Package 'ReporteRs'

November 20, 2014

Type Package

Title Microsoft Word, Microsoft Powerpoint and HTML documents generation from R

Version 0.7.0

Date 2014-11-18

Author David Gohel [aut, cre],Bootstrap [ctb, cph] (Bootstrap development team),jQuery [ctb, cph] (The jQuery Foundation),Dmitry Baranovskiy [ctb, cph] (raphael and g.raphael javascript libraries)

Maintainer David Gohel <david.gohel@lysis-consultants.fr>

Description An R package for creating Microsoft Word document (>=2007),Microsoft Powerpoint document (>=2007) and HTML documents from R. There are several features to let you format and present R outputs ; e.g. Editable Vector Graphics, functions for complex tables reporting, reuse of corporate template document (*.docx and *.pptx). You can use the package as a tool for fast reporting and as a tool for reporting automation. The package does not require any installation of Microsoft product to be able to write Microsoft files (docx and pptx).

License GPL-3

Copyright See file COPYRIGHTS.

Depends R (>= 3.0),ReporteRsjars (>= 0.0.2)

Imports rJava

Suggests ggplot2

SystemRequirements java (>= 1.6)

URL http://davidgohel.github.io/ReporteRs/index.html,http://groups.google.com/group/reporters-package

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-11-20 10:31:58

R topics documented:

`ReporteRs-package` *ReporteRs: a package to create document from R*

Description

ReporteRs is an R package for creating Microsoft (Word docx and Powerpoint pptx) and html documents.

Details

Package:	ReporteRs
Type:	Package
Version:	0.7.0
Date:	2014-11-18
License:	GPL (>= 3)
LazyLoad:	yes

To get an r document object:

- `docx` Create a Microsoft Word document object
- `pptx` Create a Microsoft PowerPoint document object
- `bsdoc` Create an HTML document object

The following functions can be used whatever the output format is (docx, pptx, bsdoc).

- `addTitle` Add a title
- `addFlexTable` Add a table (new)
- `addPlot` Add plots
- `addImage` Add external images
- `addMarkdown` Add markdown
- `addParagraph` Add paragraphs of text
- `addRScript` Add an r script
- `writeDoc` Write the document into a file or a directory

`ReporteRs` comes with an object of class `pot` to let you handle text output and format. You can associate a text with formats (font size, font color, etc.), with an hyperlink or with a `Footnote` as a reference note.

`ReporteRs` comes also with an object of class `FlexTable` that let you design and format tabular outputs.

If many text output is needed you may consider using function `addMarkdown`.

Default values:

With ReporteRs, some options can be used to reduce usage of some parameters:

- "ReporteRs-default-font" Default font family to use (default to "Helvetica"). This will be used as default values for argument fontname of addPlot and argument font.family of pot.

  Note that if you do not have Helvetica font, this options must be set to an available font.
- "ReporteRs-fontsize" Default font size to use (default to 11). This will be used as default values for argument pointsize of addPlot and argument font.size of pot.
- "ReporteRs-backtick-color" backtick font color in markdown
- "ReporteRs-backtick-shading-color" backtick shading color in markdown
- "ReporteRs-list-definition" see list.settings.
- "ReporteRs-locale.language" language encoding (for html objects). Default to "en".
- "ReporteRs-locale.region" region encoding (for html objects). Default to "US".

Author(s)

David Gohel <david.gohel@lysis-consultants.fr>

Examples

```
options("ReporteRs-fontsize"=10, "ReporteRs-default-font"="Arial")
numbering.pattern = c( "%1.", "%1. %2.", "%1. %2. %3.",
  "%4.", "%5.", "%6.", "%7.", "%8.", "%9." )

ordered.formats = rep( c( "decimal", "upperRoman", "upperLetter"), 3 )

unordered.formats = rep( c( "square", "disc", "circle"), 3 )

left.indent = seq( from = 0, by = 0.5, length.out = 9)

options("ReporteRs-list-definition" = list(
  ol.left = left.indent,
  ol.hanging = rep( 0.4, 9 ),
  ol.format = ordered.formats,
  ol.pattern = numbering.pattern,
  ul.left = left.indent,
  ul.hanging = rep( 0.4, 9 ),
  ul.format = unordered.formats
  )
)
```

+.pot *pot concatenation*

Description

"+" function is to be used for concatenation of pot elements. Concatenation of 2 pot objects returns a pot (of length 2).

Usage

```
## S3 method for class 'pot'
e1 + e2
```

Arguments

e1	a pot object or a character (vector of length 1).
e2	a pot object or a character (vector of length 1).

Details

at least one of the two objects must be a pot object. If one of the 2 parameters is a simple string, it is converted as a pot object with no associated format ; therefore, document default document style will be used (see addParagraph).

See Also

addParagraph

Examples

```
pot("My tailor", textProperties(color="red") ) + " is " + pot("rich"
, textProperties(font.weight="bold") )
```

add.plot.interactivity *add interactivity on a plot*

Description

add interactivity on elements of a raphael plot. There are three interactive features: popup text when mouse is over an element, execute javascript instructions when clicking the element and execute javascript instructions when double-clicking the element.

Usage

```
add.plot.interactivity(fun, popup.labels, click.actions, dblclick.actions, ...)
```

Arguments

fun	plot function. See details.
popup.labels	labels to display when mouse is over the elements. A character vector. Length must be the same than the number of new elements generated by the plot function.
click.actions	events to run when mouse is clicking the elements. A character vector of javascript instructions. Length must be the same than the number of new elements generated by the plot function.
dblclick.actions	events to run when mouse is double-clicking the elements. A character vector of javascript instructions. Length must be the same than the number of new elements generated by the plot function.
...	arguments for fun.

See Also

bsdoc, addPlot.bsdoc

Examples

```
plot_function = function(){
head( iris )
colorsspec = list( setosa.solid = rgb(153/255, 51/255, 0/255, 1)
, versicolor.solid = rgb(102/255, 102/255, 51/255, 1)
, virginica.solid = rgb(0/255, 51/255, 102/255, 1)
, setosa.area = rgb(153/255, 51/255, 0/255, 0.5)
, versicolor.area = rgb(102/255, 102/255, 51/255, 0.5)
, virginica.area = rgb(0/255, 51/255, 102/255, 0.5)
)
links = list( setosa = "window.open(\"http://en.wikipedia.org/wiki/Iris_(plant)\");"
, versicolor = "window.open(\"http://en.wikipedia.org/wiki/Iris_versicolor\");"
, virginica = "window.open(\"http://en.wikipedia.org/wiki/Iris_virginica\");"
)
# init plot
with( iris, plot( Sepal.Length, Petal.Length , type = "n" ) )
# loop over species
sdata = split( iris, iris$Species )
for(i in names( sdata ) ){
tempdata = sdata[[i]]
###############
# do some calculations to get, predictions and lower bands (3* se)
lo = loess(Petal.Length~Sepal.Length, data = tempdata )
min.x = min(tempdata$Sepal.Length, na.rm = TRUE)
max.x = max(tempdata$Sepal.Length, na.rm = TRUE)
newdata = data.frame( Sepal.Length = seq( min.x, max.x, length.out = 10 ) )
.pred = predict( lo, newdata = newdata, se = TRUE)
lower = .pred$fit - 3*.pred$se.fit
upper = .pred$fit + 3*.pred$se.fit
coord.x = c( newdata$Sepal.Length
```

```
  , rev( newdata$Sepal.Length )
  , NA )
coord.y = c( lower, rev(upper), NA )
# end of calculations
###############
# add interactive elts on polygons
add.plot.interactivity( fun = polygon, x = coord.x , y = coord.y
, col = colorsspec[[paste0( i, ".area")]], border = FALSE
, popup.labels = paste0( i, "\\n", "click on the area")
, click.actions = links[[i]]
)

lines( newdata$Sepal.Length, .pred$fit, col = colorsspec[[paste0( i, ".solid")]] )
# add interactive elts on points
labs = paste( i, "\\n", rep("double click on the point", nrow(tempdata) ), sep = "" )
actions = paste("alert('", format( tempdata$Petal.Length ), "');")
add.plot.interactivity( fun = points
, x = tempdata$Sepal.Length , y = tempdata$Petal.Length
, col = colorsspec[[paste0( i, ".solid")]], pch = 16
, popup.labels = labs
, dblclick.actions = actions
)
}
invisible()
}
library( ReporteRs )
doc = bsdoc( title = "title" )
doc = addPlot( doc, fun = plot_function, width = 8 )
pages = writeDoc( doc, file = "interactive_plot/example.html")
```

`add.pot` *add a paraggraph to an existing set of paragraphs of text*

Description

add a paraggraph to an existing set of paragraphs of text (`set_of_paragraphs` object).

Usage

```
add.pot(x, value)
```

Arguments

`x`	`set_of_paragraphs` object
`value`	`pot` object to add as a new paragraph

See Also

`set_of_paragraphs`, `pot`

Examples

```
pot1 = pot("My tailor", textProperties(color="red") ) + " is " + pot("rich"
, textProperties(font.weight="bold") )
my.pars = set_of_paragraphs( pot1 )
pot2 = pot("Cats", textProperties(color="red") ) + " and " + pot("Dogs"
, textProperties(color="blue") )
my.pars = add.pot( my.pars, pot2 )
```

---

addBootstrapMenu *add a* BootstrapMenu *into a* bsdoc *object.*

Description

add a BootstrapMenu into a bsdoc object.

Usage

```
addBootstrapMenu(doc, bsmenu)
```

Arguments

doc	a bsdoc object.
bsmenu	the BootstrapMenu to add into the bsdoc.

Value

an object of class BootstrapMenu.

See Also

bsdoc, BootstrapMenu

Examples

```
library( ReporteRs )

doc = bsdoc( title = "my document" )

mymenu = BootstrapMenu( title = "my title")

mydd = DropDownMenu( label = "Mon menu" )
mydd = addLinkItem( mydd, label = "GitHub", "http://github.com/")
mydd = addLinkItem( mydd, separator.after = TRUE)
mydd = addLinkItem( mydd, label = "Wikipedia", "http://www.wikipedia.fr")

mymenu = addLinkItem( mymenu, label = "ReporteRs", "http://github.com/davidgohel/ReporteRs")
mymenu = addLinkItem( mymenu, dd = mydd )
```

```
doc = addBootstrapMenu( doc, mymenu )

pages = writeDoc( doc, file = "addBoostrapMenu_example/example.html")
```

---

addColumnBreak *Add a column break into a section*

Description

Add a column break into a section

Usage

```
addColumnBreak(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

addColumnBreak only works with docx documents.

See addColumnBreak.docx for examples.

Value

a document object

See Also

docx, addColumnBreak.docx

---

addColumnBreak.docx *Insert a column break into a docx section*

Description

Insert a page break into a docx section.

Usage

```
## S3 method for class 'docx'
addColumnBreak(doc, ...)
```

Arguments

doc	Object of class docx where column break has to be added
...	further arguments, not used.

Value

an object of class docx.

See Also

docx, addColumnBreak, addSection.docx

Examples

```
doc.filename = "addColumnBreak.docx"
doc = docx( )
doc = addSection(doc, ncol = 2, columns.only = TRUE )
doc = addParagraph( doc = doc, "Text 1.", "Normal" )
doc = addColumnBreak(doc )
doc = addParagraph( doc = doc, "Text 2.", "Normal" )

# Write the object
writeDoc( doc, file = doc.filename )
```

---

addDate *Insert a date into a document object*

Description

Insert a column break

Usage

```
addDate(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

addDate only works for pptx documents. See addSlide.pptx.

See addSlide.pptx for examples.

Value

a document object

See Also

pptx, addSlide.pptx

---

addDate.pptx *Insert a date shape into a document pptx object*

Description

Insert a date into the current slide of a pptx object.

Usage

```
## S3 method for class 'pptx'
addDate(doc, value, str.format = "%Y-%m-%d", ...)
```

Arguments

doc	pptx object
value	character value to add into the date shape of the current slide. optionnal. If missing current date will be used.
str.format	character value to use to format current date (if value is missing).
...	further arguments, not used.

Value

a document object

See Also

pptx, addFooter.pptx, addPageNumber.pptx , strptime, addDate

Examples

```
# Create a new document
doc = pptx( title = "title" )
# add a slide with layout "Title Slide"
doc = addSlide( doc, slide.layout = "Title Slide" )
doc = addTitle( doc, "Presentation title" ) #set the main title
doc = addSubtitle( doc , "This document is generated with ReporteRs.")#set the sub-title

## add a date on the current slide
doc = addDate( doc )
```

```
doc = addSlide( doc, slide.layout = "Title and Content" )
## add a page number on the current slide but not the default text (slide number)
doc = addDate( doc, "Dummy date" )

# Write the object in file "presentation.pptx"
writeDoc( doc, "addDate_example.pptx" )
```

---

addFlexTable *Insert a FlexTable into a document object*

Description

Insert a FlexTable into a document object

FlexTable can be manipulated so that almost any formatting can be specified. See FlexTable for more details.

Usage

```
addFlexTable(doc, flextable, ...)
```

Arguments

doc	document object
flextable	the FlexTable object
...	further arguments passed to other methods

Details

See addFlexTable.docx or addFlexTable.pptx or addFlexTable.bsdoc for examples.

Value

a document object

See Also

FlexTable, addFlexTable.docx , addFlexTable.pptx, addFlexTable.bsdoc

addFlexTable.bsdoc *Insert a FlexTable into an bsdoc object*

Description

Insert a FlexTable into a bsdoc object

Usage

```
## S3 method for class 'bsdoc'
addFlexTable(doc, flextable,
  par.properties = parProperties(text.align = "left"), ...)
```

Arguments

doc	bsdoc object
flextable	the FlexTable object
par.properties	paragraph formatting properties of the paragraph that contains the table. An object of class parProperties
...	further arguments - not used

Value

a bsdoc object

See Also

FlexTable, bsdoc

Examples

```
doc.filename = "addFlexTable_bsdoc/example.html"

# set default font size to 11
options( "ReporteRs-fontsize" = 11 )

doc = bsdoc( )

doc = addTitle( doc, "Title example 1", level = 1 )
##########################################################################

# Create a FlexTable with data.frame mtcars, display rownames
# use different formatting properties for header and body
MyFTable = FlexTable( data = mtcars, add.rownames = TRUE,
  header.cell.props = cellProperties( background.color = "#00557F" ),
  header.text.props = textProperties( color = "white",
    font.size = 11, font.weight = "bold" ),
```

```
  body.text.props = textProperties( font.size = 10 )
)
# zebra stripes - alternate colored backgrounds on table rows
MyFTable = setZebraStyle( MyFTable, odd = "#E1EEf4", even = "white" )

# applies a border grid on table
MyFTable = setFlexTableBorders(MyFTable,
  inner.vertical = borderProperties( color="#0070A8", style="solid" ),
  inner.horizontal = borderNone(),
  outer.vertical = borderProperties( color = "#006699", style = "solid", width = 2 ),
  outer.horizontal = borderProperties( color = "#006699", style = "solid", width = 2 )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )


doc = addTitle( doc, "Title example 2", level = 1 )
#########################################################################

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

# a summary of mtcars
dataset = aggregate( mtcars[, c("disp", "mpg", "wt")]
  , by = mtcars[, c("cyl", "gear", "carb")]
  , FUN = mean )
dataset = dataset[ order(dataset$cyl, dataset$gear, dataset$carb), ]


# set cell padding defaut to 2
baseCellProp = cellProperties( padding = 2 )

# Create a FlexTable with data.frame dataset
MyFTable = FlexTable( data = dataset
  , body.cell.props = baseCellProp
  , header.cell.props = baseCellProp
  , header.par.props = parProperties(text.align = "right" )
)

# set columns widths (in inches)
MyFTable = setFlexTableWidths( MyFTable, widths = c(0.5, 0.5, 0.5, 0.7, 0.7, 0.7) )

# span successive identical cells within column 1, 2 and 3
MyFTable = spanFlexTableRows( MyFTable, j = 1, runs = as.character( dataset$cyl ) )
MyFTable = spanFlexTableRows( MyFTable, j = 2, runs = as.character( dataset$gear ) )
MyFTable = spanFlexTableRows( MyFTable, j = 3, runs = as.character( dataset$carb ) )

# overwrites some text formatting properties
MyFTable[dataset$wt < 3, 6] = textProperties( color="#003366")
MyFTable[dataset$mpg < 20, 5] = textProperties( color="#993300")

# overwrites some paragraph formatting properties
```

```
MyFTable[, 1:3] = parProperties(text.align = "center")
MyFTable[, 4:6] = parProperties(text.align = "right")


Footnote1 = Footnote(  )

par1 = pot("About this reference", textBold( ) )
par2 = pot("Omni ab coalitos pro malivolus obsecrans graviter
cum perquisitor perquisitor pericula saepeque inmunibus coalitos ut.",
  textItalic(font.size = 8) )
Footnote1 = addParagraph( Footnote1, set_of_paragraphs( par1, par2 ),
  parProperties(text.align = "justify"))

Footnote1 = addParagraph( Footnote1, set_of_paragraphs( "list item 1", "list item 2" ),
  parProperties(text.align = "left", list.style = "ordered"))

an_rscript = RScript( text = "ls()
x = rnorm(10)" )
Footnote1 = addParagraph( Footnote1, an_rscript )

MyFTable[1, 1, newpar = TRUE] = pot("a note",
  footnote = Footnote1, format = textBold(color="gray") )

# applies a border grid on table
MyFTable = setFlexTableBorders( MyFTable, footer=TRUE
  , inner.vertical = borderProperties( color = "#666666" )
  , inner.horizontal = borderProperties( color = "#666666" )
  , outer.vertical = borderProperties( width = 2, color = "#666666" )
  , outer.horizontal = borderProperties( width = 2, color = "#666666" )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )


doc = addTitle( doc, "Title example 3", level = 1 )
data = cor( cor(mtcars) )

pal = c( "#D73027", "#F46D43", "#FDAE61", "#FEE08B",
"#D9EF8B", "#A6D96A", "#66BD63", "#1A9850" )
mycut = cut( data,
breaks = c(-1,-0.75,-0.5,-0.25,0,0.25,0.5,0.75,1),
include.lowest = TRUE, label = FALSE )
mycolors = pal[ mycut ]

MyFTable = FlexTable( round(data, 3), add.rownames = TRUE )

# set computed colors
MyFTable = setFlexTableBackgroundColors( MyFTable,
j = seq_len(ncol(data)) + 1,
colors = mycolors )

# cosmetics
```

```
MyFTable = setFlexTableBackgroundColors( MyFTable, i = 1,
colors = "gray", to = "header" )
MyFTable[1, , to = "header"] = textBold(color="white")

MyFTable = setFlexTableBackgroundColors( MyFTable, j = 1, colors = "gray" )
MyFTable[,1] = textBold(color="white")

MyFTable = setFlexTableBorders( MyFTable
, inner.vertical = borderProperties( style = "dashed", color = "white" )
, inner.horizontal = borderProperties( style = "dashed", color = "white"  )
, outer.vertical = borderProperties( width = 2, color = "white"  )
, outer.horizontal = borderProperties( width = 2, color = "white"  )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )

# Write the object
writeDoc( doc, file = doc.filename )
```

