Package ‘officer’

March 13, 2020

Type Package

Title Manipulation of Microsoft Word and PowerPoint Documents

Version 0.3.8

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

License GPL-3

LazyData TRUE

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

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

Encoding UTF-8

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

RoxygenNote 7.0.2

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

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2020-03-13 16:10:02 UTC
R topics documented:

add_sheet  ........................................... 4
add_slide ........................................... 5
annotate_base ....................................... 5
block_caption ......................................... 6
block_list ............................................... 7
block_section ......................................... 8
block_table ........................................... 9
block_toc ............................................... 10
body_add ............................................... 11
body_add_blocks ...................................... 13
body_add_break ....................................... 14
body_add_docx ........................................ 15
body_add_fpar ......................................... 16
body_add_gg ........................................... 17
body_add_img ......................................... 17
body_add_par .......................................... 18
body_add_table ....................................... 19
body_add_toc .......................................... 20
body_add_xml ......................................... 21
body Bookmark ......................................... 21
body_remove ........................................... 22
body_replace_all_text ................................. 22
body_replace_text_at_bkm .............................. 25
change_styles .......................................... 26
color_scheme ........................................... 26
cursor_begin ........................................... 27
docx_body_relationship ................................. 29
docx_body_xml .......................................... 29
docx_bookmarks ......................................... 30
docx_dim ................................................ 31
docx_reference_img .................................... 31
docx_show_chunk ...................................... 32
docx_summary .......................................... 32
docx_properties ......................................... 33
empty_content .......................................... 34
external_img ............................................ 34
fortify_location ........................................ 35
fpar ...................................................... 36
fp_border ................................................ 37
fp_cell ................................................... 38
fp_par .................................................... 39
fp_text .................................................. 41
ftext ..................................................... 43
get_reference_value .................................... 43
layout_properties ...................................... 44
layout_summary ........................................ 45
### Topics Documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>length.rdocx</td>
<td>45</td>
</tr>
<tr>
<td>length.rpptx</td>
<td>46</td>
</tr>
<tr>
<td>media_extract</td>
<td>46</td>
</tr>
<tr>
<td>move_slide</td>
<td>47</td>
</tr>
<tr>
<td>officer</td>
<td>48</td>
</tr>
<tr>
<td>officer-defunct</td>
<td>49</td>
</tr>
<tr>
<td>on_slide</td>
<td>50</td>
</tr>
<tr>
<td>pack_folder</td>
<td>51</td>
</tr>
<tr>
<td>page_mar</td>
<td>51</td>
</tr>
<tr>
<td>page_size</td>
<td>52</td>
</tr>
<tr>
<td>ph_add_fpar</td>
<td>52</td>
</tr>
<tr>
<td>ph_add_par</td>
<td>54</td>
</tr>
<tr>
<td>ph_add_text</td>
<td>55</td>
</tr>
<tr>
<td>ph_empty</td>
<td>56</td>
</tr>
<tr>
<td>ph_hyperlink</td>
<td>57</td>
</tr>
<tr>
<td>ph_location</td>
<td>58</td>
</tr>
<tr>
<td>ph_location_fullsize</td>
<td>60</td>
</tr>
<tr>
<td>ph_location_label</td>
<td>60</td>
</tr>
<tr>
<td>ph_location_left</td>
<td>61</td>
</tr>
<tr>
<td>ph_location_right</td>
<td>62</td>
</tr>
<tr>
<td>ph_location_template</td>
<td>63</td>
</tr>
<tr>
<td>ph_location_type</td>
<td>64</td>
</tr>
<tr>
<td>ph_remove</td>
<td>66</td>
</tr>
<tr>
<td>ph_slidelink</td>
<td>67</td>
</tr>
<tr>
<td>ph_with</td>
<td>68</td>
</tr>
<tr>
<td>ph_with_gg_at</td>
<td>72</td>
</tr>
<tr>
<td>ph_with_img_at</td>
<td>73</td>
</tr>
<tr>
<td>ph_with_text</td>
<td>73</td>
</tr>
<tr>
<td>plot_instr</td>
<td>74</td>
</tr>
<tr>
<td>pptx_summary</td>
<td>75</td>
</tr>
<tr>
<td>print.rpptx</td>
<td>75</td>
</tr>
<tr>
<td>prop_section</td>
<td>76</td>
</tr>
<tr>
<td>read_docx</td>
<td>77</td>
</tr>
<tr>
<td>read_pptx</td>
<td>77</td>
</tr>
<tr>
<td>read_xlsx</td>
<td>78</td>
</tr>
<tr>
<td>remove_slide</td>
<td>79</td>
</tr>
<tr>
<td>run_autonum</td>
<td>80</td>
</tr>
<tr>
<td>run_columnbreak</td>
<td>80</td>
</tr>
<tr>
<td>run_linebreak</td>
<td>81</td>
</tr>
<tr>
<td>run_pagebreak</td>
<td>81</td>
</tr>
<tr>
<td>run_reference</td>
<td>82</td>
</tr>
<tr>
<td>run_seqfield</td>
<td>82</td>
</tr>
<tr>
<td>sanitize_images</td>
<td>83</td>
</tr>
<tr>
<td>sections</td>
<td>83</td>
</tr>
<tr>
<td>set_doc_properties</td>
<td>85</td>
</tr>
<tr>
<td>sheet_select</td>
<td>86</td>
</tr>
<tr>
<td>shortcuts</td>
<td>86</td>
</tr>
<tr>
<td>slide_size</td>
<td>87</td>
</tr>
</tbody>
</table>
add_sheet

Description

add a sheet into an xlsx worksheet

Usage

add_sheet(x, label)

Arguments

x rxlsx object
label sheet label

Examples

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

add a slide

Description
add a slide into a pptx presentation

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

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

See Also
print.rpptx read_pptx ph_with, layout_summary
Other functions slide manipulation: move_slide(), on_slide(), remove_slide()

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

annotate_base
PowerPoint placeholder parameters annotation

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

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

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

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

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

Value

- rpptx object of the annotated PowerPoint file

See Also

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

Examples

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

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

block_caption

caption block

Description

Create a representation of a caption that can be used for cross reference. The caption can also be an auto numbered paragraph.

