

# Package ‘officer’

March 28, 2023

**Type** Package

**Title** Manipulation of Microsoft Word and PowerPoint Documents

**Version** 0.6.2

## Description

Access and manipulate 'Microsoft Word', 'RTF' 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. The package does not require any installation of Microsoft products to be able to write Microsoft files.

**License** MIT + file LICENSE

**Imports** grDevices, stats, graphics, utils, zip (>= 2.1.0), xml2 (>= 1.1.0), openssl, R6, uuid, ragg

**URL** <https://ardata-fr.github.io/officeverse/>,  
<https://davidgohel.github.io/officer/>

**Encoding** UTF-8

**BugReports** <https://github.com/davidgohel/officer/discussions>

**RoxygenNote** 7.2.3

**Suggests** testthat, devEMF, ggplot2, rmarkdown, doconv (>= 0.3.0), knitr, rsvg, magick

**NeedsCompilation** no

**Author** David Gohel [aut, cre],  
ArData [cph],  
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] (function annotate\_base),  
John Muschelli [ctb] (google doc compatibility),  
Bill Denney [ctb] (<<https://orcid.org/0000-0002-5759-428X>>, function as.matrix.rpptx),

Nikolai Beck [ctb] (set speaker notes for .pptx documents),  
 Stefan Moog [ctb] (added functionality to set shape geometry and  
 outline),  
 Greg Leleu [ctb] (fields functionality in ppt),  
 Hongyuan Jia [ctb] (<<https://orcid.org/0000-0002-0075-8183>>)

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

**Repository** CRAN

**Date/Publication** 2023-03-28 20:30:02 UTC

## R topics documented:

add_sheet . . . . .	4
add_slide . . . . .	5
annotate_base . . . . .	5
as.matrix.rpptx . . . . .	6
block_caption . . . . .	7
block_list . . . . .	8
block_pour_docx . . . . .	9
block_section . . . . .	10
block_table . . . . .	11
block_toc . . . . .	12
body_add_blocks . . . . .	12
body_add_break . . . . .	13
body_add_caption . . . . .	14
body_add_docx . . . . .	15
body_add_fpar . . . . .	16
body_add_gg . . . . .	17
body_add_img . . . . .	18
body_add_par . . . . .	19
body_add_plot . . . . .	20
body_add_table . . . . .	21
body_add_toc . . . . .	22
body_bookmark . . . . .	23
body_end_block_section . . . . .	24
body_end_section_columns . . . . .	25
body_end_section_columns_landscape . . . . .	26
body_end_section_continuous . . . . .	27
body_end_section_landscape . . . . .	28
body_end_section_portrait . . . . .	28
body_remove . . . . .	29
body_replace_all_text . . . . .	30
body_replace_text_at_bkm . . . . .	32
body_set_default_section . . . . .	33
change_styles . . . . .	34
color_scheme . . . . .	35
cursor_begin . . . . .	35
docx_bookmarks . . . . .	38

docx_dim . . . . .	39
docx_set_character_style . . . . .	39
docx_set_paragraph_style . . . . .	40
docx_show_chunk . . . . .	41
docx_summary . . . . .	42
doc_properties . . . . .	43
empty_content . . . . .	43
external_img . . . . .	44
fpar . . . . .	45
fp_border . . . . .	47
fp_cell . . . . .	47
fp_par . . . . .	49
fp_text . . . . .	51
ftext . . . . .	53
hyperlink_ftext . . . . .	54
layout_properties . . . . .	55
layout_summary . . . . .	56
length.rdocx . . . . .	56
length.rpptx . . . . .	57
media_extract . . . . .	58
move_slide . . . . .	58
notes_location_label . . . . .	59
notes_location_type . . . . .	59
officer . . . . .	60
officer-defunct . . . . .	61
on_slide . . . . .	62
page_mar . . . . .	63
page_size . . . . .	64
ph_hyperlink . . . . .	64
ph_location . . . . .	65
ph_location_fullsize . . . . .	67
ph_location_label . . . . .	67
ph_location_left . . . . .	68
ph_location_right . . . . .	69
ph_location_template . . . . .	70
ph_location_type . . . . .	71
ph_remove . . . . .	73
ph_slidelink . . . . .	74
ph_with . . . . .	75
plot_instr . . . . .	81
plot_layout_properties . . . . .	82
pptx_summary . . . . .	82
print.rpptx . . . . .	83
print.rtf . . . . .	84
prop_section . . . . .	84
prop_table . . . . .	87
read_docx . . . . .	88
read_pptx . . . . .	89

read_xlsx . . . . .	90
remove_slide . . . . .	91
rtf_add . . . . .	92
rtf_doc . . . . .	94
run_autonum . . . . .	95
run_bookmark . . . . .	97
run_columnbreak . . . . .	97
run_footnote . . . . .	98
run_footnoteref . . . . .	99
run_linebreak . . . . .	100
run_pagebreak . . . . .	100
run_reference . . . . .	101
run_wordtext . . . . .	102
run_word_field . . . . .	102
section_columns . . . . .	103
set_autonum_bookmark . . . . .	104
set_doc_properties . . . . .	105
set_notes . . . . .	106
sheet_select . . . . .	107
shortcuts . . . . .	108
slide_size . . . . .	108
slide_summary . . . . .	109
sp_line . . . . .	110
sp_lineend . . . . .	111
styles_info . . . . .	112
table_colwidths . . . . .	113
table_conditional_formatting . . . . .	114
table_layout . . . . .	115
table_stylenames . . . . .	115
table_width . . . . .	116
unordered_list . . . . .	117

## **Index** **118**

---

add_sheet	<i>Add a sheet</i>
-----------	--------------------

---

### **Description**

Add a sheet into an xlsx worksheet.

### **Usage**

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

### **Arguments**

x	xlsx object
label	sheet label

**Examples**

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

---

add_slide	<i>Add a slide</i>
-----------	--------------------

---

**Description**

Add a slide into a pptx presentation.

**Usage**

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

**Arguments**

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

**See Also**

[print.rpptx\(\)](#), [read\\_pptx\(\)](#), [plot\\_layout\\_properties\(\)](#), [ph\\_with\(\)](#), [layout\\_summary\(\)](#)  
 Other functions slide manipulation: [move\\_slide\(\)](#), [on\\_slide\(\)](#), [remove\\_slide\(\)](#), [set\\_notes\(\)](#)

**Examples**

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

---

annotate_base	<i>Placeholder parameters annotation</i>
---------------	--

---

**Description**

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

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

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

**Usage**

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

**Arguments**

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

**Value**

rpptx object of the annotated PowerPoint file

**See Also**

Other functions for reading presentation informations: [color\\_scheme\(\)](#), [doc\\_properties\(\)](#), [layout\\_properties\(\)](#), [layout\\_summary\(\)](#), [length.rpptx\(\)](#), [plot\\_layout\\_properties\(\)](#), [slide\\_size\(\)](#), [slide\\_summary\(\)](#)

**Examples**

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

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

---

as.matrix.rpptx	<i>PowerPoint table to matrix</i>
-----------------	-----------------------------------

---

**Description**

Convert the data in an a 'PowerPoint' table to a matrix or all data to a list of matrices.

**Usage**

```
## S3 method for class 'rpptx'
as.matrix(
  x,
  ...,
  slide_id = NA_integer_,
  id = NA_character_,
  span = c(NA_character_, "fill")
)
```

**Arguments**

x	The rpptx object to convert (as created by <code>officer::read_pptx()</code> )
...	Ignored
slide_id	The slide number to load from (NA indicates first slide with a table, NULL indicates all slides and all tables)
id	The table ID to load from (ignored if <code>is.null(slide_id)</code> , NA indicates to load the first table from the <code>slide_id</code> )
span	How should <code>col_span/row_span</code> values be handled? NA means to leave the value as NA, and "fill" means to fill matrix cells with the value.

**Value**

A matrix with the data, or if `slide_id=NULL`, a list of matrices

**Examples**

```
library(officer)
pptx_file <- system.file(package="officer", "doc_examples", "example.pptx")
z <- read_pptx(pptx_file)
as.matrix(z, slide_id = NULL)
```

---

block_caption	<i>Caption block</i>
---------------	----------------------

---

**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 <a href="#">run_autonum</a>

**See Also**

Other block functions for reporting: [block\\_list\(\)](#), [block\\_pour\\_docx\(\)](#), [block\\_section\(\)](#), [block\\_table\(\)](#), [block\\_toc\(\)](#), [fpar\(\)](#), [plot\\_instr\(\)](#), [unordered\\_list\(\)](#)

## Examples

```
library(officer)

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

doc_1 <- read_docx()
doc_1 <- body_add(doc_1, "A title", style = "heading 1")
doc_1 <- body_add(doc_1, "Hello world!", style = "Normal")
doc_1 <- body_add(doc_1, caption)
doc_1 <- body_add(doc_1, mtcars, style = "table_template")

print(doc_1, target = tempfile(fileext = ".docx"))
```

---

block\_list

*List of blocks*

---

## Description

A list of blocks can be used to gather several blocks (paragraphs, tables, ...) into a single object. The result can be added into a Word document or a PowerPoint presentation.

## Usage

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

## Arguments

... a list of blocks. 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. Supported objects are: `block_caption()`, `block_pour_docx()`, `block_section()`, `block_table()`, `block_toc()`, `fpar()`, `plot_instr()`.

## See Also

`ph_with()`, `body_add_blocks()`, `fpar()`

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



**Examples**

```

# block list -----

img.file <- file.path( R.home("doc"), "html", "logo.jpg" )
fpt_blue_bold <- fp_text(color = "#006699", bold = TRUE)
fpt_red_italic <- fp_text(color = "#C32900", italic = TRUE)

## This can be only be used in a MS word output as pptx does
## not support paragraphs made of text and images.
## (actually it can be used but image will not appear in the
## pptx output)
value <- block_list(
  fpar(ftext("hello world", fpt_blue_bold)),
  fpar(ftext("hello", fpt_blue_bold), " ",
        ftext("world", fpt_red_italic)),
  fpar(
    ftext("hello world", fpt_red_italic),
    external_img(
      src = img.file, height = 1.06, width = 1.39)))
value

doc <- read_docx()
doc <- body_add(doc, value)
print(doc, target = tempfile(fileext = ".docx"))

value <- block_list(
  fpar(ftext("hello world", fpt_blue_bold)),
  fpar(ftext("hello", fpt_blue_bold), " ",
        ftext("world", fpt_red_italic)),
  fpar(
    ftext("blah blah blah", fpt_red_italic)))
value

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

```

---

block\_pour\_docx

*External Word document placeholder*


---

**Description**

Pour the content of a docx file in the resulting docx from an 'R Markdown' document.

**Usage**

```
block_pour_docx(file)
```

**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**

```
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)
```

---

block\_section

*Section for 'Word'*

---

**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 appened in the new section and finally to add a `block_section` with `page_size = page_size(orient = "landscape")`.

**Usage**

```
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**

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

---

block_table	<i>Table block</i>
-------------	--------------------

---

**Description**

Create a representation of a table

**Usage**

```
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 <a href="#">prop_table()</a> . 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**

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

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

---

block_toc	<i>Table of content for 'Word'</i>
-----------	------------------------------------

---

**Description**

Create a representation of a table of content for Word documents.

**Usage**

```
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 <a href="#">run_autonum()</a> 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**

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

---

body_add_blocks	<i>Add a list of blocks into a 'Word' document</i>
-----------------	--

---

**Description**

add a list of blocks produced by [block\\_list](#) into into an rdocx object.

**Usage**

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

**Arguments**

x	an rdocx object
blocks	set of blocks to be used as footnote content returned by function <code>block_list()</code> .
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_caption()`, `body_add_docx()`, `body_add_fpar()`, `body_add_gg()`, `body_add_img()`, `body_add_par()`, `body_add_plot()`, `body_add_table()`, `body_add_toc()`

**Examples**

```
library(officer)

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

bl <- block_list(
  fpar(ftext("hello", shortcuts$fp_bold(color = "red"))),
  fpar(
    ftext("hello world", shortcuts$fp_bold()),
    external_img(src = img.file, height = 1.06, width = 1.39),
    fp_p = fp_par(text.align = "center")
  )
)

doc_1 <- read_docx()
doc_1 <- body_add_blocks(doc_1, blocks = bl)
print(doc_1, target = tempfile(fileext = ".docx"))
```

---

body_add_break	<i>Add a page break in a 'Word' document</i>
----------------	--

---

**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\\_caption\(\)](#), [body\\_add\\_docx\(\)](#), [body\\_add\\_fpar\(\)](#), [body\\_add\\_gg\(\)](#), [body\\_add\\_img\(\)](#), [body\\_add\\_par\(\)](#), [body\\_add\\_plot\(\)](#), [body\\_add\\_table\(\)](#), [body\\_add\\_toc\(\)](#)

**Examples**

```
doc <- read_docx()
doc <- body_add_break(doc)
print(doc, target = tempfile(fileext = ".docx"))
```

---

<code>body_add_caption</code>	<i>Add Word caption in a 'Word' document</i>
-------------------------------	--

---

**Description**

Add a Word caption into an rdocx object.

**Usage**

```
body_add_caption(x, value, pos = "after")
```

**Arguments**

<code>x</code>		an rdocx object
<code>value</code>		an object returned by <a href="#">block_caption()</a>
<code>pos</code>		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\\_par\(\)](#), [body\\_add\\_plot\(\)](#), [body\\_add\\_table\(\)](#), [body\\_add\\_toc\(\)](#)

**Examples**

```
doc <- read_docx()

if (capabilities(what = "png")) {
  doc <- body_add_plot(doc,
    value = plot_instr(
      code = {
        barplot(1:5, col = 2:6)
      }
    ),
    style = "centered"
  )
}
```