`addFlexTable.docx` *Insert a FlexTable into a docx object*

Description

Insert a FlexTable into a docx object

Usage

```
## S3 method for class 'docx'
addFlexTable(doc, flextable,
  par.properties = parProperties(text.align = "left"), bookmark, ...)
```

Arguments

`doc`	docx object
`flextable`	the `FlexTable` object
`par.properties`	paragraph formatting properties of the paragraph that contains the table. An object of class `parProperties`
`bookmark`	a character vector specifying bookmark id (where to put the table). If provided, table will be add after paragraph that contains the bookmark. See `bookmark`. If not provided, table will be added at the end of the document.
`...`	further arguments - not used

Value

a docx object

See Also

FlexTable, docx

Examples

```
doc.filename = "addFlexTable_example.docx"

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

doc = docx( )

doc = addTitle( doc, "Title example 1", level = 1 )
#########################################################################

# Create a FlexTable with data.frame mtcars, display rownames
# use different formatting properties for header and body
MyFTable = FlexTable( data = mtcars, add.rownames = TRUE,
  header.cell.props = cellProperties( background.color = "#00557F" ),
  header.text.props = textProperties( color = "white",
    font.size = 11, font.weight = "bold" ),
  body.text.props = textProperties( font.size = 10 )
)
# zebra stripes - alternate colored backgrounds on table rows
MyFTable = setZebraStyle( MyFTable, odd = "#E1EEf4", even = "white" )

# applies a border grid on table
MyFTable = setFlexTableBorders(MyFTable,
  inner.vertical = borderProperties( color="#0070A8", style="solid" ),
  inner.horizontal = borderNone(),
  outer.vertical = borderProperties( color = "#006699", style = "solid", width = 2 ),
  outer.horizontal = borderProperties( color = "#006699", style = "solid", width = 2 )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )


doc = addTitle( doc, "Title example 2", level = 1 )
#########################################################################

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

# a summary of mtcars
dataset = aggregate( mtcars[, c("disp", "mpg", "wt")]
  , by = mtcars[, c("cyl", "gear", "carb")]
  , FUN = mean )
dataset = dataset[ order(dataset$cyl, dataset$gear, dataset$carb), ]
```

```
# set cell padding defaut to 2
baseCellProp = cellProperties( padding = 2 )

# Create a FlexTable with data.frame dataset
MyFTable = FlexTable( data = dataset
  , body.cell.props = baseCellProp
  , header.cell.props = baseCellProp
  , header.par.props = parProperties(text.align = "right" )
)

# set columns widths (in inches)
MyFTable = setFlexTableWidths( MyFTable, widths = c(0.5, 0.5, 0.5, 0.7, 0.7, 0.7) )

# span successive identical cells within column 1, 2 and 3
MyFTable = spanFlexTableRows( MyFTable, j = 1, runs = as.character( dataset$cyl ) )
MyFTable = spanFlexTableRows( MyFTable, j = 2, runs = as.character( dataset$gear ) )
MyFTable = spanFlexTableRows( MyFTable, j = 3, runs = as.character( dataset$carb ) )

# overwrites some text formatting properties
MyFTable[dataset$wt < 3, 6] = textProperties( color="#003366")
MyFTable[dataset$mpg < 20, 5] = textProperties( color="#993300")

# overwrites some paragraph formatting properties
MyFTable[, 1:3] = parProperties(text.align = "center")
MyFTable[, 4:6] = parProperties(text.align = "right")


Footnote1 = Footnote(  )

par1 = pot("About this reference", textBold( ) )
par2 = pot("Omni ab coalitos pro malivolus obsecrans graviter
cum perquisitor perquisitor pericula saepeque inmunibus coalitos ut.",
  textItalic(font.size = 8) )
Footnote1 = addParagraph( Footnote1, set_of_paragraphs( par1, par2 ),
  parProperties(text.align = "justify"))

Footnote1 = addParagraph( Footnote1, set_of_paragraphs( "list item 1", "list item 2" ),
  parProperties(text.align = "left", list.style = "ordered"))

an_rscript = RScript( text = "ls()
x = rnorm(10)" )
Footnote1 = addParagraph( Footnote1, an_rscript )

MyFTable[1, 1, newpar = TRUE] = pot("a note",
  footnote = Footnote1, format = textBold(color="gray") )

# applies a border grid on table
MyFTable = setFlexTableBorders( MyFTable, footer=TRUE
  , inner.vertical = borderProperties( color = "#666666" )
  , inner.horizontal = borderProperties( color = "#666666" )
  , outer.vertical = borderProperties( width = 2, color = "#666666" )
  , outer.horizontal = borderProperties( width = 2, color = "#666666" )
)
```

```
# add MyFTable into document
doc = addFlexTable( doc, MyFTable )


doc = addTitle( doc, "Title example 3", level = 1 )
data = cor( cor(mtcars) )

pal = c( "#D73027", "#F46D43", "#FDAE61", "#FEE08B",
"#D9EF8B", "#A6D96A", "#66BD63", "#1A9850" )
mycut = cut( data,
breaks = c(-1,-0.75,-0.5,-0.25,0,0.25,0.5,0.75,1),
include.lowest = TRUE, label = FALSE )
mycolors = pal[ mycut ]

MyFTable = FlexTable( round(data, 3), add.rownames = TRUE )

# set computed colors
MyFTable = setFlexTableBackgroundColors( MyFTable,
j = seq_len(ncol(data)) + 1,
colors = mycolors )

# cosmetics
MyFTable = setFlexTableBackgroundColors( MyFTable, i = 1,
colors = "gray", to = "header" )
MyFTable[1, , to = "header"] = textBold(color="white")

MyFTable = setFlexTableBackgroundColors( MyFTable, j = 1, colors = "gray" )
MyFTable[,1] = textBold(color="white")

MyFTable = setFlexTableBorders( MyFTable
, inner.vertical = borderProperties( style = "dashed", color = "white" )
, inner.horizontal = borderProperties( style = "dashed", color = "white"  )
, outer.vertical = borderProperties( width = 2, color = "white"  )
, outer.horizontal = borderProperties( width = 2, color = "white"  )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )

# Write the object
writeDoc( doc, file = doc.filename )
```

---

addFlexTable.pptx *Insert a FlexTable into a pptx object*

Description

Insert a FlexTable into a pptx object

Usage

```
## S3 method for class 'pptx'
addFlexTable(doc, flextable, offx, offy, width, height, ...)
```

Arguments

doc	docx object
flextable	the FlexTable object
offx	optional, x position of the shape (top left position of the bounding box) in inch. See details.
offy	optional, y position of the shape (top left position of the bounding box) in inch. See details.
width	optional, width of the shape in inch. See details.
height	optional, height of the shape in inch. See details.
...	further arguments - not used

Details

If arguments offx, offy, width, height are missing, position and dimensions will be defined by the width and height of the next available shape of the slide. This dimensions can be defined in the layout of the PowerPoint template used to create the pptx object.

If arguments offx, offy, width, height are provided, they become position and dimensions of the new shape.

Value

a pptx object

See Also

FlexTable, pptx

Examples

```
doc.filename = "addFlexTable_example.pptx"

# set default font size to 24
options( "ReporteRs-fontsize" = 24 )

doc = pptx( title = "title" )

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )

doc = addTitle( doc, "Title example 1" )
######################################################################
```

```
# Create a FlexTable with data.frame mtcars, display rownames
# use different formatting properties for header and body
MyFTable = FlexTable( data = mtcars, add.rownames = TRUE,
  header.cell.props = cellProperties( background.color = "#00557F" ),
  header.text.props = textProperties( color = "white",
    font.size = 11, font.weight = "bold" ),
  body.text.props = textProperties( font.size = 10 )
)
# zebra stripes - alternate colored backgrounds on table rows
MyFTable = setZebraStyle( MyFTable, odd = "#E1EEf4", even = "white" )

# applies a border grid on table
MyFTable = setFlexTableBorders(MyFTable,
  inner.vertical = borderProperties( color="#0070A8", style="solid" ),
  inner.horizontal = borderNone(),
  outer.vertical = borderProperties( color = "#006699", style = "solid", width = 2 ),
  outer.horizontal = borderProperties( color = "#006699", style = "solid", width = 2 )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )


doc = addTitle( doc, "Title example 2" )
######################################################################

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

# a summary of mtcars
dataset = aggregate( mtcars[, c("disp", "mpg", "wt")]
  , by = mtcars[, c("cyl", "gear", "carb")]
  , FUN = mean )
dataset = dataset[ order(dataset$cyl, dataset$gear, dataset$carb), ]


# set cell padding defaut to 2
baseCellProp = cellProperties( padding = 2 )

# Create a FlexTable with data.frame dataset
MyFTable = FlexTable( data = dataset
  , body.cell.props = baseCellProp
  , header.cell.props = baseCellProp
  , header.par.props = parProperties(text.align = "right" )
)

# set columns widths (in inches)
MyFTable = setFlexTableWidths( MyFTable, widths = c(0.5, 0.5, 0.5, 0.7, 0.7, 0.7) )

# span successive identical cells within column 1, 2 and 3
```

```
MyFTable = spanFlexTableRows( MyFTable, j = 1, runs = as.character( dataset$cyl ) )
MyFTable = spanFlexTableRows( MyFTable, j = 2, runs = as.character( dataset$gear ) )
MyFTable = spanFlexTableRows( MyFTable, j = 3, runs = as.character( dataset$carb ) )

# overwrites some text formatting properties
MyFTable[dataset$wt < 3, 6] = textProperties( color="#003366")
MyFTable[dataset$mpg < 20, 5] = textProperties( color="#993300")

# overwrites some paragraph formatting properties
MyFTable[, 1:3] = parProperties(text.align = "center")
MyFTable[, 4:6] = parProperties(text.align = "right")


Footnote1 = Footnote(  )

par1 = pot("About this reference", textBold( ) )
par2 = pot("Omni ab coalitos pro malivolus obsecrans graviter
cum perquisitor perquisitor pericula saepeque inmunibus coalitos ut.",
  textItalic(font.size = 8) )
Footnote1 = addParagraph( Footnote1, set_of_paragraphs( par1, par2 ),
  parProperties(text.align = "justify"))

Footnote1 = addParagraph( Footnote1, set_of_paragraphs( "list item 1", "list item 2" ),
  parProperties(text.align = "left", list.style = "ordered"))

an_rscript = RScript( text = "ls()
x = rnorm(10)" )
Footnote1 = addParagraph( Footnote1, an_rscript )

MyFTable[1, 1, newpar = TRUE] = pot("a note",
  footnote = Footnote1, format = textBold(color="gray") )

# applies a border grid on table
MyFTable = setFlexTableBorders( MyFTable, footer=TRUE
  , inner.vertical = borderProperties( color = "#666666" )
  , inner.horizontal = borderProperties( color = "#666666" )
  , outer.vertical = borderProperties( width = 2, color = "#666666" )
  , outer.horizontal = borderProperties( width = 2, color = "#666666" )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )

# add MyFTable into document
doc = addFlexTable( doc, MyFTable, offx = 7, offy = 2, width = 3, height = 3)

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )


doc = addTitle( doc, "Title example 3" )
data = cor( cor(mtcars) )
```

```
pal = c( "#D73027", "#F46D43", "#FDAE61", "#FEE08B",
"#D9EF8B", "#A6D96A", "#66BD63", "#1A9850" )
mycut = cut( data,
breaks = c(-1,-0.75,-0.5,-0.25,0,0.25,0.5,0.75,1),
include.lowest = TRUE, label = FALSE )
mycolors = pal[ mycut ]

MyFTable = FlexTable( round(data, 3), add.rownames = TRUE )

# set computed colors
MyFTable = setFlexTableBackgroundColors( MyFTable,
j = seq_len(ncol(data)) + 1,
colors = mycolors )

# cosmetics
MyFTable = setFlexTableBackgroundColors( MyFTable, i = 1,
colors = "gray", to = "header" )
MyFTable[1, , to = "header"] = textBold(color="white")

MyFTable = setFlexTableBackgroundColors( MyFTable, j = 1, colors = "gray" )
MyFTable[,1] = textBold(color="white")

MyFTable = setFlexTableBorders( MyFTable
, inner.vertical = borderProperties( style = "dashed", color = "white" )
, inner.horizontal = borderProperties( style = "dashed", color = "white"  )
, outer.vertical = borderProperties( width = 2, color = "white"  )
, outer.horizontal = borderProperties( width = 2, color = "white"  )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )

# Write the object
writeDoc( doc, file = doc.filename )
```

`addFooter` *Insert a footer into a document object*

Description

Insert a footer into a document object

Usage

```
addFooter(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

addFooter only works for pptx and bsdoc documents.

Value

a document object

See Also

pptx, addFooter.pptx , bsdoc, addFooter.bsdoc

---

addFooter.bsdoc *Add text in footer of a* bsdoc *object*

Description

Add text in footer of a bsdoc object. The function has the same behaviour than addParagraph, except that its content will be written in the footer part of the bsdoc instead of the body of the document.

Usage

```
## S3 method for class 'bsdoc'
addFooter(doc, value, par.properties = parProperties(),
  restart.numbering = FALSE, ...)
```

Arguments

doc	bsdoc object
value	text to add to in the footer as paragraphs: an object of class pot or set_of_paragraphs or a character vector.
par.properties	parProperties to apply to paragraphs.
restart.numbering	boolean value. If TRUE, next numbered list counter will be set to 1.
...	further arguments, not used.

Value

a bsdoc object

See Also

bsdoc

Examples

```
# Create a new document
doc = bsdoc( title = "title" )

doc = addFooter( doc, value = pot( "Code licensed under ",
    format = textProperties(color="gray") ) +
  pot("GPL-3", format = textProperties(color="#428bca"),
    hyperlink = "https://gnu.org/licenses/gpl.html" ) +
  pot(".", format = textProperties(color="gray") ),
par.properties = parCenter( padding = 2 )
)

# write the html object in a directory
writeDoc( doc, "addFooter/example.html")
```

---

addFooter.pptx *Insert a footer shape into a document pptx object*

Description

Insert a footer shape into the current slide of a pptx object.

Usage

```
## S3 method for class 'pptx'
addFooter(doc, value, ...)
```

Arguments

doc	pptx object
value	character value to add into the footer shape of the current slide.
...	further arguments, not used.

Value

a document object

See Also

pptx, addDate.pptx , addPageNumber.pptx

Examples

```
# Create a new document
doc = pptx( title = "title" )

# add a slide with layout "Title Slide"
doc = addSlide( doc, slide.layout = "Title Slide" )
doc = addTitle( doc, "Presentation title" ) #set the main title
doc = addSubtitle( doc , "This document is generated with ReporteRs.")#set the sub-title

## add a page number on the current slide
doc = addFooter( doc, "Hi!" )

writeDoc( doc, "addFooter_example.pptx" )
```

`addFooterRow` *add footer in a FlexTable*

Description

add a footer row in a FlexTable

Usage

