_list = integer(0),
  style = NULL)
```

Arguments

str_list	list of strings to be included in the object
level_list	list of levels for hierarchy structure
style	text style, a fp_text object list or a single fp_text objects. Use fp_text(font.size = 0, ...) to inherit from default sizes of the presentation.

See Also

ph_with

Examples

```
unordered_list(
level_list = c(1, 2, 2, 3, 3, 1),
str_list = c("Level1", "Level2", "Level2", "Level3", "Level3", "Level1"),
style = fp_text(color = "red", font.size = 0) )
```

`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

`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 allow the creation of valid wml code containing references to images.

Index

add_sheet, 4
add_slide, 4, *39, 40*, *67, 68*
annotate_base, 5, *21*, *35–37*, *72, 73*
as.data.frame.external_img *(*external_img*)*, 27
as.data.frame.fpar *(*fpar*)*, 29

block_list, *6*, 6, *42–44*, *58*, *73*
body_add_blocks, 6
body_add_break, 7
body_add_docx, 8
body_add_fpar, 9
body_add_gg, 10
body_add_img, 10
body_add_par, 11
body_add_table, 12
body_add_toc, 13
body_add_xml, 13
body_bookmark, 14
body_default_section *(*body_end_section*)*, 14
body_end_section, 14
body_end_section_columns *(*sections*)*, 69
body_end_section_columns_landscape *(*sections*)*, 69
body_end_section_continuous *(*sections*)*, 69
body_end_section_landscape *(*sections*)*, 69
body_end_section_portrait *(*sections*)*, 69
body_remove, 16
body_replace_all_text, 17, *26*
body_replace_img_at_bkm *(*body_replace_text_at_bkm*)*, 18
body_replace_text_at_bkm, 18
break_column_before, 19

change_styles, 20
color_scheme, *5*, 21, *35–37*, *72, 73*
cursor_backward *(*cursor_begin*)*, 21
cursor_begin, 21
cursor_bookmark *(*cursor_begin*)*, 21
cursor_end *(*cursor_begin*)*, 21
cursor_forward *(*cursor_begin*)*, 21
cursor_reach *(*cursor_begin*)*, 21

dim.external_img *(*external_img*)*, 27
dim.fp_par *(*fp_par*)*, 32
doc_properties, 27
docx_body_relationship, 23
docx_body_xml, 24
docx_bookmarks, 24
docx_dim, 25
docx_reference_img, 25
docx_show_chunk, *17, 18*, 26
docx_summary, 26

external_img, 27, *58*

footers_replace_all_text *(*body_replace_all_text*)*, 17
footers_replace_img_at_bkm *(*body_replace_text_at_bkm*)*, 18
footers_replace_text_at_bkm *(*body_replace_text_at_bkm*)*, 18
format.external_img *(*external_img*)*, 27
format.fp_cell *(*fp_cell*)*, 31
format.fp_text *(*fp_text*)*, 33
format.fpar *(*fpar*)*, 29
format.ftext *(*ftext*)*, 34
fortify_location, 28
fp_border, 30, *31, 32*
fp_cell, 31
fp_par, 32, *60*
fp_sign, 33
fp_text, 33, *44*
fpar, *9*, 29, *41, 42*, *58*, *60*
ftext, 34

grep, *18*

grepl, *17*
gsub, *17*

headers_replace_all_text
(body_replace_all_text), 17
headers_replace_img_at_bkm
(body_replace_text_at_bkm), 18
headers_replace_text_at_bkm
(body_replace_text_at_bkm), 18

layout_properties, *5*, *21*, 35, *36*, *37*, *72*, *73*
layout_summary, *5*, *21*, *35*, 36, *37*, *72*, *73*
length.rdocx (read_docx), 65
length.rpptx, *5*, *21*, *35*, *36*, 36, *72*, *73*
length.rxlsx (read_xlsx), 67
location_eval, 37

media_extract, 38
move_slide, *5*, 38, *40*, *68*

officer, 39
officer-package (officer), 39
on_slide, *5*, *39*, 40, *68*

pack_folder, 41
ph_add_fpar, 41
ph_add_par, 42
ph_add_text, 43
ph_empty, 45
ph_empty_at (ph_empty), 45
ph_from_xml, 46
ph_from_xml_at (ph_from_xml), 46
ph_hyperlink, 46, *54*, *56*
ph_location, *28*, *37*, *45*, 47, *49–52*, *54*, *58*, *60–64*
ph_location_fullsize, *45*, *48*, 48, *50–52*, *54*, *58*, *60–64*
ph_location_label, *45*, *48*, *49*, 49, *50–52*, *54*, *58*, *60–64*
ph_location_left, *45*, *48–50*, 50, *51*, *52*, *54*, *58*, *60–64*
ph_location_right, *45*, *48–50*, 51, *52*, *54*, *58*, *60–64*
ph_location_template, *48–51*, 51, *54*, *58*
ph_location_type, *45*, *48–52*, 53, *58*, *60–64*
ph_remove, *47*, 54, *56*
ph_slidelink, *47*, *54*, 55
ph_with, *28*, *37*, *42–44*, *47*, *54*, *56*, 56, *60–63*, *77*
ph_with_fpars_at, 59
ph_with_gg, 60
ph_with_gg_at (ph_with_gg), 60
ph_with_img, 61
ph_with_img_at (ph_with_img), 61
ph_with_table, 61
ph_with_table_at (ph_with_table), 61
ph_with_text, 62
ph_with_ul, 63
pptx_summary, 64
print.fp_cell (fp_cell), 31
print.fp_par (fp_par), 32
print.fp_text (fp_text), 33
print.ftext (ftext), 34
print.rdocx (read_docx), 65
print.rpptx, *5*, 65, *67*
print.rxlsx (read_xlsx), 67

read_docx, 65
read_pptx, *5*, *65*, 66
read_xlsx, 67
regex, *17*, *18*
remove_slide, *5*, *39*, *40*, 68

sanitize_images, 68
sections, 69
set_doc_properties, 70
sheet_select, 71
shortcuts, 71
slide_size, *5*, *21*, *35–37*, 72, *73*
slide_summary, *5*, *21*, *35–37*, *41–44*, *47*, *54–56*, *72*, 72
slip_in_column_break
(break_column_before), 19
slip_in_footnote, 73
slip_in_img, 74
slip_in_seqfield, 75
slip_in_text, 76
slip_in_xml, 76
styles_info, 77

unordered_list, *58*, 77
unpack_folder, 78
update.fp_border (fp_border), 30
update.fp_cell (fp_cell), 31
update.fp_par (fp_par), 32
update.fp_text (fp_text), 33
update.fpar (fpar), 29

wml_link_images, *25*, 78