

Package: officerExtras (via r-universe)

October 15, 2026

Title Extra Helpers for 'officer'

Version 0.0.1.9003

Description Helper and convenience functions using the 'officer' package to modify docx files.

License MIT + file LICENSE

URL <https://github.com/elipousson/officerExtras>,
<https://elipousson.github.io/officerExtras/>

BugReports <https://github.com/elipousson/officerExtras/issues>

Depends R (>= 4.1.0)

Imports cli (>= 3.4.0), glue, grDevices, lifecycle, officer (>= 0.6.3), rlang, tibble, utils, vctrs, xml2

Suggests brand.yml, covr, flextable, ggplot2, gt (>= 0.7.0), rmarkdown, testthat (>= 3.0.0), withr, yaml12

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Encoding UTF-8

Language en

Roxygen list(markdown = TRUE)

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libfribidi-dev libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libwebp-dev libxml2-dev libssl-dev

Repository <https://elipousson.r-universe.dev>

Date/Publication 2026-10-15 02:08:59 UTC

RemoteUrl <https://github.com/elipousson/officerExtras>

RemoteRef HEAD

RemoteSha c947e987987cdab719e2ceca4d92a3571990a49e

Contents

add_list_to_body	2
add_to_body	3
brand_reference_docx	7
check_office_fileext	10
check_officer	11
check_officer_summary	12
combine_docx	13
convert_docx	15
cursor_docx	16
dims_docx_ext	17
docx_sync_sections	18
fill_with_pattern	19
is_officer	20
make_block_list	21
officer_add_blocks	22
officer_media	23
officer_open	24
officer_properties	25
officer_summary	26
officer_summary_levels	27
officer_tables	28
read_docs_url	30
read_officer	31
reduce_officer	33
set_flextable_brand_defaults	34
use_doc_version	35
vec_add_to_body	37
write_officer	40
Index	41

add_list_to_body	<i>Add a formatted list of items to a rdocx object</i>
------------------	--

Description

This function assumes that the input docx object has a list style. Use `read_docx_ext()` with `allow_null = TRUE` to create a new rdocx object with the "List Bullet" and "List Number" styles. "List Paragraph" can be used but the resulting formatted list will not have bullets or numbers.

Usage

```
add_list_to_body(
  docx,
  values,
  style = "List Bullet",
  keep_na = FALSE,
  before = NULL,
  after = "",
  ...
)
```

Arguments

docx	A rdocx object.
values	Character vector with values for list.
style	Style to use for list of values, Default: 'List Bullet'
keep_na	If TRUE, keep NA values in values Default: FALSE
before	Value to insert before the list, Default: NULL
after	Value to insert after the list, Default: ""
...	Additional parameters passed to cursor_docx()

Value

A modified rdocx object.

See Also

[add_to_body\(\)](#), [vec_add_to_body\(\)](#)

add_to_body	<i>Add a xml string, text paragraph, or gt object at a specified position in a rdocx object</i>
-------------	---

Description

Wrappers for [officer::body_add_par\(\)](#), [officer::body_add_gg\(\)](#), and [officer::body_add_xml\(\)](#) that use the [cursor_docx\(\)](#) helper function to allow users to pass the value and keyword, id, or index value used to place a "cursor" within the document using a single function. If pos = NULL, [add_to_body\(\)](#) calls [officer::body_add\(\)](#) instead of [officer::body_add_par\(\)](#). If value is a gt_tbl object, value is passed as the gt_object parameter for [add_gt_to_body\(\)](#).

- [add_text_to_body\(\)](#) passes value to [glue::glue\(\)](#) to add support for glue string interpolation.
- [add_gt_to_body\(\)](#) converts gt tables to OOXML with [gt::as_word\(\)](#).
- [add_gg_to_body\(\)](#) adds a caption following the plots using the labels from the plot object.

Usage

```
add_to_body(  
  docx,  
  keyword = NULL,  
  id = NULL,  
  index = NULL,  
  value = NULL,  
  str = NULL,  
  style = NULL,  
  pos = "after",  
  ...,  
  gt_object = NULL,  
  call = caller_env()  
)
```

```
add_text_to_body(  
  docx,  
  value,  
  style = NULL,  
  pos = "after",  
  .na = "NA",  
  .null = NULL,  
  .envir = parent.frame(),  
  ...  
)
```

```
add_xml_to_body(docx, str, pos = "after", ..., call = caller_env())
```

```
add_gt_to_body(  
  docx,  
  gt_object,  
  align = "center",  
  caption_location = c("top", "bottom", "embed"),  
  caption_align = "left",  
  split = FALSE,  
  keep_with_next = TRUE,  
  pos = "after",  
  tablecontainer = TRUE,  
  ...,  
  call = caller_env()  
)
```

```
add_gg_to_body(  
  docx,  
  value,  
  caption = "title",  
  caption_style = style,  
  autonum = NULL,
```

```

    style = "Normal",
    pos = "after",
    ...
)

add_value_with_keys(docx, value, ..., .f = add_text_to_body)

add_str_with_keys(docx, str, ..., .f = add_xml_to_body)

```

Arguments

docx	A rdocx object.
keyword, id	A keyword string used to place cursor with <code>officer::cursor_reach()</code> or bookmark id with <code>officer::cursor_bookmark()</code> . Defaults to NULL. If keyword or id are not provided, the gt object is inserted at the front of the document.
index	A integer matching a doc_index value appearing in a summary of the docx object created with <code>officer::docx_summary()</code> . If index is for a paragraph value, the text of the paragraph is used as a keyword.
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.
str	a wml string
style	paragraph style name. These names are available with function <code>styles_info</code> and are the names of the Word styles defined in the base document (see argument path from <code>read_docx</code>).
pos	where to add the new element relative to the cursor, one of "after", "before", "on".
...	Additional parameters passed to <code>officer::body_add_par()</code> , <code>officer::body_add_gg()</code> , or <code>officer::body_add()</code> .
gt_object	A gt object converted to an OOXML string with <code>gt::as_word()</code> then passed to <code>add_xml_to_body()</code> as str parameter. Required for <code>add_gt_to_body()</code> .
call	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the call argument of <code>abort()</code> for more information.
.na	[character(1): 'NA'] Value to replace NA values with. If NULL missing values are propagated, that is an NA result will cause NA output. Otherwise the value is replaced by the value of .na.
.null	[character(1): 'character()'] Value to replace NULL values with. If <code>character()</code> whole output is <code>character()</code> . If NULL all NULL values are dropped (as in <code>paste0()</code>). Otherwise the value is replaced by the value of .null.
.envir	[environment: parent.frame()] Environment to evaluate each expression in. Expressions are evaluated from left to right. If .x is an environment, the expressions are evaluated in that environment and .envir is ignored. If NULL is passed, it is equivalent to <code>emptyenv()</code> .

align	<i>Table alignment</i> scalar<character> // default: "center" An option for table alignment. Can either be "center", "left", or "right".
caption_location	<i>Caption location</i> singl-kw: [top bottom embed] // default: "top" Determines where the caption should be positioned. This can either be "top", "bottom", or "embed".
caption_align	<i>Caption alignment</i> Determines the alignment of the caption. This is either "left" (the default), "center", or "right". This option is only used when caption_location is not set as "embed".
split	<i>Allow splitting of a table row across pages</i> scalar<logical> // default: FALSE A logical value that indicates whether to activate the Word option Allow row to break across pages.
keep_with_next	<i>Keeping rows together</i> scalar<logical> // default: TRUE A logical value that indicates whether a table should use Word option Keep rows together.
tablecontainer	If TRUE (default), add tables inside of a tablecontainer tag that automatically adds a table number and converts the gt title into a table caption. This feature is based on code from the gto package by Ellis Hughes to transform the gt_object to OOXML and insert the XML into the docx object.
caption	Name of the ggplot2 label to use as a caption if plot passed to value has a label for this value. Defaults to "title".
caption_style	Passed to style for officer::body_add_caption(). Defaults to same value as style.
autonum	<i>Automatic Table Numbering</i> scalar<logical> // default: TRUE A logical value that indicates whether a table should use Words built-in auto table numbering option in the caption. Automatic Table Numbering.
.f	Any function that takes a docx and value parameter and returns a rdocx object. A keyword parameter must also be supported if named is TRUE. Defaults to add_text_to_body().