```
addFooterRow(x, value, colspan, text.properties, par.properties,
  cell.properties)
```

Arguments

`x`	a `FlexTable` object
`value`	`FlexRow` object to insert as a footer row or a character vector specifying labels to use as columns labels.
`colspan`	integer vector. Optional. Applies only when argument `value` is a character vector. Vector specifying the number of columns to span for each corresponding value (in `values`).
`text.properties`	Optional. textProperties to apply to each cell. Used only if values are not missing.
`par.properties`	Optional. parProperties to apply to each cell. Used only if values are not missing.
`cell.properties`	Optional. cellProperties to apply to each cell. Used only if values are not missing.

See Also

FlexTable, addHeaderRow , alterFlexTable

Examples

```
############################
# simple example
############################

data(pbc_summary)

MyFTable = FlexTable( data = pbc_summary[,1:4], header.columns = TRUE )

# add a footer row with 1 cell that spans four columns
MyFTable = addFooterRow( MyFTable
  , value = c("Mean of serum cholesterol (mg/dl)"), colspan = 4 )
########################################
# example with FlexRow objects usage
########################################

data(pbc_summary)

# create a FlexTable
MyFTable = FlexTable( data = pbc_summary[,1:4] )

# define a complex formatted text
mytext = pot("*"
    , format = textProperties(vertical.align="superscript", font.size = 9)
  ) + pot( " Mean of serum cholesterol (mg/dl)"
    , format = textProperties(font.size = 9)
  )

# create a FlexRow - container for 1 cell
footerRow = FlexRow()
footerRow[1] = FlexCell( mytext, colspan = 4 )

# add the FlexRow to the FlexTable
MyFTable = addFooterRow( MyFTable, footerRow )
```

---

addHeaderRow *add header in a FlexTable*

Description

add a header row in a FlexTable

Usage

```
addHeaderRow(x, value, colspan, text.properties, par.properties,
  cell.properties)
```

Arguments

x	a FlexTable object
value	FlexRow object to insert as an header row or a character vector specifying labels to use as columns labels.
colspan	integer vector. Optional. Applies only when argument value is a character vector. Vector specifying the number of columns to span for each corresponding value (in values).
text.properties	Optional. textProperties to apply to each cell. Used only if values are not missing. Default is the value of argument header.text.props provided to funtion FlexTable when object has been created
par.properties	Optional. parProperties to apply to each cell. Used only if values are not missing. Default is the value of argument header.par.props provided to funtion FlexTable when object has been created
cell.properties	Optional. cellProperties to apply to each cell. Used only if values are not missing. Default is the value of argument header.cell.props provided to funtion FlexTable when object has been created

See Also

FlexTable, addFooterRow , alterFlexTable

Examples

```
############################
# simple example
############################

data(pbc_summary)

# set header.columns to FALSE so that default header row is not added in
# the FlexTable object
# We do only want the 4 first columns of the dataset
MyFTable = FlexTable( data = pbc_summary[,1:4], header.columns = FALSE )

# add an header row with 2 cells, the first one spans three columns
# and the second one spans one column (normal width)
MyFTable = addHeaderRow( MyFTable
  , value = c("By variables", "Serum cholesterol (mg/dl)")
  , colspan = c( 3, 1)
)
```

```
# add an header row with table columns labels
MyFTable = addHeaderRow( MyFTable
  , value=c("Treatment", "Sex", "Status", "Mean")
)
#######################################
# how to change default formats
#######################################
data(pbc_summary)

MyFTable = FlexTable( data = pbc_summary[,1:4], header.columns = FALSE
  , body.cell.props = cellProperties(border.color="#7895A2")
)
# add an header row with table columns labels
MyFTable = addHeaderRow( MyFTable
  , text.properties = textProperties(color = "#517281", font.weight="bold")
  , cell.properties = cellProperties(border.color="#7895A2")
  , value=c("Treatment", "Sex", "Status", "Serum cholesterol (mg/dl)")
)
#########################################
# example with FlexRow objects usage
#########################################

data(pbc_summary)

# cell styles definitions
cellProperties1 = cellProperties( border.top.width = 2
    , border.right.style="dashed"
    , border.bottom.style="dashed"
    , border.left.width = 2 )
cellProperties2 = cellProperties( border.top.width = 2
    , border.left.style="dashed"
    , border.bottom.style="dashed"
    , border.right.width = 2 )

# create a FlexTable
MyFTable = FlexTable( data = pbc_summary[,1:4]
  , header.columns = FALSE, body.text.props=textProperties() )

# create a FlexRow - container for 2 cells
headerRow = FlexRow()
headerRow[1] = FlexCell( "By variables", colspan = 3, cell.properties = cellProperties1 )
headerRow[2] = FlexCell( "Serum cholesterol (mg/dl)", cell.properties = cellProperties2 )
# add the FlexRow to the FlexTable
MyFTable = addHeaderRow( MyFTable, headerRow )


# cell styles definitions
cellProperties3 = cellProperties( border.bottom.width = 2, border.left.width = 2
    , border.right.style="dashed"
    , border.top.style="dashed"
)
cellProperties4 = cellProperties( border.bottom.width = 2
    , border.right.style="dashed", border.left.style="dashed"
```

```
    , border.top.style="dashed" )
cellProperties5 = cellProperties( border.bottom.width = 2, border.right.width = 2
    , border.left.style="dashed"
    , border.top.style="dashed"
)

# create a FlexRow - container for 4 cells
headerRow = FlexRow()
headerRow[1] = FlexCell( "Treatment", cell.properties = cellProperties3 )
headerRow[2] = FlexCell( "Sex", cell.properties = cellProperties4 )
headerRow[3] = FlexCell( "Status", cell.properties = cellProperties4 )
headerRow[4] = FlexCell( "Mean", cell.properties = cellProperties5 )
# add the FlexRow to the FlexTable
MyFTable = addHeaderRow( MyFTable, headerRow )
MyFTable = setFlexTableBorders( MyFTable
  , inner.vertical = borderProperties( style = "dashed" )
  , inner.horizontal = borderProperties( style = "dashed" )
  , outer.vertical = borderProperties( width = 2 )
  , outer.horizontal = borderProperties( width = 2 )
)
```

`addIframe` *Add an iframe into a document object*

Description

Add an iframe into a document object

Usage

```
addIframe(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Value

a document object

See Also

addIframe.bsdoc

addIframe.bsdoc *Insert an iframe into a bsdoc object*

Description

Insert an iframe into a bsdoc object

Usage

```
## S3 method for class 'bsdoc'
addIframe(doc, src, width, height, seamless = FALSE,
  par.properties = parProperties(text.align = "center", padding = 5), ...)
```

Arguments

doc	bsdoc object where iframe has to be added
src	url of the document to embed in the iframe
width	Specifies the width of an iframe
height	Specifies the height of an iframe
seamless	Specifies that the iframe should look like it is a part of the containing document
par.properties	paragraph formatting properties of the paragraph that contains iframe. An object of class parProperties
...	further arguments, not used.

Value

an object of class bsdoc.

addImage *Add an external image into a document object*

Description

Add an external image into a document object

Usage

```
addImage(doc, filename, ...)
```

Arguments

doc	document object
filename	"character" value, complete filename of the external image
...	further arguments passed to other methods

Details

See addImage.docx or addImage.pptx or addImage.bsdoc for examples.

Value

a document object

See Also

docx, addImage.docx , pptx, addImage.pptx , bsdoc, addImage.bsdoc

---

addImage.bsdoc *Insert an external image into a bsdoc object*

Description

Add an external image into a bsdoc object.

Usage

```
## S3 method for class 'bsdoc'
addImage(doc, filename, width, height,
  par.properties = parProperties(text.align = "center", padding = 5), ...)
```

Arguments

doc	bsdoc object where external image has to be added
filename	"character" value, complete filename of the external image
width	image width in pixel
height	image height in pixel
par.properties	paragraph formatting properties of the paragraph that contains images. An object of class parProperties
...	further arguments, not used.

Value

an object of class bsdoc.

See Also

bsdoc, addPlot.bsdoc , addImage

Examples

```
doc.filename = "addImage_bsdoc/example.html"

# set default font size to 11
options( "ReporteRs-fontsize" = 11 )

doc = bsdoc( )
# the file 'logo.jpg' only exists in R for Windows
img.file = file.path( Sys.getenv("R_HOME"), "doc", "html", "logo.jpg" )
doc = addImage(doc, img.file, width = 100, height = 76 )

# Write the object
writeDoc( doc, file = doc.filename )
```

---

addImage.docx *Add external image into a docx object*

Description

Add external images into a docx object.

Usage

```
## S3 method for class 'docx'
addImage(doc, filename, bookmark,
  par.properties = parProperties(text.align = "center", padding = 5), ...)
```

Arguments

doc	Object of class docx where external image has to be added
filename	"character" value, complete filename of the external image
bookmark	a character value ; id of the Word bookmark to replace by the image. optional. if missing, image is added at the end of the document. See bookmark.
par.properties	paragraph formatting properties of the paragraph that contains images. An object of class parProperties
...	further arguments, not used.

Value

an object of class docx.

See Also

docx, addPlot.docx , addImage, bookmark

Examples

```
# Create a new document
doc = docx( title = "title" )

# the file 'logo.jpg' only exists in R for Windows
img.file = file.path( Sys.getenv("R_HOME"), "doc", "html", "logo.jpg" )
doc = addImage(doc, img.file )

# Write the object in file "addImage_example.docx"
writeDoc( doc, "addImage_example.docx" )
```

`addImage.pptx` *Insert an external image into a pptx object*

Description

Add an external image into a `pptx` object.

Usage

```
## S3 method for class 'pptx'
addImage(doc, filename, offx, offy, width, height, ...)
```

Arguments

`doc`	`pptx` object where external image has to be added
`filename`	`"character"` value, complete filename of the external image
`offx`	optional, x position of the shape (top left position of the bounding box) in inch. See details.
`offy`	optional, y position of the shape (top left position of the bounding box) in inch See details.
`width`	optional, width of the shape in inch See details.
`height`	optional, height of the shape in inch See details.
`...`	further arguments, not used.

Details

Image is added to the next free 'content' shape of the current slide. See `slide.layouts.pptx` to view the slide layout.

If arguments offx, offy, width, height are missing, position and dimensions will be defined by the width and height of the next available shape of the slide. This dimensions can be defined in the layout of the PowerPoint template used to create the `pptx` object.

If arguments offx, offy, width, height are provided, they become position and dimensions of the new shape.

Value

an object of class pptx.

See Also

pptx, addPlot.pptx , addImage

Examples

```
# Create a new document
doc = pptx( title = "title" )

# add a slide with layout "Title and Content" then add an image
doc = addSlide( doc, slide.layout = "Title and Content" )

# the file 'logo.jpg' only exists in R for Windows
img.file = file.path( Sys.getenv("R_HOME"), "doc", "html", "logo.jpg" )
doc = addImage(doc, img.file )

writeDoc( doc, "addImage_example.pptx" )
```

---

addJavascript *add javascript into a bsdoc object*

Description

add javascript into a bsdoc object.

Usage

```
addJavascript(doc, file, text)
```

Arguments

doc	a bsdoc object.
file	a javascript file. Not used if text is provided.
text	character vector. The javascript text to parse. Not used if file is provided.

Value

an object of class bsdoc.

addLinkItem *add an item in a* BootstrapMenu *or a* DropDownMenu

Description

add an item in a BootstrapMenu or DropDownMenu object. An item can be a simple link associated with a label or a DropDownMenu object.

Usage

```
addLinkItem(x, label, link, dd, separator.before = FALSE,
  separator.after = FALSE, active = FALSE)
```

Arguments

x	a DropDownMenu or a BootstrapMenu object.
label	"character" value: label of a simple link. If used, argument link must be specified.
link	"character" value: hyperlink value. If used, argument label must be specified.
dd	a DropDownMenu object to insert into the menu. If used, arguments label and link will be ignored.
separator.before	if TRUE, a separator will be inserted before the new item. It only applies when x is a DropDownMenu object.
separator.after	if TRUE, a separator will be inserted after the new item. It only applies when x is a DropDownMenu object.
active	if TRUE, the item will be declared as active (highlighted).

Value

an object of class BootstrapMenu.

See Also

bsdoc, addBootstrapMenu

Examples

```
mymenu = BootstrapMenu( title = "my title")

mydd = DropDownMenu( label = "Mon menu" )
mydd = addLinkItem( mydd, label = "GitHub", "http://github.com/")
mydd = addLinkItem( mydd, separator.after = TRUE)
```

```
mydd = addLinkItem( mydd, label = "Wikipedia", "http://www.wikipedia.fr")

mymenu = addLinkItem( mymenu, label = "ReporteRs", "http://github.com/davidgohel/ReporteRs")
mymenu = addLinkItem( mymenu, dd = mydd )
```

---

addMarkdown *Add a markdown text or file*

Description

Add markdown into a document object

The markdown definition used is John Gruber documented here: http://daringfireball.net/projects/markdown/syntax.

Images are not available as addImage or addPlot is available. Pandoc footnotes have been added (see http://johnmacfarlane.net/pandoc/README.html#footnotes.

Usage

```
addMarkdown(doc, file, text, ...)
```

Arguments

doc	document object
file	markdown file. Not used if text is provided.
text	character vector. The markdown to parse.
...	further arguments passed to other methods

Value

a document object

See Also

docx, addMarkdown.docx , bsdoc, addMarkdown.bsdoc , pptx, addMarkdown.pptx

addMarkdown.bsdoc *Add a markdown text or file into an bsdoc object*

Description

Add markdown into a bsdoc object.

Usage

```
## S3 method for class 'bsdoc'
addMarkdown(doc, file, text,
  text.properties = textProperties(font.size =
  getOption("ReporteRs-fontsize")),
  default.par.properties = parProperties(text.align = "justify"),
  blockquote.par.properties = parProperties(padding.top = 0, padding.bottom =
  0, shading.color = "#eeeeee"),
  code.par.properties = parProperties(shading.color = "#eeeeee"),
  hr.border = borderSolid(width = 2, color = "gray10"), ...)
```

Arguments

doc	Object of class bsdoc where markdown has to be added
file	markdown file. Not used if text is provided.
text	character vector. The markdown text to parse.
text.properties	default textProperties object
default.par.properties	default parProperties object
blockquote.par.properties	parProperties object used for blockquote blocks.
code.par.properties	parProperties object used for code blocks.
hr.border	borderProperties object used for horizontal rules.
...	further arguments, not used.

Details

You can configure backtick rendering (single or double backtick) with options "ReporteRs-backtick-color" and "ReporteRs-backtick-shading-color".

Value

an object of class bsdoc.

See Also

bsdoc, addMarkdown

Examples

```
doc.filename = "addMarkdown_bsdoc/example.html"

# set default font size to 11
options( "ReporteRs-fontsize" = 11 )

doc = bsdoc( )
mkd = "# This is a title 1

This is a link to the [cran] (http://cran.r-project.org/).

This paragraph demonstrates note usage[^anote]. It also show an example of
reference link like this one: [DaringFireball][1].

[^anote]: Here's a note with multiple blocks.

    This paragraph is indented and belongs to the previous footnote.

    This paragraph also belongs to the previous footnote.
    In this way, multi-paragraph footnotes work like
    multi-paragraph list items.

        ls()

    * This list item belongs to the previous footnote.

## This is a title 1.1

Ex turba vero imae sortis et paupertinae in tabernis aliqui pernoctant
vinariis, non nulli velariis umbraculorum theatralium latent, quae Campanam
imitatus lasciviam Catulus in aedilitate sua suspendit omnium primus

Aut pugnaciter aleis certant turpi sono fragosis naribus introrsum reducto
spiritu concrepantes; aut quod est studiorum omnium maximum ab ortu lucis
ad vesperam sole fatiscunt vel pluviis, per minutias aurigarum equorumque
praecipua vel delicta scrutantes.

Paragraphs must be separated by a blank line. Basic formatting of *italics* and **bold** is
supported. This *can be **nested** like* so. Formatting of ``backtick`` is also
supported.

# This is a title 2

## Ordered list

1. Item 1
2. A second item
```

```
3. Number 3


## Unordered list

* An item
* Another item
* Yet another item

# Code block

    x = rnorm( 1000 )
    plot( density( x ) )

You can also make `inline code` to add code into other things.

# Quote

> Here is a quote. Quotes are indented
when used.
> > Subquotes are also supported.

# URLs

* A named link to [DaringFireball][1].
* Another named link to [DaringFireball](http://daringfireball.net/projects/markdown)
* Sometimes you just want a URL like <http://daringfireball.net/projects/markdown>.

# Miscellaneous

## Horizontal rule

A horizontal rule is a line that goes across the middle of the page.

---------

It's sometimes useful for breaking things up.

## Images

This implementation does not support images yet. Use addImage or addPlot instead.

[1]: http://daringfireball.net/projects/markdown/
"

doc = addMarkdown( doc, text = mkd,
default.par.properties = parProperties(text.align = "justify",
padding.left = 0) )

# Write the object
writeDoc( doc, file = doc.filename )
```

addMarkdown.docx *Add a markdown text or file into a docx object*

Description

Add markdown into a docx object.