```
}
run_num <- run_autonum(
  seq_id = "fig", pre_label = "Figure ",
  bkm = "barplot"
)
caption <- block_caption("a barplot",
  style = "Normal",
  autonum = run_num
)
doc <- body_add_caption(doc, caption)
print(doc, target = tempfile(fileext = ".docx"))
```

---

body\_add\_docx

*Add an external docx in a 'Word' document*

---

## 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\\_caption\(\)](#), [body\\_add\\_fpar\(\)](#), [body\\_add\\_gg\(\)](#), [body\\_add\\_img\(\)](#), [body\\_add\\_par\(\)](#), [body\\_add\\_plot\(\)](#), [body\\_add\\_table\(\)](#), [body\\_add\\_toc\(\)](#)

**Examples**

```

file1 <- tempfile(fileext = ".docx")
file2 <- tempfile(fileext = ".docx")
file3 <- tempfile(fileext = ".docx")
x <- read_docx()
x <- body_add_par(x, "hello world 1", style = "Normal")
print(x, target = file1)

x <- read_docx()
x <- body_add_par(x, "hello world 2", style = "Normal")
print(x, target = file2)

x <- read_docx(path = file1)
x <- body_add_break(x)
x <- body_add_docx(x, src = file2)
print(x, target = file3)

```

---

body\_add\_fpar

*Add fpar in a 'Word' document*


---

**Description**

Add an fpar (a formatted paragraph) into an rdocx object.

**Usage**

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

**Arguments**

x	a docx device
value	a character
style	paragraph style. 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 <code>read_docx(path = ...)</code> ); 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\\_caption\(\)](#), [body\\_add\\_docx\(\)](#), [body\\_add\\_gg\(\)](#), [body\\_add\\_img\(\)](#), [body\\_add\\_par\(\)](#), [body\\_add\\_plot\(\)](#), [body\\_add\\_table\(\)](#), [body\\_add\\_toc\(\)](#)



**Examples**

```

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()
doc <- body_add_fpar(doc, 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),
  hyperlink_ftext(
    href = "https://cran.r-project.org/index.html",
    text = "cran", prop = bold_redface
  ),
  fp_p = fp_par(text.align = "center")
)

doc <- read_docx()
doc <- body_add_fpar(doc, img_in_par)
print(doc, target = tempfile(fileext = ".docx"))

```

---

body\_add\_gg

---

*Add a 'ggplot' in a 'Word' document*


---

**Description**

add a ggplot as a png image into an rdocx object.

**Usage**

```

body_add_gg(
  x,
  value,
  width = 6,
  height = 5,
  res = 300,
  style = "Normal",
  scale = 1,
  pos = "after",
  ...
)

```

**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
scale	Multiplicative scaling factor, same as in ggsave
pos	where to add the new element relative to the cursor, one of "after", "before", "on".
...	Arguments to be passed to png function.

**See Also**

Other functions for adding content: [body\\_add\\_blocks\(\)](#), [body\\_add\\_break\(\)](#), [body\\_add\\_caption\(\)](#), [body\\_add\\_docx\(\)](#), [body\\_add\\_fpar\(\)](#), [body\\_add\\_img\(\)](#), [body\\_add\\_par\(\)](#), [body\\_add\\_plot\(\)](#), [body\\_add\\_table\(\)](#), [body\\_add\\_toc\(\)](#)

**Examples**

```
if (require("ggplot2")) {
  doc <- read_docx()

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

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

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

---

body\_add\_img

*Add an image in a 'Word' document*


---

**Description**

add an image into an rdocx object.

**Usage**

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

**Arguments**

x	an rdocx object
src	image filename, the basename of the file must not contain any blank.
style	paragraph style
width	height in inches
height	height in inches
pos	where to add the new element relative to the cursor, one of "after", "before", "on".

**See Also**

Other functions for adding content: [body\\_add\\_blocks\(\)](#), [body\\_add\\_break\(\)](#), [body\\_add\\_caption\(\)](#), [body\\_add\\_docx\(\)](#), [body\\_add\\_fpar\(\)](#), [body\\_add\\_gg\(\)](#), [body\\_add\\_par\(\)](#), [body\\_add\\_plot\(\)](#), [body\\_add\\_table\(\)](#), [body\\_add\\_toc\(\)](#)

**Examples**

```
doc <- read_docx()

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

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

---

body_add_par	<i>Add paragraphs of text in a 'Word' document</i>
--------------	--

---

**Description**

add a paragraph of text into an rdocx object

**Usage**

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

**Arguments**

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

**See Also**

Other functions for adding content: [body\\_add\\_blocks\(\)](#), [body\\_add\\_break\(\)](#), [body\\_add\\_caption\(\)](#), [body\\_add\\_docx\(\)](#), [body\\_add\\_fpar\(\)](#), [body\\_add\\_gg\(\)](#), [body\\_add\\_img\(\)](#), [body\\_add\\_plot\(\)](#), [body\\_add\\_table\(\)](#), [body\\_add\\_toc\(\)](#)

**Examples**

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

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

---

body_add_plot	<i>Add plot in a 'Word' document</i>
---------------	--------------------------------------

---

**Description**

Add a plot as a png image into an rdocx object.

**Usage**

```
body_add_plot(
  x,
  value,
  width = 6,
  height = 5,
  res = 300,
  style = "Normal",
  pos = "after",
  ...
)
```

**Arguments**

x	an rdocx object
value	plot instructions, see <a href="#">plot_instr()</a> .
width	height in inches
height	height in inches
res	resolution of the png image in ppi
style	paragraph style
pos	where to add the new element relative to the cursor, one of "after", "before", "on".
...	Arguments to be passed to png function.

**See Also**

Other functions for adding content: [body\\_add\\_blocks\(\)](#), [body\\_add\\_break\(\)](#), [body\\_add\\_caption\(\)](#), [body\\_add\\_docx\(\)](#), [body\\_add\\_fpar\(\)](#), [body\\_add\\_gg\(\)](#), [body\\_add\\_img\(\)](#), [body\\_add\\_par\(\)](#), [body\\_add\\_table\(\)](#), [body\\_add\\_toc\(\)](#)

**Examples**

```
doc <- read_docx()

if (capabilities(what = "png")) {
  doc <- body_add_plot(doc,
    value = plot_instr(
      code = {
        barplot(1:5, col = 2:6)
      }
    ),
    style = "centered"
  )
}

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

---

body\_add\_table

*Add table in a 'Word' document*

---

**Description**

Add a table into an rdocx object.

**Usage**

```
body_add_table(
  x,
  value,
  style = NULL,
  pos = "after",
  header = TRUE,
  alignment = NULL,
  align_table = "center",
  stylenames = table_stylenames(),
  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).
align_table	table alignment within document, value must be "left", "center" or "right"
stylenames	columns styles defined by <a href="#">table_stylenames()</a>
first_row	Specifies that the first column conditional formatting should be applied. Details for this and other conditional formatting options can be found at <a href="http://officeopenxml.com/WPtblLook.php">http://officeopenxml.com/WPtblLook.php</a>
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\\_caption\(\)](#), [body\\_add\\_docx\(\)](#), [body\\_add\\_fpar\(\)](#), [body\\_add\\_gg\(\)](#), [body\\_add\\_img\(\)](#), [body\\_add\\_par\(\)](#), [body\\_add\\_plot\(\)](#), [body\\_add\\_toc\(\)](#)

**Examples**

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

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

---

body\_add\_toc

*Add table of content in a 'Word' document*


---

**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\\_caption\(\)](#), [body\\_add\\_docx\(\)](#), [body\\_add\\_fpar\(\)](#), [body\\_add\\_gg\(\)](#), [body\\_add\\_img\(\)](#), [body\\_add\\_par\(\)](#), [body\\_add\\_plot\(\)](#), [body\\_add\\_table\(\)](#)

**Examples**

```
doc <- read_docx()
doc <- body_add_toc(doc)

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

---

body_bookmark	<i>Add bookmark in a 'Word' document</i>
---------------	--

---

**Description**

Add a bookmark at the cursor location. The bookmark is added on the first run of text in the current paragraph.

**Usage**

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

**Arguments**

x	an rdocx object
id	bookmark name

**Examples**

```
# cursor_bookmark ----

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

---

body\_end\_block\_section  
*Add any section*

---

### Description

Add a section to the document. You can define any section with a [block\\_section](#) object. All other `body_end_section_*` are specialized, this one is highly flexible but it's up to the user to define the section properties.

### Usage

```
body_end_block_section(x, value)
```

### Arguments

x	an rdocx object
value	a <a href="#">block_section</a> object

### Illustrations

### See Also

Other functions for Word sections: [body\\_end\\_section\\_columns\\_landscape\(\)](#), [body\\_end\\_section\\_columns\(\)](#), [body\\_end\\_section\\_continuous\(\)](#), [body\\_end\\_section\\_landscape\(\)](#), [body\\_end\\_section\\_portrait\(\)](#), [body\\_set\\_default\\_section\(\)](#)

### Examples

```
library(officer)
str1 <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
str1 <- rep(str1, 20)
str1 <- paste(str1, collapse = " ")

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

doc_1 <- read_docx()
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")

doc_1 <- body_end_block_section(doc_1, block_section(ps))

doc_1 <- body_add_par(doc_1, value = str1, style = "centered")
```



```
print(doc_1, target = tempfile(fileext = ".docx"))
```

---

body\_end\_section\_columns

*Add multi columns section*

---

## Description

A section with multiple columns is added to the document.

You may prefer to use [body\\_end\\_block\\_section\(\)](#) that is more flexible.

## Usage

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

## Arguments

x	an rdocx object
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 Word sections: [body\\_end\\_block\\_section\(\)](#), [body\\_end\\_section\\_columns\\_landscape\(\)](#), [body\\_end\\_section\\_continuous\(\)](#), [body\\_end\\_section\\_landscape\(\)](#), [body\\_end\\_section\\_portrait\(\)](#), [body\\_set\\_default\\_section\(\)](#)

## Examples

```
str1 <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit."  
str1 <- rep(str1, 5)  
str1 <- paste(str1, collapse = " ")  
  
doc_1 <- read_docx()  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
doc_1 <- body_end_section_columns(doc_1)  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
print(doc_1, target = tempfile(fileext = ".docx"))
```

---

body\_end\_section\_columns\_landscape

*Add a landscape multi columns section*

---

## Description

A landscape section with multiple columns is added to the document.

## Usage

```
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
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.
w, h	page width, page height (in inches)

## See Also

Other functions for Word sections: [body\\_end\\_block\\_section\(\)](#), [body\\_end\\_section\\_columns\(\)](#), [body\\_end\\_section\\_continuous\(\)](#), [body\\_end\\_section\\_landscape\(\)](#), [body\\_end\\_section\\_portrait\(\)](#), [body\\_set\\_default\\_section\(\)](#)

## Examples

```
str1 <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit."  
str1 <- rep(str1, 5)  
str1 <- paste(str1, collapse = " ")  
  
doc_1 <- read_docx()  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
doc_1 <- body_end_section_columns_landscape(doc_1, widths = c(6, 2))  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
print(doc_1, target = tempfile(fileext = ".docx"))
```

---

body\_end\_section\_continuous  
*Add continuous section*

---

## Description

Section break starts the new section on the same page. This type of section break is often used to change the number of columns without starting a new page.

## Usage

```
body_end_section_continuous(x)
```

## Arguments

x                    an rdocx object

## See Also

Other functions for Word sections: [body\\_end\\_block\\_section\(\)](#), [body\\_end\\_section\\_columns\\_landscape\(\)](#), [body\\_end\\_section\\_columns\(\)](#), [body\\_end\\_section\\_landscape\(\)](#), [body\\_end\\_section\\_portrait\(\)](#), [body\\_set\\_default\\_section\(\)](#)

## Examples

```
str1 <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit."  
str1 <- rep(str1, 5)  
str1 <- paste(str1, collapse = " ")  
str2 <- "Aenean venenatis varius elit et fermentum vivamus vehicula."  
str2 <- rep(str2, 5)  
str2 <- paste(str2, collapse = " ")  
  
doc_1 <- read_docx()  
doc_1 <- body_add_par(doc_1, value = "Default section", style = "heading 1")  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
doc_1 <- body_add_par(doc_1, value = str2, style = "Normal")  
doc_1 <- body_end_section_continuous(doc_1)  
  
print(doc_1, target = tempfile(fileext = ".docx"))
```

---

body\_end\_section\_landscape  
*Add landscape section*

---

**Description**

A section with landscape orientation is added to the document.

**Usage**

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

**Arguments**

x	an rdocx object
w, h	page width, page height (in inches)

**See Also**

Other functions for Word sections: [body\\_end\\_block\\_section\(\)](#), [body\\_end\\_section\\_columns\\_landscape\(\)](#), [body\\_end\\_section\\_columns\(\)](#), [body\\_end\\_section\\_continuous\(\)](#), [body\\_end\\_section\\_portrait\(\)](#), [body\\_set\\_default\\_section\(\)](#)

**Examples**

```
str1 <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit."  
str1 <- rep(str1, 5)  
str1 <- paste(str1, collapse = " ")  
  
doc_1 <- read_docx()  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
doc_1 <- body_end_section_landscape(doc_1)  
  
print(doc_1, target = tempfile(fileext = ".docx"))
```

---

body\_end\_section\_portrait  
*Add portrait section*

---

**Description**

A section with portrait orientation is added to the document.

**Usage**

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

**Arguments**

x                    an rdocx object  
w, h                page width, page height (in inches)

**See Also**

Other functions for Word sections: [body\\_end\\_block\\_section\(\)](#), [body\\_end\\_section\\_columns\\_landscape\(\)](#), [body\\_end\\_section\\_columns\(\)](#), [body\\_end\\_section\\_continuous\(\)](#), [body\\_end\\_section\\_landscape\(\)](#), [body\\_set\\_default\\_section\(\)](#)

**Examples**

```
str1 <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit."  
str1 <- rep(str1, 5)  
str1 <- paste(str1, collapse = " ")  
  
doc_1 <- read_docx()  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
doc_1 <- body_end_section_portrait(doc_1)  
doc_1 <- body_add_par(doc_1, value = str1, style = "Normal")  
print(doc_1, target = tempfile(fileext = ".docx"))
```

---

body\_remove

*Remove an element in a 'Word' document*

---

**Description**

Remove element pointed by cursor from a 'Word' document.

