Package ‘officer’

May 14, 2020

Type Package

Title Manipulation of Microsoft Word and PowerPoint Documents

Version 0.3.10

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

Imports R6, grDevices, zip (>= 2.0.3), uuid, stats, magrittr, utils, xml2 (>= 1.1.0), graphics

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

Encoding UTF-8

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

RoxygenNote 7.1.0

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

VignetteBuilder knitr

NeedsCompilation no

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 2020-05-14 11:30:03 UTC
topics documented:

- add_sheet ........................................ 4
- add_slide ........................................ 5
- annotate_base .................................... 5
- block_caption .................................... 6
- block_list ........................................ 7
- block_pour_docx .................................. 8
- block_section ..................................... 9
- block_table ....................................... 10
- block_toc ......................................... 11
- body_add .......................................... 11
- body_add_blocks .................................. 14
- body_add_break ................................... 15
- body_add_docx .................................... 15
- body_add_fpar .................................... 16
- body_add_gg ....................................... 17
- body_add_img ...................................... 18
- body_add_par ...................................... 19
- body_add_table ................................... 20
- body_add_toc ..................................... 21
- body_add_xml ...................................... 22
- body_bookmark .................................... 22
- body_remove ...................................... 23
- body_replace_all_text ......................... 23
- body_replace_text_at_bkm ...................... 26
- change_styles ..................................... 27
- color_scheme ...................................... 27
- cursor_begin ..................................... 28
- docx_body_relationship ......................... 30
- docx_body_xml .................................... 30
- docx_bookmarks ................................... 31
- docx_dim .......................................... 32
- docx_reference_img ................................ 32
- docx_show_chunk .................................. 33
- docx_summary ..................................... 33
- docx_properties ................................... 34
- empty_content .................................... 35
- external_img ..................................... 35
- fortify_location .................................. 36
- fpar ................................................ 37
- fp_border ......................................... 38
- fp_cell ............................................ 39
- fp_par ............................................. 40
- fp_text ............................................ 42
- ftext ............................................... 44
- get_reference_value ............................... 45
- layout_properties .................................. 45
layout_summary .................................................. 46
length.docx .................................................. 47
length.rptx .................................................. 47
media_extract ................................................ 48
move_slide .................................................. 49
officer ....................................................... 49
officer-defunct ............................................... 50
on_slide ....................................................... 51
opts_current_table .......................................... 52
pack_folder .................................................. 53
page_mar ....................................................... 53
page_size ...................................................... 54
ph_add_fpar .................................................. 55
ph_add_par .................................................... 56
ph_add_text ................................................... 57
ph_empty ...................................................... 59
ph_hyperlink .................................................. 60
ph_location ................................................... 61
ph_location_fullsize ........................................ 62
ph_location_label ............................................ 63
ph_location_left ............................................. 64
ph_location_right .......................................... 65
ph_location_template ....................................... 65
ph_location_type ............................................ 67
ph_remove ..................................................... 69
ph_slidelink .................................................. 70
ph_with ....................................................... 71
ph_with_gg_at ............................................... 75
ph_with_img_at .............................................. 76
ph_with_text .................................................. 76
plot_instr ..................................................... 77
plot_layout_properties ..................................... 78
pptx_summary ................................................ 79
print.rptx .................................................... 79
prop_section .................................................. 80
prop_table .................................................... 81
read_docx .................................................... 82
read_pptx .................................................... 83
read_xlsx ..................................................... 84
remove_slide .................................................. 84
run_autonum .................................................. 85
run_columnbreak ............................................. 86
run_linebreak ................................................ 86
run_pagebreak ............................................... 87
run_reference ................................................ 87
run_seqfield .................................................. 88
sanitize_images .............................................. 88
sections ....................................................... 89
add_sheet

Description

add a sheet into an xlsx worksheet

Usage

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

Arguments

- `x`: xlsx object
- `label`: sheet label

Examples

```r
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**

```r
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`, `ph_with`, `layout_summary`

Other functions slide manipulation: `move_slide()`, `on_slide()`, `remove_slide()`

**Examples**

```r
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**

```r
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(), plot_layout_properties(), slide_size(), slide_summary()

Examples

# To generate an annotation 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_caption  Caption block

Description

Create a representation of a caption that can be used for cross reference.

Usage

block_caption(label, style, autonum = NULL)

Arguments

label       a scalar character representing label to display
style       paragraph style name
autonum     an object generated with function run_autonum

See Also

Other block functions for reporting: block_list(), block_pour_docx(), block_section(), block_table(),
block_toc(), fpar(), plot_instr(), unordered_list()
Examples

```r
library(magrittr)
library(officer)

run_num <- run_autonum(seq_id = "tab", pre_label = "tab. ", bkm = "iris_table")
caption <- block_caption("iris table",
    style = "Normal",
    autonum = run_num )

doc <- read_docx() %>%
    body_add("A title", style = "heading 1") %>%
    body_add("Hello world!", style = "Normal") %>%
    body_add(caption) %>%
    body_add(iris, style = "table_template")

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

---

**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

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

Arguments

```r...
...```
a list of objects of class `fpar` or `flextable`. When output is only for Word, objects of class `external_img` can also be used in fpar construction to mix text and images in a single paragraph.

See Also

```r
ph_with(), body_add()
```

Other block functions for reporting: `block_caption()`, `block_pour_docx()`, `block_section()`, `block_table()`, `block_toc()`, `fpar()`, `plot_instr()`, `unordered_list()`

Examples

```r
# block list ------

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

Pour external Word document in the current document

**Description**

Pour the content of a docx file in the resulting docx generated by the main R Markdown document.

**Usage**

block_pour_docx(file)
**block_section**

Arguments

file  
external docx file path

See Also

Other block functions for reporting: block_caption(), block_list(), block_section(), block_table(), block_toc(), fpar(), plot_instr(), unordered_list()

Examples

```r
library(officer)

docx <- tempfile(fileext = ".docx")
doc <- read_docx()
doc <- body_add(doc, iris[1:20,], style = "table_template")
print(doc, target = docx)

target <- tempfile(fileext = ".docx")
doc_1 <- read_docx()
doc_1 <- body_add(doc_1, block_pour_docx(docx))
print(doc_1, target = target)
```

**Description**

Create a representation of a section.

A section affects preceding paragraphs or tables; i.e. 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.

When a new landscape section is needed, it is recommended to add a block_section with type = "continuous", to add the content to be appended in the new section and finally to add a block_section with page_size = page_size(orient = "landscape").

Usage

```r
block_section(property)
```

Arguments

property  
section properties defined with function prop_section

See Also

Other block functions for reporting: block_caption(), block_list(), block_pour_docx(), block_table(), block_toc(), fpar(), plot_instr(), unordered_list()
Examples

```r
ps <- prop_section(
  page_size = page_size(orient = "landscape"),
  page_margins = page_mar(top = 2),
  type = "continuous"
)
block_section(ps)
```

```
| block_table  | Table block |
```

Description

Create a representation of a table

Usage

