Package 'officer'

June 12, 2018

Type Package

Title Manipulation of Microsoft Word and PowerPoint Documents

Version 0.3.1

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, digest,uuid,utils,stats, magrittr,htmltools, xml2 (>= 1.1.0)

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

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

RoxygenNote 6.0.1.9000

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

VignetteBuilder knitr

NeedsCompilation yes

Author David Gohel [aut, cre],
Frank Hangler [ctb] (function body_replace_all_text),
Liz Sander [ctb] (several documentation fixes),
Jon Calder [ctb] (update vignettes),
John Harrold [ctb] (fuction annotate_base)

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

Repository CRAN

Date/Publication 2018-06-12 13:33:19 UTC

R topics documented:

add_sheet . . . 3
add_slide . . . 4
annotate_base . . . 4
block_list . . . 5
body_add_blocks . . . 6
body_add_break . . . 6
body_add_docx . . . 7
body_add_fpar . . . 8
body_add_gg . . . 9
body_add_img . . . 9
body_add_par . . . 10
body_add_table . . . 11
body_add_toc . . . 12
body_add_xml . . . 12
body_bookmark . . . 13
body_end_section . . . 13
body_remove . . . 15
body_replace_all_text . . . 15
body_replace_text_at_bkm . . . 17
break_column_before . . . 18
color_scheme . . . 19
cursor_begin . . . 19
docx_bookmarks . . . 21
docx_dim . . . 22
docx_reference_img . . . 23
docx_show_chunk . . . 23
docx_summary . . . 24
doc_properties . . . 24
external_img . . . 25
fpar . . . 25
fp_border . . . 27
fp_cell . . . 27
fp_par . . . 29
fp_sign . . . 30
fp_text . . . 30
ftext . . . 31
layout_properties . . . 32
layout_summary . . . 33
media_extract . . . 33
officer . . . 34
on_slide . . . 34
pack_folder . . . 35
ph_add_fpar . . . 35
ph_add_par . . . 36
ph_add_text . . . 37
ph_empty . . . 38

ph_from_xml . . . 39
ph_hyperlink . . . 40
ph_remove . . . 41
ph_slidelink . . . 41
ph_with_fpars_at . . . 42
ph_with_gg . . . 43
ph_with_img . . . 44
ph_with_table . . . 45
ph_with_text . . . 47
ph_with_ul . . . 47
pptx_summary . . . 48
read_docx . . . 49
read_pptx . . . 50
read_xlsx . . . 51
remove_slide . . . 51
sections . . . 52
set_doc_properties . . . 53
sheet_select . . . 54
shortcuts . . . 55
slide_summary . . . 55
slip_in_footnote . . . 56
slip_in_img . . . 57
slip_in_seqfield . . . 57
slip_in_text . . . 58
slip_in_xml . . . 59
stext . . . 59
styles_info . . . 60
unpack_folder . . . 60
wml_link_images . . . 61

Index **62**

`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, master)
```

Arguments

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

Examples

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

`annotate_base` *annotate PowerPoint base document*

Description

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

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

x rpptx object of the annotated PowerPoint file

Examples

```
# To generate an anotation of the default base documet with officer and place
# the output in the file annotated_layout.pptx:
  annotate_base()

# 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 *list of 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 for example.

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

x <- read_docx() %>%
  body_add_blocks( blocks = bl ) %>%
  print(target = "body_add_bl.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 = "body_add_break.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".

Examples

```
img.file <- file.path( R.home("doc"), "html", "logo.jpg" )
if( file.exists(img.file) ){
  # create a example file to be inserted later in the final doc----
  doc <- read_docx()
  doc <- body_add_img(x = doc, src = img.file, height = 1.06, width = 1.39 )
  print(doc, target = "external_file.docx")

  # insert external_file.docx in the final doc----
  final_doc <- read_docx()
  doc <- body_add_docx(x = doc, src = "external_file.docx" )
  print(doc, target = "final.docx")
}
```

body_add_fpar *add fpar*

Description

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

Usage

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

Arguments

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

See Also

fpar

Examples

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

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

# 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 = "img_in_par.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 = "body_add_gg.docx" )
}
```

body_add_img *add image*

Description

add an image into an rdocx object.

Usage

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

Arguments

x	an rdocx object
src	image filename, 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 = "body_add_img.docx" )
```

---

body_add_par *add paragraph of text*

Description

add a paragraph of text into an rdocx object

Usage

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

Arguments

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

Examples

```
library(magrittr)

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

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

---

body_add_table *add table*

Description

add a table into an rdocx object

Usage

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

Arguments

x	a docx device
value	a data.frame 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 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 = "body_add_table.docx" )
```

body_add_toc *add table of content*

Description

add a table of content into an rdocx object

Usage

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

Arguments

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

Examples

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

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

body_add_xml *add an xml string as document element*

Description

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

Usage

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

Arguments

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

body_bookmark *add bookmark*

Description

Add a bookmark at the cursor location. 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 = "section.docx")
```

body_remove *remove an element*

Description

remove element pointed by cursor from a Word document

Usage

```
body_remove(x)
```

Arguments

x	an rdocx object

Examples

```
library(officer)
library(magrittr)

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

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

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

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

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

body_replace_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_at(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 = "test_replacement_img.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".