Usage

```
## S3 method for class 'docx'
addMarkdown(doc, file, text,
  text.properties = textProperties(font.size =
  getOption("ReporteRs-fontsize")),
  default.par.properties = parProperties(text.align = "justify"),
  blockquote.par.properties = parProperties(padding = 6, shading.color =
  "#eeeeee"), code.par.properties = parProperties(shading.color = "#eeeeee"),
  hr.border = borderSolid(width = 2, color = "gray10"), ...)
```

Arguments

doc	Object of class docx where markdown has to be added
file	markdown file. Not used if text is provided.
text	character vector. The markdown text to parse.
text.properties	default textProperties object
default.par.properties	default parProperties object
blockquote.par.properties	parProperties object used for blockquote blocks.
code.par.properties	parProperties object used for code blocks.
hr.border	borderProperties object used for horizontal rules.
...	further arguments, not used.

Details

You can configure backtick rendering (single or double backtick) with options "ReporteRs-backtick-color" and "ReporteRs-backtick-shading-color".

Value

an object of class docx.

See Also

docx, addMarkdown

Examples

```
doc.filename = "addMarkdown_example.docx"

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

doc = docx( )
mkd = "# This is a title 1

This is a link to the [cran] (http://cran.r-project.org/).

This paragraph demonstrates note usage[^anote]. It also show an example of
reference link like this one: [DaringFireball][1].

[^anote]: Here's a note with multiple blocks.

    This paragraph is indented and belongs to the previous footnote.

    This paragraph also belongs to the previous footnote.
    In this way, multi-paragraph footnotes work like
    multi-paragraph list items.

        ls()

    * This list item belongs to the previous footnote.

## This is a title 1.1

Ex turba vero imae sortis et paupertinae in tabernis aliqui pernoctant
vinariis, non nulli velariis umbraculorum theatralium latent, quae Campanam
imitatus lasciviam Catulus in aedilitate sua suspendit omnium primus

Aut pugnaciter aleis certant turpi sono fragosis naribus introrsum reducto
spiritu concrepantes; aut quod est studiorum omnium maximum ab ortu lucis
ad vesperam sole fatiscunt vel pluviis, per minutias aurigarum equorumque
praecipua vel delicta scrutantes.

Paragraphs must be separated by a blank line. Basic formatting of *italics* and **bold** is
supported. This *can be **nested** like* so. Formatting of ``backtick`` is also
supported.

# This is a title 2

## Ordered list

1. Item 1
2. A second item
3. Number 3


## Unordered list
```

```
* An item
* Another item
* Yet another item

# Code block

    x = rnorm( 1000 )
    plot( density( x ) )

You can also make `inline code` to add code into other things.

# Quote

> Here is a quote. Quotes are indented
when used.
> > Subquotes are also supported.

# URLs

* A named link to [DaringFireball][1].
* Another named link to [DaringFireball](http://daringfireball.net/projects/markdown)
* Sometimes you just want a URL like <http://daringfireball.net/projects/markdown>.

# Miscellaneous

## Horizontal rule

A horizontal rule is a line that goes across the middle of the page.

---------

It's sometimes useful for breaking things up.

## Images

This implementation does not support images yet. Use addImage or addPlot instead.

[1]: http://daringfireball.net/projects/markdown/
"

doc = addMarkdown( doc, text = mkd,
default.par.properties = parProperties(text.align = "justify",
padding.left = 0) )

# Write the object
writeDoc( doc, file = doc.filename )
```

---

addMarkdown.pptx *Add a markdown text or file into a pptx object*

Description

Add markdown into a pptx object.

Usage

```
## S3 method for class 'pptx'
addMarkdown(doc, file, text,
  text.properties = textProperties(font.size =
  getOption("ReporteRs-fontsize")),
  default.par.properties = parProperties(text.align = "justify"),
  blockquote.par.properties = parProperties(padding = 6, shading.color =
  "#eeeeee"), code.par.properties = parProperties(shading.color = "#eeeeee"),
  hr.border = borderSolid(width = 2, color = "gray10"), ...)
```

Arguments

doc	Object of class pptx where markdown has to be added
file	markdown file. Not used if text is provided.
text	character vector. The markdown text to parse.
text.properties	default textProperties object
default.par.properties	default parProperties object
blockquote.par.properties	parProperties object used for blockquote blocks.
code.par.properties	parProperties object used for code blocks.
hr.border	borderProperties object used for horizontal rules.
...	further arguments, not used.

Details

You can configure backtick rendering (single or double backtick) with options "ReporteRs-backtick-color" and "ReporteRs-backtick-shading-color".

The implemented markdown for pptx objects is minimal. It does not support horizontal rules, titles and footnotes.

Value

an object of class pptx.

See Also

pptx, addMarkdown

Examples

```
doc.filename = "addMarkdown_example.pptx"

# set default font size to 24
options( "ReporteRs-fontsize" = 24 )

doc = pptx( title = "title" )

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )

doc = addTitle( doc, "Title example 1" )
mkd = "# This is a title 1

This is a link to the [cran] (http://cran.r-project.org/).

This paragraph demonstrates note usage[^anote]. It also show an example of
reference link like this one: [DaringFireball][1].

[^anote]: Here's a note with multiple blocks.

    This paragraph is indented and belongs to the previous footnote.

    This paragraph also belongs to the previous footnote.
    In this way, multi-paragraph footnotes work like
    multi-paragraph list items.

        ls()

    * This list item belongs to the previous footnote.

## This is a title 1.1

Ex turba vero imae sortis et paupertinae in tabernis aliqui pernoctant
vinariis, non nulli velariis umbraculorum theatralium latent, quae Campanam
imitatus lasciviam Catulus in aedilitate sua suspendit omnium primus

Aut pugnaciter aleis certant turpi sono fragosis naribus introrsum reducto
spiritu concrepantes; aut quod est studiorum omnium maximum ab ortu lucis
ad vesperam sole fatiscunt vel pluviis, per minutias aurigarum equorumque
praecipua vel delicta scrutantes.

Paragraphs must be separated by a blank line. Basic formatting of *italics* and **bold** is
supported. This *can be **nested** like* so. Formatting of ``backtick`` is also
supported.

# This is a title 2

## Ordered list

1. Item 1
```

```
2. A second item
3. Number 3


## Unordered list

* An item
* Another item
* Yet another item

# Code block

    x = rnorm( 1000 )
    plot( density( x ) )

You can also make `inline code` to add code into other things.

# Quote

> Here is a quote. Quotes are indented
when used.
> > Subquotes are also supported.

# URLs

* A named link to [DaringFireball][1].
* Another named link to [DaringFireball](http://daringfireball.net/projects/markdown)
* Sometimes you just want a URL like <http://daringfireball.net/projects/markdown>.

# Miscellaneous

## Horizontal rule

A horizontal rule is a line that goes across the middle of the page.

---------

It's sometimes useful for breaking things up.

## Images

This implementation does not support images yet. Use addImage or addPlot instead.

[1]: http://daringfireball.net/projects/markdown/
"

doc = addMarkdown( doc, text = mkd,
default.par.properties = parProperties(text.align = "justify",
padding.left = 0) )

# Write the object
writeDoc( doc, file = doc.filename )
```

addPageBreak *Add a page break into a document object*

Description

Add a page break into a document object

Usage

```
addPageBreak(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

addPageBreak only works with docx documents.

See addPageBreak.docx for examples.

Value

a document object

See Also

docx, addPageBreak.docx

addPageBreak.docx *Insert a page break into a docx object*

Description

Insert a page break into a docx object.

Usage

```
## S3 method for class 'docx'
addPageBreak(doc, ...)
```

Arguments

doc	Object of class docx where page break has to be added
...	further arguments, not used.

Value

an object of class docx.

See Also

docx, addPageBreak

Examples

```
doc = docx( title = "title" )
doc = addPageBreak( doc )
```

---

addPageNumber *Insert a page number into a document object*

Description

Insert a page number into a document object

Usage

```
addPageNumber(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

addPageNumber only works with pptx documents.

See addPageNumber.pptx for examples.

Value

a document object

See Also

pptx, addPageNumber.pptx

addPageNumber.pptx *Insert a page number shape into a document pptx object*

Description

Insert a page number shape into the current slide of a pptx object.

Usage

```
## S3 method for class 'pptx'
addPageNumber(doc, value, ...)
```

Arguments

doc	pptx object
value	character value to add into the page number shape of the current slide. optionnal. If missing current slide number will be used.
...	further arguments, not used.

Value

a pptx document object

See Also

addPageNumber, addDate.pptx

Examples

```
# Create a new document
doc = pptx( title = "title" )

# add a slide with layout "Title Slide"
doc = addSlide( doc, slide.layout = "Title Slide" )
doc = addTitle( doc, "Presentation title" ) #set the main title
#set the sub-title
doc = addSubtitle( doc , "This document is generated with ReporteRs.")

## add a page number on the current slide
doc = addPageNumber( doc )

doc = addSlide( doc, slide.layout = "Title and Content" )
## add a page number on the current slide but not the default text (slide number)
doc = addPageNumber( doc, value = "Page number text")

# Write the object in file "presentation.pptx"
writeDoc( doc, "addPageNumber_example.pptx" )
```

addParagraph *Add a paragraph into a document object*

Description

Add a paragraph into a document object

Usage

```
addParagraph(doc, value, ...)
```

Arguments

doc	document object
value	text to add to the document as paragraphs: an object of class pot or set_of_paragraphs or a character vector.
...	further arguments passed to other methods

Details

a paragraph is a set of text that ends with an end of line('\n' in C). Read pot to see how to get different font formats. Trying to insert a '\n' will have no effect. If an end of line is required , a new paragraph is required.

Value

a document object

See Also

docx, addParagraph.docx , pptx, addParagraph.pptx , bsdoc, addParagraph.bsdoc , pot, textProperties

addParagraph.bsdoc *Insert a paragraph into an bsdoc object*

Description

Insert paragraph(s) of text into a bsdoc object

Usage

```
## S3 method for class 'bsdoc'
addParagraph(doc, value, par.properties = parProperties(),
  restart.numbering = FALSE, ...)
```

Arguments

doc	bsdoc object where paragraph has to be added
value	text to add to the document as paragraphs: an object of class pot or set_of_paragraphs or a character vector.
par.properties	parProperties to apply to paragraphs.
restart.numbering	boolean value. If TRUE, next numbered list counter will be set to 1.
...	further arguments, not used.

Value

an object of class bsdoc.

See Also

bsdoc, addMarkdown.bsdoc, pot

Examples

```
doc.filename = "addParagraph_bsdoc/example.html"

# set default font size to 11
options( "ReporteRs-fontsize" = 11 )

doc = bsdoc( )

doc = addTitle( doc, "Title example 1", level = 1 )

# Add "Hello World" into the document doc
doc = addParagraph(doc, "Hello Word" )


doc = addTitle( doc, "Title example 2", level = 1 )

# "My tailor is rich" with formatting on some words
pot1 = pot("My tailor", textProperties(color = "red" )
  ) + " is " + pot("rich", textProperties(shading.color = "red", font.weight = "bold" ) )

# "Cats and dogs" with formatting on some words
pot2 = pot("Cats", textProperties(color = "red" ) ) +
  " and " +
  pot("dogs", textProperties( color = "blue" ),
    hyperlink = "http://www.wikipedia.org/" )

# create a set of paragraphs made of pot1 and pot2
my.pars = set_of_paragraphs( pot1, pot2 )

# Add my.pars into the document doc
doc = addParagraph(doc, my.pars )
```

```
doc = addTitle( doc, "Title example 3", level = 1 )
# define some text
text1 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
text2 = "In sit amet ipsum tellus. Vivamus dignissim arcu sit amet faucibus auctor."
text3 = "Quisque dictum tristique ligula."

# define parProperties with list properties
ordered.list.level1 = parProperties(list.style = "ordered", level = 1 )
ordered.list.level2 = parProperties(list.style = "ordered", level = 2 )

# define parProperties with list properties
unordered.list.level1 = parProperties(list.style = "unordered", level = 1 )
unordered.list.level2 = parProperties(list.style = "unordered", level = 2 )

# add ordered list items
doc = addParagraph( doc, value = text1,
par.properties = ordered.list.level1 )
doc = addParagraph( doc, value = text2,
par.properties = ordered.list.level2 )

# add ordered list items without restart renumbering
doc = addParagraph( doc, value = c( text1, text2, text3),
par.properties = ordered.list.level1 )

# add ordered list items and restart renumbering
doc = addParagraph( doc, value = c( text1, text2, text3), restart.numbering = TRUE,
par.properties = ordered.list.level1 )

# add unordered list items
doc = addParagraph( doc, value = text1,
par.properties = unordered.list.level1 )
doc = addParagraph( doc, value = text2,
par.properties = unordered.list.level2 )


# Write the object
writeDoc( doc, file = doc.filename )
```

---

addParagraph.docx *Insert a paragraph into a docx object*

Description

Insert paragraph(s) of text into a docx object

Usage

```
## S3 method for class 'docx'
addParagraph(doc, value, stylename, bookmark,
  par.properties = parProperties(), restart.numbering = FALSE, ...)
```

Arguments

doc	Object of class docx where paragraph has to be added
value	text to add to the document as paragraphs: an object of class pot or set_of_paragraphs or a character vector.
stylename	value of the named style to apply to paragraphs in the docx document. Expected value is an existing stylename of the template document used to create the docx object. see styles.docx.
bookmark	a character value ; id of the Word bookmark to replace by the table. optional. See bookmark.
par.properties	parProperties to apply to paragraphs, only used if stylename if missing.
restart.numbering	boolean value. If TRUE, next numbered list counter will be set to 1.
...	further arguments, not used.

Value

an object of class docx.

See Also

docx, addParagraph, bookmark, addMarkdown.docx, pot

Examples

```
doc.filename = "addParagraph_example.docx"

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

doc = docx( )

# returns available stylenames
styles( doc )

doc = addTitle( doc, "Title example 1", level = 1 )

# Add "Hello World" into the document doc
doc = addParagraph(doc, "Hello Word", stylename = "Normal" )


doc = addTitle( doc, "Title example 2", level = 1 )
```

```
# define some text
sometext = c( "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
, "In sit amet ipsum tellus. Vivamus dignissim arcu sit amet faucibus auctor."
, "Quisque dictum tristique ligula."
)

# add sometext with stylename BulletList
doc = addParagraph( doc, value = sometext, stylename="BulletList" )


doc = addTitle( doc, "Title example 3", level = 1 )

# "My tailor is rich" with formatting on some words
pot1 = pot("My tailor", textProperties(color = "red" )
  ) + " is " + pot("rich", textProperties(shading.color = "red", font.weight = "bold" ) )

# "Cats and dogs" with formatting on some words
pot2 = pot("Cats", textProperties(color = "red" ) ) +
  " and " +
  pot("dogs", textProperties( color = "blue" ),
    hyperlink = "http://www.wikipedia.org/" )

# create a set of paragraphs made of pot1 and pot2
my.pars = set_of_paragraphs( pot1, pot2 )

# Add my.pars into the document doc
doc = addParagraph(doc, my.pars, stylename = "Normal" )


doc = addTitle( doc, "Title example 4", level = 1 )
# define some text
text1 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
text2 = "In sit amet ipsum tellus. Vivamus dignissim arcu sit amet faucibus auctor."
text3 = "Quisque dictum tristique ligula."

# define parProperties with list properties
ordered.list.level1 = parProperties(list.style = "ordered", level = 1 )
ordered.list.level2 = parProperties(list.style = "ordered", level = 2 )

# define parProperties with list properties
unordered.list.level1 = parProperties(list.style = "unordered", level = 1 )
unordered.list.level2 = parProperties(list.style = "unordered", level = 2 )

# add ordered list items
doc = addParagraph( doc, value = text1,
par.properties = ordered.list.level1 )
doc = addParagraph( doc, value = text2,
par.properties = ordered.list.level2 )

# add ordered list items without restart renumbering
doc = addParagraph( doc, value = c( text1, text2, text3),
par.properties = ordered.list.level1 )
```

```
# add ordered list items and restart renumbering
doc = addParagraph( doc, value = c( text1, text2, text3), restart.numbering = TRUE,
par.properties = ordered.list.level1 )

# add unordered list items
doc = addParagraph( doc, value = text1,
par.properties = unordered.list.level1 )
doc = addParagraph( doc, value = text2,
par.properties = unordered.list.level2 )


# Write the object
writeDoc( doc, file = doc.filename )
```

---

addParagraph.Footnote *Insert a paragraph into a Footnote object*

Description

Insert paragraph(s) of text into a Footnote. To create a Footnote made of several paragraphs with different parProperties, add sequentially paragraphs with their associated parProperties objects with this function.

Usage

```
## S3 method for class 'Footnote'
addParagraph(doc, value, par.properties = parProperties(),
  ...)
```

Arguments

doc	Footnote object where to add paragraphs.
value	text to add to the document as paragraphs: an object of class pot or set_of_paragraphs or a character vector.
par.properties	parProperties to apply to paragraphs.
...	further arguments, not used.