**Usage**

```
body_remove(x)
```

**Arguments**

x                    an rdocx object

**Examples**

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

```

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

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

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

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

```

---

body\_replace\_all\_text *Replace text anywhere in the document*

---

## Description

Replace text anywhere in the document, or at a cursor.

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

```

body_replace_all_text(
  x,
  old_value,
  new_value,
  only_at_cursor = FALSE,
  warn = TRUE,

```

```

    ...
)

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

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

```

### Arguments

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

### header\_replace\_all\_text

Replacements will be performed in each header of all sections.

Replacements will be performed in each footer of all sections.

### Author(s)

Frank Hangler, <frank@plotandscatter.com>

### See Also

[grep](#), [regex](#), [docx\\_show\\_chunk](#)

**Examples**

```

doc <- read_docx()
doc <- body_add_par(doc, "Placeholder one")
doc <- body_add_par(doc, "Placeholder two")

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

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

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

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

```

---

body\_replace\_text\_at\_bkm

*Replace text at a bookmark location*

---

**Description**

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

**Usage**

```

body_replace_text_at_bkm(x, bookmark, value)

body_replace_img_at_bkm(x, bookmark, value)

headers_replace_text_at_bkm(x, bookmark, value)

headers_replace_img_at_bkm(x, bookmark, value)

footers_replace_text_at_bkm(x, bookmark, value)

footers_replace_img_at_bkm(x, bookmark, value)

```



**Arguments**

x	a docx device
bookmark	bookmark id
value	the replacement string, of type character

**Examples**

```

doc <- read_docx()
doc <- body_add_par(doc, "a paragraph to replace", style = "centered")
doc <- body_bookmark(doc, "text_to_replace")
doc <- body_replace_text_at_bkm(doc, "text_to_replace", "new text")

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

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

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

```

---

body\_set\_default\_section

*Define Default Section*

---

**Description**

Define default section of the document. You can define section properties (page size, orientation, ...) with a [prop\\_section](#) object.

**Usage**

```
body_set_default_section(x, value)
```

**Arguments**

x	an rdocx object
value	a <a href="#">prop_section</a> object

**Illustrations**

**See Also**

Other functions for Word sections: [body\\_end\\_block\\_section\(\)](#), [body\\_end\\_section\\_columns\\_landscape\(\)](#), [body\\_end\\_section\\_columns\(\)](#), [body\\_end\\_section\\_continuous\(\)](#), [body\\_end\\_section\\_landscape\(\)](#), [body\\_end\\_section\\_portrait\(\)](#)

**Examples**

```
default_sect_properties <- prop_section(
  page_size = page_size(orient = "landscape"), type = "continuous",
  page_margins = page_mar(bottom = .75, top = 1.5, right = 2, left = 2)
)

doc_1 <- read_docx()
doc_1 <- body_add_table(doc_1, value = mtcars[1:10, ], style = "table_template")
doc_1 <- body_add_par(doc_1, value = paste(rep(letters, 40), collapse = " "))
doc_1 <- body_set_default_section(doc_1, default_sect_properties)

print(doc_1, target = tempfile(fileext = ".docx"))
```

---

change\_styles

*Replace styles in a 'Word' Document*


---

**Description**

Replace styles with others in a 'Word' document. This function can be used for paragraph, run/character and table 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. Use <a href="#">styles_info()</a> to display available styles.

**Examples**

```
# creating a sample docx so that we can illustrate how
# to change styles
doc_1 <- read_docx()

doc_1 <- body_add_par(doc_1, "A title", style = "heading 1")
doc_1 <- body_add_par(doc_1, "Another title", style = "heading 2")
doc_1 <- body_add_par(doc_1, "Hello world!", style = "Normal")
file <- print(doc_1, target = tempfile(fileext = ".docx"))
```

```
# now we can illustrate how
# to change styles with `change_styles`
doc_2 <- read_docx(path = file)
mapstyles <- list(
  "centered" = c("Normal", "heading 2"),
  "strong" = "Default Paragraph Font"
)
doc_2 <- change_styles(doc_2, mapstyles = mapstyles)
print(doc_2, target = tempfile(fileext = ".docx"))
```

---

color_scheme	<i>Color scheme of a PowerPoint file</i>
--------------	--

---

### Description

Get the color scheme of a 'PowerPoint' master layout into a data.frame.

### Usage

```
color_scheme(x)
```

### Arguments

x                    an rpptx object

### See Also

Other functions for reading presentation informations: [annotate\\_base\(\)](#), [doc\\_properties\(\)](#), [layout\\_properties\(\)](#), [layout\\_summary\(\)](#), [length.rpptx\(\)](#), [plot\\_layout\\_properties\(\)](#), [slide\\_size\(\)](#), [slide\\_summary\(\)](#)

### Examples

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

---

cursor_begin	<i>Set cursor in a 'Word' document</i>
--------------	--

---

### Description

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

**Usage**

cursor\_begin(x)  
cursor\_bookmark(x, id)  
cursor\_end(x)  
cursor\_reach(x, keyword)  
cursor\_reach\_test(x, keyword)  
cursor\_forward(x)  
cursor\_backward(x)

**Arguments**

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

**cursor\_begin**

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

**cursor\_bookmark**

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

**cursor\_end**

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

**cursor\_reach**

Set the cursor on the first element of the document that contains text specified in argument keyword. The argument keyword is a regex pattern.

**cursor\_reach\_test**

Test if an expression has a match in the document that contains text specified in argument keyword. The argument keyword is a regex pattern.

**cursor\_forward**

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

**cursor\_backward**

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

**Examples**

```
library(officer)

# create a template ----
doc <- read_docx()
doc <- body_add_par(doc, "blah blah blah")
doc <- body_add_par(doc, "blah blah blah")
doc <- body_add_par(doc, "blah blah blah")
doc <- body_add_par(doc, "Hello text to replace")
doc <- body_add_par(doc, "blah blah blah")
doc <- body_add_par(doc, "blah blah blah")
doc <- body_add_par(doc, "blah blah blah")
doc <- body_add_par(doc, "Hello text to replace")
doc <- body_add_par(doc, "blah blah blah")
template_file <- print(
  x = doc,
  target = tempfile(fileext = ".docx")
)

# replace all pars containing "to replace" ----
doc <- read_docx(path = template_file)
while (cursor_reach_test(doc, "to replace")) {
  doc <- cursor_reach(doc, "to replace")

  doc <- body_add_fpar(
    x = doc,
    pos = "on",
    value = fpar(
      "Here is a link: ",
      hyperlink_ftext(
        text = "yopyop",
        href = "https://cran.r-project.org/"
      )
    )
  )
}

doc <- cursor_end(doc)
doc <- body_add_par(doc, "Yap yap yap yap...")

result_file <- print(
  x = doc,
  target = tempfile(fileext = ".docx")
)

# cursor_bookmark ----

doc <- read_docx()
```

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

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

---

docx\_bookmarks

*List Word bookmarks*

---

## Description

List bookmarks id that can be found in a 'Word' document.

## Usage

```
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

```
library(officer)

doc_1 <- read_docx()
doc_1 <- body_add_par(doc_1, "centered text", style = "centered")
doc_1 <- body_bookmark(doc_1, "text_to_replace_1")
doc_1 <- body_add_par(doc_1, "centered text", style = "centered")
doc_1 <- body_bookmark(doc_1, "text_to_replace_2")

docx_bookmarks(doc_1)

docx_bookmarks(read_docx())
```

---

docx_dim	<i>'Word' page layout</i>
----------	---------------------------

---

### Description

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

### Usage

```
docx_dim(x)
```

### Arguments

x                    an rdocx object

### See Also

Other functions for Word document informations: [doc\\_properties\(\)](#), [docx\\_bookmarks\(\)](#), [length.rdocx\(\)](#), [set\\_doc\\_properties\(\)](#), [styles\\_info\(\)](#)

### Examples

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

---

docx_set_character_style	<i>Add character style in a Word document</i>
--------------------------	---

---

### Description

The function lets you add or modify Word character styles.

### Usage

```
docx_set_character_style(  
  x,  
  style_id,  
  style_name,  
  base_on,  
  fp_t = fp_text_lite()  
)
```

**Arguments**

x	an rdocx object
style_id	a unique style identifier for Word.
style_name	a unique label associated with the style identifier. This label is the name of the style when Word edit the document.
base_on	the character style name used as base style
fp_t	Text formatting properties, see <a href="#">fp_text()</a> .

**Examples**

```
library(officer)
doc <- read_docx()

doc <- docx_set_character_style(
  doc,
  style_id = "newcharstyle",
  style_name = "label for char style",
  base_on = "Default Paragraph Font",
  fp_text_lite(
    shading.color = "red",
    color = "white")
)
paragraph <- fpar(
  run_wordtext("hello",
    style_id = "newcharstyle"))

doc <- body_add_fpar(doc, value = paragraph)
docx_file <- print(doc, target = tempfile(fileext = ".docx"))
docx_file
```

---

docx\_set\_paragraph\_style

*Add or replace paragraph style in a Word document*

---

**Description**

The function lets you add or replace a Word paragraph style.

**Usage**

```
docx_set_paragraph_style(
  x,
  style_id,
  style_name,
  base_on = "Normal",
  fp_p = fp_par(),
  fp_t = NULL
)
```



**Arguments**

x	an rdocx object
style_id	a unique style identifier for Word.
style_name	a unique label associated with the style identifier. This label is the name of the style when Word edit the document.
base_on	the style name used as base style
fp_p	paragraph formatting properties, see <a href="#">fp_par()</a> .
fp_t	default text formatting properties. This is used as text formatting properties, see <a href="#">fp_text()</a> . If NULL (default), the paragraph will used the default text formatting properties (defined by the base_on argument).

**Examples**

```
library(officer)

doc <- read_docx()

doc <- docx_set_paragraph_style(
  doc,
  style_id = "rightaligned",
  style_name = "Explicit label",
  fp_p = fp_par(text.align = "right", padding = 20),
  fp_t = fp_text_lite(
    bold = TRUE,
    shading.color = "#FD34F0",
    color = "white")
)

doc <- body_add_par(doc,
  value = "This is a test",
  style = "Explicit label")

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

---

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**

```
doc <- read_docx()
doc <- body_add_par(doc, "Placeholder one")
doc <- body_add_par(doc, "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 data.frame representing the document.

**Usage**

```
docx_summary(x)
```

**Arguments**

x                    an rdocx object

**Note**

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

**Examples**

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

---

doc_properties	<i>Read document properties</i>
----------------	---------------------------------

---

**Description**

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

**Usage**

```
doc_properties(x)
```

**Arguments**

x                    an rdocx or rpptx object

**Value**

a data.frame

**See Also**

Other functions for Word document informations: [docx\\_bookmarks\(\)](#), [docx\\_dim\(\)](#), [length.rdocx\(\)](#), [set\\_doc\\_properties\(\)](#), [styles\\_info\(\)](#)

Other functions for reading presentation informations: [annotate\\_base\(\)](#), [color\\_scheme\(\)](#), [layout\\_properties\(\)](#), [layout\\_summary\(\)](#), [length.rpptx\(\)](#), [plot\\_layout\\_properties\(\)](#), [slide\\_size\(\)](#), [slide\\_summary\(\)](#)

**Examples**

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

---

empty_content	<i>Empty block for 'PowerPoint'</i>
---------------	-------------------------------------

---

**Description**

Create 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 add some comments in this new content.

Empty content also works with layout fields (slide number and date) to preserve them: they are included on the slide and keep being updated by PowerPoint, i.e. update to the when the slide number when the slide moves in the deck, update to the date.

**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") )

doc <- add_slide(doc)
# add slide number as a computer field
doc <- ph_with(
  x = doc, value = empty_content(),
  location = ph_location_type(type = "sldNum"))

print(doc, target = fileout )
```

---

external\_img

*External image*

---

**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,
  unit = "in",
  guess_size = FALSE,
  alt = ""
)
```

**Arguments**

src	image file path
width, height	size of the image file. It can be ignored if parameter guess_size=TRUE, see parameter guess_size.
unit	unit for width and height, one of "in", "cm", "mm".

`guess_size` If package 'magick' is installed, this option can be used (set it to TRUE). The images will be read and width and height will be guessed.

`alt` alternative text for images

### usage

You can use this function in conjunction with [fpar](#) to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package `officetdown`.

### See Also

[ph\\_with](#), [body\\_add](#), [fpar](#)

Other run functions for reporting: [ftext\(\)](#), [hyperlink\\_ftext\(\)](#), [run\\_autonum\(\)](#), [run\\_bookmark\(\)](#), [run\\_columnbreak\(\)](#), [run\\_footnoteref\(\)](#), [run\\_footnote\(\)](#), [run\\_linebreak\(\)](#), [run\\_pagebreak\(\)](#), [run\\_reference\(\)](#), [run\\_word\\_field\(\)](#), [run\\_wordtext\(\)](#)

### 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 <- fp_text(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"))
```

---

fpar

*Formatted paragraph*

---

### Description

Create a paragraph representation by concatenating formatted text or images. The result can be inserted in a Word document or a PowerPoint presentation and can also be inserted in a [block\\_list\(\)](#) call.

All its arguments will be concatenated to create a paragraph where chunks of text and images are associated with formatting properties.

fpar supports `ftext()`, `external_img()`, `run_*` functions (i.e. `run_autonum()`, `run_word_field()`) when output is Word, and simple strings.

Default text and paragraph formatting properties can also be modified with function `update()`.