Usage

```r
block_caption(label, style, id, autonum = NULL)
```

Arguments

- label: a scalar character representing label to display
- style: paragraph style name
- id: cross reference identifier
- autonum: an object generated with function run_autonum

See Also

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

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

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

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

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

block_list  
create paragraph blocks

Description

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

Usage

`block_list(...)`

Arguments

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

See Also

`ph_with()`, `body_add()`

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

Examples

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

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

Create a representation of a section

## 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_table()`, `block_toc()`, `fpar()`, `plot_instr()`, `unordered_list()`

### Examples

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

### Description

Create a representation of a table

### Usage

```r
block_table(
  x,
  style = NULL,
  header = TRUE,
  first_row = TRUE,
  first_column = FALSE,
  last_row = FALSE,
  last_column = FALSE,
  no_hband = FALSE,
  no_vband = TRUE
)
```

### Arguments

- **x**: a data.frame to add as a table
- **style**: table style
- **header**: display header if TRUE
- **first_row**: Specifies that the first column conditional formatting should be applied. Details for this and other conditional formatting options can be found at http://officeopenxml.com/WPtblLook.php
- **first_column**: Specifies that the first column conditional formatting should be applied.
- **last_row**: Specifies that the first column conditional formatting should be applied.
- **last_column**: Specifies that the first column conditional formatting should be applied.
- **no_hband**: Specifies that there should be no horizontal bands.
- **no_vband**: Specifies that there should be no vertical bands.
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 block functions for reporting: block_caption(), block_list(), block_section(), block_toc(),
fpars(), plot_instr(), unordered_list()

Examples

block_table(x = mtcars)

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

Arguments

level max title level of the table
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 block functions for reporting: block_caption(), block_list(), block_section(), block_toc(),
fpars(), plot_instr(), unordered_list()

Examples

block_toc(level = 2)
block_toc(style = "Table title")
**body_add**  
*add content into a Word document*

**Description**

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

**Usage**

```r
body_add(x, value, ...)
```

- **S3 method for class 'character'**
  ```r
  body_add(x, value, style = NULL, ...)
  ```

- **S3 method for class 'numeric'**
  ```r
  body_add(x, value, style = NULL, format_fun = formatC, ...)
  ```

- **S3 method for class 'factor'**
  ```r
  body_add(x, value, style = NULL, format_fun = as.character, ...)
  ```

- **S3 method for class 'fpar'**
  ```r
  body_add(x, value, style = NULL, ...)
  ```

- **S3 method for class 'data.frame'**
  ```r
  body_add(
    x,
    value,
    style = NULL,
    header = TRUE,
    first_row = TRUE,
    first_column = FALSE,
    last_row = FALSE,
    last_column = FALSE,
    no_hband = FALSE,
    no_vband = TRUE,
    ...,
  )
  ```

- **S3 method for class 'block_caption'**
  ```r
  body_add(x, value, ...)
  ```

- **S3 method for class 'block_list'**
  ```r
  body_add(x, value, ...)
  ```

- **S3 method for class 'block_toc'**
  ```r
  body_add(x, value, ...)
  ```
body_add

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

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

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

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

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

Arguments

- **x**: an rdocx object
- **value**: object to add in the document. Supported objects are vectors, data.frame, graphics, block of formatted paragraphs, unordered list of formatted paragraphs, pretty tables with package flextable, 'Microsoft' charts with package mschart.
- **...**: further arguments passed to or from other methods. When adding a ggplot object or plot_instr, these arguments will be used by png function.
- **style**: paragraph style name. These names are available with function styles_info and are the names of the Word styles defined in the base document (see argument path from read_docx).
- **format_fun**: a function to be used to format values.
- **header**: display header if TRUE
- **first_row**: Specifies that the first column conditional formatting should be applied.
- **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.
- **width**: height in inches
- **height**: height in inches
- **res**: resolution of the png image in ppi

Methods (by class)

- character: add a character vector.
- numeric: add a numeric vector.
- factor: add a factor vector.
body_add_blocks

- `fpar`: add a `fpar` object. These objects enable the creation of formatted paragraphs made of formatted chunks of text.
- `data.frame`: add a `data.frame` object.
- `block_caption`: add a `block_caption` object. These objects enable the creation of set of formatted paragraphs made of formatted chunks of text.
- `block_list`: add a `block_list` object.
- `block_toc`: add a table of content (a `block_toc` object).
- `external_img`: add an image (a `external_img` object).
- `run_pagebreak`: add a `run_pagebreak` object.
- `run_columnbreak`: add a `run_columnbreak` object.
- `gg`: add a `ggplot` object.
- `plot_instr`: add a base plot

Examples

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

**Description**

add a list of blocks into a document

**Usage**

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

**Arguments**

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

```r
library(magrittr)

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

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

Description

add a page break into an rdocx object

Usage

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

Arguments

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

Examples

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

insert an external docx

Description

add content of a docx into an rdocx object.

Usage

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

Arguments

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

Note

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

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

Examples

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

Description

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

Usage

```r
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 `read_docx(path = ...)`); in that case, paragraph settings from `fpar` will be ignored.
- **pos**: where to add the new element relative to the cursor, one of "after", "before", "on".

See Also

- `fpar`

Examples

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

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

doc <- read_docx() %>% body_add_fpar(fpar_)

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

# a way of using fpar to center an image in a Word doc ----

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

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

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

add ggplot

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

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

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

Examples
if( require("ggplot2") ){  
  doc <- read_docx()
  
  gg_plot <- ggplot(data = iris ) +
    geom_point(mapping = aes(Sepal.Length, Petal.Length))
  
  if( capabilities(what = "png") )
    doc <- body_add_gg(doc, value = gg_plot, style = "centered")
  
  print(doc, target = tempfile(fileext = ".docx") )
}

body_add_img

add image

Description
add an image into an rdocx object.