---

color_scheme *color scheme*

Description

get master layout color scheme into a data.frame.

Usage

```
color_scheme(x)
```

Arguments

x	rpptx object

Examples

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

---

cursor_begin *set cursor in an rdocx object*

Description

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

Usage

```
cursor_begin(x)

cursor_bookmark(x, id)

cursor_end(x)

cursor_reach(x, keyword)

cursor_forward(x)

cursor_backward(x)
```

Arguments

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

cursor_begin

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

cursor_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 = "cursor.docx")


# cursor_bookmark ----
library(magrittr)

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

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

---

docx_bookmarks *List Word bookmarks*

Description

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

Usage

```
docx_bookmarks(x)
```

Arguments

x	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 tidy format*

Description

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

Usage

```
docx_summary(x)
```

Arguments

x	an rdocx object

Examples

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

doc_properties *read document properties*

Description

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

Usage

```
doc_properties(x)
```

Arguments

x	an rdocx or rpptx object

Examples

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

external_img *external image*

Description

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

Usage

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

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

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

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

Arguments

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

Examples

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

fpar *concatenate formatted text*

Description

Create a paragraph representation by concatenating formatted text or images.

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

Examples

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

layout_summary *presentation layouts summary*

Description

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

Usage

```
layout_summary(x)
```

Arguments

x	rpptx object

Examples

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

media_extract *Extract media from a document object*

Description

Extract files from an rdocx or rpptx object.

Usage

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

Arguments

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

Examples

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

officer *officer: Manipulate Microsoft Word and PowerPoint Documents*

Description

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

Details

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]
- Jon Calder <jonmcalder@gmail.com> (update vignettes) [contributor]
- John Harrold <john.m.harrold@gmail.com> (fuction annotate_base) [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	rpptx object
index	slide index

Examples

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

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

---

pack_folder *compress a folder*

Description

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

Usage

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

Arguments

folder	folder to compress
target	path of the archive to create

---

ph_add_fpar *append fpar*

Description

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

Usage

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

Arguments

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

See Also

fpar

Examples

```
library(magrittr)

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

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

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

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

---

ph_add_par *append paragraph*

Description

append a new empty paragraph in a placeholder

Usage

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

Arguments

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

Examples

```
library(magrittr)

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

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

print(doc, target = fileout)
```

---

ph_add_text *append text*

Description

append text in a placeholder

Usage

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

Arguments

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

style	text style, a fp_text object
pos	where to add the new element relative to the cursor, "after" or "before".
href	hyperlink 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.

Examples

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

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

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

print(my_pres, target = fileout)


fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx() %>%
  add_slide(layout = "Title and Content", master = "Office Theme") %>%
  ph_with_text(type = "title", str = "Un titre 1") %>%
  add_slide(layout = "Title and Content", master = "Office Theme") %>%
  ph_with_text(type = "title", str = "Un titre 2") %>%
  on_slide(1) %>%
  ph_empty(type = "body") %>%
  ph_add_par(type = "body", level = 2) %>%
  ph_add_text(str = "Jump here to slide 2!", type = "body",
  slide_index = 2)

print(doc, target = fileout)
```

---

ph_empty *add a new empty shape*

Description

add a new empty shape in the current slide.

Usage

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

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

Arguments

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

Examples

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

print(doc, target = fileout )

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

print(doc, target = fileout )
```

---

ph_from_xml *add an xml string as new shape*

Description

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

Usage

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

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

Arguments

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

---

ph_hyperlink *hyperlink a placeholder*

Description

add hyperlink to a placeholder in the current slide.

Usage

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

Arguments

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

Examples

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

```
doc <- ph_hyperlink(x = doc, id_chr = "2",
  href = "https://cran.r-project.org")

print(doc, target = fileout )
```

---

ph_remove *remove shape*

Description

remove a shape in a slide

Usage

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

Arguments

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

Examples

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

print(doc, target = fileout )
```

---

ph_slidelink *slide link to a placeholder*

Description

add slide link to a placeholder in the current slide.