### Usage

```
fpar(..., fp_p = fp_par(), fp_t = fp_text_lite(), values = NULL)
```

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

### Arguments

<code>...</code>	cot objects ( <code>f<sub>text</sub>()</code> , <code>external_img()</code> )
<code>fp_p</code>	paragraph formatting properties, see <code>fp_par()</code>
<code>fp_t</code>	default text formatting properties. This is used as text formatting properties when simple text is provided as argument, see <code>fp_text()</code> .
<code>values</code>	a list of cot objects. If provided, argument <code>...</code> will be ignored.
<code>object</code>	fpar object

### See Also

`block_list()`, `body_add_fpar()`, `ph_with()`

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

### Examples

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

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

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

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

---

fp_border	<i>Border properties object</i>
-----------	---------------------------------

---

**Description**

create a border properties object.

**Usage**

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

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

**Arguments**

color	border color - single character value (e.g. "#000000" or "black")
style	border style - single character value : "none" or "solid" or "dotted" or "dashed"
width	border width - an integer value : 0>= value
object	fp_border object
...	further arguments - not used

**See Also**

Other functions for defining formatting properties: [fp\\_cell\(\)](#), [fp\\_par\(\)](#), [fp\\_text\(\)](#)

**Examples**

```
fp_border()  
fp_border(color="orange", style="solid", width=1)  
fp_border(color="gray", style="dotted", width=1)  
  
# modify object -----  
border <- fp_border()  
update(border, style="dotted", width=3)
```

---

fp_cell	<i>Cell formatting properties</i>
---------	-----------------------------------

---

**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 = "lrbt",  
  rowspan = 1,  
  colspan = 1  
)  
  
## S3 method for class 'fp_cell'  
format(x, type = "wml", ...)  
  
## S3 method for class 'fp_cell'  
print(x, ...)  
  
## S3 method for class 'fp_cell'  
update(  
  object,  
  border,  
  border.bottom,  
  border.left,  
  border.top,  
  border.right,  
  vertical.align,  
  margin = 0,  
  margin.bottom,  
  margin.top,  
  margin.left,  
  margin.right,  
  background.color,  
  text.direction,  
  rowspan = 1,  
  colspan = 1,  
  ...  
)
```



**Arguments**

border	shortcut for all borders.
border.bottom, border.left, border.top, border.right	<a href="#">fp_border</a> 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 "lrb", "tblr", "btlr".
rowspan	specify how many rows the cell is spanned over
colspan	specify how many columns the cell is spanned over
x, object	fp_cell object
type	output type - one of 'wml', 'pml', 'html', 'rtf'.
...	further arguments - not used

**See Also**

Other functions for defining formatting properties: [fp\\_border\(\)](#), [fp\\_par\(\)](#), [fp\\_text\(\)](#)

**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,
  line_spacing = 1,
  border = fp_border(width = 0),
  padding.bottom,
```

```

padding.top,
padding.left,
padding.right,
border.bottom,
border.left,
border.top,
border.right,
shading.color = "transparent",
keep_with_next = FALSE,
word_style = "Normal"
)

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

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

```

### Arguments

<code>text.align</code>	text alignment - a single character value, expected value is one of 'left', 'right', 'center', 'justify'.
<code>padding</code>	paragraph paddings - 0 or positive integer value. Argument padding overwrites arguments padding.bottom, padding.top, padding.left, padding.right.
<code>line_spacing</code>	line spacing, 1 is single line spacing, 2 is double line spacing.
<code>border</code>	shortcut for all borders.
<code>padding.bottom</code> , <code>padding.top</code> , <code>padding.left</code> , <code>padding.right</code>	paragraph paddings - 0 or positive integer value.
<code>border.bottom</code> , <code>border.left</code> , <code>border.top</code> , <code>border.right</code>	<code>fp_border</code> 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.
word_style	Word paragraph style name
x, object	fp_par object
...	further arguments - not used

**Value**

a fp\_par object

**See Also**

[fpar](#)

Other functions for defining formatting properties: [fp\\_border\(\)](#), [fp\\_cell\(\)](#), [fp\\_text\(\)](#)

**Examples**

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

---

fp_text	<i>Text formatting properties</i>
---------	-----------------------------------

---

**Description**

Create a fp\_text object that describes text formatting properties.

Function fp\_text\_lite() is generating properties with only entries for the parameters users provided. The undefined properties will inherit from the default settings.

**Usage**

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

```
)

fp_text_lite(
  color = NA,
  font.size = NA,
  font.family = NA,
  cs.family = NA,
  eastasia.family = NA,
  hanshi.family = NA,
  bold = NA,
  italic = NA,
  underlined = NA,
  vertical.align = "baseline",
  shading.color = NA
)

## 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,
  italic,
  underlined,
  font.family,
  cs.family,
  eastasia.family,
  hanshi.family,
  vertical.align,
  shading.color,
  ...
)
```

### Arguments

color	font color - a single character value specifying a valid color (e.g. "#000000" or "black").
font.size	font size (in point) - 0 or positive integer value.
bold	is bold
italic	is italic
underlined	is underlined

font.family	single character value. Specifies the font to be used to format characters in the Unicode range (U+0000-U+007F).
cs.family	optional font to be used to format characters in a complex script Unicode range. For example, Arabic text might be displayed using the "Arial Unicode MS" font.
eastasia.family	optional font to be used to format characters in an East Asian Unicode range. For example, Japanese text might be displayed using the "MS Mincho" font.
hansi.family	optional. Specifies the font to be used to format characters in a Unicode range which does not fall into one of the other categories.
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', 'rtf'.
...	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](#)

Other functions for defining formatting properties: [fp\\_border\(\)](#), [fp\\_cell\(\)](#), [fp\\_par\(\)](#)

**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 the way you want.

**Usage**

```
ftext(text, prop = NULL)
```

**Arguments**

text	text value, a single character value
prop	formatting text properties returned by <a href="#">fp_text</a> . It also can be NULL in which case, no formatting is defined (the default is applied).

**usage**

You can use this function in conjunction with [fpar](#) to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package `officetdown`.

**See Also**

[fp\\_text](#)

Other run functions for reporting: [external\\_img\(\)](#), [hyperlink\\_ftext\(\)](#), [run\\_autonum\(\)](#), [run\\_bookmark\(\)](#), [run\\_columnbreak\(\)](#), [run\\_footnoteref\(\)](#), [run\\_footnote\(\)](#), [run\\_linebreak\(\)](#), [run\\_pagebreak\(\)](#), [run\\_reference\(\)](#), [run\\_word\\_field\(\)](#), [run\\_wordtext\(\)](#)

**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"))
```

---

hyperlink\_ftext

*Formatted chunk of text with hyperlink*

---

**Description**

Format a chunk of text with text formatting properties (bold, color, ...), the chunk is associated with an hyperlink.

**Usage**

```
hyperlink_ftext(text, prop = NULL, href)
```

**Arguments**

text	text value, a single character value
prop	formatting text properties returned by <code>fp_text</code> . It also can be NULL in which case, no formatting is defined (the default is applied).
href	URL value

**usage**

You can use this function in conjunction with `fpar` to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package `officedown`.

**See Also**

Other run functions for reporting: `external_img()`, `ftext()`, `run_autonum()`, `run_bookmark()`, `run_columnbreak()`, `run_footnoteref()`, `run_footnote()`, `run_linebreak()`, `run_pagebreak()`, `run_reference()`, `run_word_field()`, `run_wordtext()`

**Examples**

```
ft <- fp_text(font.size = 12, bold = TRUE)
hyperlink_ftext(
  href = "https://cran.r-project.org/index.html",
  text = "some text", prop = ft)
```

---

layout\_properties      *Slide layout properties*

---

**Description**

Get information about a particular slide layout into a data.frame.

**Usage**

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

**Arguments**

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

**See Also**

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

**Examples**

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

---

layout_summary	<i>Presentation layouts summary</i>
----------------	-------------------------------------

---

**Description**

Get informations about slide layouts and master layouts into a data.frame. This function returns a data.frame containing all layout and master names.

**Usage**

```
layout_summary(x)
```

**Arguments**

x                    an rpptx object

**See Also**

Other functions for reading presentation informations: [annotate\\_base\(\)](#), [color\\_scheme\(\)](#), [doc\\_properties\(\)](#), [layout\\_properties\(\)](#), [length.rpptx\(\)](#), [plot\\_layout\\_properties\(\)](#), [slide\\_size\(\)](#), [slide\\_summary\(\)](#)

**Examples**

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

---

length.rdocx	<i>Number of blocks inside an rdocx object</i>
--------------	--

---

**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 invisible element.

**Usage**

```
## 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**

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

---

length.rpptx	<i>Number of slides</i>
--------------	-------------------------

---

**Description**

Function length will return the number of slides.

**Usage**

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

**Arguments**

x                    an rpptx object

**See Also**

Other functions for reading presentation informations: [annotate\\_base\(\)](#), [color\\_scheme\(\)](#), [doc\\_properties\(\)](#), [layout\\_properties\(\)](#), [layout\\_summary\(\)](#), [plot\\_layout\\_properties\(\)](#), [slide\\_size\(\)](#), [slide\\_summary\(\)](#)

**Examples**

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

---

media_extract	<i>Extract media from a document object</i>
---------------	---

---

**Description**

Extract files from a rpptx object.

**Usage**

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

**Arguments**

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

**Examples**

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

---

move_slide	<i>Move a slide</i>
------------	---------------------

---

**Description**

Move a slide in a pptx presentation.

**Usage**

```
move_slide(x, index = NULL, 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**

[read\\_pptx\(\)](#)

Other functions slide manipulation: [add\\_slide\(\)](#), [on\\_slide\(\)](#), [remove\\_slide\(\)](#), [set\\_notes\(\)](#)

**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)
```

---

notes\_location\_label    *Location of a named placeholder for notes*

---

**Description**

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

**Usage**

```
notes_location_label(ph_label, ...)
```

**Arguments**

ph_label	placeholder label of the used notes master
...	unused arguments

---

notes\_location\_type    *Location of a placeholder for notes*

---

**Description**

The function will use the type name of the placeholder (e.g. body, hdr), to find the corresponding location.

**Usage**

```
notes_location_type(type = "body", ...)
```

**Arguments**

type	placeholder label of the used notes master
...	unused arguments

---

officer

---

*Manipulate Microsoft Word and PowerPoint Documents with 'officer'*


---

**Description**

The officer package facilitates access to and manipulation of 'Microsoft Word' and 'Microsoft PowerPoint' documents from R. It also supports the writing of 'RTF' documents.

Examples of usage are:

- Create Word documents with tables, titles, TOC and graphics
- Importation of Word and PowerPoint files into data objects
- Write updated content back to a PowerPoint presentation
- Clinical reporting automation
- Production of reports from a shiny application

To start with officer, read about [read\\_docx\(\)](#), [read\\_pptx\(\)](#) or [rtf\\_doc\(\)](#).

The package is also providing several objects that can be printed in 'R Markdown' documents for advanced Word or PowerPoint reporting as [run\\_autonum\(\)](#) and [block\\_caption\(\)](#).

**Author(s)**

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

Other contributors:

- ArData [copyright holder]
- 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> (function `annotate_base`) [contributor]
- John Muschelli <muschelli2@gmail.com> (google doc compatibility) [contributor]
- Bill Denney <wdenney@humanpredictions.com> ([ORCID](#)) (function `as.matrix.rpptx`) [contributor]
- Nikolai Beck <beck.nikolai@gmail.com> (set speaker notes for .pptx documents) [contributor]
- Stefan Moog <moogs@gmx.de> (added functionality to set shape geometry and outline) [contributor]
- Greg Leleu <gregoire.leleu@gmail.com> (fields functionality in ppt) [contributor]
- Hongyuan Jia <hongyuanjia@cqust.edu.cn> ([ORCID](#)) [contributor]

**See Also**

The user documentation: <https://ardata-fr.github.io/officeverse/> and manuals <https://davidgohel.github.io/officer/>

---

officer-defunct

*Defunct Functions in Package officer*

---

**Description**

Defunct Functions in Package officer

**Usage**

`slip_in_seqfield(...)`

`slip_in_column_break(...)`

`slip_in_xml(...)`

`slip_in_text(...)`

`slip_in_footnote(...)`

**Arguments**

... unused arguments

**Details**

`slip_in_seqfield()` is replaced by `run_word_field()`.

`slip_in_column_break()` is replaced by `run_columnbreak()`.

`slip_in_xml()` is replaced by `fpar()`.

`slip_in_text()` is replaced by `fpar()`.

`slip_in_footnote()` is replaced by `run_footnote()`.

---

on_slide	<i>Change current slide</i>
----------	-----------------------------

---

### Description

Change current slide index of an rpptx object.

### Usage

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

### Arguments

x	an rpptx object
index	slide index

### See Also

[read\\_pptx\(\)](#), [ph\\_with\(\)](#)

Other functions slide manipulation: [add\\_slide\(\)](#), [move\\_slide\(\)](#), [remove\\_slide\(\)](#), [set\\_notes\(\)](#)

### 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)
```

---

page\_mar

*Page margins object*


---

### Description

The margins for each page of a sectionThe 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

```
page_mar()
```

---

page_size	<i>Page size object</i>
-----------	-------------------------

---

### 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

```
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\(\)](#)

### Examples

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

---

ph_hyperlink	<i>Hyperlink a placeholder</i>
--------------	--------------------------------

---

### Description

Add hyperlink to a placeholder in the current slide.