Usage
body_add_img(x, src, style = NULL, width, height, pos = "after")
**Arguments**

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

**Examples**

```r
doc <- read_docx()

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

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

---

**Description**

add a paragraph of text into an rdocx object

**Usage**

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

**Arguments**

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

library(magrittr)

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

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

body_add_table

Description

add a table into an rdocx object

Usage

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

Arguments

x a docx device
value a data.frame to add as a table
style table style
pos where to add the new element relative to the cursor, one of "after", "before", "on".
header display header if TRUE
first_row Specifies that the first column conditional formatting should be applied. Details
first_column Specifies that the first column conditional formatting should be applied.
last_row Specifies that the first column conditional formatting should be applied.
last_column Specifies that the first column conditional formatting should be applied.
no_hband Specifies that the first column conditional formatting should be applied.
no_vband Specifies that the first column conditional formatting should be applied.
Examples

```r
library(magrittr)

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

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

body_add_toc add table of content

Description

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

Usage

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

Arguments

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

Examples

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

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

### Description

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

### Usage

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

### Arguments

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

**body_bookmark**  

### Description

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

### Usage

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

### Arguments

- **x**: an rdocx object
- **id**: bookmark name

### Examples

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

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

remove an element

Description
remove element pointed by cursor from a Word document

Usage
body_remove(x)

Arguments
x an rdocx object

Examples
library(officer)
library(magrittr)

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

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

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

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

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

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

**Description**

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

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

**Chunking of text**

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

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

**Usage**

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

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

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

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

header_replace_all_text

Replacements will be performed in each header of all sections.
Replacements will be performed in each footer of all sections.

Author(s)

Frank Hangler, <frank@plotandscatter.com>

See Also

grep, regex, docx_show_chunk

Examples

```r
library(magrittr)

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

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

# 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)

headers_replace_text_at_bkm(x, bookmark, value)

footers_replace_text_at_bkm(x, bookmark, value)

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

Examples
library(magrittr)
doc <- read_docx()

body_add_par("centered text", style = "centered")
slip_in_text("How are you", style = "strong")
body_bookmark("text_to_replace")
body_replace_text_at_bkm("text_to_replace", "not left aligned")

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

img.file <- file.path( R.home("doc"), "html", "logo.jpg" )
doc <- read_docx(path = template)
doc <- headers_replace_img_at_bkm(x = doc, bookmark = "bmk_header", value = img.file)
### change_styles

**Description**

Replace styles with others in a Word document.

**Usage**

```r
document <- read_docx() 
body_add_par("A title", style = "heading 1") 
body_add_par("Another title", style = "heading 2") 
change_styles(x, mapstyles) 
```

**Arguments**

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

**Examples**

```r
library(magrittr)

mapstyles <- list("centered" = c("Normal"), "heading 3" = c("heading 1", "heading 2") )
document <- read_docx() 
body_add_par("A title", style = "heading 1") 
body_add_par("Another title", style = "heading 2") 
body_add_par("Hello world!", style = "Normal")
change_styles( mapstyles = mapstyles )
print(document, target = tempfile(fileext = ".docx"))
```

### color_scheme

**Description**

generate master layout color scheme into a data.frame.

**Usage**

```r
color_scheme(x)
```
**Arguments**

- `x` an rpptx object

**See Also**

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

**Examples**

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

---

**Description**

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

**Usage**

- `cursor_begin(x)`
- `cursor_bookmark(x, id)`
- `cursor_end(x)`
- `cursor_reach(x, keyword)`
- `cursor_forward(x)`
- `cursor_backward(x)`

**Arguments**

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

**cursor_begin**

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

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

**cursor_end**

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

**cursor_reach**

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

**cursor_forward**

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

**cursor_backward**

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

**Examples**

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

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

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

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

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

# move the cursor at the end of the document
cursor_end() %>%
  body_add_par("The document ends now", style = "Normal")
```
print(doc, target = tempfile(fileext = ".docx"))

# cursor_bookmark ----
library(magrittr)

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

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

---

docx_body_relationship

body xml document

Description

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

Usage

docx_body_relationship(x)

Arguments

x an rdocx object

Examples

doc <- read_docx()
docx_body_relationship(doc)

---

docx_body_xml

body xml document

Description

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


Usage

docx_body_xml(x)

Arguments

x  an rdocx object

Examples

    doc <- read_docx()
    docx_body_xml(doc)

---

docx_bookmarks  List Word bookmarks

Description

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

Usage

    docx_bookmarks(x)

Arguments

x  an rdocx object

See Also

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

Examples

    library(magrittr)

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

    docx_bookmarks(doc)

    docx_bookmarks(read_docx())
**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())

---

**Description**

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

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

**Usage**

docx_reference_img(x, src)

**Arguments**

x an rdocx object

src a vector of character containing image filenames.
**docx_show_chunk**  
*Show underlying text tag structure*

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

**Usage**
```r
docx_show_chunk(x)
```

**Arguments**
- `x`: a docx device

**See Also**
- `body_replace_all_text`

**Examples**
```r
library(magrittr)

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

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

---

**docx_summary**  
*get Word content in a data.frame*

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

**Usage**
```r
docx_summary(x)
```

**Arguments**
- `x`: an rdocx object
**Note**

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

**Examples**

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

---

**doc_properties**

*read document properties*

**Description**

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

**Usage**

```r
doc_properties(x)
```

**Arguments**

- `x` an `rdocx` or `rpptx` object

**See Also**

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

**Examples**

```r
x <- read_docx()
doc_properties(x)
```
empty_content  
create empty blocks

Description

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

Usage

empty_content()

See Also

ph_with(), body_add_blocks()

Examples

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

external_img  
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)

Arguments

src  
image file path
width  
height in inches.
height  
height in inches
**fortify_location**

**See Also**

`ph_with`, `body_add`, `fpar`

Other run functions for reporting: `ftext()`, `run_linebreak()`, `run_pagebreak()`

**Examples**

```r
# 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"))
```

---

**Description**

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

**Usage**

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

**Arguments**

- `x` a location for a placeholder.
- `doc` an rpptx object
- `...` unused arguments
See Also

`ph_location, ph_with`

Examples

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

---

`fpar`  

`concatenate formatted text as a paragraph`

Description

Create a paragraph representation by concatenating formatted text or images.  

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

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

Usage

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

### S3 method for class 'fpar'

```r
update(object, fp_p = NULL, fp_t = NULL, ...)
```

Arguments

- `...`  
  cot objects (ftext, external_img)
- `fp_p`  
  paragraph formatting properties
- `fp_t`  
  default text formatting properties. This is used as text formatting properties when simple text is provided as argument.
- `object`  
  fpar object

Details

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

See Also

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

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

---

**fp_border**

**border properties object**

---

**Description**

create a border properties object.

**Usage**

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

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

**Arguments**

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

**Examples**

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

---

**fp_cell**  
*Cell formatting properties*

### Description

Create a `fp_cell` object that describes cell formatting properties.

### Usage

```r
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 = "ltrb"
)
```

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

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

## S3 method for class 'fp_cell'
`update(
  object,
  border,
  border.bottom,
  border.left,
  border.top,
  border.right,
  vertical.align,
  margin = 0,
  margin.bottom,
  margin.top,
```
Arguments

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

Examples

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

Description

Create a fp_par object that describes paragraph formatting properties.