Details

Using [add_value_with_keys\(\)](#) or [add_str_with_keys\(\)](#)

[add_value_with_keys\(\)](#) supports value vectors of length 1 or longer. If value is named, the names are assumed to be keywords indicating the cursor position for adding each value in the vector. If value is not named, a keyword parameter with the same length as value must be provided. When named = FALSE, no keyword parameter is required. Add [add_str_with_keys\(\)](#) works identically but uses a str parameter and .f defaults to [add_xml_to_body\(\)](#).

Note, as of July 2023, both [add_value_with_keys\(\)](#) and [add_str_with_keys\(\)](#) are superseded by [vec_add_to_body\(\)](#).

Value

A rdocx object with xml, gt tables, or paragraphs of text added.

Author(s)

Ellis Hughes <ellis.h.hughes@gsk.com> ([ORCID](#))

Examples

```
if (rlang::is_installed("gt")) {
  docx_example <- read_docx_ext(
    filename = "example.docx",
    path = system.file("doc_examples", package = "officer")
  )

  tab_1 <-
    gt::gt(
      gt::exibble,
      rowname_col = "row",
      groupname_col = "group"
    )

  add_gt_to_body(
    docx_example,
    tab_1,
    keyword = "Title 1"
  )

  tab_str <- gt::as_word(tab_1)

  add_str_with_keys(
    docx_example,
    str = c("Title 1" = tab_str, "Title 2" = tab_str)
  )
}
```

brand_reference_docx *Create a Word reference document from a brand.yml file*

Description

brand_reference_docx() creates a Word document for use as a Quarto or Pandoc reference document (the reference-doc option for docx output) using the colors and typography from a [brand.yml](#) file.

By default, the function starts from Pandoc's default reference.docx (so every style Pandoc uses exists) and merges the brand settings into the existing style definitions. Styles are edited with xml2, rather than with [officer::docx_set_paragraph_style\(\)](#), because that function replaces a style wholesale and drops properties the brand doesn't set (e.g. the heading outline levels Word uses for the table of contents).

`pandoc_reference_docx()` writes Pandoc's default reference.docx to a file using Quarto's bundled Pandoc (if Quarto is installed) or Pandoc.

Usage

```
brand_reference_docx(
  brand = NULL,
  path = NULL,
  reference = NULL,
  overwrite = TRUE,
  quiet = FALSE
)

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

Arguments

brand	Path to a brand.yml file or a directory containing one, a brand_yml object from <code>brand.yml::read_brand_yml()</code> , or a list with the brand settings (e.g. from <code>yaml12::read_yaml()</code>). If NULL (default), the path is taken from the BRAND_YML_PATH environment variable or, if that is not set, found by searching the working directory and its parent directories for <code>_brand.yml</code> or <code>_brand.yaml</code> , either directly or in a brand or _brand subdirectory (the same search used by Quarto and <code>brand.yml::read_brand_yml()</code>).
path	Optional file path for the output docx file. If NULL (default), the document is not written to a file.
reference	Optional path to a docx file or a rdocx object to use as the starting reference document. If NULL (default), Pandoc's default reference.docx is used (which requires Quarto or Pandoc). Styles named in <code>defaults.docx.styles</code> must exist in the reference document. Note that a rdocx object is modified in place (as with other officer functions).
overwrite	If TRUE (default), remove file at path if it already exists. If FALSE and file exists, this function aborts.
quiet	If TRUE, suppress the message reporting the path of the written file. Defaults to FALSE.

Value

`brand_reference_docx()` returns a rdocx object (invisibly if path is supplied). `pandoc_reference_docx()` returns path.

Brand settings

The following brand.yml settings are used:

- `color`: palette and color roles (e.g. primary or foreground). Wherever a color is used, it can be a hex value, a CSS color name, an `rgb()` value, a palette name, or a color role.
- `typography.base`: family, size, weight, and line-height for the document defaults and the Normal style.

- `typography.headings`: family, weight, style, color, and line-height for the Heading 1-9, TOC Heading, and Title styles.
- `typography.link`: color, background-color, weight, and decoration for the Hyperlink style.
- `typography.monospace-inline` (inheriting from `typography.monospace`): family, size, weight, color, and background-color for the Verbatim Char style.
- `typography.monospace-block` (inheriting from `typography.monospace`): background-color (as paragraph shading), line-height, color, family, size, and weight for the Source Code style used for code blocks. Pandoc's default reference document has no Source Code style, so it is created if `monospace-block` sets background-color, line-height, or color; otherwise, Pandoc creates the style when rendering (with shading from the highlight style).

Pandoc formats the text in code blocks with the Verbatim Char style (or syntax highlighting styles based on it), which takes precedence over the Source Code style. As a result, `monospace-block` values for family, size, weight, and color only apply if `monospace-inline` doesn't set a different value, and highlighted code keeps the token colors and shading from the highlight style (set with the Quarto `highlight-style` option).

`brand.yml` has no Word-specific options, so additional settings are read from `defaults.docx` (the `brand.yml` section for format-specific options):

- `defaults.docx.styles`: a list of style settings named by style ID (e.g. `Heading1` or `BodyText`) or style name (e.g. `Body Text`). Each style supports the `typography` fields family, size, weight, style, color, background-color, decoration, and line-height plus the paragraph fields `space-before`, `space-after`, `indent-left`, and `keep-with-next`. The Table style supports `border-color`, `border-width`, `cell-padding` (with top, bottom, left, and right), and `header` (`typography` fields for the first row).
- `defaults.docx.page`: size ("letter", "legal", or "a4"), margins (with top, bottom, left, right, header, footer, and gutter), and footer (with text, align, size, color, and family). In the footer text, `{page}` and `{pages}` are replaced with the page number and page count. Margins that aren't set keep the reference document's value (or Word's default of 1in for the page and 0.5in for the header and footer).

Field names can use hyphens (as in the `brand.yml` spec) or underscores (as in objects from `brand.yml::read_brand_yaml()`).

Sizes can use pt, in, cm, mm, px, rem, or em units (bare numbers are treated as pt). Following CSS, 1rem and 1em are 16px (12pt). A unitless line-height is a multiple of the font size; a line-height with units sets an exact line height. Font weights can be numbers (100-900) or CSS weight names (e.g. "semi-bold"); a weight of 600 or more is bold.

These `brand.yml` settings are not used: `color.background` (Word documents don't have a page color by default), `typography.fonts` (fonts are not embedded, so the font families must be installed wherever the document is opened), and `decoration: overline` (Word has no overline).

See `system.file("extdata", "_brand.yml", package = "officerExtras")` for an example.

Table borders and other details that the brand doesn't cover can be edited by hand in Word afterwards (but re-creating the reference document overwrites those edits).

Page settings and sections

The defaults.docx page settings (page size, margins, and footer) are stored in the reference document's section properties. Pandoc copies them to the last section of the rendered document, so they don't always apply to every page:

- Quarto's .landscape divs add section breaks with no footer, no margins, and an A4 landscape page size, so the pages before and in a landscape section lose the footer and page settings.
- A custom Pandoc OpenXML template (the `template` option for docx output, e.g. as used by the better-word Quarto extension) replaces the section properties if it doesn't use the `$sectpr$` variable.

Use `docx_sync_sections()` on the rendered document (e.g. in a Quarto post-render script) to copy the footer and page settings from the last section to every other section. The reference document only controls styles and these defaults, not the document structure (e.g. the title block or table of contents), which a custom template controls.

Examples

```
if (nzchar(Sys.which("quarto")) || nzchar(Sys.which("pandoc"))) {
  brand_path <- system.file("extdata", "_brand.yml", package = "officerExtras")

  withr::with_tempdir({
    brand_reference_docx(brand_path, path = "reference.docx")
  })
}
```

check_office_fileext *Check file path for a docx, pptx, or xlsx file extension*

Description

Check file path for a docx, pptx, or xlsx file extension

Usage