```r
block_table(x, header = TRUE, properties = prop_table(), alignment = NULL)
```

Arguments

- `x` a data.frame to add as a table
- `header` display header if TRUE
- `properties` table properties, see `prop_table()`. Table properties are not handled identically between Word and PowerPoint output format. They are fully supported with Word but for PowerPoint (which does not handle as many things as Word for tables), only conditional formatting properties are supported.
- `alignment` alignment for each columns, 'l' for left, 'r' for right and 'c' for center. Default to NULL.

See Also

`prop_table()`

Other block functions for reporting: `block_caption()`, `block_list()`, `block_pour_docx()`, `block_section()`, `block_toc()`, `fpar()`, `plot_instr()`, `unordered_list()`

Examples

```r
block_table(x = head(iris))
```

```r
block_table(x = mtcars, header = TRUE,
  properties = prop_table(
    tcf = table_conditional_formatting(
      first_row = TRUE, first_column = TRUE)
  ))
```
### block_toc

**Table of content**

#### Description

Create a representation of a table of content.

#### Usage

```r
block_toc(level = 3, style = NULL, seq_id = NULL, separator = ";")
```

#### Arguments

- **level**: max title level of the table
- **style**: optional. If not NULL, its value is used as style in the document that will be used to build entries of the TOC.
- **seq_id**: optional. If not NULL, its value is used as sequence identifier in the document that will be used to build entries of the TOC. See also `run_autonum()` to specify a sequence identifier.
- **separator**: optional. Some configurations need "," (i.e. from Canada) separator instead of ";"

#### See Also

Other block functions for reporting: `block_caption()`, `block_list()`, `block_pour_docx()`, `block_section()`, `block_table()`, `fpar()`, `plot_instr()`, `unordered_list()`

#### Examples

```r
block_toc(level = 2)
block_toc(style = "Table Caption")
```

---

### body_add

**Add content into a Word document**

#### Description

This function add objects into a Word document. Values are added as new paragraphs or tables.

This function is experimental and will replace the `body_add_*` functions later. For now it is only to be used for successive additions and cannot be used in conjunction with the `body_add_*` functions.
Usage

body_add(x, value, ...)

## S3 method for class 'character'
body_add(x, value, style = NULL, ...)

## S3 method for class 'numeric'
body_add(x, value, style = NULL, format_fun = formatC, ...)

## S3 method for class 'factor'
body_add(x, value, style = NULL, format_fun = as.character, ...)

## S3 method for class 'fpar'
body_add(x, value, style = NULL, ...)

## S3 method for class 'data.frame'
body_add(
  x,
  value,
  style = NULL,
  header = TRUE,
  tcf = table_conditional_formatting(),
  alignment = NULL,
  ...
)

## S3 method for class 'block_caption'
body_add(x, value, ...)

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

## S3 method for class 'block_toc'
body_add(x, value, ...)

## S3 method for class 'external_img'
body_add(x, value, style = "Normal", ...)

## S3 method for class 'run_pagebreak'
body_add(x, value, style = NULL, ...)

## S3 method for class 'run_columnbreak'
body_add(x, value, style = NULL, ...)

## S3 method for class 'gg'
body_add(x, value, width = 6, height = 5, res = 300, style = "Normal", ...)

## S3 method for class 'plot_instr'
body_add(x, value, width = 6, height = 5, res = 300, style = "Normal", ...)  

## S3 method for class 'block_pour_docx'
body_add(x, value, ...)  

Arguments

- **x**: an rdocx object  
- **value**: object to add in the document. Supported objects are vectors, data.frame, graphics, block of formatted paragraphs, unordered list of formatted paragraphs, pretty tables with package flextable, "Microsoft" charts with package mschart.  
- **...**: further arguments passed to or from other methods. When adding a ggplot object or plot_instr, these arguments will be used by png function. See method signatures to see what arguments can be used.  
- **style**: paragraph style name. These names are available with function styles_info and are the names of the Word styles defined in the base document (see argument path from read_docx).  
- **format_fun**: a function to be used to format values.  
- **header**: display header if TRUE  
- **tcf**: conditional formatting settings defined by table_conditional_formatting()  
- **alignment**: columns alignment, argument length must match with columns length, values must be "l" (left), "r" (right) or "c" (center).  
- **width**: height in inches  
- **height**: height in inches  
- **res**: resolution of the png image in ppi

Methods (by class)

- character: add a character vector.  
- numeric: add a numeric vector.  
- factor: add a factor vector.  
- fpar: add a fpar object. These objects enable the creation of formatted paragraphs made of formatted chunks of text.  
- data.frame: add a data.frame object with block_table().  
- block_caption: add a block_caption object. These objects enable the creation of set of formatted paragraphs made of formatted chunks of text.  
- block_list: add a block_list object.  
- block_toc: add a table of content (a block_toc object).  
- external_img: add an image (a external_img object).  
- run_pagebreak: add a run_pagebreak object.  
- run_columnbreak: add a run_columnbreak object.  
- gg: add a ggplot object.  
- plot_instr: add a base plot with a plot_instr object.  
- block_pour_docx: pour content of an external docx file with with a block_pour_docx object.
Illustrations

Examples

```r
# Examples

doc_1 <- read_docx()
doc_1 <- body_add(doc_1, "Table of content", style = "heading 1")
doc_1 <- body_add(doc_1, block_toc())
doc_1 <- body_add(doc_1, run_pagebreak())
doc_1 <- body_add(doc_1, "A title", style = "heading 1")
doc_1 <- body_add(doc_1, head(iris), style = "table_template")
doc_1 <- body_add(doc_1, "Another title", style = "heading 1")
doc_1 <- body_add(doc_1, letters, style = "Normal")
print(doc_1, target = tempfile(fileext = "docx"))
# print(doc_1, target = "test.docx")
```

**body_add_blocks**  
add a list of blocks into a document

Description

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

Usage

```r
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".

See Also

Other functions for adding content: `body_add_break()`, `body_add_docx()`, `body_add_fpar()`, `body_add_gg()`, `body_add_img()`, `body_add_par()`, `body_add_table()`, `body_add_toc()`

Examples

```r
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()),
  )
)
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".

See Also

Other functions for adding content: body_add_blocks(), body_add_docx(), body_add_fpar(), body_add_gg(), body_add_img(), body_add_par(), body_add_table(), body_add_toc()

Examples

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

See Also

Other functions for adding content: `body_add_blocks()`, `body_add_break()`, `body_add_fpar()`, `body_add_gg()`, `body_add_img()`, `body_add_par()`, `body_add_table()`, `body_add_toc()`

Examples

```r
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)
```

Description

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

Usage

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

**Arguments**

- `x`: a docx device
- `value`: a character
- `style`: paragraph style. If NULL, paragraph settings from `fpar` will be used. If not NULL, it must be a paragraph style name (located in the template provided as `read_docx(path = ...)`); in that case, paragraph settings from `fpar` will be ignored.
- `pos`: where to add the new element relative to the cursor, one of "after", "before", "on".

**See Also**

- `fpar`
- Other functions for adding content: `body_add_blocks()`, `body_add_break()`, `body_add_docx()`, `body_add_gg()`, `body_add_img()`, `body_add_par()`, `body_add_table()`, `body_add_toc()`

**Examples**

```r
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"))
```

---

**add ggplot**

**Description**

add a ggplot as a png image into an rdocx object

**Usage**

```r
body_add_gg(x, value, width = 6, height = 5, res = 300, style = "Normal", ...)
```
Arguments

- `x`: an rdocx object
- `value`: ggplot object
- `width`: height in inches
- `height`: height in inches
- `res`: resolution of the png image in ppi
- `style`: paragraph style
- `...`: Arguments to be passed to png function.

See Also

Other functions for adding content: `body_add_blocks()`, `body_add_break()`, `body_add_docx()`, `body_add_fpar()`, `body_add_img()`, `body_add_par()`, `body_add_table()`, `body_add_toc()`

Examples

```r
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") )
}
```

---

**Description**

add an image into an rdocx object.

**Usage**

```r
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".
See Also

Other functions for adding content: body_add_blocks(), body_add_break(), body_add_docx(), body_add_fpar(), body_add_gg(), body_add_img(), body_add_table(), body_add_toc()

Examples

```r
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_par**

add paragraph of text

**Description**

add a paragraph of text into an rdocx object

**Usage**

```r
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".