Usage

fp_par(
  text.align = "left",
  padding = 0,
  border = fp_border(width = 0),
  padding.bottom,
## S3 method for class 'fp_par'
print(x, ...)

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

**Arguments**

- **text.align**
  text alignment - a single character value, expected value is one of 'left', 'right', 'center', 'justify'.

- **padding**
  paragraph paddings - 0 or positive integer value. Argument padding overwrites arguments padding.bottom, padding.top, padding.left, padding.right.

- **border**
  shortcut for all borders.

- **padding.bottom**, **padding.top**, **padding.left**, **padding.right**
  paragraph paddings - 0 or positive integer value.

- **border.bottom**, **border.left**, **border.top**, **border.right**
  **fp_border** for borders. overwrite other border properties.

- **shading.color**
  shading color - a single character value specifying a valid color (e.g. "#000000" or "black").

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

`x`, object  fp_par object

... further arguments - not used

**Value**

a fp_par object

**Examples**

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

---

**fp_text**  Text formatting properties

**Description**

Create a fp_text object that describes text formatting properties.

**Usage**

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

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

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

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

Arguments

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

Value

- a `fp_text` object

See Also

- `ftext`, `fpar`

Examples

```r
fp_text()
fp_text(color = "red")
fp_text(bold = TRUE, shading.color = "yellow")
print(f fp_text (color="red", font.size = 12) )
```
**ftext**  
*formatted chunk of text*

**Description**

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

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

**Usage**

```r
ftext(text, prop)
```

**Arguments**

- `text`: text value, a string.
- `prop`: formatting text properties returned by **fp_text**.

**See Also**

- **fp_text**
- Other run functions for reporting: **external_img()**, **run_linebreak()**, **run_pagebreak()**

**Examples**

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

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

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

---

**get_reference_value**  
*Get the document being used as a template*

**Description**

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

Usage

get_reference_value(format = NULL)

Arguments

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

Value

a name file

Author(s)

Noam Ross

layout_properties  slide layout properties

Description

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

Usage

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

Arguments

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

See Also

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

Examples

x <- read_pptx()
layout_properties ( x = x, layout = "Title Slide", master = "Office Theme" )
layout_properties ( x = x, master = "Office Theme" )
layout_properties ( x = x, layout = "Two Content" )
layout_properties ( x = x )
Description
get informations about slide layouts and master layouts into a data.frame. This function returns a
data.frame containing all layout and master names.

Usage
layout_summary(x)

Arguments
x an rpptx object

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

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

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

Usage
## 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() )

<table>
<thead>
<tr>
<th>length.rpptx</th>
<th>number of slides</th>
</tr>
</thead>
</table>

Description

Function `length` will return the number of slides.

Usage

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

Arguments

- `x` an rpptx object

See Also

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

Examples

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

---

媒体_extract

Extract media from a document object

Description

Extract files from an rdocx or rpptx object.

Usage

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

Arguments

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

Examples

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

move_slide move a slide

Description

move a slide in a pptx presentation

Usage

move_slide(x, index, to)

Arguments

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

Note

cursor is set on the last slide.

See Also

Other functions slide manipulation: add_slide(), on_slide(), remove_slide()

Examples

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

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

Details

Examples of manipulations are:

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

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

Author(s)

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

Other contributors:

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

See Also

https://davidgohel.github.io/officer/
Defunct Functions in Package officer

Usage

\texttt{ph\_from\_xml(...)}
\texttt{ph\_from\_xml\_at(...)}
\texttt{ph\_with\_table(...)}
\texttt{ph\_with\_img(...)}
\texttt{ph\_with\_gg(...)}
\texttt{ph\_with\_ul(...)}
\texttt{ph\_with\_table\_at(...)}
\texttt{ph\_with\_fpars\_at(...)}
\texttt{body\_end\_section(...)}
\texttt{body\_default\_section(...)}
\texttt{break\_column\_before(...)}

Arguments

... unused arguments

Details

\texttt{ph\_from\_xml()} is replaced by \texttt{ph\_with.xml\_document}.
\texttt{ph\_from\_xml\_at()} is replaced by \texttt{ph\_with.xml\_document}.
\texttt{ph\_with\_table()} is replaced by \texttt{ph\_with.xml\_document}.
\texttt{ph\_with\_img()} is replaced by \texttt{ph\_with.xml\_document}.
\texttt{ph\_with\_gg()} is replaced by \texttt{ph\_with.xml\_document}.
\texttt{ph\_with\_ul()} is replaced by \texttt{ph\_with.xml\_document}.
\texttt{ph\_with\_table\_at()} is replaced by \texttt{ph\_with.xml\_document}.
\texttt{ph\_with\_fpars\_at()} is replaced by \texttt{ph\_with.xml\_document}.
on_slide

body_end_section() is replaced by function body_end_section_.*.
body_default_section() is replaced by function body_end_section_.*.
break_column_before() is replaced by function slip_in_column_break.

---

**on_slide**

*change current slide*

**Description**

change current slide index of an rpptx object.

**Usage**

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

**Arguments**

- `x` an rpptx object
- `index` slide index

**See Also**

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

**Examples**

```r
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"))

doc <- on_slide( doc, index = 3)

tmpfile(fileext = ".pptx")
print(doc, target = file )
```
**pack_folder**

*compress a folder*

---

**Description**

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

**Usage**

pack_folder(folder, target)

**Arguments**

- **folder**: folder to compress
- **target**: path of the archive to create

---

**page_mar**

*page margins object*

---

**Description**

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

**Usage**

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

**Arguments**

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

Examples

page.mar()

---

page_size  

**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'.

**Examples**

page.size(orient = "landscape")