```
check_office_fileext(
  x,
  arg = caller_arg(x),
  fileext = c("docx", "pptx", "xlsx"),
  allow_null = FALSE,
  call = caller_env(),
  ...
)

check_docx_fileext(x, arg = caller_arg(x), call = caller_env(), ...)

check_pptx_fileext(x, arg = caller_arg(x), call = caller_env(), ...)

check_xlsx_fileext(x, arg = caller_arg(x), call = caller_env(), ...)
```

Arguments

<code>x</code>	Object to check.
<code>arg</code>	Argument name of object to check. Used to improve <code>cli::cli_abort()</code> messages. Defaults to <code>caller_arg(x)</code> .
<code>fileext</code>	File extensions to allow without error. Defaults to "docx", "pptx", "xlsx".
<code>allow_null</code>	If TRUE and <code>x</code> is NULL, invisibly return NULL. If FALSE (default), error unless <code>x</code> is a character vector with file extension matching the supplied <code>fileext</code> value.
<code>call</code>	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.
<code>...</code>	Additional parameters passed to <code>cli::cli_abort()</code>

Value

Invisibly returns NULL if `x` passes the check (or is NULL and `allow_null` is TRUE). Otherwise, errors.

<code>check_officer</code>	<i>Check if <code>x</code> is a <code>rdocx</code>, <code>rpptx</code>, or <code>rxlsx</code> object</i>
----------------------------	--

Description

Check if `x` is a `rdocx`, `rpptx`, or `rxlsx` object

Usage

```
check_officer(
  x,
  arg = caller_arg(x),
  what = c("rdocx", "rpptx", "rxlsx"),
  call = caller_env(),
  ...
)

check_docx(x, arg = caller_arg(x), call = caller_env(), ...)

check_pptx(x, arg = caller_arg(x), call = caller_env(), ...)

check_xlsx(x, arg = caller_arg(x), call = caller_env(), ...)

check_block_list(
  x,
  arg = caller_arg(x),
  allow_empty = FALSE,
  allow_null = FALSE,
  call = caller_env()
)
```

Arguments

x	Object to check.
arg	Argument name of object to check. Used to improve <code>cli::cli_abort()</code> messages. Defaults to <code>caller_arg(x)</code> .
what	Class names to check
call	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.
...	Additional parameters passed to <code>cli::cli_abort()</code>
allow_empty	If TRUE, <code>check_block_list()</code> allows an empty block list. Defaults to FALSE.
allow_null	If FALSE (default), error if x is NULL.

Value

Invisibly returns NULL if x passes the check. Otherwise, errors.

`check_officer_summary` Check a summary data.frame created from a rdocx or rpptx object

Description

Check an object and error if an object is not a data.frame with the required column names to be a summary data.frame created from a rdocx or rpptx object. Optionally check for the number of rows, a specific content_type value, or if tables are included in the document the summary was created from.

Usage

```
check_officer_summary(
  x,
  n = NULL,
  content_type = NULL,
  summary_type = "doc",
  tables = FALSE,
  ...,
  arg = caller_arg(x),
  call = parent.frame()
)
```

Arguments

x	An object to check if it is a data.frame object created with <code>officer::docx_summary()</code> or another summary function.
---	--

n	Required number of rows. Optional. If n is more than length 1, checks to make sure the number of rows is within the range of <code>max(n)</code> and <code>min(n)</code> . Defaults to NULL.
content_type	Required content_type, e.g. "paragraph", "table cell", or "image". Optional. Defaults to NULL.
summary_type	Summary type. Options "doc", "docx", "pptx", "slide", or "layout". "doc" requires the data.frame include a "content_type" column but allows columns for either a docx or pptx summary.
tables	If TRUE, require that the summary include the column names indicated a table is present in the rdocx or rpptx summary.
...	Additional parameters passed to <code>cli::cli_abort()</code>
arg	Argument name to use in error messages. Defaults to <code>caller_arg(x)</code>
call	The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be NULL or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages.

Value

Invisibly returns NULL if x passes the checks. Otherwise, errors.

See Also

Other summary functions: [officer_summary\(\)](#), [officer_summary_levels\(\)](#)

combine_docx

Combine multiple rdocx objects or docx files

Description

[combine_docx\(\)](#) is a variant of [officer::body_add_docx\(\)](#) that allows any number of input files and supports rdocx objects as well as Word file paths. Optionally use a separator between files or objects.

Please note that when you create a new rdocx object with this function (or [officer::body_add_docx\(\)](#)) the added content will not appear in a summary data frame created with [officer_summary\(\)](#) and is not accessible to other functions until the document is *opened and edited* with Microsoft Word. This is part of how the OOXML [AltChunk Class](#) works and can't be avoided.

Usage

```
combine_docx(
  ...,
  docx = NULL,
  .list = list2(...),
  pos = "after",
  sep = NULL,
  call = caller_env()
)
```

Arguments

<code>...</code>	Any number of additional rdocx objects or docx file paths.
<code>docx</code>	A rdocx object or a file path with a docx file extension. Defaults to <code>NULL</code> .
<code>.list</code>	Any number of additional rdocx objects or docx file paths passed as a list. Defaults to <code>list2()</code>
<code>pos</code>	where to add the new element relative to the cursor, one of "after", "before", "on".
<code>sep</code>	Separator to use docx files. A bare function that takes a rdocx object as the only parameter, such as <code>officer::body_add_break</code> or another object passed to <code>add_to_body()</code> as the value parameter. Optional. Defaults to <code>NULL</code> .
<code>call</code>	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.

Value

A rdocx object.

See Also

[officer::body_add_docx\(\)](#)

Examples

```
## Not run:
if (interactive()) {
  docx_path <- system.file("doc_examples", "example.docx", package = "officer")
  docx <- read_officer(docx_path)

  combine_docx(
    docx,
    docx_path
  )
}

## End(Not run)
```

convert_docx	<i>Convert an rdocx object or Word file to another format with pandoc</i>
--------------	---

Description

convert_docx() uses `rmarkdown::pandoc_convert()` to convert a rdocx object or Word file to another format with pandoc. If the "to" parameter contains a file extension, it is assumed to be an output file name. If you want to convert a file to a Word document, use the input parameter for the path to the Markdown, HTML, or other file.

Usage

```
convert_docx(
  docx = NULL,
  to = NULL,
  input = NULL,
  output = NULL,
  path = NULL,
  options = NULL,
  extract_media = TRUE,
  overwrite = TRUE,
  quiet = TRUE,
  ...
)
```

Arguments

docx	A rdocx object or path to a docx file. Optional if input is provided.
to	Format to convert to (if not specified, you must specify output)
input	Character vector containing paths to input files (files must be UTF-8 encoded)
output	Output file (if not specified then determined based on format being converted to).
path	File path for converted file passed to wd parameter of <code>rmarkdown::pandoc_convert()</code> . If docx is a rdocx object, path defaults to current working directory instead of the base directory of the input file path.
options	Character vector of command line options to pass to pandoc.
extract_media	If TRUE, append "--extract-media=" to options. Defaults to FALSE.
overwrite	If TRUE (default), remove file at path if it already exists. If FALSE and file exists, this function aborts.
quiet	If TRUE, suppress alert messages and pass FALSE to verbose parameter of <code>rmarkdown::pandoc_convert()</code> . Defaults TRUE.
...	Arguments passed on to <code>rmarkdown::pandoc_convert</code>

from Format to convert from (if not specified then the format is determined based on the file extension of input).

citeproc TRUE to run the pandoc-citeproc filter (for processing citations) as part of the conversion.

Details

Supported input and output formats are described in the [pandoc user guide](#).

The system path as well as the version of pandoc shipped with RStudio (if running under RStudio) are scanned for pandoc and the highest version available is used.

Value

Executes a call to pandoc using `rmarkdown::pandoc_convert()` to create a file from an officer object or a docx file.

See Also

`rmarkdown::pandoc_convert()`

Examples