See Also

Other functions for adding content: body_add_blocks(), body_add_break(), body_add_docx(), body_add_fpar(), body_add_gg(), body_add_img(), body_add_table(), body_add_toc()

Examples

```r
library(magrittr)

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"))
```
Description

add a table into an rdocx object

Usage

body_add_table(
  x,
  value,
  style = NULL,
  pos = "after",
  header = TRUE,
  alignment = NULL,
  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
alignment columns alignment, argument length must match with columns length, values must be "l" (left), "r" (right) or "c" (center).
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.

See Also

Other functions for adding content: body_add_blocks(), body_add_break(), body_add_docx(), body_add_fpar(), body_add_gg(), body_add_img(), body_add_par(), body_add_toc()
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 ":"

See Also

Other functions for adding content: body_add_blocks(), body_add_break(), body_add_docx(), body_add_fpar(), body_add_gg(), body_add_img(), body_add_par(), body_add_table()

Examples

library(magrittr)

    doc <- read_docx() %>% body_add_toc()

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

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

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

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

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

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

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

```r
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, "\b\n\.*\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**

```r
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**

```r
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",
```
change_styles

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"))

---

change_styles  replace paragraphs styles

Description

Replace styles with others in a Word document. This function is to be used for paragraph styles.

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 \texttt{rpptx} object

See Also

Other functions for reading presentation informations: \texttt{annotate_base()}, \texttt{layout_properties()}, \texttt{layout_summary()}, \texttt{length.rpptx()}, \texttt{plot_layout_properties()}, \texttt{slide_size()}, \texttt{slide_summary()}

Examples

\begin{verbatim}
x <- read_pptx()
color_scheme ( x = x )
\end{verbatim}

---

\texttt{cursor_begin} \hspace{1cm} \textit{set cursor in an \texttt{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

\begin{verbatim}
cursor_begin(x)
cursor_bookmark(x, id)
cursor_end(x)
cursor_reach(x, keyword)
cursor_forward(x)
cursor_backward(x)
\end{verbatim}

Arguments

\begin{verbatim}
x \hspace{1cm} a \texttt{docx} device
id \hspace{1cm} bookmark id
keyword \hspace{1cm} keyword to look for as a regular expression
\end{verbatim}

\texttt{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_begin**

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**

```r
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() %>%

# 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.
**docx_bookmarks**

**Usage**

```r
docx_body_xml(x)
```

**Arguments**

- `x`: an *rdocx* object

**Examples**

```r
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**

```r
docx_bookmarks(x)
```

**Arguments**

- `x`: an *rdocx* object

**See Also**

Other functions for Word document informations: `doc_properties()`, `docx_dim()`, `length.rdocx()`, `set_doc_properties()`, `styles_info()`

**Examples**

```r
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**

```r
docx_dim(x)
```

**Arguments**

- `x`: an `rdocx` object

**See Also**

Other functions for Word document informations: `doc_properties()`, `docx_bookmarks()`, `length.rdocx()`, `set_doc_properties()`, `styles_info()`

**Examples**

```r
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**

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

**Arguments**

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

**See Also**

Other functions for officer extensions: `fortify_location()`, `get_reference_value()`, `opts_current_table()`, `to_pml()`, `to_wml()`, `wml_link_images()`
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

```r
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

```r
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

```r
doc_properties(x)
```

Arguments

- `x` an rdocx or rpptx object

See Also

Other functions for Word document informations: docx_bookmarks(), docx_dim(), length.rdocx(), set_doc_properties(), styles_info()

Examples

```r
x <- read_docx()
doc_properties(x)
```
empty_content

Description

an empty object to include as an empty placeholder shape in a presentation. This comes in handy when presentation are updated through R, but a user still wants to write the takeaway statements in PowerPoint.

Usage

empty_content()

See Also

ph_with(), body_add_blocks()

Examples

fileout <- tempfile(fileext = ".pptx")
doc <- read_pptx()
doc <- add_slide(doc, layout = "Two Content",
    master = "Office Theme")
doc <- ph_with(x = doc, value = empty_content(),
    location = ph_location_type(type = "title") )
print(doc, target = fileout )

external_img

Description

Wraps an image in an object that can then be embedded in a PowerPoint slide or within a Word paragraph.

The image is added as a shape in PowerPoint (it is not possible to mix text and images in a PowerPoint form). With a Word document, the image will be added inside a paragraph.

Usage

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

Arguments

src image file path
width height in inches.
height height in inches
See Also

ph_with, body_add, fpar

Other run functions for reporting: ftext(), run_autonum(), run_columnbreak(), run_linebreak(), run_pagebreak(), run_reference(), run_seqfield()

Examples

# wrap r logo with external_img ----
srcfile <- file.path(R.home("doc"), "html", "logo.jpg")
extimg <- external_img(src = srcfile, height = 1.06/2,
                        width = 1.39/2)

# pptx example ----
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(x = doc, value = extimg,
                location = ph_location_type(type = "body"),
                use_loc_size = FALSE )
print(doc, target = tempfile(fileext = ".pptx"))

fp_t <- fpar(font.size = 20, color = "red")
an_fpar <- fpar(extimg, ftext(" is cool!", fp_t))

# docx example ----
x <- read_docx()
x <- body_add(x, an_fpar)
print(x, target = tempfile(fileext = ".docx"))

---

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`

Other functions for officer extensions: `docx_reference_img()`, `get_reference_value()`, `opts_current_table()`, `to_pml()`, `to_wml()`, `wml_link_images()`

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

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

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

Arguments

... cot objects (`ftext`, `external_img`)
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.
object fpar object

Details

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

See Also

Other block functions for reporting: `block_caption()`, `block_list()`, `block_pour_docx()`, `block_section()`, `block_table()`, `block_toc()`, `plot_instr()`, `unordered_list()`
**Examples**

```r
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**

```r
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: \(0\geq value\)
- **object**: fp_border object
- **...**: further arguments - not used