Value

an object of class Footnote.

See Also

Footnote, parProperties, pot , set_of_paragraphs

addParagraph.pptx *Insert a paragraph into a pptx object*

Description

Insert paragraph(s) of text into a pptx object

Usage

```
## S3 method for class 'pptx'
addParagraph(doc, value, offx, offy, width, height,
  par.properties = parProperties(), append = FALSE,
  restart.numbering = FALSE, ...)
```

Arguments

doc	pptx object where paragraph is added
value	text to add to the document as paragraphs: an object of class pot or set_of_paragraphs or a character vector.
offx	optional, x position of the shape (top left position of the bounding box) in inch. See details.
offy	optional, y position of the shape (top left position of the bounding box) in inch. See details.
width	optional, width of the shape in inch. See details.
height	optional, height of the shape in inch. See details.
par.properties	parProperties to apply to paragraphs. Shading and border settings will have no effect.
restart.numbering	boolean value. If TRUE, next numbered list counter will be set to 1.
append	boolean default to FALSE. If TRUE, paragraphs will be appened in the current shape instead of beeing sent into a new shape. Paragraphs can only be appended on shape containing paragraphs (i.e. you can not add paragraphs after a FlexTable).
...	further arguments, not used.

Details

If arguments offx, offy, width, height are missing, position and dimensions will be defined by the width and height of the next available shape of the slide. This dimensions can be defined in the layout of the PowerPoint template used to create the pptx object.

If arguments offx, offy, width, height are provided, they become position and dimensions of the new shape.

Value

an object of class pptx.

See Also

pptx, addParagraph addMarkdown.pptx, pot

Examples

```
doc.filename = "addParagraph_example.pptx"

# set default font size to 24
options( "ReporteRs-fontsize" = 24 )

doc = pptx( title = "title" )

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )

doc = addTitle( doc, "Title example 1" )

# Add "Hello World" into the document doc
doc = addParagraph(doc, "Hello Word" )

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )


doc = addTitle( doc, "Title example 2" )

# "My tailor is rich" with formatting on some words
pot1 = pot("My tailor", textProperties(color = "red" )
  ) + " is " + pot("rich", textProperties(shading.color = "red", font.weight = "bold" ) )

# "Cats and dogs" with formatting on some words
pot2 = pot("Cats", textProperties(color = "red" ) ) +
  " and " +
  pot("dogs", textProperties( color = "blue" ),
    hyperlink = "http://www.wikipedia.org/" )

# create a set of paragraphs made of pot1 and pot2
my.pars = set_of_paragraphs( pot1, pot2 )

# Add my.pars into the document doc
doc = addParagraph(doc, my.pars )

# Add my.pars into the document doc
doc = addParagraph(doc, my.pars, offx = 3, offy = 3, width = 2, height = 0.5
, par.properties=parProperties(text.align="center", padding=0) )

# add a slide with layout "Title and Content"
```

```
doc = addSlide( doc, slide.layout = "Title and Content" )


doc = addTitle( doc, "Title example 3" )

# "My tailor is rich" with formatting on some words
pot1 = pot("My tailor", textProperties(color = "red" )
  ) + " is " + pot("rich", textProperties(shading.color = "red", font.weight = "bold" ) )

# "Cats and dogs" with formatting on some words
pot2 = pot("Cats", textProperties(color = "red" ) ) +
  " and " +
  pot("dogs", textProperties( color = "blue" ),
    hyperlink = "http://www.wikipedia.org/" )

# create a set of paragraphs made of pot1 and pot2
my.pars = set_of_paragraphs( pot1, pot2 )

# Add my.pars into the document doc
doc = addParagraph(doc, my.pars
, par.properties=parProperties(text.align="center", padding=24) )

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )

doc = addTitle( doc, "Title example 1" )
# define some text
text1 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
text2 = "In sit amet ipsum tellus. Vivamus dignissim arcu sit amet faucibus auctor."
text3 = "Quisque dictum tristique ligula."

# define parProperties with list properties
ordered.list.level1 = parProperties(list.style = "ordered", level = 1 )
ordered.list.level2 = parProperties(list.style = "ordered", level = 2 )

# define parProperties with list properties
unordered.list.level1 = parProperties(list.style = "unordered", level = 1 )
unordered.list.level2 = parProperties(list.style = "unordered", level = 2 )

# add ordered list items
doc = addParagraph( doc, value = text1,
par.properties = ordered.list.level1 )
doc = addParagraph( doc, value = text2, append = TRUE,
par.properties = ordered.list.level2 )

doc = addParagraph(doc, "This paragraph has no list attribute", append = TRUE )

# add ordered list items without restart renumbering
doc = addParagraph( doc, value = c( text1, text2, text3), append = TRUE,
par.properties = ordered.list.level1 )

# add ordered list items and restart renumbering
doc = addParagraph( doc, value = c( text1, text2, text3),
```

```
append = TRUE, restart.numbering = TRUE,
par.properties = ordered.list.level1 )

# add unordered list items
doc = addParagraph( doc, value = text1,
append = TRUE,
par.properties = unordered.list.level1 )
doc = addParagraph( doc, value = text2,
append = TRUE,
par.properties = unordered.list.level2 )


# Write the object
writeDoc( doc, file = doc.filename )
```

---

addPlot *Add a plot into a document object*

Description

Add a plot into a document object

Usage

```
addPlot(doc, fun, pointsize = 12, vector.graphic = F, ...)
```

Arguments

doc	document object
fun	plot function
vector.graphic	logical scalar, if TRUE, vector graphics are produced instead of PNG images. SVG will be produced for bsdoc objects and DrawingML instructions for docx and pptx objects. DrawingML instructions offer advantage to provide editable graphics (forms and text colors , text contents, moving and resizing is disabled).
pointsize	the default pointsize of plotted text in pixels, default to 12.
...	further arguments passed to or from other methods..

Details

Plot parameters are specified with the ... argument. However, the most convenient usage is to wrap the plot code into a function whose parameters will be specified as '...'.

If you want to add ggplot2 or lattice plot, use print function.

vector.graphic: if document is a pptx or bsdoc document, vector graphics will always be displayed. Don't use vector graphics if document is a docx and MS Word version used to open the document is 2007.

See addPlot.docx or addPlot.pptx or addPlot.bsdoc for examples.

Value

a document object

See Also

docx, addPlot.docx , pptx, addPlot.pptx , bsdoc, addPlot.bsdoc

---

addPlot.bsdoc *Add a plot into an bsdoc object*

Description

Add a plot into the bsdoc object.

Usage

```
## S3 method for class 'bsdoc'
addPlot(doc, fun, pointsize = getOption("ReporteRs-fontsize"),
  vector.graphic = T, width = 6, height = 6,
  fontname = getOption("ReporteRs-default-font"),
  par.properties = parCenter(padding = 5), ...)
```

Arguments

doc	Object of class bsdoc where paragraph has to be added
fun	plot function. The function will be executed to produce graphics. For grid or lattice or ggplot object, the function should just be print and an extra argument x should specify the object to plot. For traditionnal plots, the function should contain plot instructions. See examples.
width	plot width in inches (default value is 6).
height	plot height in inches (default value is 6).
vector.graphic	logical scalar, default to FALSE. If TRUE, vector graphics are produced instead of PNG images. If TRUE, vector graphics are RaphaelJS instructions(transformed as SVG).
pointsize	the default pointsize of plotted text in pixels, default to 12.
fontname	the default font family to use, default to getOption("ReporteRs-default-font").
par.properties	paragraph formatting properties of the paragraph that contains plot(s). An object of class parProperties
...	arguments for fun.

Value

an object of class bsdoc.

See Also

bsdoc, addPlot , add.plot.interactivity

Examples

```
doc.filename = "addPlot_bsdoc/example.html"

# set default font size to 11
options( "ReporteRs-fontsize" = 11 )

doc = bsdoc( )

doc = addTitle( doc, "Title example 1", level = 1 )
# Add a base plot
# set vector.graphic to FALSE if Word version
#   used to read the file is <= 2007
doc = addPlot( doc, fun = plot
  , x = rnorm( 100 ), y = rnorm (100 )
  , main = "base plot main title"
  , vector.graphic = TRUE
  , width = 5, height = 7
  , par.properties = parProperties(text.align = "left")
)


doc = addTitle( doc, "Title example 2", level = 1 )
# load ggplot2
require( ggplot2 )

# create a ggplot2 plot
myplot = qplot(Sepal.Length, Petal.Length, data = iris
  , color = Species, size = Petal.Width, alpha = I(0.7) )

# Add myplot into object doc
#   myplot is assigned to argument 'x' because function 'print' on ggplot
#   objects is expecting argument 'x'.
doc = addPlot( doc = doc, fun = print, x = myplot )


doc = addTitle( doc, "Title example 3", level = 1 )
############################################################
# Create lm.D9, a lm object
ctl = c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt = c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group = gl(2, 10, 20, labels = c("Ctl","Trt"))
weight = c(ctl, trt)
lm.D9 = lm(weight ~ group)

# add the 6 plots into the document
doc = addPlot( doc, plot, x = lm.D9, width = 6, height = 7 )
```

```
# Write the object
writeDoc( doc, file = doc.filename )
```

---

addPlot.docx *Add a plot into a docx object*

Description

Add a plot into the docx object.

Usage

```
## S3 method for class 'docx'
addPlot(doc, fun, pointsize = getOption("ReporteRs-fontsize"),
  vector.graphic = F, width = 6, height = 6,
  fontname = getOption("ReporteRs-default-font"), editable = TRUE, bookmark,
  par.properties = parProperties(text.align = "center", padding = 5), ...)
```

Arguments

doc	the docx to use
fun	plot function. The function will be executed to produce graphics. For grid or lattice or ggplot object, the function should just be print and an extra argument x should specify the object to plot. For traditionnal plots, the function should contain plot instructions. See examples.
width	plot width in inches (default value is 6).
height	plot height in inches (default value is 6).
vector.graphic	logical scalar, default to FALSE. DrawingML instructions cannot be read by MS Word 2007.
bookmark	id of the Word bookmark to replace by the plot. optional. bookmark is a character vector specifying bookmark id to replace by the plot(s). If provided, plot(s) will replace the paragraph that contains the bookmark. See bookmark. If not provided, plot(s) will be added at the end of the document.
par.properties	paragraph formatting properties of the paragraph that contains plot(s). An object of class parProperties
pointsize	the default pointsize of plotted text in pixels, default to getOption("ReporteRs-fontsize").
fontname	the default font family to use, default to getOption("ReporteRs-default-font").
editable	logical value - if TRUE vector graphics elements (points, text, etc.) are editable.
...	arguments for fun.

Value

an object of class docx.

See Also

docx, addPlot, bookmark.

Examples

```
doc.filename = "addPlot_example.docx"

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

doc = docx( )

doc = addTitle( doc, "Title example 1", level = 1 )
# Add a base plot
# set vector.graphic to FALSE if Word version
#   used to read the file is <= 2007
doc = addPlot( doc, fun = plot
  , x = rnorm( 100 ), y = rnorm (100 )
  , main = "base plot main title"
  , vector.graphic = TRUE
  , width = 5, height = 7
  , par.properties = parProperties(text.align = "left")
)


doc = addTitle( doc, "Title example 2", level = 1 )
# load ggplot2
require( ggplot2 )

# create a ggplot2 plot
myplot = qplot(Sepal.Length, Petal.Length, data = iris
  , color = Species, size = Petal.Width, alpha = I(0.7) )

# Add myplot into object doc
#   myplot is assigned to argument 'x' because function 'print' on ggplot
#   objects is expecting argument 'x'.
doc = addPlot( doc = doc, fun = print, x = myplot )

# Write the object
writeDoc( doc, file = doc.filename )
```

addPlot.pptx *Add a plot into a pptx object*

Description

Add a plot to the current slide of an existing pptx object.

Usage

```
## S3 method for class 'pptx'
addPlot(doc, fun, pointsize = 11, vector.graphic = TRUE,
  fontname = getOption("ReporteRs-default-font"), editable = TRUE, offx,
  offy, width, height, ...)
```

Arguments

doc	pptx object
fun	plot function. The function will be executed to produce graphics. For grid or lattice or ggplot object, the function should just be print and an extra argument x should specify the object to plot. For traditionnal plots, the function should contain plot instructions. See examples.
pointsize	the default pointsize of plotted text, interpreted as big points (1/72 inch) at res ppi.
vector.graphic	logical scalar, default to TRUE. If TRUE, vector graphics are produced instead of PNG images. Vector graphics in pptx document are DrawingML instructions.
fontname	the default font family to use, default to getOption("ReporteRs-default-font").
editable	logical value - if TRUE vector graphics elements (points, text, etc.) are editable.
offx	optional, x position of the shape (top left position of the bounding box) in inch. See details.
offy	optional, y position of the shape (top left position of the bounding box) in inch. See details.
width	optional, width of the shape in inch. See details.
height	optional, height of the shape in inch. See details.
...	arguments for fun.

Details

If arguments offx, offy, width, height are missing, position and dimensions will be defined by the width and height of the next available shape of the slide. This dimensions can be defined in the layout of the PowerPoint template used to create the pptx object.

If arguments offx, offy, width, height are provided, they become position and dimensions of the new shape.

Value

an object of class pptx.

See Also

pptx, addPlot

Examples

```
doc.filename = "addPlot_example.pptx"

# set default font size to 24
options( "ReporteRs-fontsize" = 24 )

doc = pptx( title = "title" )

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )

doc = addTitle( doc, "Title example 1" )
# Add a base plot
doc = addPlot( doc, fun = plot
  , x = rnorm( 100 ), y = rnorm (100 )
  , main = "base plot main title"
)
# Add a base plot at a specified location
doc = addPlot( doc, fun = plot
  , x = rnorm( 100 ), y = rnorm (100 ), col = "red"
  , main = "small shape", pointsize=5
  , offx = 7, offy = 0, width = 3, height = 2
)

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )


doc = addTitle( doc, "Title example 2" )
# load ggplot2
require( ggplot2 )

# create a ggplot2 plot
myplot = qplot(Sepal.Length, Petal.Length, data = iris
  , color = Species, size = Petal.Width, alpha = I(0.7) )

# Add myplot into object doc
#   myplot is assigned to argument 'x' because function 'print' on ggplot
#   objects is expecting argument 'x'.
doc = addPlot( doc = doc, fun = print, x = myplot )

# Write the object
writeDoc( doc, file = doc.filename )
```

addPostCommand *add post plot commands*

Description

internal use only

Usage

```
addPostCommand(labels, ids, env)
```

Arguments

labels	labels
ids	ids
env	environment

addRScript *Add R script into a document object*

Description

Add R script into a document object

Usage

```
addRScript(doc, rscript, file, text, ...)
```

Arguments

doc	document object
file	R script file. Not used if text or rscript is provided.
text	character vector. The text to parse. Not used if file or rscript is provided.
rscript	an object of class RScript. Not used if file or text is provided.
...	further arguments passed to other methods

Details

You have to one of the following argument: file or text or rscript.

Value

a document object

See Also

addRScript.bsdoc, addRScript.docx , addRScript.pptx

---

addRScript.bsdoc *Add R script into a bsdoc object*

Description

Add R script into a bsdoc object.

Usage

```
## S3 method for class 'bsdoc'
addRScript(doc, rscript, file, text, ...)
```

Arguments

doc	bsdoc object where expressions have to be added
file	R script file. Not used if text or rscript is provided.
text	character vector. The text to parse. Not used if file or rscript is provided.
rscript	an object of class RScript. Not used if file or text is provided.
...	further arguments, not used.

Details

You have to one of the following argument: file or text or rscript.

Value

an object of class bsdoc.

See Also

bsdoc, addRScript

Examples

```
doc.filename = "addRScript_bsdoc/example.html"

# set default font size to 11
options( "ReporteRs-fontsize" = 11 )

doc = bsdoc( )
doc = addRScript(doc, text = "x = rnorm(100)
plot(density( x ) )" )
```

```
# Write the object
writeDoc( doc, file = doc.filename )
```

---

addRScript.docx *Add R script into a docx object*

Description

Add R script into a docx object.

Usage

```
## S3 method for class 'docx'
addRScript(doc, rscript, file, text, bookmark,
  par.properties = parProperties(), ...)
```

Arguments

doc	Object of class docx where expressions have to be added
file	R script file. Not used if text or rscript is provided.
text	character vector. The text to parse. Not used if file or rscript is provided.
rscript	an object of class RScript. Not used if file or text is provided.
par.properties	paragraph formatting properties of the paragraphs that contain rscript. An object of class parProperties
bookmark	a character value ; id of the Word bookmark to replace by the script. optional. See bookmark.
...	further arguments, not used.

Details

You have to one of the following argument: file or text or rscript.

Value

an object of class docx.