```
docx_example <- read_officer(
  system.file("doc_examples/example.docx", package = "officer")
)

convert_docx(
  docx_example,
  to = "markdown"
)

withr::with_tempdir({
  convert_docx(
    docx_example,
    output = "docx_example.html"
  )
})
```

cursor_docx

Set cursor position in rdocx object based on keyword, id, or index

Description

A combined function for setting cursor position with `officer::cursor_reach()`, `officer::cursor_bookmark()`, or using a doc_index value from `officer::docx_summary()`. Defaults to using `officer::cursor_end()`, `officer::cursor_begin()`, `officer::cursor_backward()`, or `officer::cursor_forward()` if keyword, id, and index are all NULL.

Usage

```
cursor_docx(
  docx,
  keyword = NULL,
  id = NULL,
```

```

    index = NULL,
    default = "end",
    quiet = FALSE,
    call = caller_env()
  )

```

Arguments

docx	A rdocx object.
keyword, id	A keyword string used to place cursor with <code>officer::cursor_reach()</code> or bookmark id with <code>officer::cursor_bookmark()</code> . Defaults to NULL. If keyword or id are not provided, the gt object is inserted at the front of the document.
index	A integer matching a doc_index value appearing in a summary of the docx object created with <code>officer::docx_summary()</code> . If index is for a paragraph value, the text of the paragraph is used as a keyword.
default	Character string with one of the following options: c("end", "begin", "backward", "forward") to set cursor position. Only used if keyword, id, and index are all NULL.
quiet	If FALSE (default) warn when keyword is not found.
call	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the call argument of <code>abort()</code> for more information.

Value

A rdocx object with the cursor repositioned. If keyword can't be found, docx is returned unmodified.

See Also

`officer::cursor_begin()`, `officer::docx_summary()`

dims_docx_ext

Get page and block layout dimensions from an rdocx object

Description

This function extends `officer::docx_dim()` by also returning the body text dimensions within the margins, the aspect ratio of the page and body, and the page orientation as a string ("landscape" or "portrait").

Usage

```
dims_docx_ext(docx)
```

Arguments

docx A rdocx object to get dimensions for.

Value

A list with page, landscape, margins, orientation, and body elements describing the page and body text dimensions.

See Also

`officer::docx_dim()`

docx_sync_sections	<i>Copy footer and page settings from the last section to every section</i>
--------------------	---

Description

`docx_sync_sections()` copies the header and footer references, page margins, and page size from the last section of a Word document to the other sections, filling in what each section is missing.

This fixes documents rendered by Quarto or Pandoc where section breaks don't carry the reference document's page settings, for example when using Quarto's `.landscape` divs (which add section breaks with no footer, no margins, and an A4 landscape page size) or a custom OpenXML template. See the "Page settings and sections" section of [brand_reference_docx\(\)](#).

For each section other than the last:

- If the section has no header or footer references, the last section's references are added.
- If the section has no page margins, the last section's margins are added.
- If the section has no page size, the last section's page size is added. If `page_size = TRUE`, landscape sections also get the last section's page size with the width and height swapped (and portrait sections get the last section's page size).

To use `docx_sync_sections()` in a Quarto project, add a post-render script to `_quarto.yml`:

```
project:
  post-render: post-render.R
```

With a `post-render.R` file:

```
output_files <- strsplit(Sys.getenv("QUARTO_PROJECT_OUTPUT_FILES"), "\n")[[1]]

for (file in output_files[grepl("\\.docx$", output_files)]) {
  officerExtras::docx_sync_sections(file)
}
```

Usage

```
docx_sync_sections(x, path = NULL, page_size = TRUE)
```

Arguments

<code>x</code>	A rdocx object or a path to a docx file.
<code>path</code>	Output file path. If NULL (default) and <code>x</code> is a file path, <code>x</code> is overwritten. If NULL and <code>x</code> is a rdocx object, the document is not written to a file.
<code>page_size</code>	If TRUE (default), set the page size of every section to the last section's page size (with the width and height swapped for landscape sections). If FALSE, only add a page size to sections without one.

Value

A rdocx object (invisibly if a file is written).

Examples

```
docx <- officer::read_docx()
docx <- officer::body_add_par(docx, "Portrait")
docx <- officer::body_end_section_portrait(docx)
docx <- officer::body_add_par(docx, "Landscape")
docx <- officer::body_end_section_landscape(docx)

docx_sync_sections(docx)
```

<code>fill_with_pattern</code>	<i>Fill a new column based on an existing column depending on a pattern</i>
--------------------------------	---

Description

This function uses `vctrs::vec_fill_missing()` to convert hierarchically nested headings and text into a rectangular data.frame. It is an experimental function that may be modified or removed. At present, it is only used by `officer_tables()`.

Usage

```
fill_with_pattern(
  x,
  pattern = "^heading",
  pattern_col = "style_name",
  fill_col = "text",
  col = "heading",
  direction = c("down", "up", "downup", "updown"),
  call = caller_env()
)
```

Arguments

x	A input data.frame (assumed to be from <code>officer_summary()</code> for default values).
pattern	Passed to <code>grepl()</code> as the pattern identifying which rows of the fill_col should have values pulled into the new column named by col. Defaults to "^heading" which matches the default heading style names.
pattern_col	Name of column to use for pattern matching, Defaults to "style_name".
fill_col	Name of column to fill , Defaults to "text".
col	Name of new column to fill with values from fill_col, Default: Defaults to "heading"
direction	Direction of fill passed to <code>vctrs::vec_fill_missing()</code> , Default: <code>c("down", "up", "downup", "updown")</code>
call	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the call argument of <code>abort()</code> for more information.

Value

A data.frame with an additional column taking the name from col and the values from the column named in fill_col.

See Also

`vctrs::vec_fill_missing()`

is_officer	<i>Is this object an officer class object?</i>
------------	--

Description

`is_officer()` and variants are basic wrappers for `inherits()` to check for object classes created by functions from the officer package. These functions are intended for internal use but are exposed for use by other extension developers.

Usage

```
is_officer(x, what = c("rdocx", "rpptx", "rxlsx"))

is_rdocx(x)

is_rpptx(x)

is_block_list(x)
```

Arguments

x	A object to test
what	Class or classes passed to <code>inherits()</code>

Value

TRUE or FALSE.

make_block_list	<i>Make a list of blocks to add to a Word document or PowerPoint presentation</i>
-----------------	---

Description**[Experimental]**

`make_block_list()` extends `officer::block_list()` by supporting a list of inputs and optionally combining parameters with an existing block list (using the `blocks` parameter). Unlike `officer::block_list()`, `make_block_list()` errors if no input parameters are provided.

`combine_blocks()` takes any number of `block_list` objects and combined them into a single `block_list`. Both functions are not yet working as expected.

Usage

```
make_block_list(blocks = NULL, ..., allow_empty = FALSE, call = caller_env())
```

```
combine_blocks(...)
```

Arguments

blocks	A list of parameters to pass to <code>officer::block_list()</code> or a <code>block_list</code> object. If parameters are provided to both <code>...</code> and <code>blocks</code> is a <code>block_list</code> , the additional parameters are appended to the end of <code>blocks</code> .
...	For <code>make_block_list()</code> , these parameters are passed to <code>officer::block_list()</code> and must <i>not</i> include <code>block_list</code> objects. For <code>combine_blocks()</code> , these parameters must all be <code>block_list</code> objects.
allow_empty	If TRUE, <code>check_block_list()</code> allows an empty block list. Defaults to FALSE.
call	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.

Value

A `block_list` object.

See Also

Other block list functions: `officer_add_blocks()`

officer_add_blocks *Add a list of blocks into a Word document or PowerPoint presentation*

Description

[Experimental]

`officer_add_blocks()` supports adding a list of blocks to rdocx (using `add_blocks_to_body()`) or rpptx objects (using `officer::ph_with()`). `add_blocks_to_body()` is a variant of `officer::body_add_blocks()` that allows users to set the cursor position before adding the block list using `cursor_docx()`.

Usage