**Examples**

```r
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  

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,
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 "ltrb", "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,
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").

- **keep_with_next**
  - a scalar logical. Specifies that the paragraph (or at least part of it) should be rendered on the same page as the next paragraph when possible.
fp_text

x, object  fp_par object
...

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)

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,
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

See Also

ftext, fpar

Examples

fp_text()
fp_text(color = "red")
fp_text(bold = TRUE, shading.color = "yellow")
print( fp_text (color="red", font.size = 12) )
ftext  

formatted chunk of text

Description

Format a chunk of text with text formatting properties (bold, color, ...).

The function allows you to create pieces of text formatted in a certain way. You should use this function in conjunction with fpar to create paragraphs consisting of differently formatted text parts.

Usage

ftext(text, prop)

Arguments

text  

text value, a string.

prop  

formatting text properties returned by fp_text.

See Also

fp_text

Other run functions for reporting: external_img(), run_autonum(), run_columnbreak(), run_linebreak(), run_pagebreak(), run_reference(), run_seqfield()

Examples

ftext("hello", fp_text())

properties1 <- fp_text(color = "red")
properties2 <- fp_text(bold = TRUE, shading.color = "yellow")
ftext1 <- ftext("hello", properties1)
ftext2 <- ftext("World", properties2)
paragraph <- fpar(ftext1, " ", ftext2)

x <- read_docx()
x <- body_add(x, paragraph)
print(x, target = tempfile(fileext = ".docx"))
get_reference_value

Get the document being used as a template

Description
Get filename of the document being used as a template in an R Markdown document rendered as HTML, PowerPoint presentation or Word document. It requires packages rmarkdown >= 1.10.14 and knitr.

Usage
get_reference_value(format = NULL)

Arguments
format document format, one of 'pptx', 'docx' or 'html'

Value
a name file

See Also
Other functions for officer extensions: docx_reference_img(), fortify_location(), opts_current_table(), to_pml(), to_wml(), wml_link_images()

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()`, `plot_layout_properties()`, `slide_size()`, `slide_summary()`

Examples

```r
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 )
```

```r
layout_summary ( x = my_pres )
```
**length.rdocx**

`length.x`  

**number of blocks inside an rdocx object**

**Description**

return the number of blocks inside an rdocx object. This number also include the default section definition of a Word document - default Word section is an uninvisible element.

**Usage**

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

**Arguments**

- `x`  
an rdocx object

**See Also**

Other functions for Word document informations: `doc_properties()`, `docx_bookmarks()`, `docx_dim()`, `set_doc_properties()`, `styles_info()`

**Examples**

```r
# how many elements are there in an new document produced  
# with the default template.
length( read_docx() )
```

---

**length.rpptx**  

`length.x`  

**number of slides**

**Description**

Function length will return the number of slides.

**Usage**

```r
## 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()`, `plot_layout_properties()`, `slide_size()`, `slide_summary()`

Examples

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

---

**media_extract**

*Extract media from a document object*

Description

Extract files from an `rdocx` or `rpptx` object.

Usage

```r
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

```r
eample_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

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

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/

---

officer-defunct      Defunct Functions in Package officer

Description

Defunct Functions in Package officer

Usage

ph_from_xml(...)

ph_from_xml_at(...)

ph_with_table(...)

ph_with_img(...)

ph_with_gg(...)

on_slide

ph_with_ul(...)
ph_with_table_at(...)
ph_with_fpars_at(...)
body_end_section(...)
body_default_section(...)
break_column_before(...)

Arguments
...

Details
ph_from_xml() is replaced by ph_with.xml_document.
ph_from_xml_at() is replaced by ph_with.xml_document.
ph_with_table() is replaced by ph_with.xml_document.
ph_with_img() is replaced by ph_with.xml_document.
ph_with_gg() is replaced by ph_with.xml_document.
ph_with_ul() is replaced by ph_with.xml_document.
ph_with_table_at() is replaced by ph_with.xml_document.
ph_with_fpars_at() is replaced by ph_with.xml_document.
body_end_section() is replaced by function body_end_section_*.
body_default_section() is replaced by function body_end_section_*.
break_column_before() is replaced by function slip_in_column_break.

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)

opts_current_table

Get table options in a 'knitr' context

Description

Get options for table rendering.

It should not be used by the end user. The function is a utility to facilitate the retrieval of table
options supported by the 'flextable', 'officedown' and of course 'officer' packages.

Usage

opts_current_table()

Value

a list with following elements:

• cap.style (default: NULL)
• cap.pre (default: "Table ")
• cap.sep (default: ":")
• id (default: NULL)
• cap (default: NULL)
• style (default: NULL)
• tab.lp (default: "tab:")
• table_layout (default: "autofit")
• table_width (default: 1)
pack_folder

- first_row (default: TRUE)
- first_column (default: FALSE)
- last_row (default: FALSE)
- last_column (default: FALSE)
- no_hband (default: TRUE)
- no_vband (default: TRUE)

See Also
Other functions for officer extensions: docx_reference_img(), fortify_location(), get_reference_value(), to_pml(), to_wml(), wml_link_images()

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

page_mar  page margins object

Description
The margins for each page of a section The function creates a representation of the dimensions of a page. The dimensions are defined by length, width and orientation. If the orientation is in landscape mode then the length becomes the width and the width becomes the length.

Usage
page_mar(
  bottom = 1,
  top = 1,
  right = 1,
  left = 1,
  header = 0.5,
  footer = 0.5,
  gutter = 0.5
)
**Arguments**

- **bottom, top**
  distance (in inches) between the bottom/top of the text margin and the bottom/top of the page. The text is placed at the greater of the value of this attribute and the extent of the header/footer text. A negative value indicates that the content should be measured from the bottom/top of the page regardless of the footer/header, and so will overlap the footer/header. For example, header=-0.5, bottom=1 means that the footer must start one inch from the bottom of the page and the main document text must start a half inch from the bottom of the page. In this case, the text and footer overlap since bottom is negative.

- **left, right**
  distance (in inches) from the left/right edge of the page to the left/right edge of the text.

- **header**
  distance (in inches) from the top edge of the page to the top edge of the header.

- **footer**
  distance (in inches) from the bottom edge of the page to the bottom edge of the footer.

- **gutter**
  page gutter (in inches).

**See Also**

Other functions for section definition: `page_size()`, `prop_section()`, `section_columns()`

**Examples**

```r
page_mar()
```

---

**Description**

The function creates a representation of the dimensions of a page. The dimensions are defined by length, width and orientation. If the orientation is in landscape mode then the length becomes the width and the width becomes the length.

**Usage**

```r
page_size(width = 21/2.54, height = 29.7/2.54, orient = "portrait")
```

**Arguments**

- **width, height**
  page width, page height (in inches).

- **orient**
  page orientation, either 'landscape', either 'portrait'.

**See Also**

Other functions for section definition: `page_mar()`, `prop_section()`, `section_columns()`
**ph_add_fpar**

**Examples**

```r
page_size(orient = "landscape")
```

---

**Description**

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

**Usage**

```r
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_with("", 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 existing shape, existing paragraphs will be preserved.

Usage

```r
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

```r
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)
```

Description

append text in a placeholder. The function let you add text to an existing content in an existing shape, existing text will be preserved.

Usage

```r
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

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

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

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_with(doc, "",
location = ph_location(rotation = 90, bg = "red",
...)
```
`ph_empty`  

```r
  newlabel = "myph")
  doc <- ph_add_text(doc, str = "dummy text",
                      ph_label = "myph")

  print(doc, target = fileout)
```

---

**Description**

add a new empty shape in the current slide. This function is deprecated, function `ph_with` should be used instead.