See Also

docx, addRScript, bookmark

Examples

```
doc.filename = "addRScript_example.docx"
# Create a new document
doc = docx( title = "title" )

an_rscript = RScript( text = "ls()
x = rnorm(10)" )
doc = addRScript(doc, an_rscript )

doc = addPageBreak( doc )

doc = addRScript(doc, text = "ls()" )

# Write the object
writeDoc( doc, file = doc.filename )
```

`addRScript.pptx` *Add R script into a pptx object*

Description

Add R script into a `pptx` object.

Usage

```
## S3 method for class 'pptx'
addRScript(doc, rscript, file, text, append = FALSE, ...)
```

Arguments

`doc`	`pptx` object where expressions have to be added
`file`	R script file. Not used if text or rscript is provided.
`text`	character vector. The text to parse. Not used if file or rscript is provided.
`rscript`	an object of class `RScript`. Not used if file or text is provided.
`append`	boolean default to FALSE. If TRUE, paragraphs will be appened in the current shape instead of beeing sent into a new shape. Paragraphs can only be appended on shape containing paragraphs (i.e. you can not add paragraphs after a FlexTable).
`...`	further arguments, not used.

Details

You have to one of the following argument: file or text or rscript.

Value

an object of class pptx.

See Also

pptx, addRScript

Examples

```
doc.filename = "addRScript_example.pptx"
# Create a new document
doc = pptx( title = "title" )
doc = addSlide( doc, slide.layout = "Title and Content" )
an_rscript = RScript( text = "ls()
x = rnorm(10)", par.properties = parProperties() )
doc = addRScript(doc, an_rscript )

# Write the object
writeDoc( doc, file = doc.filename )
```

---

addSection *Add a section into a document object*

Description

Add a section into a document object

Usage

```
addSection(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

addSection only works with docx documents. See addSection.docx for examples.

Value

a document object

See Also

docx, addSection.docx

addSection.docx *Insert a slide into a pptx object*

Description

Add a slide into a pptx object.

Usage

```
## S3 method for class 'docx'
addSection(doc, landscape = FALSE, ncol = 1,
  space_between = 0.3, columns.only = FALSE, ...)
```

Arguments

doc	Object of class docx where section has to be added
landscape	logical value. Specify TRUE to get a section with horizontal page.
ncol	integer number to specify how many columns the section should contains.
space_between	width in inches of the space between columns of the section.
columns.only	logical value, if set to TRUE, no break page will (continuous section).
...	further arguments, not used.

Value

an object of class docx.

See Also

docx, addSection

Examples

```
doc.filename = "addSection.docx"

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

doc = docx( )
doc = addSection(doc, landscape = TRUE, ncol = 2 )
doc = addPlot( doc = doc, fun = function() {
barplot( 1:8, col = 1:8 )
}, width = 3, height = 3, pointsize = 5)

doc = addColumnBreak(doc )
doc = addFlexTable(doc, FlexTable(head(iris) ) )
```

```
doc = addSection(doc, ncol = 2 )
doc = addParagraph( doc = doc, "Text 1.", "Normal" )
doc = addColumnBreak(doc )
doc = addParagraph( doc = doc, "Text 2.", "Normal" )


doc = addSection(doc, ncol = 2, columns.only = TRUE )
doc = addFlexTable(doc, FlexTable(head(iris) ) )
doc = addColumnBreak(doc )
doc = addParagraph( doc = doc, "Text 3.", "Normal" )


doc = addSection(doc, ncol = 1, columns.only = TRUE )
doc = addFlexTable(doc, FlexTable(mtcars, add.rownames = TRUE) )
doc = addParagraph( doc = doc, "Text 4.", "Normal" )

# Write the object
writeDoc( doc, file = doc.filename )
```

---

addSlide *Add a slide into a document object*

Description

Add a slide into a document object

Usage

```
addSlide(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

addSlide only works with pptx documents. See addSlide.pptx for examples.

Value

a document object

See Also

pptx, addSlide.pptx

addSlide.pptx *Insert a slide into a pptx object*

Description

Add a slide into a pptx object.

Usage

```
## S3 method for class 'pptx'
addSlide(doc, slide.layout, bookmark, ...)
```

Arguments

doc	pptx object where slide has to be added
slide.layout	layout name of the slide to create. See slide.layouts.pptx
bookmark	"integer" page number to specify where slide has to be replaced with a new empty one.
...	further arguments, not used.

Details

This function is a key function ; if no slide has been added into the document object no content (tables, plots, images, text) can be added.

If creating a slide of type "Title and Content", only one content can be added because there is only one content shape in the layout. If creating a slide of type "Two Content", two content can be added because there are 2 content shapes in the layout.

Content shapes are boxes with dotted borders that hold content in its place on a slide layout. If you need a new layout, create it in PowerPoint :

On the View tab, in the Presentation Views group, click Slide Master.

read http://office.microsoft.com/en-us/powerpoint-help/create-a-new-custom-layout-HA010079650.aspx

read http://office.microsoft.com/en-us/powerpoint-help/change-a-placeholder-HA010064940.aspx

Function slide.layouts returns available layout names of the template used when pptx object has been created. It is important to know that when using addParagraph.pptx, paragraph and defaut font formats will be defined by the properties of the shape of the slide.layout where content will be added. For example, if you set the shape formatting properties to a 'no bullet', paragraphs of text won't have any bullet.

Also when using addPlot, plot dimensions will be the shape dimensions. It means that if you want to change plot dimensions , this has to be done in the PowerPoint template used when creating the pptx object.

Value

an object of class pptx.

Note

The layout names must only contain letters (upper or lower case) from 'a' to 'z', numbers (from 0 to 9) and spaces.

See Also

addTitle.pptx, slide.layouts , pptx, addSlide

Examples

```
# Create a new document
doc = pptx( title = "title" )

# add a slide with layout "Title Slide"
doc = addSlide( doc, slide.layout = "Title Slide" )
doc = addTitle( doc, "Presentation title" ) #set the main title
doc = addSubtitle( doc , "This document is generated with ReporteRs.")#set the sub-title


# add a slide with layout "Title and Content" then add content
doc = addSlide( doc, slide.layout = "Title and Content" )
doc = addTitle( doc, "Iris sample dataset", level = 1 )
doc = addFlexTable( doc, vanilla.table( iris[ 1:10,] ) )


# add a slide with layout "Two Content" then add content
doc = addSlide( doc, slide.layout = "Two Content" )
doc = addTitle( doc, "Two Content demo", level = 1 )
doc = addFlexTable( doc, vanilla.table( iris[ 46:55,] ) )
doc = addParagraph(doc, "Hello Word!" )

# to see available layouts :
slide.layouts( doc )

# Write the object in file "addSlide_example.pptx"
writeDoc( doc, "addSlide_example.pptx" )
```

---

addSubtitle *Add a subtitle shape into a document object*

Description

Add a subtitle shape into a document object

Usage

```
addSubtitle(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

addSubtitle only works with pptx documents. See addSubtitle.pptx for examples.

Value

a document object

See Also

pptx, addSubtitle.pptx

---

addSubtitle.pptx *Insert a addSubtitle shape into a pptx object*

Description

Add a addSubtitle shape into a pptx object.

Usage

```
## S3 method for class 'pptx'
addSubtitle(doc, value, ...)
```

Arguments

doc	pptx object
value	"character" value to use as subtitle text
...	further arguments, not used.

Details

Subtitle shape only exist in slide of type 'Title Slide'.

Value

an object of class pptx.

See Also

pptx, addSubtitle

Examples

```
# Create a new document
doc = pptx( title = "title" )

# add a slide with layout "Title Slide"
doc = addSlide( doc, slide.layout = "Title Slide" )
doc = addTitle( doc, "Presentation title" ) #set the main title
doc = addSubtitle( doc , "This document is generated with ReporteRs.")#set the sub-title

# Write the object in file "addSubtitle_example.pptx"
writeDoc( doc, "addSubtitle_example.pptx" )
```

---

addTitle *Add a title into a document object*

Description

Add a title into a document object

Usage

```
addTitle(doc, value, ...)
```

Arguments

doc	document object
value	"character" value to use as title text
...	further arguments passed to or from other methods..

Details

See addTitle.docx or addTitle.pptx or addTitle.bsdoc for examples.

Value

a document object

See Also

docx, addTitle.docx, pptx , addTitle.pptx, bsdoc, addTitle.bsdoc

addTitle.bsdoc *Insert a title into a bsdoc object*

Description

Add a title into a bsdoc object.

Usage

```
## S3 method for class 'bsdoc'
addTitle(doc, value, level = 1, ...)
```

Arguments

doc	bsdoc object
value	"character" value to use as title text
level	"integer" positive value to use as heading level. 1 for title1, 2 for title2, etc. Default to 1.
...	further arguments, not used.

Value

an object of class bsdoc.

See Also

bsdoc, addTitle

Examples

```
doc.filename = "addTitle_bsdoc/example.html"

# set default font size to 11
options( "ReporteRs-fontsize" = 11 )

doc = bsdoc( )

doc = addTitle( doc, "Title example 1", level = 1 )


doc = addTitle( doc, "Title example 2", level = 1 )

# Write the object
writeDoc( doc, file = doc.filename )
```

addTitle.docx *Insert a title into a docx object*

Description

Add a title into a docx object.

Usage

```
## S3 method for class 'docx'
addTitle(doc, value, level = 1, ...)
```

Arguments

doc	Object of class docx
value	"character" value to use as title text
level	"integer" positive value to use as heading level. 1 for title1, 2 for title2, etc. Default to 1.
...	further arguments, not used.

Details

In MS Word, you can use whatever style you want as title formatting style. But to be considered as entries for a Table of Content, used styles must be 'title' styles. Theses are always available in MS Word list styles. When template is read, ReporteRs try to guess what are theses styles. If it does not succeed, you will see that error when addTitle will be called:

Error in addHeader(...
You must defined title styles via declareTitlesStyles first.

You have to use function declareTitlesStyles.docx to indicate which available styles are meant to be used as titles styles. A side effect is that you will be able then to add a table of content in your Word document.

Value

an object of class docx.

See Also

docx, addParagraph.docx , declareTitlesStyles.docx, styles.docx

Examples

```
# Create a new document
doc = docx( title = "title" )

# add a title (level 1)
doc = addTitle( doc, "My first title", level = 1 )

# add another title (level 2)
doc = addTitle( doc, "My first sub-title", level = 2 )
doc = addParagraph(doc, "Hello Word!", stylename = "Normal")

# Write the object in file "addTitle_example.docx"
writeDoc( doc, "addTitle_example.docx" )
```

---

addTitle.pptx *Insert a title into a pptx object*

Description

Add a title into a pptx object.

Usage

```
## S3 method for class 'pptx'
addTitle(doc, value, ...)
```

Arguments

doc	pptx object
value	"character" value to use as title text
...	further arguments, not used.

Value

an object of class pptx.

See Also

pptx, addTitle, addSlide.pptx

Examples

```
# Create a new document
doc = pptx( title = "title" )

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )

# Here we fill the title shape with "My title"
doc = addTitle( doc, "My title" )

# Write the object in file "addTitle_example.pptx"
writeDoc( doc, "addTitle_example.pptx" )
```

---

addTOC *Add a table of contents into a document object*

Description

Add a table of contents into a document object

Usage

```
addTOC(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

addTOC only works with docx documents.

See addTOC.docx for examples.

Value

a document object

See Also

docx, addTOC.docx, styles.docx

addTOC.docx *Insert a table of contents into a docx object*

Description

Insert a table of contents into a docx object.

Usage

```
## S3 method for class 'docx'
addTOC(doc, stylename, ...)
```

Arguments

doc	Object of class docx where table of content has to be added
stylename	optional. Stylename in the document that will be used to build entries of the TOC.
...	further arguments, not used.

Details

If stylename is not used, a classical table of content will be produced.
If stylename is used, a custom table of contents will be produced, pointing to entries that have been formated with stylename. For example, this can be used to produce a toc with only plots.

Value

an object of class docx.

See Also

docx, addTitle.docx , styles.docx, addParagraph.docx

Examples

```
require( ggplot2 )

### example 1
# Create a new document
doc = docx( title = "title" )
#leave the first page blank and add a page break
doc = addPageBreak(doc)
# add a TOC (to be refresh when document is opened)
# and add a page break
doc = addTOC(doc)
doc = addPageBreak(doc)
```

```
# add titles that will be entries in the TOC
doc = addTitle( doc, "My first title", level = 1 )
doc = addTitle( doc, "My second title", level = 1 )

# Write the object in file "addTOC_example1.docx"
writeDoc( doc, "addTOC_example1.docx" )


### example 2
# Create a new document
doc = docx( title = "title" )
#leave the first page blank and add a page break
doc = addPageBreak(doc)#'

doc = addTitle( doc, "Plots", level = 1 )
doc = addPlot( doc
, fun = plot
, x = rnorm( 100 )
, y = rnorm (100 )
, main = "base plot main title"
)
doc = addParagraph( doc, value="graph example 1", stylename = "rPlotLegend" )

myplot = qplot(Sepal.Length, Petal.Length, data = iris, color = Species
, size = Petal.Width, alpha = I(0.7))
doc = addPlot( doc = doc
, fun = print
, x = myplot #this argument MUST be named, print is expecting argument 'x'
)
doc = addParagraph( doc, value="graph example 2", stylename = "rPlotLegend" )

# Because we used "rPlotLegend" as legend in plot
# , addTOC will use this stylename to define
# entries in the generated TOC
doc = addTOC(doc, stylename = "rPlotLegend")

# Write the object in file "addTOC_example2.docx"
writeDoc( doc, "addTOC_example2.docx" )
```

`as.html` *get HTML code*

Description

Get HTML code in a character vector.

Usage

```
as.html(object, ...)
```

Arguments

object	object to get HTML from
...	further arguments passed to other methods

Details

See FlexTable or raphael.html for examples.

Value

a character value

See Also

FlexTable, raphael.html

---

as.html.FlexTable *get HTML code from a FlexTable*

Description

get HTML code from a FlexTable

Usage

```
## S3 method for class 'FlexTable'
as.html(object, ...)
```

Arguments

object	the FlexTable object
...	further arguments passed to other methods

Value

a character value

See Also

FlexTable

Examples

```
#####################################################################

# Create a FlexTable with data.frame mtcars, display rownames
# use different formatting properties for header and body
MyFTable = FlexTable( data = mtcars, add.rownames = TRUE,
  header.cell.props = cellProperties( background.color = "#00557F" ),
  header.text.props = textProperties( color = "white",
    font.size = 11, font.weight = "bold" ),
  body.text.props = textProperties( font.size = 10 )
)
# zebra stripes - alternate colored backgrounds on table rows
MyFTable = setZebraStyle( MyFTable, odd = "#E1EEf4", even = "white" )

# applies a border grid on table
MyFTable = setFlexTableBorders(MyFTable,
  inner.vertical = borderProperties( color="#0070A8", style="solid" ),
  inner.horizontal = borderNone(),
  outer.vertical = borderProperties( color = "#006699", style = "solid", width = 2 ),
  outer.horizontal = borderProperties( color = "#006699", style = "solid", width = 2 )
)

# get HTML of the FlexTable
as.html( MyFTable )
```

`as.html.pot` *get HTML code from a pot*

Description

get HTML code from a pot

Usage

```
## S3 method for class 'pot'
as.html(object, ...)
```

Arguments

`object`	the `pot` object
`...`	further arguments passed to other methods

Value

a character value

See Also

pot

Examples

```
my_pot = pot("My tailor", textProperties(color="red") ) + " is " + pot("rich"
, textProperties(font.weight="bold") )
as.html( my_pot )
```

---

as.html.RScript *get HTML code from a RScript object*

Description

get HTML code from a RScript object

Usage

```
## S3 method for class 'RScript'
as.html(object, ...)
```

Arguments

object	the RScript object
...	further arguments passed to other methods - not used.

Value

a character value

See Also

RScript

Examples

```
my_rscript = RScript( text = "ls()" )
as.html( my_rscript )
```

BootstrapMenu *Create a bootstrap DropDownMenu*

Description

Create a DropDownMenu object. This object is to be used with BootstrapMenu to define menu links.

Usage

```
BootstrapMenu(title, link = "#", bg.active.color = "#34495E",
  bg.color = "#2C3E50", text.emphasis.color = "white",
  text.color = "#ecf0f1")
```

Arguments

title	"character" value: label of the title.
link	url to use as link associated with the title.
bg.active.color	active background color - a single character value specifying a valid color (e.g. "#000000" or "black").
bg.color	background color - a single character value specifying a valid color (e.g. "#000000" or "black").
text.emphasis.color	text emphasis color - a single character value specifying a valid color (e.g. "#000000" or "black").
text.color	text color - a single character value specifying a valid color (e.g. "#000000" or "black").

Value

an object of class BootstrapMenu.