```
officer_add_blocks(
  x,
  blocks,
  pos = "after",
  location = NULL,
  level_list = integer(0),
  ...,
  call = caller_env()
)

add_blocks_to_body(
  docx,
  blocks,
  pos = "after",
  keyword = NULL,
  id = NULL,
  index = NULL,
  ...
)
```

Arguments

<code>x</code>	A rdocx or rpptx object. Required.
<code>blocks</code>	set of blocks to be used as footnote content returned by function <code>block_list()</code> .
<code>pos</code>	where to add the new element relative to the cursor, one of "after", "before", "on".
<code>location</code>	If NULL, location defaults to <code>officer::ph_location_type()</code>
<code>level_list</code>	The list of levels for hierarchy structure as integer values. If used the object is formatted as an unordered list. If 1 and 2, item 1 level will be 1, item 2 level will be 2.
<code>...</code>	further arguments passed to or from other methods. When adding a ggplot object or plot_instr, these arguments will be used by the png function.

call	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.
docx	A <code>rdocx</code> object.
keyword, id	A keyword string used to place cursor with <code>officer::cursor_reach()</code> or bookmark id with <code>officer::cursor_bookmark()</code> . Defaults to <code>NULL</code> . If keyword or id are not provided, the <code>gt</code> object is inserted at the front of the document.
index	A integer matching a <code>doc_index</code> value appearing in a summary of the <code>docx</code> object created with <code>officer::docx_summary()</code> . If index is for a paragraph value, the text of the paragraph is used as a keyword.

Value

A `rdocx` or `rpptx` object with the block list added.

See Also

Other block list functions: `make_block_list()`

officer_media	<i>Copy media from a docx or pptx file or a rdocx or rpptx object to a target folder</i>
---------------	--

Description

Unzip a `docx` or `pptx` file to a temporary directory, check if the directory contains a media folder, and copy media files to the directory set by `dir`. If a `rdocx` or `rpptx` object is provided, files are copied from the temporary `package_dir` associated with the object (accessible via `x[["package_dir"]]`).

Usage

```
officer_media(
  filename = NULL,
  path = NULL,
  x = NULL,
  target = "media",
  list = FALSE,
  overwrite = TRUE
)
```

Arguments

filename, path	File name and path for a <code>docx</code> or <code>pptx</code> file. One of the two must be provided. Ignored if <code>x</code> is provided. Defaults to <code>NULL</code> .
x	A <code>rdocx</code> or <code>rpptx</code> object containing one or more media file.

target	Folder name or path to copy media files. dir is created if no folder exists at that location.
list	If TRUE, display a message listing files contained in the docx or pptx file but do not copy the files to dir. Defaults to FALSE.
overwrite	If TRUE (default), overwrite any files with the same names at target path. If FALSE, abort if files with the same names already exist.

Value

Invisibly returns the paths of the copied (or, if list is TRUE, listed) media files, or NULL if no media files are found.

See Also

`officer::media_extract()`

Examples

```
officer_media(
  system.file("doc_examples/example.pptx", package = "officer"),
  list = TRUE
)
```

officer_open	<i>Preview a rdocx, rpptx, or rxlsx object in local default applications</i>
--------------	--

Description

`officer_open()` uses `officer::open_file()` to open a file ceated from a rdocx, rpptx, or rxlsx object.

Usage

```
officer_open(
  x,
  path = NULL,
  ...,
  overwrite = FALSE,
  interactive = is_interactive()
)
```

Arguments

x	A rdocx, rpptx, or rxlsx object to save. Document properties won't be set for rxlsx objects.
path	Optional. Set as temporary file with file extension matching type of input object x.

...	Arguments passed on to write_officer
modified_by	If the withr package is installed, modified_by overrides the default value for the lastModifiedBy property assigned to the output file by officer. Defaults to Sys.getenv("USER") (the same value used by officer).
overwrite	Defaults to FALSE.
interactive	If FALSE, warn the user and return x.

Value

Input object x without modification.

officer_properties	<i>Get doc properties for a rdocx or rpptx object as a list</i>
--------------------	---

Description

[officer_properties\(\)](#) is a variant on [officer::doc_properties\(\)](#) that will warn instead of error if document properties can't be found

Usage

```
officer_properties(x, values = list(), keep.null = FALSE, call = caller_env())
```

Arguments

x	A rdocx or rpptx object.
values	A named list with new properties to replace existing document properties before they are returned as a named list.
keep.null	Passed to utils::modifyList() . If TRUE, retain properties in returned list even if they have NULL values.
call	The execution environment of a currently running function, e.g. caller_env() . The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

A named list of existing document properties or (if values is supplied) modified document properties.

officer_summary	<i>Summarize a rdocx or rpptx object</i>
-----------------	--

Description

officer_summary() extends officer::docx_summary() and other officer summary functions by handling multiple input types within a single function.

Usage

```
officer_summary(
  x,
  summary_type = "doc",
  index = NULL,
  preserve = FALSE,
  as_tibble = TRUE,
  call = caller_env()
)
```

Arguments

x	A rdocx or rpptx object passed to officer::docx_summary() , officer::pptx_summary() , officer::slide_summary() , or officer::layout_summary() . If x is a data frame created with one of those functions, it is returned as is (this feature may be removed in the future).
summary_type	Summary type. Options "doc", "docx", "pptx", "slide", or "layout". "doc" requires the data.frame include a "content_type" column but allows columns for either a docx or pptx summary.
index	slide index
preserve	If FALSE (default), text in table cells is collapsed into a single line. If TRUE, line breaks in table cells are preserved as a "\n" character. This feature is adapted from docxtractr::docx_extract_tbl() published under a MIT licensed in the 'docxtractr' package by Bob Rudis.
as_tibble	If TRUE (default), return a tibble data frame.
call	The execution environment of a currently running function, e.g. call = caller_env(). The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply call when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be NULL or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages.

Value

A tibble or data frame object.

See Also

Other summary functions: [check_officer_summary\(\)](#), [officer_summary_levels\(\)](#)

Examples

```
docx_example <- read_officer(  
  system.file("doc_examples", "example.docx", package = "officer")  
)  
  
officer_summary(docx_example)  
  
pptx_example <- read_officer(  
  "example.pptx", system.file("doc_examples", package = "officer")  
)  
  
officer_summary(pptx_example)  
  
officer_summary(pptx_example, "slide", 1)
```

officer_summary_levels

Create new columns to an officer summary data.frame based on specific levels

Description

[officer_summary_levels\(\)](#) works with [fill_with_pattern\(\)](#) to help convert hierarchically organized text, e.g. text with leveled headings or other styles into a data.frame where new columns hold the value of the preceding (or succeeding) heading text.

Usage

```
officer_summary_levels(  
  x,  
  levels = NULL,  
  levels_from = "style_name",  
  exclude_levels = NULL,  
  fill_col = "text",  
  direction = c("down", "up", "downup", "updown"),  
  ...,  
  strict = FALSE,  
  call = caller_env()  
)
```

Arguments

x	A input data.frame (assumed to be from officer_summary() for default values).
levels	Levels to use. If NULL (default), levels is set to unique, non-NA values from the levels_from column.
levels_from	Column name to use for identifying levels. Defaults to "style_name".
exclude_levels	Levels to exclude from process of adding new columns.
fill_col	Name of column to fill , Defaults to "text".
direction	Direction of fill passed to vctrs::vec_fill_missing() , Default: c("down", "up", "downup", "updown")
...	Arguments passed on to officer_summary
as_tibble	If TRUE (default), return a tibble data frame.
summary_type	Summary type. Options "doc", "docx", "pptx", "slide", or "layout". "doc" requires the data.frame include a "content_type" column but allows columns for either a docx or pptx summary.
index	slide index
preserve	If FALSE (default), text in table cells is collapsed into a single line. If TRUE, line breaks in table cells are preserved as a "\n" character. This feature is adapted from docxtractr::docx_extract_tbl() published under a MIT licensed in the 'docxtractr' package by Bob Rudis.
strict	If TRUE, error unless all values provided to level are present in the levels_from column. Defaults to FALSE which warns if invalid values are present.
call	The execution environment of a currently running function, e.g. caller_env() . The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

Value

A summary data frame with additional columns based on the supplied levels.

See Also

Other summary functions: [check_officer_summary\(\)](#), [officer_summary\(\)](#)

officer_tables	<i>Get tables from a rdocx or rpptx object</i>
----------------	--

Description

Get one or more tables from a rdocx or rpptx object. [officer_tables\(\)](#) returns a list of data frames and [officer_table\(\)](#) returns a single table as a data frame. These functions are based on example code on extracting Word document and PowerPoint slides in the [officeverse documentation](#). Some additional features including the type_convert parameter and the addition of doc_index values as the default names for the returned list of tables are based on [this blog post by Matt Dray](#).

Usage