### Usage

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

### Arguments

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



id_chr	deprecated.
ph_label	label associated to the placeholder. Use column ph_label of result returned by <a href="#">slide_summary</a> .
href	hyperlink (do not forget http or https prefix)

**See Also**[ph\\_with](#)Other functions for placeholders manipulation: [ph\\_remove\(\)](#), [ph\\_slidelink\(\)](#)**Examples**

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

print(doc, target = fileout )
```

---

ph_location	<i>Location for a placeholder from scratch</i>
-------------	--

---

**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,
  ln = NULL,
  geom = NULL,
  ...
)
```

**Arguments**

left, top, width, height	placeholder coordinates in inches.
newlabel	a label for the placeholder. See section details.
bg	background color
rotation	rotation angle
ln	a <code>sp_line()</code> object specifying the outline style.
geom	shape geometry, see <a href="http://www.datypic.com/sc/ooxml/t-a_ST_ShapeType.html">http://www.datypic.com/sc/ooxml/t-a_ST_ShapeType.html</a>
...	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") )

# Set geometry and outline
doc <- read_pptx()
doc <- add_slide(doc)
loc <- ph_location(left = 1, top = 1, width = 4, height = 3, bg = "steelblue",
  ln = sp_line(color = "red", lwd = 2.5),
  geom = "trapezoid")
doc <- ph_with(doc, "", loc = loc)
print(doc, target = tempfile(fileext = ".pptx") )
```

---

ph\_location\_fullsize    *Location of a full size element*

---

**Description**

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

**Usage**

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

**Arguments**

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

**See Also**

Other functions for placeholder location: [ph\\_location\\_label\(\)](#), [ph\\_location\\_left\(\)](#), [ph\\_location\\_right\(\)](#), [ph\\_location\\_template\(\)](#), [ph\\_location\\_type\(\)](#), [ph\\_location\(\)](#)

**Examples**

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

---

ph\_location\_label    *Location of a named placeholder*

---

**Description**

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

**Usage**

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

**Arguments**

ph_label	placeholder label of the used layout. It can be read in PowerPoint or with function <code>layout_properties()</code> in column <code>ph_label</code> .
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

```
# ph_location_label demo ----

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

# all ph_label can be read here
layout_properties(doc, layout = "Title and Content")

doc <- ph_with(doc, head(iris),
  location = ph_location_label(ph_label = "Content Placeholder 2") )
doc <- ph_with(doc, format(Sys.Date()),
  location = ph_location_label(ph_label = "Date Placeholder 3") )
doc <- ph_with(doc, "This is a title",
  location = ph_location_label(ph_label = "Title 1") )

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

---

ph_location_left	<i>Location of a left body element</i>
------------------	--

---

## Description

The function will return the location corresponding to a left bounding box. The function assume the layout 'Two Content' is existing. This is an helper function, if you don't have a layout named 'Two Content', use [ph\\_location\\_type\(\)](#) and set arguments to your specific needs.

**Usage**

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

**Arguments**

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

**See Also**

Other functions for placeholder location: [ph\\_location\\_fullsize\(\)](#), [ph\\_location\\_label\(\)](#), [ph\\_location\\_right\(\)](#), [ph\\_location\\_template\(\)](#), [ph\\_location\\_type\(\)](#), [ph\\_location\(\)](#)

**Examples**

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

---

ph\_location\_right      *Location of a right body element*

---

**Description**

The function will return the location corresponding to a right bounding box. The function assume the layout 'Two Content' is existing. This is an helper function, if you don't have a layout named 'Two Content', use [ph\\_location\\_type\(\)](#) and set arguments to your specific needs.

**Usage**

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

**Arguments**

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

**See Also**

Other functions for placeholder location: [ph\\_location\\_fullsize\(\)](#), [ph\\_location\\_label\(\)](#), [ph\\_location\\_left\(\)](#), [ph\\_location\\_template\(\)](#), [ph\\_location\\_type\(\)](#), [ph\\_location\(\)](#)

**Examples**

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

---

ph\_location\_template    *Location for a placeholder based on a template*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**left** left coordinate of the bounding box

**top** top coordinate of the bounding box

**width** width of the bounding box

**height** height of the bounding box

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

### See Also

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

### Examples

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

---

ph_location_type	<i>Location of a placeholder based on a type</i>
------------------	--

---

### Description

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

### Usage

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

**Arguments**

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

**Details**

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

**left** left coordinate of the bounding box

**top** top coordinate of the bounding box

**width** width of the bounding box

**height** height of the bounding box

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

**See Also**

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

**Examples**

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

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

```

---

ph\_remove

*Remove a shape*

---

### Description

Remove a shape in a slide.

### Usage

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

### Arguments

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

### See Also

[ph\\_with](#)

Other functions for placeholders manipulation: [ph\\_hyperlink\(\)](#), [ph\\_slidelink\(\)](#)

**Examples**

```

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

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

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

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

print(doc, target = fileout )

```

---

ph\_slidelink

*Slide link to a placeholder*

---

**Description**

Add slide link to a placeholder in the current slide.

**Usage**

```

ph_slidelink(
  x,
  type = "body",
  id = 1,
  id_chr = NULL,
  ph_label = NULL,
  slide_index
)

```

**Arguments**

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

**See Also**

[ph\\_with](#)

Other functions for placeholders manipulation: [ph\\_hyperlink\(\)](#), [ph\\_remove\(\)](#)

**Examples**

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

print(doc, target = fileout )
```

---

ph\_with

*Add objects 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. See section **Methods (by class)** to see supported outputs.

**Usage**

```
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, alt_text, scale = 1, ...)  
  
## 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 <a href="#">table_conditional_formatting()</a>
alignment	alignment for each columns, 'l' for left, 'r' for right and 'c' for center. Default to NULL.
res	resolution of the png image in ppi
alt_text	Alt-text for screen-readers
scale	Multiplicative scaling factor, same as in ggsave
use_loc_size	if set to FALSE, external_img width and height will be used.

### Methods (by class)

- `ph_with(character)`: add a character vector to a new shape on the current slide, values will be added as paragraphs.
- `ph_with(numeric)`: add a numeric vector to a new shape on the current slide, values will be first formatted then added as paragraphs.
- `ph_with(factor)`: add a factor vector to a new shape on the current slide, values will be converted as character and then added as paragraphs.
- `ph_with(block_list)`: add a [block\\_list](#) made of [fpar](#) to a new shape on the current slide.
- `ph_with(unordered_list)`: add a [unordered\\_list](#) made of [fpar](#) to a new shape on the current slide.
- `ph_with(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.
- `ph_with(gg)`: add a ggplot object to a new shape on the current slide. Use package rvg for more advanced graphical features.
- `ph_with(plot_instr)`: add an R plot to a new shape on the current slide. Use package rvg for more advanced graphical features.

- `ph_with(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`.
- `ph_with(fpar)`: add an `fpar` to a new shape on the current slide as a single paragraph in a `block_list`.
- `ph_with(empty_content)`: add an `empty_content` to a new shape on the current slide.
- `ph_with(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.

## Illustrations

## See Also

[ph\\_location\\_type](#), [ph\\_location](#), [ph\\_location\\_label](#), [ph\\_location\\_left](#), [ph\\_location\\_right](#), [ph\\_location\\_fullsize](#), [ph\\_location\\_template](#)

## Examples

```
# 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(
    ftext("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 -----
fpt <- fp_text(
  bold = TRUE, font.family = "Bradley Hand",
  font.size = 150, color = "#F5595B"
)
hw <- fpar(
  ftext("hello ", fpt),
  hyperlink_ftext(
    href = "https://cran.r-project.org/index.html",
    text = "cran", prop = fpt
  )
)
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)

```

---

plot\_instr

*Wrap plot instructions for png plotting in Powerpoint or Word*


---

## 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\(\)](#) and [body\\_add\\_plot\(\)](#).

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\\_plot\(\)](#)

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()
doc <- body_add(doc, anyplot, width = 5, height = 4)
print(doc, target = tempfile(fileext = ".docx"))

doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(
  doc, anyplot,
  location = ph_location_fullsize(),
  bg = "#00000066", pointsize = 12)
print(doc, target = tempfile(fileext = ".pptx"))

```

plot\_layout\_properties

*Slide layout properties plot*

---

### 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\(\)](#), [doc\\_properties\(\)](#), [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

*PowerPoint content in a data.frame*

---

### Description

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

### Usage

```
pptx_summary(x)
```

**Arguments**

x                    an rpptx object

**Examples**

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

---

print.rpptx	<i>Write a 'PowerPoint' file.</i>
-------------	-----------------------------------

---

**Description**

Write a 'PowerPoint' file with an object of class 'rpptx' (created with [read\\_pptx\(\)](#)).

**Usage**

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

**Arguments**

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

**See Also**

[read\\_pptx](#)

**Examples**

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

---

print.rtf	<i>Write an 'RTF' document to a file</i>
-----------	--

---

### Description

Write the RTF object and its content to a file.

### Usage

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

### Arguments

x	an 'rtf' object created with <a href="#">rtf_doc()</a>
target	path to the RTF file to write
...	unused

### See Also

[rtf\\_doc\(\)](#)

### Examples

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

---

prop_section	<i>Section properties</i>
--------------	---------------------------

---

### 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**

```
prop_section(
  page_size = NULL,
  page_margins = NULL,
  type = NULL,
  section_columns = NULL,
  header_default = NULL,
  header_even = NULL,
  header_first = NULL,
  footer_default = NULL,
  footer_even = NULL,
  footer_first = NULL
)
```

**Arguments**

page_size	page dimensions, an object generated with function <a href="#">page_size</a> .
page_margins	page margins, an object generated with function <a href="#">page_mar</a> .
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 <a href="#">section_columns</a> . Use NULL (default value) for no content.
header_default	content as a <a href="#">block_list()</a> for the default page header. Use NULL (default value) for no content.
header_even	content as a <a href="#">block_list()</a> for the even page header. Use NULL (default value) for no content.
header_first	content as a <a href="#">block_list()</a> for the first page header. Use NULL (default value) for no content.
footer_default	content as a <a href="#">block_list()</a> for the default page footer. Use NULL (default value) for no content.
footer_even	content as a <a href="#">block_list()</a> for the even page footer. Use NULL (default value) for no content.
footer_first	content as a <a href="#">block_list()</a> for the default page footer. Use NULL (default value) for no content.

**Illustrations****See Also**

[block\\_section](#)

Other functions for section definition: [page\\_mar\(\)](#), [page\\_size\(\)](#), [section\\_columns\(\)](#)

**Examples**

```

library(officer)

landscape_one_column <- block_section(
  prop_section(
    page_size = page_size(orient = "landscape"), type = "continuous"
  )
)
landscape_two_columns <- block_section(
  prop_section(
    page_size = page_size(orient = "landscape"), type = "continuous",
    section_columns = section_columns(widths = c(4.75, 4.75))
  )
)

doc_1 <- read_docx()
# there starts section with landscape_one_column
doc_1 <- body_add_table(doc_1, value = mtcars[1:10,], style = "table_template")
doc_1 <- body_end_block_section(doc_1, value = landscape_one_column)
# there stops section with landscape_one_column

# there starts section with landscape_two_columns
doc_1 <- body_add_par(doc_1, value = paste(rep(letters, 50), collapse = " "))
doc_1 <- body_end_block_section(doc_1, value = landscape_two_columns)
# there stops section with landscape_two_columns

doc_1 <- body_add_table(doc_1, value = mtcars[1:25,], style = "table_template")

print(doc_1, target = tempfile(fileext = ".docx"))

# an example with headers and footers -----
txt_lorem <- rep(
  "Purus lectus eros metus turpis mattis platea praesent sed. ",
  50)
txt_lorem <- paste0(txt_lorem, collapse = "")

header_first <- block_list(fpar(ftext("text for first page header")))
header_even <- block_list(fpar(ftext("text for even page header")))
header_default <- block_list(fpar(ftext("text for default page header")))
footer_first <- block_list(fpar(ftext("text for first page footer")))
footer_even <- block_list(fpar(ftext("text for even page footer")))
footer_default <- block_list(fpar(ftext("text for default page footer")))

ps <- prop_section(
  header_default = header_default, footer_default = footer_default,
  header_first = header_first, footer_first = footer_first,
  header_even = header_even, footer_even = footer_even
)
x <- read_docx()
for (i in 1:20) {

```

```

    x <- body_add_par(x, value = txt_lorem)
  }
  x <- body_set_default_section(
    x,
    value = ps
  )
  print(x, target = tempfile(fileext = ".docx"))

```

---

prop\_table

*Table properties*


---

### Description

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

### Usage

```

prop_table(
  style = NA_character_,
  layout = table_layout(),
  width = table_width(),
  stylenames = table_stylenames(),
  colwidths = table_colwidths(),
  tcf = table_conditional_formatting(),
  align = "center",
  word_title = NULL,
  word_description = NULL
)

```

### Arguments

style	table style to be used to format table
layout	layout defined by <a href="#">table_layout()</a> ,
width	table width in the document defined by <a href="#">table_width()</a>
stylenames	columns styles defined by <a href="#">table_stylenames()</a>
colwidths	column widths defined by <a href="#">table_colwidths()</a>
tcf	conditional formatting settings defined by <a href="#">table_conditional_formatting()</a>
align	table alignment (one of left, center or right)
word_title	alternative text for Word table (used as title of the table)
word_description	alternative text for Word table (used as description of the table)

### See Also

Other functions for table definition: [table\\_colwidths\(\)](#), [table\\_conditional\\_formatting\(\)](#), [table\\_layout\(\)](#), [table\\_stylenames\(\)](#), [table\\_width\(\)](#)

**Examples**

```
prop_table()
to_wml(prop_table())
```

---

read_docx	<i>Create a 'Word' document object</i>
-----------	--

---

**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. dotx file are supported.
x	an rdocx object
target	path to the docx file to write
...	unused

**Value**

an object of class rdocx.

**Functions**

- `print(rdocx)`: write docx to a file. It returns the path of the result file.

**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 also use the styles from the original document.

See `body_add_*` functions to add content.

**Illustrations**



**See Also**

[body\\_add\\_par](#), [body\\_add\\_plot](#), [body\\_add\\_table](#)

**Examples**

```
library(officer)

pinst <- plot_instr({
  z <- c(rnorm(100), rnorm(50, mean = 5))
  plot(density(z))
})

doc_1 <- read_docx()
doc_1 <- body_add_par(doc_1, "This is a table", style = "heading 2")
doc_1 <- body_add_table(doc_1, value = mtcars, style = "table_template")
doc_1 <- body_add_par(doc_1, "This is a plot", style = "heading 2")
doc_1 <- body_add_plot(doc_1, pinst)
docx_file_1 <- print(doc_1, target = tempfile(fileext = ".docx"))

template <- system.file(package = "officer",
  "doc_examples", "landscape.docx")
doc_2 <- read_docx(path = template)
doc_2 <- body_add_par(doc_2, "This is a table", style = "heading 2")
doc_2 <- body_add_table(doc_2, value = mtcars)
doc_2 <- body_add_par(doc_2, "This is a plot", style = "heading 2")
doc_2 <- body_add_plot(doc_2, pinst)
docx_file_2 <- print(doc_2, target = tempfile(fileext = ".docx"))
```

---

read\_pptx

*Create a 'PowerPoint' document object*

---

**Description**

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

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

**Usage**

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

**Arguments**

`path` path to the pptx file to use as base document. potx file are supported.