Usage

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

Arguments

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

Examples

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

print(doc, target = fileout )
```

---

ph_with_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'.

Examples

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

bold_face <- shortcuts$fp_bold(font.size = 0)
bold_redface <- update(bold_face, color = "red")

fpar_1 <- fpar(
  ftext("Hello ", prop = bold_face), ftext("World", prop = bold_redface ),
  ftext(", \r\nhow are you?", prop = bold_face ) )

fpar_2 <- fpar(
  ftext("Hello ", prop = bold_face), ftext("World", prop = bold_redface ),
  ftext(", \r\nhow are you again?", prop = bold_face ) )

doc <- ph_with_fpars_at(x = doc, fpars = list(fpar_1, fpar_2),
  fp_pars = list(NULL, fp_par(text.align = "center")),
  left = 1, top = 2, width = 7, height = 4)
doc <- ph_with_fpars_at(x = doc, fpars = list(fpar_1, fpar_2),
  template_type = "body", template_index = 1,
  left = 4, top = 5, width = 4, height = 3)

print(doc, target = fileout )
```

---

ph_with_gg *add ggplot to a pptx presentation*

Description

add a ggplot as a png image into an rpptx object

Usage

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

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

Arguments

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

Examples

```
if( require("ggplot2") ){
  doc <- read_pptx()
  doc <- add_slide(doc, layout = "Title and Content",
    master = "Office Theme")

  gg_plot <- ggplot(data = iris ) +
    geom_point(mapping = aes(Sepal.Length, Petal.Length), size = 3) +
    theme_minimal()

  if( capabilities(what = "png") ){
    doc <- ph_with_gg(doc, value = gg_plot )
    doc <- ph_with_gg_at(doc, value = gg_plot,
      height = 4, width = 8, left = 4, top = 4 )
  }

  print(doc, target = "ph_with_gg.pptx" )
}
```

---

ph_with_img *add image*

Description

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

Usage

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

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

Arguments

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

Examples

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

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

print(doc, target = fileout )

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

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

print(doc, target = fileout )
```

---

ph_with_table *add table*

Description

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

Usage

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

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

Arguments

x	a pptx device
value	data.frame
type	placeholder type
index	placeholder index (integer). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body'.
header	display header if TRUE
first_row, last_row, first_column, last_column	logical for PowerPoint table options
left, top	location of the new shape on the slide
width, height	shape size in inches

Examples

```
library(magrittr)

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

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

library(magrittr)

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

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

ph_with_text *add text into a new shape*

Description

add text into a new shape in a slide.

Usage

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

Arguments

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

Examples

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

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

print(doc, target = fileout )
```

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.

Usage

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

Arguments

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

Examples

```
library(magrittr)
pptx <- read_pptx()
pptx <- add_slide(x = pptx, layout = "Title and Content", master = "Office Theme")
pptx <- ph_with_text(x = pptx, type = "title", str = "Example title")
pptx <- ph_with_ul(
  x = pptx, type = "body", index = 1,
  str_list = c("Level1", "Level2", "Level2", "Level3", "Level3", "Level1"),
  level_list = c(1, 2, 2, 3, 3, 1),
  style = fp_text(color = "red", font.size = 0) )
print(pptx, target = "example2.pptx") %>%
  invisible()
```

---

pptx_summary *get PowerPoint content in a tidy format*

Description

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

Usage

```
pptx_summary(x)
```

Arguments

x	an rpptx object

Examples

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

read_docx *open a 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 = "out.docx")
  # full path of produced file is returned
  print(.Last.value)
}

# 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.

Usage

```
read_pptx(path = NULL)

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

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

Arguments

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

number of slides

Function length will return the number of slides.

Examples

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

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 rpptx 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 = "out.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	rpptx object
index	slide index, default to current slide position.

Note

cursor is set on the last slide.

Examples

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

my_pres <- remove_slide(my_pres)
```

---

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 = "section.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_summary *get PowerPoint slide content in a tidy format*

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	rpptx object
index	slide index

Examples

```
library(magrittr)

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

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

`slip_in_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$fp_bold())),
  fpar(
    ftext("hello", shortcuts$fp_bold()),
    stext(" world", "strong"),
    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 = "footnote.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 = "append_img.docx")
```

slip_in_seqfield *append seq field*

Description

append seq field into a paragraph of an rdocx object

Usage

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

Arguments

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

Examples

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

  body_add_par(" - This is a figure title", style = "centered") %>%
  slip_in_seqfield(str = "SEQ Figure \u005C* roman",
    style = 'Default Paragraph Font', pos = "before") %>%
  slip_in_text("Figure: ", style = "strong", pos = "before") %>%

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

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

---

slip_in_text *append text*

Description

append text into a paragraph of an rdocx object

Usage

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

Arguments

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

Examples

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

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

---

slip_in_xml *add a wml string into a Word document*

Description

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

Usage

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

Arguments

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

---

stext *Word styled text*

Description

Format a chunk of text with a referenced style.

Usage

```
stext(text, style_id = NULL, style, doc)
```

Arguments

text	text value
style_id	style id
style	style name. Use only when you don't know what is the id associated with a stylename.
doc	rdocx object where the text will be inserted. This is only used to get the style id from the style name (use only when you don't know what is the id associated with a stylename).

Examples