**Usage**

```r
ph_empty(x, type = "body", index = 1, 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 recommended 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
**Description**

add hyperlink to a placeholder in the current slide.

**Usage**

```r
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**

```r
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 function Ph_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.
b
  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") )
The function will use the label of a placeholder to find the corresponding location.

**Usage**

```r
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**

```r
# ph_location_label demo ----

doc <- read_pptx()
doc <- add_slide(doc, layout = "Title and Content")

# all ph_label can be read here
```
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

Description

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

Usage

```r
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

```r
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

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

Arguments

- `left`, `top`, `width`, `height`  
  placeholder 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', 'fr', 'sldNum'. It will be used as a template placeholder.
- `id`  
  index of the placeholder template. If two body placeholders, 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

```r
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 criteria to find the corresponding location.

**Usage**

```r
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', 'fr', '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**

```r
# ph_location_type demo ----

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_crttitle <- 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_crttitle)

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

remove a shape in a slide

**Usage**

```r
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**

```r
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  

### Description

add slide link to a placeholder in the current slide.

### Usage

```r
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

```r
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**

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

## 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, level_list = integer(0), ...)`

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

## S3 method for class 'data.frame'  
`ph_with(
  x,
```
value, location, header = TRUE, tcf = table_conditional_formatting(), alignment = NULL, ...

## S3 method for class 'gg'
ph_with(x, value, location, res = 300, ...)

## S3 method for class 'plot_instr'
ph_with(x, value, location, res = 300, ...)

## 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 'empty_content'
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.
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".
... further arguments passed to or from other methods. When adding a ggplot object or plot_instr, these arguments will be used by png function.
format_fun format function for non character vectors
level_list The list of levels for hierarchy structure as integer values. If used the object is formatted as an unordered list. If 1 and 2, item 1 level will be 1, item 2 level will be 2.
header display header if TRUE
tcf conditional formatting settings defined by table_conditional_formatting()
alignment alignment for each columns, 'l' for left, 'r' for right and 'c' for center. Default to NULL.
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 first formatted then added as paragraphs.
- **factor**: add a factor vector to a new shape on the current slide, values will 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 with function block_table(). 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.
- **plot_instr**: add an R plot 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.
- **empty_content**: add an empty_content to a new shape on the current slide.
- **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.

Examples

```r
# this name will be used to print the file
# change it to "youfile.pptx" to write the pptx
# file in your working directory.
fileout <- tempfile(fileext = "pptx")

doc_1 <- read_pptx()
```
sz <- slide_size(doc_1)
# add text and a table ----
doc_1 <- add_slide(doc_1, layout = "Two Content", master = "Office Theme")
doc_1 <- ph_with(x = doc_1, value = c("Table cars"),
location = ph_location_type(type = "title") )
doc_1 <- ph_with(x = doc_1, value = names(cars),
location = ph_location_left() )
doc_1 <- ph_with(x = doc_1, value = cars,
location = ph_location_right() )

# add a base plot ----
anyplot <- plot_instr(code = {
  col <- c("#440154FF", ",#443A83FF", ",#31688EFF",
"#21908CFF", ",#35B779FF", ",#8FD744FF", ",#FDE725FF")
  barplot(1:7, col = col, yaxt="n")
})
doc_1 <- add_slide(doc_1)
doc_1 <- ph_with( doc_1, anyplot,
location = ph_location_fullsize(),
bg = "#006699")

# add a ggplot2 plot ----
if( require("ggplot2") ){
  doc_1 <- add_slide(doc_1)
  gg_plot <- ggplot(data = iris ) +
      geom_point(mapping = aes(Sepal.Length, Petal.Length),
  size = 3) +
      theme_minimal()
  doc_1 <- ph_with(x = doc_1, value = gg_plot,
      location = ph_location_type(type = "body"),
      bg = "transparent" )
  doc_1 <- ph_with(x = doc_1, value = "graphic title",
      location = ph_location_type(type="title") )
}

# add a external images ----
doc_1 <- add_slide(doc_1, layout = "Title and Content",
master = "Office Theme")
doc_1 <- ph_with(x = doc_1, value = empty_content(),
location = ph_location(left = 0, top = 0,
width = sz$width, height = sz$height, bg = "black") )

svg_file <- file.path(R.home(component = "doc"), "html/Rlogo.svg")
if( require("rsvg") ){
  doc_1 <- ph_with(x = doc_1, value = "External images",
  location = ph_location_type(type = "title") )
  doc_1 <- ph_with(x = doc_1, external_img(svg_file, 100/72, 76/72),
  location = ph_location_right(), use_loc_size = FALSE )
  doc_1 <- ph_with(x = doc_1, external_img(svg_file),
  location = ph_location_left(),
  use_loc_size = TRUE )
}
# add a block_list ----
dummy_text <- readLines(system.file(package = "officer", "doc_examples/text.txt"))
fp_1 <- fp_text(bold = TRUE, color = "pink", font.size = 0)
fp_2 <- fp_text(bold = TRUE, font.size = 0)
fp_3 <- fp_text(italic = TRUE, color="red", font.size = 0)
bl <- block_list(
  fpar(ftext("hello world", fp_1)),
  fpar("hello", fp_2),
  ftext("hello", fp_3)
),
  dummy_text
)
doc_1 <- add_slide(doc_1)
doc_1 <- ph_with(x = doc_1, value = bl,
  location = ph_location_type(type="body") )

# fpar ------
hw <- fpar(ftext("hello world",
  fp_text(bold = TRUE, font.family = "Bradley Hand",
  font.size = 150, color = "#F5595B"))
doc_1 <- add_slide(doc_1)
doc_1 <- ph_with(x = doc_1, 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_1 <- add_slide(doc_1)
doc_1 <- ph_with(x = doc_1, value = ul,
  location = ph_location_type() )

print(doc_1, target = fileout )

---

### ph_with_gg_at

*add ggplot to a pptx presentation*

**Description**

Add a ggplot as a png image into an rppx object. This function is deprecated in favor of `ph_with`.

**Usage**

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

- **x**: an pptx object
- **src**: ggplot object
- **width, height**: image size in inches
- **left, top**: location of the new shape on the slide
- **...**: Arguments to be passed to png function.

---

**Description**

add an image as a new shape in the current slide. This function is deprecated in favor of `ph_with`.

**Usage**

```r
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.
- **left, top**: location of the new shape on the slide
- **width, height**: image size in inches
- **rot**: rotation angle

---

**Description**

add text into a new shape in a slide. This function is deprecated in favor of `ph_with`.

**Usage**

```r
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 recommended 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`.

Description

A simple wrapper to capture plot instructions that will be executed and copied in a document. It produces an object of class 'plot_instr' with a corresponding method `ph_with()`.

The function enable usage of any R plot with argument `code`. Wrap your code between curly bracket if more than a single expression.