**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\(\)](#), [plot\\_layout\\_properties\(\)](#), [ph\\_with\(\)](#)

**Examples**

```
read_pptx()
```

---

<code>read_xlsx</code>	<i>Create an 'Excel' document object</i>
------------------------	--

---

**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**

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

**Examples**

```
read_xlsx()  
x <- read_xlsx()  
print(x, target = tempfile(fileext = ".xlsx"))
```

---

remove_slide	<i>Remove a slide</i>
--------------	-----------------------

---

**Description**

Remove a slide from a pptx presentation.

**Usage**

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

**Arguments**

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

**Note**

cursor is set on the last slide.

**See Also**

[read\\_pptx\(\)](#), [ph\\_with\(\)](#), [ph\\_remove\(\)](#)

Other functions slide manipulation: [add\\_slide\(\)](#), [move\\_slide\(\)](#), [on\\_slide\(\)](#), [set\\_notes\(\)](#)

**Examples**

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

---

`rtf_add`*Add content into an RTF document*

---

**Description**

This function add 'officer' objects into an RTF document. Values are added as new paragraphs. See section 'Methods (by class)' that list supported objects.

**Usage**

```
rtf_add(x, value, ...)

## S3 method for class 'block_section'
rtf_add(x, value, ...)

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

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

## S3 method for class 'double'
rtf_add(x, value, formatter = formatC, ...)

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

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

## S3 method for class 'gg'
rtf_add(
  x,
  value,
  width = 6,
  height = 5,
  res = 300,
  scale = 1,
  ppr = fp_par(text.align = "center"),
  ...
)

## S3 method for class 'plot_instr'
rtf_add(
  x,
  value,
  width = 6,
```

```

    height = 5,
    res = 300,
    scale = 1,
    ppr = fp_par(text.align = "center"),
    ...
  )

```

### Arguments

x	rtf object, created by <code>rtf_doc()</code> .
value	object to add in the document. Supported objects are vectors, graphics, block of formatted paragraphs. Use package 'flextable' to add tables.
...	further arguments passed to or from other methods. When adding a ggplot object or <code>plot_instr</code> , these arguments will be used by <code>png</code> function. See section 'Methods' to see what arguments can be used.
formatter	function used to format the numerical values
width	height in inches
height	height in inches
res	resolution of the png image in ppi
scale	Multiplicative scaling factor, same as in <code>ggsave</code>
ppr	<code>fp_par()</code> to apply to paragraph.

### Methods (by class)

- `rtf_add(block_section)`: add a new section definition
- `rtf_add(character)`: add characters as new paragraphs
- `rtf_add(factor)`: add a factor vector as new paragraphs
- `rtf_add(double)`: add a double vector as new paragraphs
- `rtf_add(fpar)`: add an `fpar()`
- `rtf_add(block_list)`: add an `block_list()`
- `rtf_add(gg)`: add a `ggplot2`
- `rtf_add(plot_instr)`: add a `plot_instr()` object

### Examples

```

library(officer)

def_text <- fp_text_lite(color = "#006699", bold = TRUE)
center_par <- fp_par(text.align = "center", padding = 3)

doc <- rtf_doc(
  normal_par = fp_par(line_spacing = 1.4, padding = 3)
)

doc <- rtf_add(

```

```

x = doc,
value = fpar(
  ftext("how are you?", prop = def_text),
  fp_p = fp_par(text.align = "center")
)
)

a_paragraph <- fpar(
  ftext("Here is a date: ", prop = def_text),
  run_word_field(field = "Date \@ \\"MMMM d yyyy\\""),
  fp_p = center_par
)
doc <- rtf_add(
  x = doc,
  value = block_list(
    a_paragraph,
    a_paragraph,
    a_paragraph
  )
)

if (require("ggplot2")) {
  gg <- gg_plot <- ggplot(data = iris) +
    geom_point(mapping = aes(Sepal.Length, Petal.Length))
  doc <- rtf_add(doc, gg,
    width = 3, height = 4,
    ppr = center_par
  )
}
anyplot <- plot_instr(code = {
  barplot(1:5, col = 2:6)
})
doc <- rtf_add(doc, anyplot,
  width = 5, height = 4,
  ppr = center_par
)

print(doc, target = tempfile(fileext = ".rtf"))

```

---

rtf\_doc

*Create an RTF document object*


---

## Description

Creation of the object representing an RTF document which can then receive contents with the `rtf_add()` function and be written to a file with the `print(x, target="doc.rtf")` function.

## Usage

```
rtf_doc(
```

```

def_sec = prop_section(),
normal_par = fp_par(),
normal_chunk = fp_text(font.family = "Arial", font.size = 11)
)

```

### Arguments

def\_sec            a [block\\_section](#) object used to defined default section.

normal\_par        an object generated by [fp\\_par\(\)](#)

normal\_chunk     an object generated by [fp\\_text\(\)](#)

### Value

an object of class `rtf` representing an empty RTF document.

### See Also

[read\\_docx\(\)](#), [print.rtf\(\)](#), [rtf\\_add\(\)](#)

### Examples

```
rtf_doc(normal_par = fp_par(padding = 3))
```

---

run_autonom	<i>Auto number</i>
-------------	--------------------

---

### 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_autonom(
  seq_id = "table",
  pre_label = "Table ",
  post_label = ": ",
  bkm = NULL,
  bkm_all = FALSE,
  prop = NULL,
  start_at = NULL,
  tnd = 0,
  tns = "-"
)

```

**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 whole 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).
prop	formatting text properties returned by <a href="#">fp_text</a> .
start_at	If not NULL, it must be a positive integer, it specifies the new number to use, at which number the auto numbering will restart.
tnd	<i>title number depth</i> , a positive integer (only applies if positive) that specify the depth (or heading of level <i>depth</i> ) to use for prefixing the caption number with this last reference number. For example, setting tnd=2 will generate numbered captions like '4.3-2' (figure 2 of chapter 4.3).
tns	separator to use between title number and table number. Default is "-".

**usage**

You can use this function in conjunction with [fpar](#) to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package [officetdown](#).

**See Also**

Other run functions for reporting: [external\\_img\(\)](#), [ftext\(\)](#), [hyperlink\\_ftext\(\)](#), [run\\_bookmark\(\)](#), [run\\_columnbreak\(\)](#), [run\\_footnoteref\(\)](#), [run\\_footnote\(\)](#), [run\\_linebreak\(\)](#), [run\\_pagebreak\(\)](#), [run\\_reference\(\)](#), [run\\_word\\_field\(\)](#), [run\\_wordtext\(\)](#)

Other Word computed fields: [run\\_reference\(\)](#), [run\\_word\\_field\(\)](#)

**Examples**

```
run_autonom()
run_autonom(seq_id = "fig", pre_label = "fig. ")
run_autonom(seq_id = "tab", pre_label = "Table ", bkm = "anytable")
run_autonom(seq_id = "tab", pre_label = "Table ", bkm = "anytable",
  tnd = 2, tns= " ")
```



---

run_bookmark	<i>Bookmark for 'Word'</i>
--------------	----------------------------

---

**Description**

Add a bookmark on a run object.

**Usage**

```
run_bookmark(bkm, run)
```

**Arguments**

bkm	bookmark id to associate with run. Value can only be made of alpha numeric characters, '-' and '_'.
run	a run object, made with a call to one of the "run functions for reporting".

**usage**

You can use this function in conjunction with [fpar](#) to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package `officedown`.

**See Also**

Other run functions for reporting: [external\\_img\(\)](#), [ftext\(\)](#), [hyperlink\\_ftext\(\)](#), [run\\_autonum\(\)](#), [run\\_columnbreak\(\)](#), [run\\_footnoteref\(\)](#), [run\\_footnote\(\)](#), [run\\_linebreak\(\)](#), [run\\_pagebreak\(\)](#), [run\\_reference\(\)](#), [run\\_word\\_field\(\)](#), [run\\_wordtext\(\)](#)

**Examples**

```
ft <- fp_text(font.size = 12, bold = TRUE)
run_bookmark("par1", ftext("some text", ft))
```

---

run_columnbreak	<i>Column break for 'Word'</i>
-----------------	--------------------------------

---

**Description**

Create a representation of a column break.

**Usage**

```
run_columnbreak()
```

**usage**

You can use this function in conjunction with `fpar` to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package `officetdown`.

**See Also**

Other run functions for reporting: `external_img()`, `ftext()`, `hyperlink_ftext()`, `run_autonum()`, `run_bookmark()`, `run_footnoteref()`, `run_footnote()`, `run_linebreak()`, `run_pagebreak()`, `run_reference()`, `run_word_field()`, `run_wordtext()`

**Examples**

```
run_columnbreak()
```

---

```
run_footnote      Footnote for 'Word'
```

---

**Description**

Wraps a footnote in an object that can then be inserted as a run/chunk with `fpar()` or within an R Markdown document.

**Usage**

```
run_footnote(x, prop = NULL)
```

**Arguments**

`x` a set of blocks to be used as footnote content returned by function `block_list()`.  
`prop` formatting text properties returned by `fp_text_lite()` or `fp_text()`. It also can be `NULL` in which case, no formatting is defined (the default is applied).

**See Also**

Other run functions for reporting: `external_img()`, `ftext()`, `hyperlink_ftext()`, `run_autonum()`, `run_bookmark()`, `run_columnbreak()`, `run_footnoteref()`, `run_linebreak()`, `run_pagebreak()`, `run_reference()`, `run_word_field()`, `run_wordtext()`

**Examples**

```
library(officer)

fp_bold <- fp_text_lite(bold = TRUE)
fp_refnote <- fp_text_lite(vertical.align = "superscript")

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

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

a_par <- fpar(
  "this paragraph contains a note ",
  run_footnote(x = bl, prop = fp_refnote),
  "."
)

doc <- read_docx()
doc <- body_add_fpar(doc, value = a_par, style = "Normal")

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

---

run\_footnoteref

*Word footnote reference*

---

## Description

Wraps a footnote reference in an object that can then be inserted as a run/chunk with `fpar()` or within an R Markdown document.

## Usage

```
run_footnoteref(prop = NULL)
```

## Arguments

`prop` formatting text properties returned by `fp_text_lite()` or `fp_text()`. It also can be `NULL` in which case, no formatting is defined (the default is applied).

## See Also

Other run functions for reporting: `external_img()`, `ftext()`, `hyperlink_ftext()`, `run_autonum()`, `run_bookmark()`, `run_columnbreak()`, `run_footnote()`, `run_linebreak()`, `run_pagebreak()`, `run_reference()`, `run_word_field()`, `run_wordtext()`

## Examples

```
run_footnoteref()
to_wml(run_footnoteref())
```

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

```
run_linebreak()
```

### usage

You can use this function in conjunction with [fpar](#) to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package `officedown`.

### See Also

Other run functions for reporting: [external\\_img\(\)](#), [ftext\(\)](#), [hyperlink\\_ftext\(\)](#), [run\\_autonum\(\)](#), [run\\_bookmark\(\)](#), [run\\_columnbreak\(\)](#), [run\\_footnoteref\(\)](#), [run\\_footnote\(\)](#), [run\\_pagebreak\(\)](#), [run\\_reference\(\)](#), [run\\_word\\_field\(\)](#), [run\\_wordtext\(\)](#)

### Examples

```
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.

### Usage

```
run_pagebreak()
```

### usage

You can use this function in conjunction with [fpar](#) to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package `officedown`.

**See Also**

Other run functions for reporting: [external\\_img\(\)](#), [ftext\(\)](#), [hyperlink\\_ftext\(\)](#), [run\\_autonum\(\)](#), [run\\_bookmark\(\)](#), [run\\_columnbreak\(\)](#), [run\\_footnoteref\(\)](#), [run\\_footnote\(\)](#), [run\\_linebreak\(\)](#), [run\\_reference\(\)](#), [run\\_word\\_field\(\)](#), [run\\_wordtext\(\)](#)

**Examples**

```
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

*Cross reference*

---

**Description**

Create a representation of a reference

**Usage**

```
run_reference(id, prop = NULL)
```

**Arguments**

id	reference id, a string
prop	formatting text properties returned by <a href="#">fp_text</a> .

**usage**

You can use this function in conjunction with [fpar](#) to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package `officetdown`.

**See Also**

Other run functions for reporting: [external\\_img\(\)](#), [ftext\(\)](#), [hyperlink\\_ftext\(\)](#), [run\\_autonum\(\)](#), [run\\_bookmark\(\)](#), [run\\_columnbreak\(\)](#), [run\\_footnoteref\(\)](#), [run\\_footnote\(\)](#), [run\\_linebreak\(\)](#), [run\\_pagebreak\(\)](#), [run\\_word\\_field\(\)](#), [run\\_wordtext\(\)](#)

Other Word computed fields: [run\\_autonum\(\)](#), [run\\_word\\_field\(\)](#)

**Examples**

```
run_reference('a_ref')
```

---

run_wordtext	<i>Word chunk of text with a style</i>
--------------	--

---

### Description

Format a chunk of text associated with a 'Word' character style. The style is defined with its unique identifier.

### Usage

```
run_wordtext(text, style_id = NULL)
```

### Arguments

text	text value, a single character value
style_id	'Word' unique style identifier associated with the style to use.