```
doc <- read_docx()
stext("hello", "strong", doc)
```

`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()
```

`unpack_folder` *Extract files from a zip file*

Description

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

Usage

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

Arguments

`file`	path of the archive to unzip
`folder`	folder to create

Details

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

`wml_link_images` *transform an xml string with images references*

Description

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

Usage

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

Arguments

`x`	an rdocx object
`str`	wml string

Details

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

Index

add_sheet, 3
add_slide, 4
annotate_base, 4
as.data.frame.external_img (external_img), 25
as.data.frame.fpar (fpar), 25

block_list, 5, *6*, *56*
body_add_blocks, 6
body_add_break, 6
body_add_docx, 7
body_add_fpar, 8
body_add_gg, 9
body_add_img, 9
body_add_par, 10
body_add_table, 11
body_add_toc, 12
body_add_xml, 12
body_bookmark, 13
body_default_section (body_end_section), 13
body_end_section, 13
body_end_section_columns (sections), 52
body_end_section_columns_landscape (sections), 52
body_end_section_continuous (sections), 52
body_end_section_landscape (sections), 52
body_end_section_portrait (sections), 52
body_remove, 15
body_replace_all_text, 15, *23*
body_replace_at (body_replace_text_at_bkm), 17
body_replace_img_at_bkm (body_replace_text_at_bkm), 17
body_replace_text_at_bkm, 17
break_column_before, 18

color_scheme, 19
cursor_backward (cursor_begin), 19
cursor_begin, 19
cursor_bookmark (cursor_begin), 19
cursor_end (cursor_begin), 19
cursor_forward (cursor_begin), 19
cursor_reach (cursor_begin), 19

dim.external_img (external_img), 25
dim.fp_par (fp_par), 29
doc_properties, 24
docx_bookmarks, 21
docx_dim, 22
docx_reference_img, 23
docx_show_chunk, *16, 17*, 23
docx_summary, 24

external_img, 25

footers_replace_all_text (body_replace_all_text), 15
footers_replace_img_at_bkm (body_replace_text_at_bkm), 17
footers_replace_text_at_bkm (body_replace_text_at_bkm), 17
format.external_img (external_img), 25
format.fp_cell (fp_cell), 27
format.fp_text (fp_text), 30
format.fpar (fpar), 25
format.ftext (ftext), 31
fp_border, 27, *28, 29*
fp_cell, 27
fp_par, 29, *42*
fp_sign, 30
fp_text, 30, *38*
fpar, *8*, 25, *36*, *42*
ftext, 31

grep, *17*
grepl, *16*
gsub, *16*

headers_replace_all_text
(body_replace_all_text), 15
headers_replace_img_at_bkm
(body_replace_text_at_bkm), 17
headers_replace_text_at_bkm
(body_replace_text_at_bkm), 17

layout_properties, 32
layout_summary, 33
length.rdocx (read_docx), 49
length.rpptx (read_pptx), 50
length.rxlsx (read_xlsx), 51

media_extract, 33

officer, 34
officer-package (officer), 34
on_slide, 34

pack_folder, 35
ph_add_fpar, 35
ph_add_par, 36
ph_add_text, 37
ph_empty, 38
ph_empty_at (ph_empty), 38
ph_from_xml, 39
ph_from_xml_at (ph_from_xml), 39
ph_hyperlink, 40
ph_remove, 41
ph_slidelink, 41
ph_with_fpars_at, 42
ph_with_gg, 43
ph_with_gg_at (ph_with_gg), 43
ph_with_img, 44
ph_with_img_at (ph_with_img), 44
ph_with_table, 45
ph_with_table_at (ph_with_table), 45
ph_with_text, 47
ph_with_ul, 47
pptx_summary, 48
print.fp_cell (fp_cell), 27
print.fp_par (fp_par), 29
print.fp_text (fp_text), 30
print.ftext (ftext), 31
print.rdocx (read_docx), 49
print.rpptx (read_pptx), 50
print.rxlsx (read_xlsx), 51

read_docx, 49
read_pptx, 50
read_xlsx, 51
regex, *16, 17*
remove_slide, 51

sections, 52
set_doc_properties, 53
sheet_select, 54
shortcuts, 55
slide_summary, *36, 37*, *40–42*, 55
slip_in_column_break
(break_column_before), 18
slip_in_footnote, 56
slip_in_img, 57
slip_in_seqfield, 57
slip_in_text, 58
slip_in_xml, 59
stext, 59
styles_info, 60

unpack_folder, 60
unzip, *60*
update.fp_border (fp_border), 27
update.fp_cell (fp_cell), 27
update.fp_par (fp_par), 29
update.fp_text (fp_text), 30
update.fpar (fpar), 25

wml_link_images, *23*, 61