Usage

```
plot_instr(code)
```

Arguments

- `code`: plotting instructions

See Also

`ph_with()`, `body_add()`

Other block functions for reporting: `block_caption()`, `block_list()`, `block_pour_docx()`, `block_section()`, `block_table()`, `block_toc()`, `fpar()`, `unordered_list()`

Examples

```
# plot_instr demo ----

anyplot <- plot_instr(code = {
    barplot(1:5, col = 2:6)
})

doc <- read_docx()
```
 plot_layout_properties

 Plot slide layout properties

Description

Plot slide layout properties and print informations into defined placeholders. This can be useful to help visualise placeholders locations and identifier.

Usage

plot_layout_properties(x, layout = NULL, master = NULL, labels = TRUE)

Arguments

x an rpptx object
layout slide layout name to use
master master layout name where layout is located
labels if TRUE, placeholder labels will be printed, if FALSE placeholder types and identifiers will be printed.

See Also

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

Examples

x <- read_pptx()
plot_layout_properties( x = x, layout = "Title Slide",
  master = "Office Theme" )
plot_layout_properties( x = x, layout = "Two Content" )
### pptx_summary

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

**Description**

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

**Usage**

```r
pptx_summary(x)
```

**Arguments**

- `x`  
an rpptx object

**Examples**

```r
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**

```r
## 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

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

Description

A section is a grouping of blocks (ie. paragraphs and tables) that have a set of properties that define pages on which the text will appear.

A Section properties object stores information about page composition, such as page size, page orientation, borders and margins.

Usage

```r
prop_section(
  page_size = NULL,
  page_margins = NULL,
  type = NULL,
  section_columns = NULL
)
```

Arguments

- `page_size`: page dimensions, an object generated with function `page_size`.
- `page_margins`: page margins, an object generated with function `page_mar`.
- `type`: Section type. It defines how the contents of the section will be placed relative to the previous section. Available types are "continuous" (begins the section on the next paragraph), "evenPage" (begins on the next even-numbered page), "nextColumn" (begins on the next column on the page), "nextPage" (begins on the following page), "oddPage" (begins on the next odd-numbered page).
- `section_columns`: section columns, an object generated with function `section_columns`.

See Also

`block_section`

Other functions for section definition: `page_mar()`, `page_size()`, `section_columns()`

Examples

```r
prop_section(
  page_size = page_size(orient = "landscape"),
  page_margins = page_mar(top = 2),
  type = "continuous"
)```
Description

Define table properties such as fixed or autofit layout, table width in the document, eventually column widths.

Usage

```r
prop_table(
    style = NA_character_,
    layout = table_layout(),
    width = table_width(),
    colwidths = table_colwidths(),
    tcf = table_conditional_formatting(),
    align = "center"
)
```

Arguments

- **style**: table style to be used to format table
- **layout**: layout defined by `table_layout()`. 
- **width**: table width in the document defined by `table_width()` 
- **colwidths**: column widths defined by `table_colwidths()` 
- **tcf**: conditional formatting settings defined by `table_conditional_formatting()` 
- **align**: table alignment (one of left, center or right)

See Also

Other functions for table definition: `table_colwidths()`, `table_conditional_formatting()`, `table_layout()`, `table_width()`

Examples

```r
prop_table()
```
Create a 'Word' document object

Description

read and import a docx file as an R object representing the document. When no file is specified, it uses a default empty file.

Use then this object to add content to it and create Word files from R.

Usage

read_docx(path = NULL)

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

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

styles

read_docx() uses a Word file as the initial document. This is the original Word document from which the document layout, paragraph styles, or table styles come.

You will be able to add formatted text, change the paragraph style with the R api, but it will always be easier to use the styles from the original document.

See body_add_* functions to add content.

See Also

print.rdocx, body_add_par, body_add

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"))
}
**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**

```r
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**

```r
read_pptx()
```
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)
}

Description

remove a slide from a pptx presentation

Usage

remove_slide(x, index = NULL)
run_autonum

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)

Description
Create an autonumbered chunk, i.e. a string representation of a sequence, each item will be numbered. These runs can also be bookmarked and be used later for cross references.

Usage

run_autonum(
  seq_id = "table",
  pre_label = "Table ",
  post_label = ": ",
  bkm = NULL,
  bkm_all = FALSE
)

Arguments

seq_id  sequence identifier
pre_label, post_label  text to add before and after number
bkm  bookmark id to associate with autonumber run. If NULL, no bookmark is added. Value can only be made of alpha numeric characters, `-` and `_`.

bkm_all  if TRUE, the bookmark will be set on the while string, if FALSE, the bookmark will be set on the number only. Default to FALSE. As an effect when a reference to this bookmark is used, the text can be like “Table 1” or “1” (pre_label is not included in the referenced text).
See Also

Other run functions for reporting: `external_img()`, `ftext()`, `run_columnbreak()`, `run_linebreak()`, `run_pagebreak()`, `run_reference()`, `run_seqfield()`

Examples

```r
run_autonum()
run_autonum(seq_id = "fig", pre_label = "fig. ")
run_autonum(seq_id = "tab", pre_label = "Table ", bkm = "anytable")
```

---

**run_columnbreak**  
*column break*

Description

Create a representation of a column break

Usage

```r
run_columnbreak()
```

See Also

Other run functions for reporting: `external_img()`, `ftext()`, `run_autonum()`, `run_linebreak()`, `run_pagebreak()`, `run_reference()`, `run_seqfield()`

Examples

```r
run_columnbreak()
```

---

**run_linebreak**  
*page break for Word*

Description

Object representing a line break for a Word document. The result must be used within a call to `fpar`.

Usage

```r
run_linebreak()
```

See Also

Other run functions for reporting: `external_img()`, `ftext()`, `run_autonum()`, `run_columnbreak()`, `run_pagebreak()`, `run_reference()`, `run_seqfield()`
run_pagebreak

Examples

```r
fp_t <- fp_text(font.size = 12, bold = TRUE)
an_fpar <- fpar("let's add a line break", run_linebreak(), ftext("and blah blah!", fp_t))

x <- read_docx()
x <- body_add(x, an_fpar)
print(x, target = tempfile(fileext = ".docx"))
```

---

run_pagebreak  

**page break for Word**

Description

Object representing a page break for a Word document. The result must be used within a call to `fpar`.

Usage

```r
run_pagebreak()
```

See Also

Other run functions for reporting: `external_img()`, `ftext()`, `run_autonum()`, `run_columnbreak()`, `run_linebreak()`, `run_reference()`, `run_seqfield()`

Examples

```r
fp_t <- fp_text(font.size = 12, bold = TRUE)
an_fpar <- fpar("let's add a break page", run_pagebreak(), ftext("and blah blah!", fp_t))

x <- read_docx()
x <- body_add(x, an_fpar)
print(x, target = tempfile(fileext = ".docx"))
```

---

run_reference  

**reference**

Description

Create a representation of a reference

Usage

```r
run_reference(id)
```
sanitize_images

Arguments

id reference id, a string

See Also

Other run functions for reporting: external_img(), ftext(), run_autonum(), run_columnbreak(), run_linebreak(), run_pagebreak(), run_seqfield()

Examples

run_reference('a_ref')

run_seqfield seqfield

Description

Create a seqfield

Usage

run_seqfield(seqfield)

Arguments

seqfield seqfield string

See Also

Other run functions for reporting: external_img(), ftext(), run_autonum(), run_columnbreak(), run_linebreak(), run_pagebreak(), run_reference()

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
Description

Add sections in a Word document. A section affects preceding paragraphs or tables.