```

officer_tables(
  x,
  index = NULL,
  has_header = TRUE,
  col = NULL,
  preserve = FALSE,
  ...,
  stack = FALSE,
  type_convert = FALSE,
  nm = NULL,
  call = caller_env()
)

officer_table(
  x,
  index = NULL,
  has_header = TRUE,
  col = NULL,
  ...,
  call = caller_env()
)

```

Arguments

<code>x</code>	A rdocx or rpptx object or a data frame created with officer_summary() .
<code>index</code>	A index value matching a doc_index value for a table in the summary data frame, Default: NULL
<code>has_header</code>	If TRUE (default), tables are expected to have implicit headers even if the Word table does not have an explicit header row. If FALSE, only explicit header rows will be used as column names.
<code>col</code>	If col is supplied, officer_table() passes col and the additional parameters in ... to fill_with_pattern() . This allows the addition of preceding headings or captions as a column within the data.frame returned by officer_tables() . This is an experimental feature and may be modified or removed. Defaults to NULL.
<code>preserve</code>	If FALSE (default), text in table cells is collapsed into a single line. If TRUE, line breaks in table cells are preserved as a "\n" character. This feature is adapted from <code>docxtractr::docx_extract_tbl()</code> published under a MIT licensed in the 'docxtractr' package by Bob Rudis.
<code>...</code>	Additional parameters passed to fill_with_pattern() .
<code>stack</code>	If TRUE and all tables share the same number of columns, return a single combined data frame instead of a list. Defaults to FALSE.
<code>type_convert</code>	If TRUE, convert columns for the returned data frames to the appropriate type using utils::type.convert() .

nm	Names to use for returned list of tables. If NULL (default), the names are set to the table_index values using the pattern "table_index_<table_index_number>".
call	The execution environment of a currently running function, e.g. call = caller_env(). The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply call when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be NULL or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages.

Value

A list of data frames or, if stack is TRUE, a single data frame.

A single data.frame, or (if the table has more than one header row) a list of two data.frames: the table body and the table header.

See Also

[docxtractr::docx_extract_all\(\)](#)

Examples

```
docx_example <- read_docx_ext(
  filename = "example.docx",
  path = system.file("doc_examples", package = "officer")
)

officer_tables(docx_example)

pptx_example <- read_pptx_ext(
  filename = "example.pptx",
  path = system.file("doc_examples", package = "officer")
)

officer_tables(pptx_example)[[1]]
```

read_docs_url	<i>Read a Google Docs document, Slides, or Sheets URL to a rdocx, rpptx, or rxlsx object</i>
---------------	--

Description

Uses the Google Docs, Slides, or Sheets URL to export a file locally, read to an officer object. If filename is NULL, exported file is removed after export.

Usage

```
read_docs_url(
  url,
  format = NULL,
  filename = NULL,
  path = NULL,
  overwrite = TRUE,
  quiet = TRUE
)
```

Arguments

url	A URL for a Google Doc, Google Slides presentation, or Google Sheets.
format	File format to use for export URL (typically set automatically). Options are NULL (default), "doc", "pptx", or "xlsx". "pdf" and "csv" be supported in the future.
filename	Destination file name. Optional. If filename is NULL, downloaded file is removed as part of the function execution.
path	Folder path. Optional.
overwrite	If TRUE (default), overwrite any existing file at the same location specified by filename and path.
quiet	If TRUE, suppress messages when downloading file.

Value

A rdocx, rpptx, or rxlsx object.

See Also

[read_officer\(\)](#)

read_officer	<i>Read a docx, pptx, potx, or xlsx file or use an existing object from officer if provided</i>
--------------	---

Description

[read_officer\(\)](#) is a variant of [officer::read_docx\(\)](#), [officer::read_pptx\(\)](#), and [officer::read_xlsx\(\)](#) that allows users to read different Microsoft Office file types with a single function. [read_docx_ext\(\)](#), [read_pptx_ext\(\)](#), and [read_xlsx_ext\(\)](#) are wrappers for [read_officer\(\)](#) that require the matching input file type. All versions allow both a filename and path (the officer functions only use a path). If a rdocx, rpptx, or rxlsx class object is provided to x, the object is checked based on the fileext parameter and then returned as is.

Usage

```

read_officer(
  filename = NULL,
  path = NULL,
  fileext = c("docx", "pptx", "xlsx"),
  x = NULL,
  arg = caller_arg(x),
  allow_null = TRUE,
  quiet = TRUE,
  call = parent.frame(),
  ...
)

read_docx_ext(
  filename = NULL,
  path = NULL,
  docx = NULL,
  allow_null = FALSE,
  quiet = TRUE
)

read_pptx_ext(
  filename = NULL,
  path = NULL,
  pptx = NULL,
  allow_null = FALSE,
  quiet = TRUE
)

read_xlsx_ext(
  filename = NULL,
  path = NULL,
  xlsx = NULL,
  allow_null = FALSE,
  quiet = TRUE
)

```

Arguments

filename, path	File name and path. Default: NULL. Must include a "docx", "pptx", or "xlsx" file path. "dotx" and "potx" files are also supported.
fileext	File extensions to allow without error. Defaults to "docx", "pptx", "xlsx".
x	A rdocx, rpptx, or rxlsx class object If x is provided, filename and path are ignored. Default: NULL
arg	Argument name of object to check. Used to improve <code>cli::cli_abort()</code> messages. Defaults to <code>caller_arg(x)</code> .

allow_null	If TRUE, function supports the default behavior of <code>officer::read_docx()</code> , <code>officer::read_pptx()</code> , or <code>officer::read_xlsx()</code> and returns an empty document if x, filename, and path are all NULL. If FALSE, one of the three parameters must be supplied.
quiet	If FALSE, warn if docx is provided when filename and/or path are also provided. Default: TRUE.
call	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the call argument of <code>abort()</code> for more information.
...	Additional parameters passed to <code>cli::cli_abort()</code>
docx, pptx, xlsx	A rdocx, rpptx, or rxlsx class object passed to the x parameter of <code>read_officer()</code> by the variant functions. Defaults to NULL.

Value

A rdocx, rpptx, or rxlsx object.

See Also

`officer::read_docx()`

reduce_officer	<i>Reduce a list to a single officer object</i>
----------------	---

Description

`reduce_officer()` is a wrapper for `purrr::reduce()`.

Usage

```
reduce_officer(
  x = NULL,
  .f = function(x, value, ...) {
    vec_add_to_body(x, value = value, ...)
  },
  value = NULL,
  ...,
  .path = NULL
)
```

Arguments

x	File path or officer object.
.f	Any function taking an officer object as the first parameter and a value as the second parameter, Defaults to anonymous function: <code>function(x, value, ...) { vec_add_to_body(x, value = value, ...) }</code>
value	A vector of values that are support by the function passed to .f, Default: NULL

... Additional parameters passed to `purrr::reduce()`.

.path If .path not NULL, it should be a file path that is passed to `write_officer()`, allowing you to modify a docx file and write it back to a file in a single function call.

Value

A rdocx, rpptx, or rxlsx object.

See Also

`purrr::reduce()`, `vec_add_to_body()`

Examples