---

ph_add_fpar  

**Description**

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

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

**Arguments**

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

**Usage**

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

**See Also**

`fpar`

**Examples**

```r
library(magrittr)

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

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

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

Usage

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

Arguments

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

Usage

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

Examples

library(magrittr)

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

doc <- read_pptx() %>%
  add_slide(layout = "Title and Content", master = "Office Theme") %>%
  ph_with("", location = ph_location(bg = "wheat", newlabel = "myph")) %>%
  ph_add_fpar(value = fpar_, ph_label = "myph", level = 2)

print(doc, target = tempfile(fileext = ".pptx"))
```r
ph_add_par(level = 2) %>%
  ph_add_text(str = "and another, ", style = default_text) %>%
ph_add_par(level = 3) %>%
  ph_add_text(str = "and another!",
    style = update(default_text, color = "blue"))

print(doc, target = fileout)
```

---

### Description

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

### Usage

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

### Arguments

- **x**: an rpptx object
- **str**: text to add
- **type**: placeholder type
- **id**: placeholder index (integer) for a duplicated type. This is to be used when a placeholder type is not unique in the layout of the current slide, e.g. two placeholders with type 'body'. To add onto the first, use `id = 1` and `id = 2` for the second one. Values can be read from `slide_summary`.
- **id_chr**: deprecated.
- **ph_label**: label associated to the placeholder. Use column `ph_label` of result returned by `slide_summary`.
- **style**: text style, a `fp_text` object
- **pos**: where to add the new element relative to the cursor, "after" or "before".
add a new empty shape in the current slide. This function is deprecated, function `ph_with` should be used instead.

```r
ph_empty
```

**Description**

- **href**: hyperlink to reach when clicking the text
- **slide_index**: slide index to reach when clicking the text. It will be ignored if `href` is not NULL.

**Usage**

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

**Examples**

```r
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Un titre 2",
location = ph_location_type(type = "title"))
doc <- ph_with(doc, ",",
location = ph_location(rotation = 90, bg = "red",
newlabel = "myph"))
doc <- ph_add_text(doc, str = "dummy text",
ph_label = "myph")
print(doc, target = fileout)
```

```r
doc <- read_pptx()
doc <- add_slide(doc)
doc <- ph_with(doc, "Un titre 2",
location = ph_location_type(type = "title"))
doc <- ph_with(doc, "",
location = ph_location(rotation = 90, bg = "red",
newlabel = "myph"))
doc <- ph_add_text(doc, str = "dummy text",
ph_label = "myph")
print(doc, target = fileout)
```
Usage

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

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

Arguments

- **x**: an pptx object
- **type**: placeholder type (i.e. 'body', 'title')
- **index**: placeholder index (integer). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body', the first one will be added with index 1 and the second one with index 2. It is recommended to use argument `location` instead of `type` and `index`.
- **location**: a placeholder location object. This is a convenient argument that can replace usage of arguments `type` and `index`. See `ph_location_type`, `ph_location`, `ph_location_label`, `ph_location_left`, `ph_location_right`, `ph_location_fullsize`.
- **left, top**: location of the new shape on the slide
- **width, height**: shape size in inches
- **bg**: background color
- **rot**: rotation angle
- **template_type**: placeholder template type. If used, the new shape will inherit the style from the placeholder template. If not used, no text property is defined and for example text lists will not be indented.
- **template_index**: placeholder template index (integer). To be used when a placeholder template type is not unique in the current slide, e.g. two placeholders with type 'body'.

```
ph_hyperlink
```

hyperlink a placeholder

Description

add hyperlink to a placeholder in the current slide.
Usage

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

Arguments

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

See Also

`ph_with`

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

Examples

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

---

**ph_location**

create a location for a placeholder

Description

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

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

Arguments

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

Details

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

- `left`: left coordinate of the bounding box
- `top`: top coordinate of the bounding box
- `width`: width of the bounding box
- `height`: height of the bounding box

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

See Also

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

Examples

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

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

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

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

See Also
Other functions for placeholder location: ph_location_label(), ph_location_left(), ph_location_right(), ph_location_template(), ph_location_type(), ph_location()

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

ph_location_label  location of a named placeholder

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

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

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

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

- **left** left coordinate of the bounding box
- **top** top coordinate of the bounding box
- **width** width of the bounding box
- **height** height of the bounding box

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

See Also

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

Examples

```r
# 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** | **location of a left body element** |

Description

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

\texttt{ph\_location\_left(newlabel = NULL, ...)}

Arguments

\begin{description}
\item [newlabel] a label to associate with the placeholder.
\item [...] unused arguments
\end{description}

See Also

Other functions for placeholder location: \texttt{ph\_location\_fullsize()}, \texttt{ph\_location\_label()}, \texttt{ph\_location\_right()}, \texttt{ph\_location\_template()}, \texttt{ph\_location\_type()}, \texttt{ph\_location()}

Examples

\begin{verbatim}
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") )
\end{verbatim}

\begin{verbatim}
ph\_location\_right location of a right body element
\end{verbatim}

Description

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

Usage

\texttt{ph\_location\_right(newlabel = NULL, ...)}

Arguments

\begin{description}
\item [newlabel] a label to associate with the placeholder.
\item [...] unused arguments
\end{description}

See Also

Other functions for placeholder location: \texttt{ph\_location\_fullsize()}, \texttt{ph\_location\_label()}, \texttt{ph\_location\_left()}, \texttt{ph\_location\_template()}, \texttt{ph\_location\_type()}, \texttt{ph\_location()}
Examples

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

---

**ph_location_template**

Create a location for a placeholder based on a template

**Description**

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

**Usage**

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

**Arguments**

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

**See Also**

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

**Examples**

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

---

**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**

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

- **type**
  - Placeholders 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**

```r
# ph_location_type demo ----

loc_title <- ph_location_type(type = "title")
loc_footer <- ph_location_type(type = "ftr")
loc_dt <- ph_location_type(type = "dt")
loc_slidenum <- ph_location_type(type = "sldNum")
loc_body <- ph_location_type(type = "body")

doc <- read_pptx()
```
```r
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**

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

**Arguments**

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

**See Also**

- `ph_with`

Other functions for placeholders manipulation: `ph_hyperlink()`, `ph_slidelink()`
Examples

```r
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
}
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**