### See Also

[ftext\(\)](#)

Other run functions for reporting: [external\\_img\(\)](#), [ftext\(\)](#), [hyperlink\\_ftext\(\)](#), [run\\_autonum\(\)](#), [run\\_bookmark\(\)](#), [run\\_columnbreak\(\)](#), [run\\_footnoteref\(\)](#), [run\\_footnote\(\)](#), [run\\_linebreak\(\)](#), [run\\_pagebreak\(\)](#), [run\\_reference\(\)](#), [run\\_word\\_field\(\)](#)

### Examples

```
run1 <- run_wordtext("hello", "DefaultParagraphFont")
paragraph <- fpar(run1)

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

---

run_word_field	<i>'Word' computed field</i>
----------------	------------------------------

---

### Description

Create a 'Word' computed field.

### Usage

```
run_word_field(field, prop = NULL, seqfield = NULL)

run_seqfield(field, prop = NULL, seqfield = NULL)
```

**Arguments**

field	Value for a "Word Computed Field" as a string.
prop	formatting text properties returned by <code>fp_text</code> .
seqfield	deprecated in favor of field.

**usage**

You can use this function in conjunction with `fpar` to create paragraphs consisting of differently formatted text parts. You can also use this function as an *r chunk* in an R Markdown document made with package `officedown`.

**Note**

In the previous version, this function was called `run_seqfield` but the name was wrong and should have been `run_word_field`.

**See Also**

Other run functions for reporting: `external_img()`, `ftext()`, `hyperlink_ftext()`, `run_autonum()`, `run_bookmark()`, `run_columnbreak()`, `run_footnoteref()`, `run_footnote()`, `run_linebreak()`, `run_pagebreak()`, `run_reference()`, `run_wordtext()`

Other Word computed fields: `run_autonum()`, `run_reference()`

**Examples**

```
run_word_field(field = "PAGE \\* MERGEFORMAT")
run_word_field(field = "Date \@ \"MMMM d yyyy\"")
```

---

section\_columns      *Section columns*

---

**Description**

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\(\)](#)

**Examples**

```
section_columns()
```

---

set\_autonum\_bookmark *Update bookmark of an autonumber run*

---

**Description**

This function lets recycling a object made by [run\\_autonum\(\)](#) by changing the bookmark value. This is useful to avoid calling [run\\_autonum\(\)](#) several times because of many tables.

**Usage**

```
set_autonum_bookmark(x, bkm = NULL)
```

**Arguments**

x	an object of class <a href="#">run_autonum()</a>
bkm	bookmark id to associate with autonumber run. Value can only be made of alpha numeric characters, '.', '-' and '_'.

**See Also**

[run\\_autonum\(\)](#)

**Examples**

```
z <- run_autonum(seq_id = "tab", pre_label = "Table ",  
  bkm = "anytable")  
set_autonum_bookmark(z, bkm = "anothertable")
```



---

set\_doc\_properties      *Set document properties*

---

### Description

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

Any character property can be added as a document property. It provides an easy way to insert arbitrary fields. Given the challenges that can be encountered with find-and-replace in word with officer, the use of document fields and quick text fields provides a much more robust approach to automatic document generation from R.

### Usage

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

### Arguments

x	an rdocx or rpptx object
title, subject, creator, description	text fields
created	a date object
...	named arguments (names are field names), each element is a single character value specifying value associated with the corresponding field name.
values	a named list (names are field names), each element is a single character value specifying value associated with the corresponding field name. If values is provided, argument ... will be ignored.

### 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**

```
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(),
  yoyo = "yok yok",
  glop = "pas glop")
x <- doc_properties(x)
```

---

 set\_notes

*Set notes for current slide*


---

**Description**

Set speaker notes for the current slide in a pptx presentation.

**Usage**

```
set_notes(x, value, location, ...)
```

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

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

**Arguments**

x	an rpptx object
value	text to be added to notes
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 notes_ph functions. See section "see also".
...	further arguments passed to or from other methods.

**Methods (by class)**

- `set_notes(character)`: add a character vector to a place holder in the notes on the current slide, values will be added as paragraphs.
- `set_notes(block_list)`: add a `block_list()` to a place holder in the notes on the current slide.

**See Also**

[print.rpptx\(\)](#), [read.pptx\(\)](#), [add\\_slide\(\)](#), [notes\\_location\\_label\(\)](#), [notes\\_location\\_type\(\)](#)

Other functions slide manipulation: [add\\_slide\(\)](#), [move\\_slide\(\)](#), [on\\_slide\(\)](#), [remove\\_slide\(\)](#)

**Examples**

```

# 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")
fpt_blue_bold <- fp_text_lite(color = "#006699", bold = TRUE)
doc <- read_pptx()
# add a slide with some text ----
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_with(x = doc, value = "Slide Title 1",
  location = ph_location_type(type = "title") )
# set speaker notes for the slide ----
doc <- set_notes(doc, value = "This text will only be visible for the speaker.",
  location = notes_location_type("body"))

# add a slide with some text ----
doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
doc <- ph_with(x = doc, value = "Slide Title 2",
  location = ph_location_type(type = "title") )
bl <- block_list(
  fpar(ftext("hello world", fpt_blue_bold)),
  fpar(ftext("Turlututu chapeau pointu", fpt_blue_bold))
)
doc <- set_notes(doc, value = bl,
  location = notes_location_type("body"))

print(doc, target = fileout)

```

---

sheet\_select

*Select sheet*


---

**Description**

Set a particular sheet selected when workbook will be edited.

**Usage**

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

**Arguments**

x	xlsx object
sheet	sheet name

**Examples**

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

---

shortcuts

*shortcuts for formatting properties*


---

**Description**

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

**Usage**

```
shortcuts
```

**Examples**

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

---

slide\_size

*Slides width and height*


---

**Description**

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

**Usage**

```
slide_size(x)
```

**Arguments**

x                    an `rpptx` object

**See Also**

Other functions for reading presentation informations: [annotate\\_base\(\)](#), [color\\_scheme\(\)](#), [doc\\_properties\(\)](#), [layout\\_properties\(\)](#), [layout\\_summary\(\)](#), [length.rpptx\(\)](#), [plot\\_layout\\_properties\(\)](#), [slide\\_summary\(\)](#)

**Examples**

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

---

slide_summary	<i>Slide content in a data.frame</i>
---------------	--------------------------------------

---

### Description

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

### Usage

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

### Arguments

x	an rpptx object
index	slide index

### Note

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

### See Also

Other functions for reading presentation informations: [annotate\\_base\(\)](#), [color\\_scheme\(\)](#), [doc\\_properties\(\)](#), [layout\\_properties\(\)](#), [layout\\_summary\(\)](#), [length.rpptx\(\)](#), [plot\\_layout\\_properties\(\)](#), [slide\\_size\(\)](#)

### Examples

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

---

 sp\_line

*Line properties*


---

### Description

Create a sp\_line object that describes line properties.

### Usage

```
sp_line(
  color = "transparent",
  lwd = 1,
  lty = "solid",
  linecompd = "sng",
  lineend = "rnd",
  linejoin = "round",
  headend = sp_lineend(type = "none"),
  tailend = sp_lineend(type = "none")
)
```

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

```
## S3 method for class 'sp_line'
update(
  object,
  color,
  lwd,
  lty,
  linecompd,
  lineend,
  linejoin,
  headend,
  tailend,
  ...
)
```

### Arguments

color	line color - a single character value specifying a valid color (e.g. "#000000" or "black").
lwd	line width (in point) - 0 or positive integer value.
lty	single character value specifying the line type. Expected value is one of the following: default 'solid' or 'dot' or 'dash' or 'lgDash' or 'dashDot' or 'lgDashDot' or 'lgDashDotDot' or 'sysDash' or 'sysDot' or 'sysDashDot' or 'sysDashDotDot'.

linecmpd	single character value specifying the compound line type. Expected value is one of the following : default 'sng' or 'dbl' or 'tri' or 'thinThick' or 'thickThin'
lineend	single character value specifying the line end style Expected value is one of the following : default 'rnd' or 'sq' or 'flat'
linejoin	single character value specifying the line join style Expected value is one of the following : default 'round' or 'bevel' or 'miter'
headend	a sp_lineend object specifying line head end style
tailend	a sp_lineend object specifying line tail end style
x, object	sp_line object
...	further arguments - not used

**Value**

a sp\_line object

**See Also**

[sp\\_lineend](#)

Other functions for defining shape properties: [sp\\_lineend\(\)](#)

**Examples**

```
sp_line()
sp_line(color = "red", lwd = 2)
sp_line(lty = "dot", linecmpd = "dbl")
print( sp_line (color="red", lwd = 2) )
obj <- sp_line (color="red", lwd = 2)
update( obj, linecmpd = "dbl" )
```

---

sp\_lineend

*Line end properties*


---

**Description**

Create a sp\_lineend object that describes line end properties.

**Usage**

```
sp_lineend(type = "none", width = "med", length = "med")
```

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

```
## S3 method for class 'sp_lineend'
update(object, type, width, length, ...)
```

**Arguments**

type	single character value specifying the line end type. Expected value is one of the following : default 'none' or 'triangle' or 'stealth' or 'diamond' or 'oval' or 'arrow'
width	single character value specifying the line end width Expected value is one of the following : default 'sm' or 'med' or 'lg'
length	single character value specifying the line end length Expected value is one of the following : default 'sm' or 'med' or 'lg'
x, object	sp_lineend object
...	further arguments - not used

**Value**

a sp\_lineend object

**See Also**

[sp\\_line](#)

Other functions for defining shape properties: [sp\\_line\(\)](#)

**Examples**

```
sp_lineend()
sp_lineend(type = "triangle")
sp_lineend(type = "arrow", width = "lg", length = "lg")
print( sp_lineend (type="triangle", width = "lg") )
obj <- sp_lineend (type="triangle", width = "lg")
update( obj, type = "arrow" )
```

---

styles\_info

*Read 'Word' styles*

---

**Description**

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

**Usage**

```
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()
styles_info(x)
styles_info(x, type = "paragraph", is_default = TRUE)
```

---

table_colwidths	<i>Column widths of a table</i>
-----------------	---------------------------------

---

**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\\_stylenames\(\)](#), [table\\_width\(\)](#)

---

`table_conditional_formatting`*Table conditional formatting*

---

### 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.

### Note

You must define a format for `first_row`, `first_column` and other properties if you need to use them. The format is defined in a docx template.

### See Also

Other functions for table definition: [prop\\_table\(\)](#), [table\\_colwidths\(\)](#), [table\\_layout\(\)](#), [table\\_stylenames\(\)](#), [table\\_width\(\)](#)

### Examples

```
table_conditional_formatting(first_row = TRUE, first_column = TRUE)
```

---

table_layout	<i>Algorithm for table layout</i>
--------------	-----------------------------------

---

**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**

```
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\\_stylenames\(\)](#), [table\\_width\(\)](#)

---

table_stylenames	<i>Paragraph styles for columns</i>
------------------	-------------------------------------

---

**Description**

The function defines the paragraph styles for columns.

**Usage**

```
table_stylenames(stylenames = list())
```

**Arguments**

stylenames            a named character vector, names are column names, values are paragraph styles associated with each column. If a column is not specified, default value 'Normal' is used. Another form is as a named list, the list names are the styles and the contents are column names to be formatted with the corresponding style.

**See Also**

Other functions for table definition: [prop\\_table\(\)](#), [table\\_colwidths\(\)](#), [table\\_conditional\\_formatting\(\)](#), [table\\_layout\(\)](#), [table\\_width\(\)](#)

**Examples**

```

library(officer)

stylenames <- c(
  vs = "centered", am = "centered",
  gear = "centered", carb = "centered"
)

doc_1 <- read_docx()
doc_1 <- body_add_table(doc_1,
  value = mtcars, style = "table_template",
  stylenames = table_stylenames(stylenames = stylenames)
)

print(doc_1, target = tempfile(fileext = ".docx"))

stylenames <- list(
  "centered" = c("vs", "am", "gear", "carb")
)

doc_2 <- read_docx()
doc_2 <- body_add_table(doc_2,
  value = mtcars, style = "table_template",
  stylenames = table_stylenames(stylenames = stylenames)
)

print(doc_2, target = tempfile(fileext = ".docx"))

```

---

table\_width

*Preferred width for a table*


---

**Description**

Define the preferred width for a table.