```
## Not run:
if (interactive()) {
  x <- reduce_officer(value = LETTERS)

  officer_summary(x)
}

## End(Not run)
```

set_flextable_brand_defaults

Set flextable defaults from a brand.yml file

Description

`set_flextable_brand_defaults()` sets `flextable::set_flextable_defaults()` from a brand.yml file, so tables created with flextable match the Table style in a reference document created with `brand_reference_docx()`. flextable formats each table directly (rather than using the reference document's Table style), so the defaults are needed for tables to match.

Usage

```
set_flextable_brand_defaults(brand = NULL, theme_fun = NULL, ...)
```

Arguments

brand Path to a brand.yml file or a directory containing one, a brand_yml object from `brand.yml::read_brand_yml()`, or a list with the brand settings (e.g. from `yaml12::read_yaml()`). If NULL (default), the path is taken from the BRAND_YML_PATH environment variable or, if that is not set, found by searching the working directory and its parent directories for _brand.yml or _brand.yaml, either directly or in a brand or _brand subdirectory (the same search used by Quarto and `brand.yml::read_brand_yml()`).

theme_fun	A flextable theme function applied to each new table, before the header settings. Defaults to <code>flextable::theme_box()</code> (borders on every cell, as in the reference document Table style).
...	Additional arguments passed to <code>flextable::set_flextable_defaults()</code> . These take precedence over the brand settings.

Details

The following brand.yml settings are used:

- `typography.base`: family, size, and line-height (if unitless) for the table text.
- `color.foreground` for the text color.
- `defaults.docx.styles.Table`: border-color, border-width, cell-padding, and header (family, size, weight, style, color, and background-color for the header row).

Value

The previous flextable defaults, invisibly. Pass them to `do.call(flextable::set_flextable_defaults, old)` to restore them.

Examples

```
if (rlang::is_installed(c("flextable", "yaml12"))) {
  brand_path <- system.file("extdata", "_brand.yml", package = "officerExtras")

  old <- set_flextable_brand_defaults(brand_path)
  flextable::flextable(head(mtcars, 3))

  do.call(flextable::set_flextable_defaults, old)
}
```

use_doc_version	<i>Update the version of a document and optionally save a new version of the input document</i>
-----------------	---

Description

[Experimental]

`use_doc_version()` is inspired by `usethis::use_version()` and designed to support semantic versioning for Microsoft Word or PowerPoint documents. The document version is tracked as a custom file property and a component of the document filename. If filename is supplied and the filename contains a version number, the function ignores any version number stored as a document property and overwrites the property with the new incremented version number.

`doc_version()` is a helper function that returns the document version based on a supplied filename or document properties.

The internals for this function are adapted from the internal `idesc_bump_version()` function authored by Csárdi Gábor for the `{desc}` package.

Usage

```

use_doc_version(
  filename = NULL,
  x = NULL,
  which = NULL,
  save = TRUE,
  sep = ".",
  property = "version",
  prefix = NULL,
  path = NULL,
  ...,
  call = caller_env()
)

doc_version(
  filename = NULL,
  x = NULL,
  sep = ".",
  property = "version",
  allow_new = TRUE,
  .default = c(0, 1, 0),
  call = caller_env()
)

```

Arguments

filename	Filename for a Word document or PowerPoint presentation. Optional if x is supplied, however, it is required to locally save a file with an updated version name.
x	A rdocx or rpptx object. Optional if filename is supplied.
which	A string specifying which level to increment, one of: "major", "minor", "patch", "dev".
save	If TRUE (default) and filename is supplied, write a new file replacing the existing version number.
sep	Character separating version components. Defaults to ".". "-" is also supported.
property	Property name optionally containing a version number. Defaults to "version".
prefix	Property name to use as prefix for new filename, e.g. "modified" to use modified date/time as the new filename prefix. If prefix is identical to property and the filename does not already include a version number, version is added to the filename as a prefix instead of a postfix. Defaults to NULL. Note handling of this parameter is expected to change in future versions of this function.
path	Passed to write_officer() .
...	Additional parameters passed to write_officer() if save is TRUE.
call	The execution environment of a currently running function, e.g. caller_env() . The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.

allow_new	If TRUE (default), return "0.1.0" if version can't be found in the filename or the properties of the input rdocx or rpptx object x.
.default	Specification for initial version.

Value

Invisibly return a rdocx or rpptx object with an updated version property.

A string with the document version.

vec_add_to_body	<i>Add a vector of objects to a rdocx object using add_to_body()</i>
-----------------	--

Description

`vec_add_to_body()` is a vectorized variant `add_to_body()` that allows users to supply a vector of values or str inputs to add multiple blocks of text, images, plots, or tables to a document. Alternatively, the function also supports adding a single object at multiple locations or using multiple different styles. All parameters are recycled using `vctrs::vec_recycle_common()` so inputs must be length 1 or match the length of the longest input vector. Optionally, the function can apply a separator between each element by passing a value to `add_to_body()` or passing docx to a function, such as `officer::body_add_break()`.

Usage

```
vec_add_to_body(
  docx,
  ...,
  .sep = NULL,
  .pos = "after",
  .size = NULL,
  .call = caller_env()
)
```

Arguments

docx	A rdocx object.
...	Arguments passed on to <code>add_to_body</code>
gt_object	A gt object converted to an OOXML string with <code>gt::as_word()</code> then passed to <code>add_xml_to_body()</code> as str parameter. Required for <code>add_gt_to_body()</code> .
tablecontainer	If TRUE (default), add tables inside of a tablecontainer tag that automatically adds a table number and converts the gt title into a table caption. This feature is based on code from the gto package by Ellis Hughes to transform the gt_object to OOXML and insert the XML into the docx object.
caption	Name of the ggplot2 label to use as a caption if plot passed to value has a label for this value. Defaults to "title".

caption_style Passed to style for `officer::body_add_caption()`. Defaults to same value as style.

.f Any function that takes a docx and value parameter and returns a rdocx object. A keyword parameter must also be supported if named is TRUE. Defaults to `add_text_to_body()`.

keyword, id A keyword string used to place cursor with `officer::cursor_reach()` or bookmark id with `officer::cursor_bookmark()`. Defaults to NULL. If keyword or id are not provided, the gt object is inserted at the front of the document.

index A integer matching a doc_index value appearing in a summary of the docx object created with `officer::docx_summary()`. If index is for a paragraph value, the text of the paragraph is used as a keyword.

call The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the call argument of `abort()` for more information.

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.

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()`).

str a wml string

pos where to add the new element relative to the cursor, one of "after", "before", "on".

.envir [environment: parent.frame()]
Environment to evaluate each expression in. Expressions are evaluated from left to right. If .x is an environment, the expressions are evaluated in that environment and .envir is ignored. If NULL is passed, it is equivalent to `emptyenv()`.

.na [character(1): 'NA']
Value to replace NA values with. If NULL missing values are propagated, that is an NA result will cause NA output. Otherwise the value is replaced by the value of .na.

.null [character(1): 'character()']
Value to replace NULL values with. If `character()` whole output is `character()`. If NULL all NULL values are dropped (as in `paste0()`). Otherwise the value is replaced by the value of .null.

align *Table alignment*
scalar<character> // default: "center"
An option for table alignment. Can either be "center", "left", or "right".

caption_location *Caption location*
singl-kw:[top|bottom|embed] // default: "top"
Determines where the caption should be positioned. This can either be "top", "bottom", or "embed".

caption_align *Caption alignment*

	Determines the alignment of the caption. This is either "left" (the default), "center", or "right". This option is only used when caption_location is not set as "embed".
<code>split</code>	<i>Allow splitting of a table row across pages</i> scalar<logical> // default: FALSE A logical value that indicates whether to activate the Word option Allow row to break across page
<code>keep_with_next</code>	<i>Keeping rows together</i> scalar<logical> // default: TRUE A logical value that indicates whether a table should use Word option Keep rows together.
<code>autonum</code>	<i>Automatic Table Numbering</i> scalar<logical> // default: TRUE A logical value that indicates whether a table should use Words built-in auto table numbering option in the caption. Automatic Table Numbering.
<code>.sep</code>	A bare function, such as <code>officer::body_add_break</code> or another object passed to <code>add_to_body()</code> as the value parameter.
<code>.pos</code>	String passed to pos parameter if <code>add_to_body()</code> with .sep if .sep is not a function. Defaults to "after".
<code>.size</code>	Desired output size.
<code>.call</code>	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the call argument of <code>abort()</code> for more information.

Value

A rdocx object with each recycled element added in sequence.

Examples