See Also

bsdoc, addBootstrapMenu

Examples

```
mymenu = BootstrapMenu( title = "my title")

mydd = DropDownMenu( label = "Mon menu" )
mydd = addLinkItem( mydd, label = "GitHub", "http://github.com/")
mydd = addLinkItem( mydd, separator.after = TRUE)
mydd = addLinkItem( mydd, label = "Wikipedia", "http://www.wikipedia.fr")

mymenu = addLinkItem( mymenu, label = "ReporteRs", "http://github.com/davidgohel/ReporteRs")
mymenu = addLinkItem( mymenu, dd = mydd )
```

borderDashed *shortcut for dashed border*

Description

shortcut for a dashed border borderProperties()

Usage

```
borderDashed(...)
```

Arguments

...	arguments passed to borderProperties

See Also

textNormal , textBold , textItalic , textBoldItalic , parRight , parLeft , parCenter , parJustify , borderDotted , borderNone , borderSolid

Examples

```
borderDashed()
```

borderDotted *shortcut for dotted border*

Description

shortcut for a dotted border borderProperties()

Usage

```
borderDotted(...)
```

Arguments

...	arguments passed to borderProperties

See Also

textNormal , textBold , textItalic , textBoldItalic , parRight , parLeft , parCenter , parJustify , borderDashed , borderNone , borderSolid

Examples

```
borderDotted()
```

borderNone *shortcut for no border*

Description

shortcut for no border borderProperties()

Usage

```
borderNone(...)
```

Arguments

...	arguments passed to borderProperties

See Also

textNormal , textBold , textItalic , textBoldItalic , parRight , parLeft , parCenter , parJustify , borderDotted , borderDashed , borderSolid

Examples

```
borderNone()
```

borderProperties *border properties object*

Description

create a border properties object.

Usage

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

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

See Also

chprop.borderProperties, alterFlexTable, setFlexTableBorders

Examples

```
borderProperties()
borderProperties(color="orange", style="solid", width=1)
borderProperties(color="gray", style="dotted", width=1)
```

`borderSolid` *shortcut for solid border*

Description

shortcut for solid border borderProperties()

Usage

```
borderSolid(...)
```

Arguments

`...`	arguments passed to borderProperties

See Also

`textNormal` , `textBold` , `textItalic` , `textBoldItalic` , `parRight` , `parLeft` , `parCenter` , `parJustify` , `borderDotted` , `borderDashed` , `borderNone`

Examples

```
borderSolid()
```

`bsdoc` *Create an object representation of a bootstrap html document*

Description

Create a `bsdoc` object

Usage

```
bsdoc(title = "untitled",
  list.definition = getOption("ReporteRs-list-definition"), keywords = "",
  description = "")
```

Arguments

title	"character" value: title of the document.
list.definition	a list definition to specify how ordered and unordered lists have to be formated. See list.settings. Default to getOption("ReporteRs-list-definition").
keywords	"character" value: keywords metadata value to set in the html page
description	"character" value: description metadata value to set in the html page

Details

Several methods can used to send R output into an object of class bsdoc.

- addTitle.bsdoc add titles
- addParagraph.bsdoc add text
- addPlot.bsdoc add plots
- addFlexTable.bsdoc add tables. See FlexTable
- addImage.bsdoc add external images
- addMarkdown.bsdoc add markdown text
- addRScript.bsdoc add highlighted r script
- addBootstrapMenu add a bootstrap menu to the html page
- addFooter.bsdoc add text into the footer of the html page

Once object has content, user can write the docx into a ".html" file, see writeDoc.bsdoc.

Value

an object of class bsdoc.

See Also

docx, pptx

Examples

```
doc = bsdoc( title = "full example" )


doc = addTitle( doc, "Plot example", level = 1 )
# load ggplot2
require( ggplot2 )

# create a ggplot2 plot
myplot = qplot(Sepal.Length, Petal.Length, data = iris
  , color = Species, size = Petal.Width, alpha = I(0.7) )

# Add myplot into object doc
```

```
#   myplot is assigned to argument 'x' because function 'print' on ggplot
#   objects is expecting argument 'x'.
doc = addPlot( doc = doc, fun = print, x = myplot )



doc = addTitle( doc, "Text example", level = 1 )

# "My tailor is rich" with formatting on some words
pot1 = pot("My tailor", textProperties(color = "red" ) ) +
  " is " +
  pot("rich", textProperties(shading.color = "red", font.weight = "bold" ) )

# "Cats and dogs" with formatting on some words
pot2 = pot("Cats", textProperties(color = "red" ) ) +
  " and " +
  pot("dogs", textProperties( color = "blue" ),
    hyperlink = "http://www.wikipedia.org/" )

# create a set of paragraphs made of pot1 and pot2
my.pars = set_of_paragraphs( pot1, pot2 )

# Add my.pars into the document doc
doc = addParagraph(doc, my.pars )


doc = addTitle( doc, "List example", level = 1 )
# define some text
text1 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
text2 = "In sit amet ipsum tellus. Vivamus dignissim arcu sit amet faucibus auctor."
text3 = "Quisque dictum tristique ligula."

# define parProperties with list properties
ordered.list.level1 = parProperties(list.style = "ordered", level = 1 )
ordered.list.level2 = parProperties(list.style = "ordered", level = 2 )

# define parProperties with list properties
unordered.list.level1 = parProperties(list.style = "unordered", level = 1 )
unordered.list.level2 = parProperties(list.style = "unordered", level = 2 )

# add ordered list items
doc = addParagraph( doc, value = text1,
  par.properties = ordered.list.level1 )
doc = addParagraph( doc, value = text2,
  par.properties = ordered.list.level2 )

# add ordered list items without restart renumbering
doc = addParagraph( doc, value = c( text1, text2, text3),
  par.properties = ordered.list.level1 )

# add ordered list items and restart renumbering
doc = addParagraph( doc, value = c( text1, text2, text3), restart.numbering = TRUE,
```

```
  par.properties = ordered.list.level1 )

# add unordered list items
doc = addParagraph( doc, value = text1,
  par.properties = unordered.list.level1 )
doc = addParagraph( doc, value = text2,
  par.properties = unordered.list.level2 )

doc = addTitle( doc, "Table example", level = 1 )
#######################################################################

# Create a FlexTable with data.frame mtcars, display rownames
# use different formatting properties for header and body
MyFTable = FlexTable( data = mtcars, add.rownames = TRUE,
  header.cell.props = cellProperties( background.color = "#00557F" ),
  header.text.props = textProperties( color = "white",
    font.size = 11, font.weight = "bold" ),
  body.text.props = textProperties( font.size = 10 )
)
# zebra stripes - alternate colored backgrounds on table rows
MyFTable = setZebraStyle( MyFTable, odd = "#E1EEf4", even = "white" )

# applies a border grid on table
MyFTable = setFlexTableBorders(MyFTable,
  inner.vertical = borderProperties( color="#0070A8", style="solid" ),
  inner.horizontal = borderNone(),
  outer.vertical = borderProperties( color = "#006699", style = "solid", width = 2 ),
  outer.horizontal = borderProperties( color = "#006699", style = "solid", width = 2 )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )


#######################################################################
# add a menu
mymenu = BootstrapMenu( title = "my title")

mydd = DropDownMenu( label = "my menu" )
mydd = addLinkItem( mydd, label = "GitHub", "http://github.com/")
mydd = addLinkItem( mydd, separator.after = TRUE)
mydd = addLinkItem( mydd, label = "Wikipedia", "http://www.wikipedia.fr")

mymenu = addLinkItem( mymenu, label = "ReporteRs", "http://github.com/davidgohel/ReporteRs")
mymenu = addLinkItem( mymenu, dd = mydd )

doc = addBootstrapMenu( doc, mymenu )

#######################################################################
# add a footer
doc = addFooter( doc, pot( "Hello world",
  format = textProperties(color="gray") ), parCenter( padding = 0 ) )
```

```
#####################################################################
# write the doc
pages = writeDoc( doc, file = "bsdoc_example/example.html")
```

`cellProperties` *Cell formatting properties*

Description

Create a `cellProperties` object that describes cell formatting properties. This objects are used by `FlexTable`.

Usage

```
cellProperties(padding, border.width, border.style, border.color, border.bottom,
  border.left, border.top, border.right, border.bottom.color = "black",
  border.bottom.style = "solid", border.bottom.width = 1,
  border.left.color = "black", border.left.style = "solid",
  border.left.width = 1, border.top.color = "black",
  border.top.style = "solid", border.top.width = 1,
  border.right.color = "black", border.right.style = "solid",
  border.right.width = 1, vertical.align = "middle", padding.bottom = 1,
  padding.top = 1, padding.left = 1, padding.right = 1,
  background.color = "white")
```

Arguments

`padding`	cell padding - 0 or positive integer value. Argument `padding` overwrites arguments `padding.bottom`, `padding.top`, `padding.left`, `padding.right`.
`border.width`	border width - 0 or positive integer value. Argument `border.width` overwrites arguments `border.bottom.width`, `border.top.width` , `border.left.width`, `border.right.width`.
`border.style`	border style - a single character value, expected value is one of "none" , "solid", "dotted", "dashed". Argument `border.style` overwrites arguments `border.bottom.style`, `border.top.style`, `border.left.style` , `border.right.style`.
`border.color`	border color - a single character value specifying a valid color (e.g. "#000000" or "black"). Argument `border.color` overwrites arguments `border.bottom.color` , `border.top.color`, `border.left.color`, `border.right.color`.
`border.bottom`	`borderProperties` for bottom border. overwrite all border.bottom.* if specified.
`border.left`	`borderProperties` for left border. overwrite all border.left.* if specified.
`border.top`	`borderProperties` for top border. overwrite all border.top.* if specified.
`border.right`	`borderProperties` for right border. overwrite all border.right.* if specified.

`border.bottom.color`
: border bottom color - a single character value specifying a valid color (e.g. "#000000" or "black").

`border.bottom.style`
: border bottom style - a single character value, expected value is one of "none", "solid", "dotted", "dashed".

`border.bottom.width`
: border bottom width - 0 or positive integer value

`border.left.color`
: border left color - a single character value specifying a valid color (e.g. "#000000" or "black").

`border.left.style`
: border left style - a single character value, expected value is one of "none", "solid", "dotted", "dashed".

`border.left.width`
: border left width - 0 or positive integer value

`border.top.color`
: border top color - a single character value specifying a valid color (e.g. "#000000" or "black").

`border.top.style`
: border top style - a single character value, expected value is one of "none", "solid", "dotted", "dashed".

`border.top.width`
: border top width - 0 or positive integer value

`border.right.color`
: border right color - a single character value specifying a valid color (e.g. "#000000" or "black").

`border.right.style`
: border right style - a single character value, expected value is one of "none", "solid", "dotted", "dashed".

`border.right.width`
: border right width - 0 or positive integer value

`vertical.align`
: cell content vertical alignment - a single character value , expected value is one of "center" or "top" or "bottom"

`padding.bottom`
: cell bottom padding - 0 or positive integer value.

`padding.top`
: cell top padding - 0 or positive integer value.

`padding.left`
: cell left padding - 0 or positive integer value.

`padding.right`
: cell right padding - 0 or positive integer value.

`background.color`
: cell background color - a single character value specifying a valid color (e.g. "#000000" or "black").

Details

Default values are:

- `border.bottom.color` "black"
- `border.bottom.style` "solid"
- `border.bottom.width` 1
- `border.left.color` "black"
- `border.left.style` "solid"
- `border.left.width` 1
- `border.top.color` "black"
- `border.top.style` "solid"
- `border.top.width` 1
- `border.right.color` "black"
- `border.right.style` "solid"
- `border.right.width` 1
- `vertical.align` "middle"
- `padding.bottom` 1
- `padding.top` 1
- `padding.left` 1
- `padding.right` 1
- `background.color` "white"

See Also

`borderProperties`, `chprop.cellProperties` , `FlexTable`

Examples

```
cellProp01 = cellProperties( border.color = "gray", border.width = 2 )
cellProp02 = cellProperties(border.left.width = 0, border.right.width = 0
  , border.bottom.width = 2, border.top.width = 0
  , padding.bottom = 2, padding.top = 2
  , padding.left = 2, padding.right = 2 )
```

chprop *Change a formatting properties object*

Description

Change a formatting properties object

Usage

```
chprop(object, ...)
```

Arguments

object	formatting properties object
...	further arguments passed to other methods

Details

See chprop.textProperties or chprop.parProperties or chprop.cellProperties for examples.

Value

a formatting properties object

See Also

cellProperties, textProperties , parProperties

chprop.borderProperties *Modify border formatting properties*

Description

Modify an object of class borderProperties.

Usage

```
## S3 method for class 'borderProperties'
chprop(object, color, style, width, ...)
```

Arguments

object	borderProperties object to modify
...	further arguments - not used
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

Value

a borderProperties object

See Also

borderProperties

Examples

```
x = borderProperties()
chprop(x, color="orange", style="dashed", width=1)
chprop(x, width=5)
```

---

chprop.cellProperties *Modify a cell formatting properties object*

Description

Modify an object of class cellProperties.

Usage

```
## S3 method for class 'cellProperties'
chprop(object, border.bottom, border.left, border.top,
  border.right, padding, border.bottom.color, border.bottom.style,
  border.bottom.width, border.left.color, border.left.style, border.left.width,
  border.top.color, border.top.style, border.top.width, border.right.color,
  border.right.style, border.right.width, vertical.align, padding.bottom,
  padding.top, padding.left, padding.right, background.color, ...)
```

Arguments

object	cellProperties object to modify
border.bottom	borderProperties for bottom border. Overwrite all border.bottom.* argument values.
border.left	borderProperties for left border. Overwrite all border.left.* argument values.
border.top	borderProperties for top border. Overwrite all border.top.* argument values.

`border.right` borderProperties for right border. Overwrite all border.right.* argument values.

`padding` cell padding - 0 or positive integer value. Argument `padding` overwrites arguments `padding.bottom`, `padding.top`, `padding.left`, `padding.right`.

`border.bottom.color`
border bottom color - a single character value specifying a

`border.bottom.style`
border bottom style - a single character value, expected value is one of "none", "solid", "dotted", "dashed".

`border.bottom.width`
border bottom width - 0 or positive integer value

`border.left.color`
border left color - a single character value specifying a valid color (e.g. "#000000" or "black").

`border.left.style`
border left style - a single character value, expected value is one of "none", "solid", "dotted", "dashed".

`border.left.width`
border left width - 0 or positive integer value

`border.top.color`
border top color - a single character value specifying a valid color (e.g. "#000000" or "black").

`border.top.style`
border top style - a single character value, expected value is one of "none", "solid", "dotted", "dashed".

`border.top.width`
border top width - 0 or positive integer value

`border.right.color`
border right color - a single character value specifying a valid color (e.g. "#000000" or "black").

`border.right.style`
border right style - a single character value, expected value is one of "none", "solid", "dotted", "dashed".

`border.right.width`
border right width - 0 or positive integer value

`vertical.align` cell content vertical alignment - a single character value , expected value is one of "center" or "top" or "bottom"

`padding.bottom` cell bottom padding - 0 or positive integer value.

`padding.top` cell top padding - 0 or positive integer value.

`padding.left` cell left padding - 0 or positive integer value.

`padding.right` cell right padding - 0 or positive integer value.

`background.color`
cell background color - a single character value specifying a valid color (e.g. "#000000" or "black").

`...` further arguments - not used

Value

a cellProperties object

See Also

borderProperties, cellProperties , FlexTable

Examples

```
cellProp = cellProperties()

cellProp01 = chprop( cellProp, border.bottom.color = "#8A949B" )
cellProp02 = chprop( cellProp, border.right.color = "#8A949B" )
cellProp03 = chprop( cellProp, border.left.color = "#8A949B" )
cellProp04 = chprop( cellProp, border.top.color = "#8A949B" )
cellProp05 = chprop( cellProp, border.color = "#8A949B" )

cellProp06 = chprop( cellProp, border.bottom.width = 2 )
cellProp07 = chprop( cellProp, border.left.width = 2 )
cellProp08 = chprop( cellProp, border.top.width = 2 )
cellProp09 = chprop( cellProp, border.right.width = 2 )
cellProp10 = chprop( cellProp, border.width = 2 )

cellProp11 = chprop( cellProp, padding.bottom = 5 )
cellProp12 = chprop( cellProp, padding.top = 5 )
cellProp13 = chprop( cellProp, padding.left = 5 )
cellProp14 = chprop( cellProp, padding.right = 5 )
cellProp15 = chprop( cellProp, padding = 5 )

cellProp16 = chprop( cellProp, border.bottom = borderProperties( style = "dotted" ) )
cellProp17 = chprop( cellProp, border.left.style = "dotted" )
cellProp18 = chprop( cellProp, border.top.style = "dotted" )
cellProp19 = chprop( cellProp, border.right.style = "dotted" )
cellProp20 = chprop( cellProp, border.style = "dotted" )

cellProp21 = chprop( cellProp, vertical.align = "middle" )
cellProp22 = chprop( cellProp, background.color = "#517281" )

cellProp23 = chprop( cellProp, background.color = "#517281"
  , border.color = "#F37257", border.width = 2  )
```

---

chprop.parProperties *Modify paragraph formatting properties*

Description

Modify an object of class parProperties.