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

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

See Also

- `ph_with`

Other functions for placeholders manipulation: `ph_hyperlink()`, `ph_remove()`

Examples

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

print(doc, target = fileout )
```

Description

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

ph_with(x, value, 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, is_list = FALSE, ...)

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

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

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

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

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

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

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

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

### Arguments

- **x**: an rpptx object
- **value**: object to add as a new shape. Supported objects are vectors, data.frame, graphics, block of formatted paragraphs, unordered list of formatted paragraphs, pretty tables with package flextable, editable graphics with package rvg, 'Microsoft' charts with package mschart.
- **location**: a placeholder location object. It will be used to specify the location of the new shape. This location can be defined with a call to one of the ph_location functions. See section "see also".
- **...**: further arguments passed to or from other methods. When adding a ggplot object or plot_instr, these arguments will be used by png function.
- **format_fun**: format function for non character vectors
- **is_list**: experimental paramater to make block_list formated as an unordered list. This should evolve in the next versions.
- **header**: display header if TRUE
- **first_row, last_row, first_column, last_column**: logical for PowerPoint table options
- **res**: resolution of the png image in ppi
- **use_loc_size**: if set to FALSE, external_img width and height will be used.

### Methods (by class)

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

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

• `empty_content`: add an `empty_content` to a new shape on the current slide.

• `xml_document`: add an `xml_document` object to a new shape on the current slide. This function is to be used to add custom openxml code.

See Also

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

Examples

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

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

doc <- add_slide(doc)
doc <- ph_with(
  doc, anyplot,
  location = ph_location_fullsize(),
  bg = "#00000066", pointsize = 12)

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

doc <- add_slide(doc, layout = "Title and Content",
master = "Office Theme")
img.file <- file.path( R.home("doc"), "html", "logo.jpg" )
doc <- ph_with(x = doc, external_img(img.file, 100/72, 76/72),
location = ph_location_right(), use_loc_size = FALSE )

svg_file <- file.path(R.home(component = "doc"), "html/Rlogo.svg")
```
if( require("rsvg") ){  
  doc <- ph_with(x = doc, external_img(svg_file),  
               location = ph_location_left(),  
               use_loc_size = TRUE )  
}

# block list -----
bl <- block_list(  
  fpar(ftext("hello world", shortcuts$fp_bold(color = "pink"))),  
  fpar(    
    ftext("hello", shortcuts$fp_bold()),  
    ftext("hello", shortcuts$fp_italic(color="red"))  
  )
)

doc <- add_slide(doc)
doc <- ph_with(x = doc, value = bl,  
               location = ph_location_type(type="body") )

# fpar ------
hw <- fpar(ftext("hello world", shortcuts$fp_bold(color = "pink")))
doc <- add_slide(doc)
doc <- ph_with(x = doc, value = hw,  
               location = ph_location_type(type="body") )

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

doc <- add_slide(doc)
doc <- ph_with(x = doc, value = ul,  
               location = ph_location_fullsize() )

print(doc, target = fileout)

---

### ph_with_gg_at

**add ggplot to a pptx presentation**

#### Description

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

#### Usage

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

#### Arguments

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

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

**Usage**

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

**Arguments**

- `x`: an pptx object
- `src`: image filename, the basename of the file must not contain any blank.
- `left`, `top`: location of the new shape on the slide
- `width`, `height`: image size in inches
- `rot`: rotation angle

**Description**

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

**Usage**

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

**Arguments**

- `x`: an pptx object
- `str`: text to add
- `type`: placeholder type (i.e. 'body', 'title')
- `index`: placeholder index (integer). This is to be used when a placeholder type is not unique in the current slide, e.g. two placeholders with type 'body', the first one will be added with index 1 and the second one with index 2. It is recommended to use argument `location` instead of type and index.
- `location`: a placeholder location object. This is a convenient argument that can replace usage of arguments `type` and `index`. See `ph_location_type`, `ph_location`, `ph_location_label`, `ph_location_left`, `ph_location_right`, `ph_location_fullsize`. 

```r
ph_location_type`, `ph_location`, `ph_location_label`, `ph_location_left`, `ph_location_right`, `ph_location_fullsize`.```
plot_instr

Description

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

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

Usage

```r
plot_instr(code)
```

Arguments

- `code` plotting instructions

See Also

`ph_with()`, `body_add()`

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

Examples

```r
# 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"))
```
### pptx_summary

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

**Description**

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

**Usage**

```r
pptx_summary(x)
```

**Arguments**

- `x` an rpptx object

**Examples**

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

---

### print.rpptx

**write a 'PowerPoint' file.**

**Description**

write a 'PowerPoint' file.

**Usage**

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

**Arguments**

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

**See Also**

- `read_pptx`
Examples

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

---

prop_section     section properties

Description

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

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

Usage

```r
prop_section(page_size, page_margins, type)
```

Arguments

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

See Also

`block_section`

Examples

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

**open a connection to a 'Word' file**

**Description**

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

**Usage**

`read_docx(path = NULL)`

```r
t## S3 method for class 'rdocx'
print(x, target = NULL, ...)
```

**Arguments**

- `path`: path to the docx file to use as base document.
- `x`: an rdocx object
- `target`: path to the docx file to write
- `...`: unused

**See Also**

`print.rdocx, body_add`

**Examples**

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

`print(read_docx())`

# write a rdocx object in a docx file ----
`if( require(magrittr) ){`
  `read_docx() %>% print(target = tempfile(fileext = ".docx"))`
`}`

---

**read_pptx**

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

**Description**

read and import a pptx file as an R object representing the document. The function is called `read_pptx` because it allows you to initialize an object of class `rpptx` from an existing PowerPoint file. Content will be added to the existing presentation. By default, an empty document is used.
Usage

read_pptx(path = NULL)

Arguments

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

master layouts and slide layouts

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

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

See Also

print.rpptx add_slide

Examples

read_pptx()

---

read_xlsx open a connexion to an 'Excel' file

Description

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

Usage

read_xlsx(path = NULL)

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

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

Arguments

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

Examples

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

Description

  remove a slide from a pptx presentation

Usage

  remove_slide(x, index = NULL)

Arguments

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

Note

  cursor is set on the last slide.

See Also

  Other functions slide manipulation: add_slide(), move_slide(), on_slide()

Examples

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

Create a string representation of a number

Usage

\texttt{run\_autonum(seq\_id = "table", pre\_label = "TABLE \", post\_label = ": \")}

Arguments

- seq\_id: sequence identifier
- pre\_label, post\_label: text to add before and after number

Examples

\texttt{run\_autonum()}
\texttt{run\_autonum(seq\_id = "fig", pre\_label = "fig. ")}

Description

Create a representation of a column break

Usage

\texttt{run\_columnbreak()}

Examples

\texttt{run\_columnbreak()}
run_linebreak

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

Usage

run_linebreak()

See Also

Other run functions for reporting: `external_img()`, `ftext()`, `run_pagebreak()`

Examples

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

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

run_pagebreak

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

Usage

run_pagebreak()

See Also

Other run functions for reporting: `external_img()`, `ftext()`, `run_linebreak()`
Examples

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

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

run_reference reference

Description
Create a representation of a reference

Usage
run_reference(id)

Arguments
id reference id, a string

Examples
run_reference('a_ref')

run_seqfield seqfield

Description
Create a seqfield

Usage
run_seqfield(seqfield)

Arguments
seqfield seqfield string
sanitize_images

remove unused media from a document

Description

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

Usage

sanitize_images(x)

Arguments

x  rdocx or rpptx object

sections

Description

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

Usage

body_end_section_continuous(x)

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

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

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

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

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

Details

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

Examples