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. " %>%
section_columns

The function creates a representation of the columns of a section.

Usage

section_columns(widths = c(2.5, 2.5), space = 0.25, sep = FALSE)

Arguments

- `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.

See Also

Other functions for section definition: `page_mar()`, `page_size()`, `prop_section()`
**set_doc_properties**

**Examples**

```r
section_columns()
```

---

**Description**

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

**Usage**

```r
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.

**See Also**

Other functions for Word document informations: `doc_properties()`, `docx_bookmarks()`, `docx_dim()`, `length.rdocx()`, `styles_info()`

**Examples**

```r
x <- read_docx()
x <- set_doc_properties(x, title = "title",
subject = "document subject", creator = "Me me me",
description = "this document is empty",
created = Sys.time())
x <- doc_properties(x)
```
**Description**

set a particular sheet selected when workbook will be edited.

**Usage**

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

**Arguments**

- `x` : rxlsx object
- `sheet` : sheet name

**Examples**

```r
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**

**Description**

Shortcuts for `fp_text`, `fp_par`, `fp_cell` and `fp_border`.

**Usage**

```
shortcuts
```

**Examples**

```r
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

```r
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()`, `plot_layout_properties()`, `slide_summary()`

Examples

```r
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

```r
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(), plot_layout_properties(), slide_size()  

**Examples**

```r
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_column_break**  
*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**

```r
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".
slip_in_footnote
append a footnote

slip_in_footnote 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
slip_in_seqfield

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".

Description

append seq field into a paragraph of an rdocx object. This feature is only available when documents 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

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

```r
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**

```r
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**

```r
styles_info(  
x,  
  type = c("paragraph", "character", "table", "numbering"),  
  is_default = c(TRUE, FALSE)
)
```
Arguments

x an rdocx object
type, is_default
subsets for types (i.e. paragraph) and default style (when is_default is TRUE or FALSE)

See Also

Other functions for Word document informations: doc_properties(), docx_bookmarks(), docx_dim(), length.rdocx(), set_doc_properties()

Examples

x <- read_docx()
styless_info(x)
styless_info(x, type = "paragraph", is_default = TRUE)

---

table_colwidths Column widths of a table

Description

The function defines the size of each column of a table.

Usage

table_colwidths(widths = NULL)

Arguments

widths Column widths expressed in inches.

See Also

Other functions for table definition: prop_table(), table_conditional_formatting(), table_layout(), table_width()
Description

Tables can be conditionally formatted based on few properties as whether the content is in the first row, last row, first column, or last column, or whether the rows or columns are to be banded.

Usage

```
table_conditional_formatting(
    first_row = TRUE,
    first_column = FALSE,
    last_row = FALSE,
    last_column = FALSE,
    no_hband = FALSE,
    no_vband = TRUE
)
```

Arguments

- **first_row, last_row**
  - apply or remove formatting from the first or last row in the table.
- **first_column, last_column**
  - apply or remove formatting from the first or last column in the table.
- **no_hband, no_vband**
  - don’t display odd and even rows or columns with alternating shading for ease of reading.

See Also

Other functions for table definition: `prop_table()`, `table_colwidths()`, `table_layout()`, `table_width()`

Examples

```
table_conditional_formatting(first_row = TRUE, first_column = TRUE)
```
**table_layout**  
*Algorithm for table layout*

### Description

When a table is displayed in a document, it can either be displayed using a fixed width or autofit layout algorithm:

- **fixed**: uses fixed widths for columns. The width of the table is not changed regardless of the contents of the cells.
- **autofit**: uses the contents of each cell and the table width to determine the final column widths.

### Usage

```r
table_layout(type = "autofit")
```

### Arguments

- **type**: 'autofit' or 'fixed' algorithm. Default to 'autofit'.

### See Also

Other functions for table definition: `prop_table()`, `table_colwidths()`, `table_conditional_formatting()`, `table_width()`

---

**table_width**  
*Preferred width for a table*

### Description

Define the preferred width for a table.

### Usage

```r
table_width(width = 1, unit = "pct")
```

### Arguments

- **width**: value of the preferred width of the table.
- **unit**: unit of the width. Possible values are 'in' (inches) and 'pct' (percent)

### Word

All widths in a table are considered preferred because widths of columns can conflict and the table layout rules can require a preference to be overridden.
See Also

Other functions for table definition: `prop_table()`, `table_colwidths()`, `table_conditional_formatting()`, `table_layout()`

---

**to_html**

Convert officer objects to HTML

**Description**

Convert an object made with package officer to HTML. The result is a string.

**Usage**

```r
to_html(x, ...)
```

**Arguments**

- `x`: object
- `...`: Arguments to be passed to methods

---

**to_pml**

Convert officer objects to PresentationML

**Description**

Convert an object made with package officer to PresentationML. The result is a string.

**Usage**

```r
to_pml(x, add_ns = FALSE, ...)
```

**Arguments**

- `x`: object
- `add_ns`: should namespace be added to the top tag
- `...`: Arguments to be passed to methods

**See Also**

Other functions for officer extensions: `docx_reference_img()`, `fortify_location()`, `get_reference_value()`, `opts_current_table()`, `to_wml()`, `wml_link_images()`
**to_wml**  
*Convert officer objects to WordprocessingML*

**Description**

Convert an object made with package officer to WordprocessingML. The result is a string.

**Usage**

```r
to_wml(x, add_ns = FALSE, ...)
```

**Arguments**

- `x`: object
- `add_ns`: should namespace be added to the top tag
- `...`: Arguments to be passed to methods

**See Also**

Other functions for officer extensions: `docx_reference_img()`, `fortify_location()`, `get_reference_value()`, `opts_current_table()`, `to_pml()`, `wml_link_images()`

---

**unordered_list**  
*Unordered list*

**Description**

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

**Usage**

```r
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**

Other block functions for reporting: `block_caption()`, `block_list()`, `block_pour_docx()`, `block_section()`, `block_table()`, `block_toc()`, `fpar()`, `plot_instr()`
Examples

```r
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)
)
unordered_list(
  level_list = c(1, 2, 1),
  str_list = c("Level1", "Level2", "Level1"),
  style = list(
    fp_text(color = "red", font.size = 0),
    fp_text(color = "pink", font.size = 0),
    fp_text(color = "orange", 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.