```
docx_example <- read_officer()

docx_example <- vec_add_to_body(
  docx_example,
  value = c("Sample text 1", "Sample text 2", "Sample text 3"),
  style = c("heading 1", "heading 2", "Normal")
)

docx_example <- vec_add_to_body(
  docx_example,
  value = rep("Text", 5),
  style = "Normal",
  .sep = officer::body_add_break
)

officer_summary(docx_example)

if (rlang::is_installed("gt")) {
  gt_tbl <- gt::gt(gt::gtcars[1:2, 1:2])

  # list inputs such as gt tables must be passed within a list to avoid
```

```

# issues
docx_example <- vec_add_to_body(
  docx_example,
  gt_object = list(gt_tbl, gt_tbl),
  keyword = c("Sample text 1", "Sample text 2")
)

officer_summary(docx_example)
}

```

write_officer

Write a rdocx, rpptx, or rxlsx object to a file

Description

Write a rdocx, rpptx, or rxlsx object to a file

Usage

```
write_officer(x, path, overwrite = TRUE, modified_by = Sys.getenv("USER"), ...)
```

Arguments

x	A rdocx, rpptx, or rxlsx object to save. Document properties won't be set for rxlsx objects.
path	File path.
overwrite	If TRUE (default), remove file at path if it already exists. If FALSE and file exists, this function aborts.
modified_by	If the withr package is installed, modified_by overrides the default value for the lastModifiedBy property assigned to the output file by officer. Defaults to Sys.getenv("USER") (the same value used by officer).
...	Arguments passed on to officer::set_doc_properties title, subject, creator, description text fields created a date object hyperlink_base a string specifying the base URL for relative hyperlinks in the document (only for rdocx). values a named list (names are field names), each element is a single character value specifying value associated with the corresponding field name. If values is provided, argument ... will be ignored.

Value

Returns the input object (invisibly) and writes the rdocx, rpptx, or rxlsx object to a file with a name and location matching the provided path.

Index

- * **block list functions**
 - make_block_list, 21
 - officer_add_blocks, 22
- * **summary functions**
 - check_officer_summary, 12
 - officer_summary, 26
 - officer_summary_levels, 27
- abort(), 5, 11, 12, 14, 17, 20, 21, 23, 25, 28, 33, 36, 38, 39
- add_blocks_to_body
 - (officer_add_blocks), 22
- add_blocks_to_body(), 22
- add_gg_to_body (add_to_body), 3
- add_gg_to_body(), 3
- add_gt_to_body (add_to_body), 3
- add_gt_to_body(), 3, 5, 37
- add_list_to_body, 2
- add_str_with_keys (add_to_body), 3
- add_str_with_keys(), 6
- add_text_to_body (add_to_body), 3
- add_text_to_body(), 3, 6, 38
- add_to_body, 3, 37
- add_to_body(), 3, 14, 37, 39
- add_value_with_keys (add_to_body), 3
- add_value_with_keys(), 6
- add_xml_to_body (add_to_body), 3
- add_xml_to_body(), 5, 6, 37
- block_list(), 22
- brand.yml::read_brand_yaml(), 8, 9, 34
- brand_reference_docx, 7
- brand_reference_docx(), 18, 34
- check_block_list (check_officer), 11
- check_block_list(), 12, 21
- check_docx (check_officer), 11
- check_docx_fileext
 - (check_office_fileext), 10
- check_office_fileext, 10
- check_officer, 11
- check_officer_summary, 12
- check_officer_summary(), 27, 28
- check_pptx (check_officer), 11
- check_pptx_fileext
 - (check_office_fileext), 10
- check_xlsx (check_officer), 11
- check_xlsx_fileext
 - (check_office_fileext), 10
- cli::cli_abort(), 11–13, 32, 33
- combine_blocks (make_block_list), 21
- combine_blocks(), 21
- combine_docx, 13
- combine_docx(), 13
- convert_docx, 15
- cursor_docx, 16
- cursor_docx(), 3, 22
- defused function call, 13, 26, 30
- dims_docx_ext, 17
- doc_version (use_doc_version), 35
- doc_version(), 35
- docx_sync_sections, 18
- docx_sync_sections(), 10
- docxtractr::docx_extract_all(), 30
- emptyenv(), 5, 38
- fill_with_pattern, 19
- fill_with_pattern(), 27, 29
- flextable::set_flextable_defaults(), 34, 35
- flextable::theme_box(), 35
- glue::glue(), 3
- grepl(), 20
- gt::as_word(), 3, 5, 37
- Including function calls in error messages, 13, 26, 30
- inherits(), 20, 21

is_block_list(is_officer), 20
 is_officer, 20
 is_officer(), 20
 is_rdocx(is_officer), 20
 is_rpptx(is_officer), 20

 list2(), 14

 make_block_list, 21
 make_block_list(), 21, 23

 officer::block_list(), 21
 officer::body_add(), 3, 5
 officer::body_add_blocks(), 22
 officer::body_add_break, 14, 39
 officer::body_add_break(), 37
 officer::body_add_caption(), 6, 38
 officer::body_add_docx(), 13, 14
 officer::body_add_gg(), 3, 5
 officer::body_add_par(), 3, 5
 officer::body_add_xml(), 3
 officer::cursor_backward(), 16
 officer::cursor_begin(), 16, 17
 officer::cursor_bookmark(), 5, 16, 17, 23, 38
 officer::cursor_end(), 16
 officer::cursor_forward(), 16
 officer::cursor_reach(), 5, 16, 17, 23, 38
 officer::doc_properties(), 25
 officer::docx_dim(), 17, 18
 officer::docx_set_paragraph_style(), 7
 officer::docx_summary(), 5, 12, 16, 17, 23, 26, 38
 officer::layout_summary(), 26
 officer::media_extract(), 24
 officer::ph_location_type(), 22
 officer::ph_with(), 22
 officer::pptx_summary(), 26
 officer::read_docx(), 31, 33
 officer::read_pptx(), 31, 33
 officer::read_xlsx(), 31, 33
 officer::set_doc_properties, 40
 officer::slide_summary(), 26
 officer_add_blocks, 22
 officer_add_blocks(), 21, 22
 officer_media, 23
 officer_open, 24
 officer_properties, 25
 officer_properties(), 25

 officer_summary, 26, 28
 officer_summary(), 13, 20, 28, 29
 officer_summary_levels, 27
 officer_summary_levels(), 13, 27
 officer_table(officer_tables), 28
 officer_table(), 28, 29
 officer_tables, 28
 officer_tables(), 19, 28, 29

 pandoc_reference_docx
 (brand_reference_docx), 7
 purrr::reduce(), 33, 34

 read_docs_url, 30
 read_docx, 5, 38
 read_docx_ext(read_officer), 31
 read_docx_ext(), 2, 31
 read_officer, 31
 read_officer(), 31, 33
 read_pptx_ext(read_officer), 31
 read_pptx_ext(), 31
 read_xlsx_ext(read_officer), 31
 read_xlsx_ext(), 31
 reduce_officer, 33
 reduce_officer(), 33
 rmarkdown::pandoc_convert, 15
 rmarkdown::pandoc_convert(), 15, 16

 set_flextable_brand_defaults, 34
 styles_info, 5, 38

 use_doc_version, 35
 use_doc_version(), 35
 usethis::use_version(), 35
 utils::modifyList(), 25
 utils::type.convert(), 29

 vctrs::vec_fill_missing(), 19, 20, 28
 vctrs::vec_recycle_common(), 37
 vec_add_to_body, 37
 vec_add_to_body(), 3, 6, 34, 37

 write_officer, 25, 40
 write_officer(), 34, 36

 yaml12::read_yaml(), 8, 34