```r
library(magrittr)

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

my_doc <- read_docx() %>%
  body_add_par(value = "Default section", style = "heading 1") %>%
  body_add_par(value = str1, style = "centered") %>%
  body_add_par(value = str2, style = "centered") %>%
  body_end_section_continuous() %>%
  body_add_par(value = "Landscape section", style = "heading 1") %>%
  body_add_par(value = str1, style = "centered") %>%
  body_add_par(value = str2, style = "centered") %>%
  body_end_section_landscape() %>%
  body_add_par(value = "Columns", style = "heading 1") %>%
  body_end_section_continuous() %>%
  body_add_par(value = str1, style = "centered") %>%
  body_add_par(value = str2, style = "centered") %>%
  slip_in_column_break() %>%
  body_add_par(value = str1, style = "centered") %>%
  body_end_section_columns(widths = c(2,2), sep = TRUE, space = 1) %>%
  body_add_par(value = str1, style = "Normal") %>%
  body_add_par(value = str2, style = "Normal") %>%
  slip_in_column_break() %>%
  body_end_section_columns_landscape(widths = c(3,3), sep = TRUE, space = 1)

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

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

Usage

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

Arguments

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

Note

The "last modified" and "last modified by" fields will be automatically be updated when the file is written.

See Also

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

Examples

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

**Description**
set a particular sheet selected when workbook will be edited.

**Usage**
`sheet_select(x, sheet)`

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

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

---

**shortcuts**  
*shortcuts for formatting properties*

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

**Usage**
`shortcuts`

**Examples**
```r
shortcuts$fp_bold()  
shortcuts$fp_italic()  
shortcuts$b_null()
```
slide_size

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

slide_summary

Example:
my_pres <- read_pptx()
my_pres <- add_slide(my_pres,
                                 layout = "Two Content", master = "Office Theme")
slide_summary(my_pres)
Note

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

See Also

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

Examples

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

Description

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

Usage

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

Arguments

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

append a footnote

Description
append a new footnote into a paragraph of an rdocx object

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

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

Examples
library(magrittr)

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

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

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

slip_in_img
append an image

Description
append an image into a paragraph of an rdocx object
Usage

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

Arguments

- **x**: an `rdocx` object
- **src**: image filename, the basename of the file must not contain any blank.
- **style**: text style
- **width**: height in inches
- **height**: height in inches
- **pos**: where to add the new element relative to the cursor, "after" or "before".

Examples

```r
library(magrittr)
img.file <- file.path( R.home("doc"), "html", "logo.jpg" )
x <- read_docx() %>%
  body_add_par("R logo: ", style = "Normal") %>%
  slip_in_img(src = img.file, style = "strong", width = .3, height = .3)
print(x, target = tempfile(fileext = ".docx"))
```

---

**slip_in_seqfield append seq field**

Description

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

Usage

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

Arguments

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

```r
library(magrittr)
x <- read_docx() %>%
  body_add_par("Time is: ", style = "Normal") %>%
  slip_in_seqfield(
    str = "TIME \u005C@ "HH:mm:ss" \u005C* MERGEFORMAT",
    style = 'strong') %>%
  body_add_par(" - This is a figure title", style = "centered") %>%
  slip_in_seqfield(str = "SEQ Figure \u005C* roman",
    style = 'Default Paragraph Font', pos = "before") %>%
  slip_in_text("Figure: ", style = "strong", pos = "before") %>%
  body_add_par(" - This is another figure title", style = "centered") %>%
  slip_in_seqfield(str = "SEQ Figure \u005C* roman",
    style = 'strong', pos = "before") %>%
  slip_in_text("Figure: ", style = "strong", pos = "before") %>%
  body_add_par("This is a symbol: ", style = "Normal") %>%
  slip_in_seqfield(str = "SYMBOL 100 \u005Cf Wingdings",
    style = 'strong')

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

---

**slip_in_text**

*append text*

---

**Description**

append text into a paragraph of an rdocx object

**Usage**

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

**Arguments**

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

```r
library(magrittr)
x <- read_docx() %>%
  body_add_par("Hello ", style = "Normal") %>%
  slip_in_text("world", style = "strong") %>%
  slip_in_text("Message is", style = "strong", pos = "before") %>%
  slip_in_text("with a link", style = "strong",
              pos = "after", hyperlink = "https://davidgohel.github.io/officer/")

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

---

**slip_in_xml**

*add a wml string into a Word document*

**Description**

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

**Usage**

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

**Arguments**

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

---

**styles_info**

*read Word styles*

**Description**

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

**Usage**

```r
styles_info(x)
```

**Arguments**

- `x` an rdocx object

**See Also**

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

```r
x <- read_docx()
styles_info(x)
```

---

### to_html

Convert officer objects to HTML

#### Description

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

#### Usage

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

#### Arguments

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

---

### to_pml

Convert officer objects to PresentationML

#### Description

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

#### Usage

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

#### Arguments

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

Convert officer objects to WordprocessingML

Description

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

Usage

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

Arguments

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

unordered_list

unordered list

Description

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

Usage

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

Arguments

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

See Also

ph_with

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

Extract files from a zip file

Description

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

Usage

unpack_folder(file, folder)

Arguments

file path of the archive to unzip
folder folder to create

wml_link_images transform an xml string with images references

Description

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

Usage

wml_link_images(x, str)

