Package: wordup (via r-universe)

August 27, 2024

Title Convert Word to Govspeak

Version 0.0.0.9000

Description Try to convert a Word document (docx) to the equivalent Govspeak Markdown, ready for upload to the UK government's publishing platform.

License MIT + file LICENSE

URL https://matt-dray.github.io/wordup/

BugReports https://github.com/matt-dray/wordup/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports cli, clipr, fs, xml2

Repository https://matt-dray.r-universe.dev

RemoteUrl https://github.com/matt-dray/wordup

RemoteRef HEAD

RemoteSha e1ec880c326f2d6188b27e8d426e013a48e1bdec

Contents

table_to_govspeak *Convert a Copy-Pasted Word Table to Govspeak*

Description

Provide a copied table from a Word document and be returned a Govspeak Markdown version of it. Some post-editing may be necessary for more complex tables.

Usage

```
table_to_govspeak(
  word_table = NULL,
  guess_types = TRUE,
  ignore_regex = ",|%|\\[.\\]",
  has_row_titles = FALSE,
  totals_rows = NULL,
  to_clipboard = TRUE
)
```

Arguments

word_table	Character. A table copy-pasted from a Microsoft Word document. If NULL (default) the table will be read from the clipboard so that you don't have to paste it.
guess_types	Logical. Should data types be guessed for each column based on their content? Defaults to TRUE. If FALSE, all columns will be returned as character type.
ignore_regex	Character. A regular expression of strings to ignore when trying to guess column types. See details.
has_row_titles	Logical. Should the first column be treated as though it contains titles for each row? Defaults to FALSE. If TRUE, the first column will be marked-up as bold.
totals_rows	Integer. A vector of indices to identify rows that contain totals. These will marked up as bold.
to_clipboard	Logical. Should the output be copied to your clipboard? Defaults to TRUE.

Details

If guess_types is TRUE, then utils::type.convert() is used to coerce each column to the appropriate data type. For example, a column containing numbers will be coerced to numeric. This will fail if the numbers in a given column are formatted to contain non-numeric characters, like '1,234' (comma) or '10%' (percentage symbol). Use ignore_regex so that the process of guessing the data types will ignore these characters.

Value

Character. A string that contains Govspeak Markdown that represents the copy-pasted table.

Examples

```
word_table <- c(
  "Column 1 Column 2 Column 3 Column 4 Column 5
  X 100 1,000 1% 15
  Y 200 2,000 2% 12
  Z 300 3,000 3% [c]"
)

table_to_govspeak(word_table, to_clipboard = FALSE)
```

---

wu_body *Extract Specific Body Elements*

Description

Extract Specific Body Elements

Usage

```
wu_body(doc_list, element = c("p", "tbl"))
```

Arguments

doc_list	List. Output from wu_read.
element	Character. The elements you want to return.

Value

A list with an element for each instance of the desired element.

Examples

```
path <- system.file("examples/simple.docx", package = "wordup")
doc_list <- wu_read(path)
p_list <- wu_body(doc_list, "p")
str(p_list, give.attr = FALSE, max.level = 1)
```

wu_p *Extract All 'p' Body Text and Style to a Dataframe*

Description

Extract All 'p' Body Text and Style to a Dataframe

Usage

```
wu_p(p_list)
```

Arguments

p_list	List. Output from wu_body with argument element = "p".

Value

A data.frame with a row per 'p' element and columns with text and possibly style information.

Examples

```
path <- system.file("examples/simple.docx", package = "wordup")
doc_list <- wu_read(path)
p_list <- wu_body(doc_list, "p")
wu_p(p_list)
```

wu_read *Read a Word File to a List*

Description

Unzips a docx file, reads the XML from /word/document.xml and converts it to a list object for further processing.

Usage

```
wu_read(docx_path)
```

Arguments

docx_path	Character. A path to a docx file.

Value

A nested list.

Examples

```
path <- system.file("examples/simple.docx", package = "wordup")
body_list <- wu_read(path)
str(body_list, give.attr = FALSE, max.level = 3)
```

Index