**See Also**

Other functions for officer extensions: `docx_reference_img()`, `fortify_location()`, `get_reference_value()`, `opts_current_table()`, `to_pml()`, `to_wml()`
add_sheet, 4
add_slide, 5, 49, 52, 83, 85
annotate_base, 5, 28, 46, 48, 78, 93, 94
block_caption, 6, 7, 9–11, 13, 37, 77, 103
block_list, 6, 7, 9–11, 13, 14, 37, 55, 57, 58, 73, 77, 95, 103
block_pour_docx, 6, 7, 9, 10, 11, 13, 37, 77, 103
block_section, 6, 7, 8, 9–11, 13, 37, 77, 103
block_table, 6, 7, 9, 10, 11, 13, 37, 77, 80, 103
body_add, 11, 36, 82
body_add_blocks, 14, 15–21
body_add_break, 14, 15, 16–21
body_add_docx, 14, 15, 16, 17–21
body_add_gg, 14–17, 17, 19–21
body_add_img, 14–18, 18, 19–21
body_add_par, 14–19, 19, 20, 21, 82
body_add_table, 14–19, 20, 21
body_add_toc, 14–20, 21
body_add_xml, 22
body_bookmark, 22
body_default_section (officer-defunct), 50
body_end_section (officer-defunct), 50
body_end_section_columns (sections), 89
body_end_section_columns_landscape (sections), 89
body_end_section_continuous (sections), 89
body_end_section_landscape (sections), 89
body_end_section_portrait (sections), 89
body_remove, 23
body_replace_all_text, 23, 33
body_replace_img_at_bkm
   (body_replace_text_at_bkm), 26
body_replace_text_at_bkm, 26
break_column_before (officer-defunct), 50
change_styles, 27
color_scheme, 6, 27, 46, 48, 78, 93, 94
cursor_backward (cursor_begin), 28
cursor_begin, 28
cursor_bookmark (cursor_begin), 28
cursor_end (cursor_begin), 28
cursor_forward (cursor_begin), 28
cursor_reach (cursor_begin), 28
doc_properties, 31, 32, 34, 47, 91, 99
docx_body_relationship, 30
docx_body_xml, 30
docx_bookmarks, 31, 32, 34, 47, 91, 99
docx_dim, 31, 32, 34, 47, 91, 99
docx_reference_img, 32, 37, 45, 53, 102, 103, 105
docx_show_chunk, 24, 25, 33
docx_summary, 33
empty_content, 35, 73
external_img, 7, 13, 35, 44, 73, 86–88
footers_replace_all_text
   (body_replace_all_text), 23
footers_replace_img_at_bkm
   (body_replace_text_at_bkm), 26
footers_replace_text_at_bkm
   (body_replace_text_at_bkm), 26
format.fp_cell (fp_cell), 39
format.fp_text (fp_text), 42
fortify_location, 32, 36, 45, 53, 102, 103, 105
fp_border, 38, 40, 41
INDEX

fp_cell, 39
fp_par, 40
fp_text, 42, 44, 58
fpar, 6, 7, 9–11, 13, 17, 36, 37, 43, 44, 55, 56, 73, 77, 86, 87, 103
ftext, 36, 43, 44, 86–88
get_reference_value, 32, 37, 45, 53, 102, 103, 105
grep, 25
grepl, 24
gsub, 24
headers_replace_all_text
  (body_replace_all_text), 23
headers_replace_img_at_bkm
  (body_replace_text_at_bkm), 26
headers_replace_text_at_bkm
  (body_replace_text_at_bkm), 26
layout_properties, 6, 28, 45, 46, 48, 78, 93, 94
layout_summary, 5, 6, 28, 45, 46, 48, 78, 93, 94
length.rdocx, 31, 32, 34, 47, 91, 99
length.rpptx, 6, 28, 46, 47, 78, 93, 94
length.xlsx (read.xlsx), 84
media_extract, 48
move_slide, 5, 49, 52, 85
officer, 49
officer-defunct, 50
officer-package (officer), 49
on_slide, 5, 49, 51, 85
opts_current_table, 32, 37, 45, 52, 102, 103, 105
pack_folder, 53
page_mar, 53, 54, 80, 90
page_size, 54, 54, 80, 90
ph_add_fpar, 55
ph_add_par, 56
ph_add_text, 57
ph_empty, 59
ph_empty_at (ph_empty), 59
ph_from_xml (officer-defunct), 50
ph_from_xml_at (officer-defunct), 50
ph_hyperlink, 60, 69, 70
ph_location, 37, 59, 61, 62–66, 68, 73, 77
ph_location_fullsize, 59, 62, 62, 63–66, 68, 73, 77
ph_location_label, 59, 62, 63, 64–66, 68, 73, 77
ph_location_left, 59, 62, 63, 64, 65, 66, 68, 73, 77
ph_location_right, 59, 62–64, 65, 66, 68, 73, 77
ph_location_template, 62–65, 65, 68, 73
ph_location_type, 59, 62–66, 67, 73, 77
ph_remove, 60, 69, 70
ph_slidelink, 60, 69, 70
ph_with, 5, 36, 37, 55, 57, 58, 60, 69, 70, 71, 75, 76, 103
ph_with(), 7, 35, 77
ph_with_fpars_at (officer-defunct), 50
ph_with_gg (officer-defunct), 50
ph_with_gg_at, 75
ph_with_img (officer-defunct), 50
ph_with_img_at, 76
ph_with_table (officer-defunct), 50
ph_with_table_at (officer-defunct), 50
ph_with_text, 76
ph_with_ul (officer-defunct), 50
plot_instr, 6, 7, 9–11, 13, 37, 77, 103
plot_layout_properties, 6, 28, 46, 48, 78, 93, 94
pptx_summary, 79
print.fp_cell (fp_cell), 39
print.fp_par (fp_par), 40
print.fp_text (fp_text), 42
print.rdocx, 82
print.rdocx (read_docx), 82
print.rpptx, 5, 79, 83
print.xlsx (readxlsx), 84
prop_section, 9, 54, 80, 90
prop_table, 81, 99–102
prop_table(), 10
read_docx, 13, 82
read_pptx, 5, 79, 83
read_xlsx, 84
regex, 24, 25
remove_slide, 5, 49, 52, 84
run_autonum, 6, 36, 44, 85, 86–88
run_autonum(), 11
run_columnbreak, 13, 36, 44, 86, 86, 87, 88
run_linebreak, 36, 44, 86, 87, 88
run_pagebreak, 13, 36, 44, 86, 87, 88
run_reference, 36, 44, 86, 87, 88
run_seqfield, 36, 44, 86–88, 88

sanitize_images, 88
sections_columns, 54, 80, 90
sections, 89
set_doc_properties, 31, 32, 34, 47, 91, 99
sheet_select, 92
shortcuts, 92
slide_size, 6, 28, 46, 48, 78, 93, 94
slide_summary, 6, 28, 46, 48, 55, 56, 58, 60, 69, 70, 78, 93, 94
slip_in_column_break, 94
slip_in_footnote, 95
slip_in_img, 95
slip_in_seqfield, 96
slip_in_text, 97
slip_in_xml, 98
styles_info, 13, 31, 32, 34, 47, 91, 98

table_colwidths, 81, 99, 100–102
table_colwidths(), 81
table_conditional_formatting, 81, 99, 100, 101, 102
table_conditional_formatting(), 13, 72, 81
table_layout, 81, 99, 100, 101, 102
table_layout(), 81
table_width, 81, 99–101, 101
table_width(), 81
to_html, 102
to_pml, 32, 37, 45, 53, 102, 103, 105
to_wml, 32, 37, 45, 53, 102, 103, 105

unordered_list, 6, 7, 9–11, 37, 73, 77, 103
unpack_folder, 104
update.fp_border (fp_border), 38
update.fp_cell (fp_cell), 39
update.fp_par (fp_par), 40
update.fp_text (fp_text), 42
update.fpar (fpar), 37

wml_link_images, 32, 37, 45, 53, 102, 103, 104