**Usage**

```
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\(\)](#), [table\\_stylenames\(\)](#)

---

unordered_list	<i>Unordered list</i>
----------------	-----------------------

---

**Description**

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

**Usage**

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

**Arguments**

str_list	list of strings to be included in the object
level_list	list of levels for hierarchy structure. Use 0 for 'no bullet', 1 for level 1, 2 for level 2 and so on.
style	text style, a fp_text object list or a single fp_text objects. Use fp_text(font.size = 0, ...) to inherit from default sizes of the presentation.

**See Also**

[ph\\_with](#)

Other block functions for reporting: [block\\_caption\(\)](#), [block\\_list\(\)](#), [block\\_pour\\_docx\(\)](#), [block\\_section\(\)](#), [block\\_table\(\)](#), [block\\_toc\(\)](#), [fpar\(\)](#), [plot\\_instr\(\)](#)

**Examples**

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

# Index

- \* **Word computed fields**
  - run\_autonum, 95
  - run\_reference, 101
  - run\_word\_field, 102
- \* **block functions for reporting**
  - block\_caption, 7
  - block\_list, 8
  - block\_pour\_docx, 9
  - block\_section, 10
  - block\_table, 11
  - block\_toc, 12
  - fpar, 45
  - plot\_instr, 81
  - unordered\_list, 117
- \* **functions for Word document informations**
  - doc\_properties, 43
  - docx\_bookmarks, 38
  - docx\_dim, 39
  - length.rdocx, 56
  - set\_doc\_properties, 105
  - styles\_info, 112
- \* **functions for Word sections**
  - body\_end\_block\_section, 24
  - body\_end\_section\_columns, 25
  - body\_end\_section\_columns\_landscape, 26
  - body\_end\_section\_continuous, 27
  - body\_end\_section\_landscape, 28
  - body\_end\_section\_portrait, 28
  - body\_set\_default\_section, 33
- \* **functions for adding content**
  - body\_add\_blocks, 12
  - body\_add\_break, 13
  - body\_add\_caption, 14
  - body\_add\_docx, 15
  - body\_add\_fpar, 16
  - body\_add\_gg, 17
  - body\_add\_img, 18
  - body\_add\_par, 19
  - body\_add\_plot, 20
  - body\_add\_table, 21
  - body\_add\_toc, 22
- \* **functions for defining formatting properties**
  - fp\_border, 47
  - fp\_cell, 47
  - fp\_par, 49
  - fp\_text, 51
- \* **functions for defining shape properties**
  - sp\_line, 110
  - sp\_lineend, 111
- \* **functions for placeholder location**
  - ph\_location, 65
  - ph\_location\_fullsize, 67
  - ph\_location\_label, 67
  - ph\_location\_left, 68
  - ph\_location\_right, 69
  - ph\_location\_template, 70
  - ph\_location\_type, 71
- \* **functions for placeholders manipulation**
  - ph\_hyperlink, 64
  - ph\_remove, 73
  - ph\_slidelink, 74
- \* **functions for reading presentation informations**
  - annotate\_base, 5
  - color\_scheme, 35
  - doc\_properties, 43
  - layout\_properties, 55
  - layout\_summary, 56
  - length.rpptx, 57
  - plot\_layout\_properties, 82
  - slide\_size, 108
  - slide\_summary, 109
- \* **functions for section definition**
  - page\_mar, 63
  - page\_size, 64

- prop\_section, 84
- section\_columns, 103
- \* **functions for table definition**
  - prop\_table, 87
  - table\_colwidths, 113
  - table\_conditional\_formatting, 114
  - table\_layout, 115
  - table\_stylenames, 115
  - table\_width, 116
- \* **functions slide manipulation**
  - add\_slide, 5
  - move\_slide, 58
  - on\_slide, 62
  - remove\_slide, 91
  - set\_notes, 106
- \* **run functions for reporting**
  - external\_img, 44
  - ftext, 53
  - hyperlink\_ftext, 54
  - run\_autonum, 95
  - run\_bookmark, 97
  - run\_columnbreak, 97
  - run\_footnote, 98
  - run\_footnoteref, 99
  - run\_linebreak, 100
  - run\_pagebreak, 100
  - run\_reference, 101
  - run\_word\_field, 102
  - run\_wordtext, 102
- add\_sheet, 4
- add\_slide, 5, 59, 62, 91, 106
- add\_slide(), 90, 106
- annotate\_base, 5, 35, 43, 55–57, 82, 108, 109
- as.matrix.rpptx, 6
- block\_caption, 7, 8, 10–12, 46, 81, 117
- block\_caption(), 8, 14, 60
- block\_list, 7, 8, 10–12, 46, 77, 78, 81, 117
- block\_list(), 13, 45, 46, 85, 93, 98, 106
- block\_pour\_docx, 7, 8, 9, 10–12, 46, 81, 117
- block\_pour\_docx(), 8
- block\_section, 7, 8, 10, 10, 11, 12, 24, 46, 81, 85, 95, 117
- block\_section(), 8
- block\_table, 7, 8, 10, 11, 12, 46, 81, 117
- block\_table(), 8, 77
- block\_toc, 7, 8, 10, 11, 12, 46, 81, 117
- block\_toc(), 8
- body\_add, 45
- body\_add\_blocks, 12, 14–16, 18–23
- body\_add\_blocks(), 8, 44
- body\_add\_break, 13, 13, 14–16, 18–23
- body\_add\_caption, 13, 14, 14, 15, 16, 18–23
- body\_add\_docx, 13, 14, 15, 16, 18–23
- body\_add\_fpar, 13–15, 16, 18–23
- body\_add\_fpar(), 46
- body\_add\_gg, 13–16, 17, 19–23
- body\_add\_img, 13–16, 18, 18, 20–23
- body\_add\_par, 13–16, 18, 19, 19, 21–23, 89
- body\_add\_plot, 13–16, 18–20, 20, 22, 23, 89
- body\_add\_plot(), 81
- body\_add\_table, 13–16, 18–21, 21, 23, 89
- body\_add\_toc, 13–16, 18–22, 22
- body\_bookmark, 23
- body\_end\_block\_section, 24, 25–29, 34
- body\_end\_block\_section(), 25
- body\_end\_section\_columns, 24, 25, 26–29, 34
- body\_end\_section\_columns\_landscape, 24, 25, 26, 27–29, 34
- body\_end\_section\_continuous, 24–26, 27, 28, 29, 34
- body\_end\_section\_landscape, 24–27, 28, 29, 34
- body\_end\_section\_portrait, 24–28, 28, 34
- body\_remove, 29
- body\_replace\_all\_text, 30, 41, 42
- body\_replace\_img\_at\_bkm
  - (body\_replace\_text\_at\_bkm), 32
- body\_replace\_text\_at\_bkm, 32
- body\_set\_default\_section, 24–29, 33
- change\_styles, 34
- color\_scheme, 6, 35, 43, 55–57, 82, 108, 109
- cursor\_backward(cursor\_begin), 35
- cursor\_begin, 35
- cursor\_bookmark(cursor\_begin), 35
- cursor\_end(cursor\_begin), 35
- cursor\_forward(cursor\_begin), 35
- cursor\_reach(cursor\_begin), 35
- cursor\_reach\_test(cursor\_begin), 35
- doc\_properties, 6, 35, 38, 39, 43, 55–57, 82, 105, 108, 109, 113
- docx\_bookmarks, 38, 39, 43, 57, 105, 113
- docx\_dim, 38, 39, 43, 57, 105, 113
- docx\_set\_character\_style, 39

- docx\_set\_paragraph\_style, 40
- docx\_show\_chunk, 30, 31, 41
- docx\_summary, 42
- empty\_content, 43, 78
- external\_img, 44, 54, 55, 78, 96–103
- external\_img(), 8, 46
- footers\_replace\_all\_text
  - (body\_replace\_all\_text), 30
- footers\_replace\_img\_at\_bkm
  - (body\_replace\_text\_at\_bkm), 32
- footers\_replace\_text\_at\_bkm
  - (body\_replace\_text\_at\_bkm), 32
- format.fp\_cell(fp\_cell), 47
- format.fp\_text(fp\_text), 51
- fp\_border, 47, 49–51, 53
- fp\_cell, 47, 47, 51, 53
- fp\_par, 47, 49, 49, 53
- fp\_par(), 41, 46, 93, 95
- fp\_text, 47, 49, 51, 51, 54, 55, 96, 101, 103
- fp\_text(), 40, 41, 46, 95, 98, 99
- fp\_text\_lite(fp\_text), 51
- fp\_text\_lite(), 98, 99
- fpar, 7, 8, 10–12, 16, 45, 45, 51, 53–55, 77, 78, 81, 96–98, 100, 101, 103, 117
- fpar(), 8, 93, 98, 99
- ftext, 45, 53, 53, 55, 96–103
- ftext(), 46, 102
- grep, 31
- grepl, 30
- gsub, 30
- headers\_replace\_all\_text
  - (body\_replace\_all\_text), 30
- headers\_replace\_img\_at\_bkm
  - (body\_replace\_text\_at\_bkm), 32
- headers\_replace\_text\_at\_bkm
  - (body\_replace\_text\_at\_bkm), 32
- hyperlink\_ftext, 45, 54, 54, 96–103
- layout\_properties, 6, 35, 43, 55, 56, 57, 82, 108, 109
- layout\_summary, 6, 35, 43, 55, 56, 57, 82, 108, 109
- layout\_summary(), 5
- length.rdocx, 38, 39, 43, 56, 105, 113
- length.rpptx, 6, 35, 43, 55, 56, 57, 82, 108, 109
- length.xlsx(read\_xlsx), 90
- media\_extract, 58
- move\_slide, 5, 58, 62, 91, 106
- notes\_location\_label, 59
- notes\_location\_label(), 106
- notes\_location\_type, 59
- notes\_location\_type(), 106
- officer, 60
- officer-defunct, 61
- officer-package(officer), 60
- on\_slide, 5, 59, 62, 91, 106
- page\_mar, 63, 64, 85, 104
- page\_size, 63, 64, 85, 104
- ph\_hyperlink, 64, 73, 75
- ph\_location, 65, 67–69, 71, 72, 78
- ph\_location\_fullsize, 66, 67, 68, 69, 71, 72, 78
- ph\_location\_label, 66, 67, 67, 69, 71, 72, 78
- ph\_location\_left, 66–68, 68, 69, 71, 72, 78
- ph\_location\_right, 66–69, 69, 71, 72, 78
- ph\_location\_template, 66–69, 70, 72, 78
- ph\_location\_type, 66–69, 71, 71, 78
- ph\_location\_type(), 68, 69
- ph\_remove, 65, 73, 75
- ph\_remove(), 91
- ph\_slidelink, 65, 73, 74
- ph\_with, 45, 65, 73, 75, 75, 117
- ph\_with(), 5, 8, 44, 46, 62, 81, 90, 91
- plot\_instr, 7, 8, 10–12, 46, 81, 93, 117
- plot\_instr(), 8, 20, 93
- plot\_layout\_properties, 6, 35, 43, 55–57, 82, 108, 109
- plot\_layout\_properties(), 5, 90
- pptx\_summary, 82
- print.fp\_cell(fp\_cell), 47
- print.fp\_par(fp\_par), 49
- print.fp\_text(fp\_text), 51
- print.rdocx(read\_docx), 88
- print.rpptx, 83
- print.rpptx(), 5, 90, 106
- print.rtf, 84
- print.rtf(), 95
- print.xlsx(read\_xlsx), 90
- print.sp\_line(sp\_line), 110
- print.sp\_lineend(sp\_lineend), 111



- prop\_section, [10](#), [33](#), [63](#), [64](#), [84](#), [104](#)
- prop\_table, [87](#), [113–115](#), [117](#)
- prop\_table(), [11](#)
- read\_docx, [88](#)
- read\_docx(), [60](#), [95](#)
- read\_pptx, [83](#), [89](#)
- read\_pptx(), [5](#), [59](#), [60](#), [62](#), [83](#), [91](#), [106](#)
- read\_xlsx, [90](#)
- regex, [30](#), [31](#)
- remove\_slide, [5](#), [59](#), [62](#), [91](#), [106](#)
- rtf\_add, [92](#)
- rtf\_add(), [94](#), [95](#)
- rtf\_doc, [94](#)
- rtf\_doc(), [60](#), [84](#), [93](#)
- run\_autonum, [7](#), [45](#), [54](#), [55](#), [95](#), [97–103](#)
- run\_Autonum(), [12](#), [46](#), [60](#), [104](#)
- run\_bookmark, [45](#), [54](#), [55](#), [96](#), [97](#), [98–103](#)
- run\_ColumnBreak, [45](#), [54](#), [55](#), [96](#), [97](#), [97](#), [98–103](#)
- run\_footnote, [45](#), [54](#), [55](#), [96–98](#), [98](#), [99–103](#)
- run\_footnoteref, [45](#), [54](#), [55](#), [96–98](#), [99](#), [100–103](#)
- run\_linebreak, [45](#), [54](#), [55](#), [96–99](#), [100](#), [101–103](#)
- run\_pagebreak, [45](#), [54](#), [55](#), [96–100](#), [100](#), [101–103](#)
- run\_reference, [45](#), [54](#), [55](#), [96–101](#), [101](#), [102](#), [103](#)
- run\_seqfield(run\_word\_field), [102](#)
- run\_word\_field, [45](#), [54](#), [55](#), [96–102](#), [102](#)
- run\_word\_field(), [46](#)
- run\_wordtext, [45](#), [54](#), [55](#), [96–101](#), [102](#), [103](#)
- section\_columns, [63](#), [64](#), [85](#), [103](#)
- set\_AutonumBookmark, [104](#)
- set\_doc\_properties, [38](#), [39](#), [43](#), [57](#), [105](#), [113](#)
- set\_notes, [5](#), [59](#), [62](#), [91](#), [106](#)
- sheet\_select, [107](#)
- shortcuts, [108](#)
- slide\_size, [6](#), [35](#), [43](#), [55–57](#), [82](#), [108](#), [109](#)
- slide\_summary, [6](#), [35](#), [43](#), [55–57](#), [64](#), [65](#), [73](#), [75](#), [82](#), [108](#), [109](#)
- slip\_in\_column\_break(officer-defunct), [61](#)
- slip\_in\_footnote(officer-defunct), [61](#)
- slip\_in\_seqfield(officer-defunct), [61](#)
- slip\_in\_text(officer-defunct), [61](#)
- slip\_in\_xml(officer-defunct), [61](#)
- sp\_line, [110](#), [112](#)
- sp\_line(), [66](#)
- sp\_lineend, [111](#), [111](#)
- styles\_info, [38](#), [39](#), [43](#), [57](#), [105](#), [112](#)
- styles\_info(), [34](#)
- table\_colwidths, [87](#), [113](#), [114](#), [115](#), [117](#)
- table\_colwidths(), [87](#)
- table\_conditional\_formatting, [87](#), [113](#), [114](#), [115](#), [117](#)
- table\_conditional\_formatting(), [77](#), [87](#)
- table\_layout, [87](#), [113–115](#), [115](#), [117](#)
- table\_layout(), [87](#)
- table\_stylenames, [87](#), [113–115](#), [115](#), [117](#)
- table\_stylenames(), [22](#), [87](#)
- table\_width, [87](#), [113–115](#), [116](#)
- table\_width(), [87](#)
- unordered\_list, [7](#), [8](#), [10–12](#), [46](#), [77](#), [81](#), [117](#)
- update.fp\_border(fp\_border), [47](#)
- update.fp\_cell(fp\_cell), [47](#)
- update.fp\_par(fp\_par), [49](#)
- update.fp\_text(fp\_text), [51](#)
- update.fpar(fpar), [45](#)
- update.sp\_line(sp\_line), [110](#)
- update.sp\_lineend(sp\_lineend), [111](#)