Arguments

x an rdocx object
str wml string
Details

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

add_sheet, 4
add_slide, 5, 47, 50, 78, 79
annotate_base, 5, 27, 44–46, 87, 88

block_caption, 6, 7, 9, 10, 13, 36, 74, 94
block_list, 6, 7, 9, 10, 13, 36, 53, 54, 56, 70, 71, 74, 89, 94
block_section, 6, 7, 8, 10, 36, 74, 76, 94
block_toc, 6, 7, 9, 9, 10, 13, 36, 74, 94
body_add, 11, 35, 77
body_add_blocks, 15
body_add_blocks(), 34
body_add_break, 14
body_add_docx, 15
body_add_fpar, 16
body_add_gg, 17
body_add_img, 17
body_add_par, 18
body_add_table, 19
body_add_toc, 20
body_add_xml, 21
body_bookmark, 21
body_default_section (officer-defunct), 49
body_end_section (officer-defunct), 49
body_end_section_columns (sections), 83
body_end_section_columns_landscape (sections), 83
body_end_section_continuous (sections), 83
body_end_section_landscape (sections), 83
body_end_section_portrait (sections), 83
break_column_before (officer-defunct), 49
body_remove, 22
body_replace_all_text (body_replace_all_text), 22
body_replace_all_text (body_replace_text_at_bkm), 25
body_replace_img_at_bkm (body_replace_text_at_bkm), 25
body_replace_text_at_bkm (body_replace_text_at_bkm), 25
change_styles, 26
color_scheme, 6, 26, 44–46, 87, 88
cursor_backward (cursor_begin), 27
cursor_begin, 27
cursor_bookmark (cursor_begin), 27
cursor_end (cursor_begin), 27
cursor_forward (cursor_begin), 27
cursor_reach (cursor_begin), 27
doc_properties, 30, 31, 33, 45, 85, 92
docx_body_relationship, 29
docx_body_xml, 29
docx_bookmarks, 30, 31, 33, 45, 85, 92
docx_dim, 30, 31, 33, 45, 85, 92
docx_reference_img, 31
docx_show_chunk, 23, 24, 32

docx_summary, 32

docx_body_xml, 29

empty_content, 34, 71
external_img, 7, 13, 34, 43, 70, 81

footers_replace_all_text
(body_replace_all_text), 22
footers_replace_img_at_bkm
(body_replace_text_at_bkm), 25
footers_replace_text_at_bkm
(body_replace_text_at_bkm), 25
format.fp_cell (fp_cell), 38
format.fp_text (fp_text), 41
fortify_location, 35
fp_border, 37, 39, 40
fp_cell, 38
fp_par, 39
fp_text, 41, 43, 55
fpar, 6, 7, 9, 10, 13, 16, 35, 36, 42, 43, 52, 53, 70, 71, 74, 81, 94
ftext, 35, 42, 43, 81
get_reference_value, 43
grep, 24
grep, 23
gsub, 23
headers_replace_all_text
  (body_replace_all_text), 22
headers_replace_img_at_bkm
  (body_replace_text_at_bkm), 25
headers_replace_text_at_bkm
  (body_replace_text_at_bkm), 25
layout_properties, 6, 27, 44, 45, 46, 87, 88
layout_summary, 5, 6, 27, 44, 45, 46, 87, 88
length.rdocx, 30, 31, 33, 45, 85, 92
length.rpptx, 6, 27, 44, 45, 46, 87, 88
length.xlsx (read.xlsx), 78
media_extract, 46
move_slide, 5, 47, 50, 79
officer, 48
officer-defunct, 49
officer-package (officer), 48
on_slide, 5, 47, 50, 79
pack_folder, 51
page_mar, 51, 76
page_size, 52, 76
ph_add_fpar, 52
ph_add_par, 54
ph_add_text, 55
ph_empty, 56
ph_empty_at (ph_empty), 56
ph_from_xml (officer-defunct), 49
ph_from_xml_at (officer-defunct), 49
ph_hyperlink, 57, 66, 68
ph_location, 36, 57, 58, 57, 58, 60–62, 64, 65, 71, 73
ph_location_fullsize, 57, 59, 60, 61, 62,
  64, 65, 71, 73
ph_location_label, 57, 59, 60, 62, 64,
  71, 73
ph_location_left, 57, 59–61, 61, 62, 64, 65,
  71, 73
ph_location_right, 57, 59–62, 62, 64, 65,
  71, 73
ph_location_template, 59–62, 63, 65, 71
ph_location_type, 57, 59–62, 64, 65, 71, 73
ph_remove, 58, 66, 68
ph_slideml, 58, 66, 67
ph_with, 5, 35, 36, 53, 54, 56, 58, 66, 68, 68,
  72, 73, 74
ph_with(), 7, 34, 74
ph_with_fpars_at (officer-defunct), 49
ph_with_gg (officer-defunct), 49
ph_with_gg_at, 72
ph_with_img (officer-defunct), 49
ph_with_img_at, 73
ph_with_table (officer-defunct), 49
ph_with_table_at (officer-defunct), 49
ph_with_text, 73
ph_with_ul (officer-defunct), 49
plot_instr, 6, 7, 9, 10, 36, 74, 94
pptx_summary, 75
print.fp_cell (fp_cell), 38
print.fp_par (fp_par), 39
print.fp_text (fp_text), 41
print.rdocx, 77
print.rdocx (read_docx), 77
print.rpptx, 5, 75, 78
print.xlsx (read.xlsx), 78
prop_section, 9, 76
read_docx, 12, 77
read_pptx, 5, 75, 77
read.xlsx, 78
regex, 23, 24
remove_slide, 5, 47, 50, 79
run_autonum, 6, 80
run_columnbreak, 13, 80
run_linebreak, 35, 43, 51, 81
run_pagebreak, 13, 35, 43, 81
run_reference, 82
run_seqfield, 82
sanitize_images, 83
sections, 83
set_doc_properties, 30, 31, 33, 45, 85, 92
sheet_select, 86
shortcuts, 86
slide_size, 6, 27, 44–46, 87, 88
slide_summary, 6, 27, 44–46, 53–55, 58, 66,
  68, 87, 88
slip_in_column_break, 88
slip_in_footnote, 89
slip_in_img, 89
slip_in_seqfield, 90
slip_in_text, 91
slip_in_xml, 92
styles_info, 12, 30, 31, 33, 45, 85, 92

to_html, 93
to_pml, 93
to_wml, 94

unordered_list, 6, 7, 9, 10, 36, 70, 74, 94
unpack_folder, 95
update.fp_border (fp_border), 37
update.fp_cell (fp_cell), 38
update.fp_par (fp_par), 39
update.fp_text (fp_text), 41
update.fpar (fpar), 36

wml_link_images, 31, 95