Usage

```
## S3 method for class 'parProperties'
chprop(object, text.align, padding.bottom, padding.top,
  padding.left, padding.right, padding, list.style, level, border.bottom,
  border.left, border.top, border.right, shading.color, ...)
```

Arguments

object	parProperties object to modify
text.align	text alignment - a single character value, expected value is one of 'left', 'right', 'center', 'justify'.
padding.bottom	paragraph bottom padding - 0 or positive integer value.
padding.top	paragraph top padding - 0 or positive integer value.
padding.left	paragraph left padding - 0 or positive integer value.
padding.right	paragraph right padding - 0 or positive integer value.
padding	paragraph padding - 0 or positive integer value. Argument padding overwrites arguments padding.bottom, padding.top, padding.left, padding.right.
list.style	list style - a single character value, expected value is one of 'none' (default), 'unordered', 'ordered', 'blockquote'.
level	list level if argument list is not 'none'.
border.bottom	borderProperties for bottom border. overwrite all border.bottom.* if specified.
border.left	borderProperties for left border. overwrite all border.left.* if specified.
border.top	borderProperties for top border. overwrite all border.top.* if specified.
border.right	borderProperties for right border. overwrite all border.right.* if specified.
shading.color	shading color - a single character value specifying a valid color (e.g. "#000000" or "black").
...	further arguments - not used

Value

a parProperties object

See Also

parProperties

Examples

```
parProp = parProperties()

parProp01 = chprop( parProp, text.align = "center" )
parProp02 = chprop( parProp, padding.bottom = 2 )
parProp03 = chprop( parProp, padding.top = 2 )
```

```
parProp04 = chprop( parProp, padding.left = 2 )
parProp05 = chprop( parProp, padding = 2 )

parProp06 = chprop( parProp, padding = 2, text.align = "center" )
```

`chprop.textProperties` *Modify text formatting properties*

Description

Modify an object of class `textProperties`.

Usage

```
## S3 method for class 'textProperties'
chprop(object, color, font.size, font.weight,
  font.style, underlined, font.family, vertical.align, shading.color, ...)
```

Arguments

`object`	`textProperties` object to modify
`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.
`font.weight`	single character value specifying font weight (expected value is `normal` or `bold`).
`font.style`	single character value specifying font style (expected value is `normal` or `italic`).
`underlined`	single logical value specifying if the font is underlined.
`font.family`	single character value specifying font name (it has to be an existing font in the OS).
`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").
`...`	further arguments - not used

Value

a `textProperties` object

See Also

`textProperties`

Examples

```
textProp = textProperties()

textProp01 = chprop( textProp, color = "red" )
textProp02 = chprop( textProp, font.size = 12 )
textProp03 = chprop( textProp, font.weight = "bold" )
textProp04 = chprop( textProp, font.style = "italic" )
textProp05 = chprop( textProp, underlined = TRUE )
textProp06 = chprop( textProp, font.family = "Arial" )
textProp07 = chprop( textProp, vertical.align = "superscript" )

textProp08 = chprop( textProp, font.size = 12, font.weight = "bold", shading.color = "red" )
```

---

declareTitlesStyles *Set manually headers'styles of a document object*

Description

Set manually titles'styles of a document object

Usage

```
declareTitlesStyles(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

`declareTitlesStyles` only works with docx documents.

See `declareTitlesStyles.docx` for examples.

Value

a document object

See Also

`docx`, `styles.docx` , `declareTitlesStyles.docx`, `addTOC.docx`

declareTitlesStyles.docx *Set manually headers'styles of a docx object*

Description

Set manually headers'styles of a docx object

Usage

```
## S3 method for class 'docx'
declareTitlesStyles(doc, stylenames, ...)
```

Arguments

doc	docx object to be used with declareTitlesStyles.
stylenames	existing styles (character vector) where first element represents the style to use for title 1, second element represents the style to use for title 2, etc.
...	further arguments, not used.

Details

Function addTitle need to know which styles are corresponding to which title level (1 ; 1.1 ; 1.1.1 ; etc.). When template is read, function docx try to guess what are theses styles. If he do not succeed, an error occured saying 'You must defined header styles via declareTitlesStyles first.'. In that case, run `styles(...)` to see what are available styles, then declareTitlesStyles to indicate which available styles are meant to be used as header styles.

See Also

docx,styles.docx,addTitle.docx ,declareTitlesStyles

Examples

```
## Not run:
doc = docx( title = "My example" )
styles( doc )
# [1] "Normal"                "Title1"                "Title2"
# [4] "Title3"                "Title4"                "Title5"
# [7] "Title6"                "Title7"                "Title8"
#[10] "Title9"                "Defaut"                ...
doc = declareTitlesStyles(doc
, stylenames = c("Title1", "Title2", "Title3"
, "Title4", "Title5", "Title6", "Title7", "Title8", "Title9" ) )
doc = addTitle( doc, "title 1", 1 )

## End(Not run)
```

deleteBookmark *delete a bookmark into a docx object*

Description

delete a bookmark into a docx object

Usage

```
deleteBookmark(doc, bookmark)
```

Arguments

doc	Object of class docx
bookmark	a character vector specifying bookmark id to delete

See Also

docx

deleteBookmarkNextContent *delete first content after a bookmark into a docx object*

Description

delete first content after a bookmark into a docx object

Usage

```
deleteBookmarkNextContent(doc, bookmark)
```

Arguments

doc	Object of class docx
bookmark	a character vector specifying bookmark id to delete

See Also

docx

dim.docx *Get page layout dimensions of a Word document*

Description

Returns page width and height and page margins of a docx object.

Usage

```
## S3 method for class 'docx'
dim(x)
```

Arguments

x	Object of class docx

See Also

docx, dim.pptx

Examples

```
doc = docx( title = "title" )
dim( doc )
```

dim.pptx *Get layout information on a PowerPoint slide*

Description

Returns slide width and height, position and dimension of the next available shape in the current slide.

Usage

```
## S3 method for class 'pptx'
dim(x)
```

Arguments

x	Object of class pptx

See Also

pptx, dim.docx

Examples

```
doc = pptx( title = "title" )
doc = addSlide( doc, "Title and Content" )
dim(doc)
```

`doc-list-settings` *format ordered and unordered lists*

Description

Create a description used to format ordered and unordered lists in object documents.

Arguments

`ol.left`	left indent values (in inches) for each level of an ordered list. Length must be 9 as there are 9 elements to define (from level1 to level9).
`ol.hanging`	space values (in inches) between numbering label (argument `ol.format`) and content for each level of an ordered list. Length must be 9 as there are 9 elements to define (from level1 to level9).
`ol.format`	type of numbering for ordered levels, values can be 'decimal' or 'upperRoman' or 'lowerRoman' or 'upperLetter' or 'lowerLetter'. Length must be 9 as there are 9 elements to define (from level1 to level9).
`ol.pattern`	numbering pattern for ordered levels. A level numbering has the following syntax: `"%1"` (numbering of level1), `"%2"` (numbering of level2), ..., `"%9"` (numbering of level9).
`ul.left`	left indent values for each level of an unordered list. Length must be 9 as there are 9 elements to define (from level1 to level9). Length must be 9 as there are 9 elements to define (from level1 to level9).
`ul.hanging`	space values (in inches) between bullet symbol (argument `ul.format`) and content for each level of an unordered list. Length must be 9 as there are 9 elements to define (from level1 to level9).
`ul.format`	type of bullet for unordered levels, values can be 'disc' or 'circle' or 'square'. Length must be 9 as there are 9 elements to define (from level1 to level9).

Details

List settings are used to configure formatting of list in documents.

It can be set in R session options or as a parameter in `docx` or `pptx` or `bsdoc`.

See Also

`addParagraph.docx`, `addParagraph.pptx`, `addParagraph.bsdoc`, `ReporteRs`

Examples

```
numbering.pattern = c( "%1.", "%1. %2.", "%1. %2. %3.",
  "%4.", "%5.", "%6.", "%7.", "%8.", "%9." )

ordered.formats = rep( c( "decimal", "upperRoman", "upperLetter"), 3 )

unordered.formats = rep( c( "square", "disc", "circle"), 3 )

left.indent = seq( from = 0, by = 0.5, length.out = 9)

options("ReporteRs-list-definition" = list(
  ol.left = left.indent,
  ol.hanging = rep( 0.4, 9 ),
  ol.format = ordered.formats,
  ol.pattern = numbering.pattern,
  ul.left = left.indent,
  ul.hanging = rep( 0.4, 9 ),
  ul.format = unordered.formats
  )
)
```

---

docx *Create Microsoft Word document object representation*

Description

Create a docx object

Usage

```
docx(title = "untitled", template,
  list.definition = getOption("ReporteRs-list-definition"))
```

Arguments

title	"character" value: title of the document (in the doc properties).
template	"character" value, it represents the filename of the docx file used as a template.
list.definition	a list definition to specify how ordered and unordered lists have to be formated. See list.settings. Default to getOption("ReporteRs-list-definition").

Details

Several methods can used to send R output into an object of class docx.

- addTitle.docx add titles

- addParagraph.docx add text
- addPlot.docx add plots
- addFlexTable.docx add tables. See FlexTable
- addImage.docx add external images
- addMarkdown.docx add markdown text
- addTOC.docx add table of content
- addPageBreak.docx add page break
- addSection.docx add section

R outputs (tables, plots, paragraphs and images) can be inserted (and not added at the end) in a document if a bookmark exists in the template file. See bookmark.

Once object has content, user can write the docx into a ".docx" file, see writeDoc.

Value

an object of class docx.

Note

Word 2007-2013 (*.docx) file formats are the only supported files.
Document are manipulated in-memory ; a docx's document is not written to the disk unless the writeDoc method has been called on the object.

References

Wikipedia: Office Open XML
http://en.wikipedia.org/wiki/Office_Open_XML

See Also

bsdoc, pptx, bookmark

Examples

```
require( ggplot2 )

# Word document to write
docx.file = "document_example.docx"

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

# Create a new document
doc = docx( title = "title" )

# display available styles
styles( doc )
```

```
# add title
doc = addParagraph( doc, "Document title", stylename = "TitleDoc" )

# add a paragraph
doc = addParagraph( doc , "This document is generated with ReporteRs."
  , stylename="Citationintense")

# add page break
doc = addPageBreak( doc )

# add a title
doc = addTitle( doc, "Table of contents", level =  1 )

################ TOC DEMO ################
# add a table of content
doc = addTOC( doc )

# add page break and then tables of contents for produced plots and tables
doc = addPageBreak( doc )
doc = addTitle( doc, "List of graphics", level =  1 )
doc = addTOC( doc, stylename = "rPlotLegend" )
doc = addTitle( doc, "List of tables", level =  1 )
doc = addTOC( doc, stylename = "rTableLegend" )

# add page break
doc = addPageBreak( doc )

################ TEXT DEMO ################

# add a title
doc = addTitle( doc, "Text demo", level =  1 )


sometext = c( "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
  , "In sit amet ipsum tellus. Vivamus dignissim arcu sit amet faucibus auctor."
  , "Quisque dictum tristique ligula."
)
# add simple text with 'Normal' style
doc = addParagraph( doc, value = sometext, stylename="Normal" )
# add simple text with 'BulletList' style
doc = addParagraph( doc, value = sometext, stylename="BulletList" )

# Add "My tailor is rich" and "Cats and Dogs"
# format some of the pieces of text
pot1 = pot("My tailor"
  , textProperties(color="red", shading.color = "#CCCCCC" ) ) + " is " + pot("rich"
, textProperties(font.weight="bold") )
pot2 = pot("Cats"
, textProperties(color="red" )
) + " and " + pot("Dogs"
, textProperties(color="blue" ) )
doc = addParagraph(doc, set_of_paragraphs( pot1, pot2 ), stylename="Normal" )
```

```
doc = addParagraph(doc, "Silentium tractibus per minimis ne excita
ut temptentur generalibus quam primordiis per clades post delictis
iuge exitium silentium per et.",
par.properties = parProperties( padding.left = 25, padding.right = 25) )


doc = addParagraph(doc, pot("Gallus necem refert singula modum quae
est quae quorum leo quae non cadaveribus ut quod.", format = textItalic( ) ),
par.properties = parProperties(list.style = "blockquote") )


ordered.list.level1 = parProperties(list.style = "ordered", level = 1 )
ordered.list.level2 = parProperties(list.style = "ordered", level = 2 )

doc = addParagraph( doc, value = sometext, par.properties = ordered.list.level1 )
doc = addParagraph( doc, value = sometext, par.properties = ordered.list.level2 )

################ PLOT DEMO ################

myplot = qplot(Sepal.Length, Petal.Length
  , data = iris, color = Species
  , size = Petal.Width, alpha = I(0.7)
)
# Add titles and then 'myplot'
doc = addTitle( doc, "Plot examples", level =  1 )
doc = addPlot( doc, function( ) print( myplot ) )
# Add a legend below the plot
doc = addParagraph( doc, value = "my first plot", stylename = "rPlotLegend")

################ FLEXTABLE DEMO ################

doc = addTitle( doc, "FlexTable example", level = 1 )

# Create a FlexTable with data.frame mtcars, display rownames
# use different formatting properties for header and body
MyFTable = FlexTable( data = mtcars, add.rownames = TRUE,
header.cell.props = cellProperties( background.color = "#00557F" ),
header.text.props = textProperties( color = "white",
font.size = 11, font.weight = "bold" ),
body.text.props = textProperties( font.size = 10 )
)
# zebra stripes - alternate colored backgrounds on table rows
MyFTable = setZebraStyle( MyFTable, odd = "#E1EEf4", even = "white" )

# applies a border grid on table
MyFTable = setFlexTableBorders(MyFTable,
inner.vertical = borderProperties( color="#0070A8", style="solid" ),
inner.horizontal = borderNone(),
outer.vertical = borderProperties( color = "#006699", style = "solid", width = 2 ),
outer.horizontal = borderProperties( color = "#006699", style = "solid", width = 2 )
)
```

```
# add MyFTable into document
doc = addFlexTable( doc, MyFTable )
doc = addParagraph( doc, value = "my first table", stylename = "rTableLegend")

# write the doc
writeDoc( doc, file = docx.file)
```

---

docx-bookmark *docx bookmarks*

Description

docx can generate Word documents using bookmarks as placeholders to insert contents. Read MS documentation about bookmark here:

http://office.microsoft.com/en-us/word-help/add-or-delete-bookmarks-HP001226532.aspx#BM1

Functions addFlexTable, addPlot , addParagraph and addImage can send respective outputs into these bookmarks.

These functions have an optional argument named bookmark.

When used with addPlot, addParagraph and addImage, content (plots, paragraphs or images) will replace the whole paragraph containing the bookmark.

When used with addFlexTable content (table) will be inserted after the paragraph containing the bookmark.

To be used with a docx object, bookmark must be placed into a single paragraph, if placed along 1 or more paragraphs side effects could occur and insertion of a content could fail.

You can insert the bookmark at the beginning of the paragraph (see the file bookmark_example.docx in the templates directory of the package for an example) or on a portion of a text in a paragraph.

See Also

docx

Examples

```
require( ReporteRs )

# Word document to write
docx.file = "document_new.docx"

# create document
doc = docx( title = "My example"
  , template = file.path( find.package("ReporteRs"), "templates/bookmark_example.docx")
  )

# replace bookmarks 'AUTHOR' and 'REVIEWER'
# by dummy values
```

```
doc = addParagraph( doc
, value = c( "James Sonny Crockett", "Ricardo Rico Tubbs" )
, stylename = "Normal"
, bookmark = "AUTHOR" )
doc = addParagraph( doc
, value = c( "Martin Marty Castillo" )
, stylename = "Normal"
, bookmark = "REVIEWER" )



MyFTable = FlexTable( data = mtcars[1:10, ]
, add.rownames=TRUE
)

# replace bookmarks 'DATA' and 'CONFINT' located in 'ttest_example.docx'
# by data.frame objects 'data' and 'conf.int'
doc = addFlexTable( doc
, MyFTable
, bookmark = "DATA1" )

# replace bookmarks 'DATA' and 'CONFINT' located in 'ttest_example.docx'
# by data.frame objects 'data' and 'conf.int'
doc = addFlexTable( doc
, vanilla.table( head( iris ) )
, bookmark = "DATA2" )

doc = addPlot( doc, vector.graphic = TRUE
, fun = function(){
require(stats)
sale5 <- c(6, 4, 9, 7, 6, 12, 8, 10, 9, 13)
plot(sale5)
abline(lsfit(1:10, sale5))
abline(lsfit(1:10, sale5, intercept = FALSE), col = 4)
}
, bookmark = "PLOT")


doc = addParagraph( doc, value = c( "Header 1" )
, stylename = "NAMESTYLE", bookmark = "COLNAME1" )

doc = addParagraph( doc, value = c( "Header 2" )
, stylename = "NAMESTYLE", bookmark = "COLNAME2" )

doc = addParagraph( doc, value = c( "Header 3" )
, stylename = "NAMESTYLE", bookmark = "COLNAME3" )

doc = addParagraph( doc, value = c( "Row name 1" )
, stylename = "NAMESTYLE", bookmark = "ROWNAME1" )

doc = addParagraph( doc, value = c( "Row name 2" )
, stylename = "NAMESTYLE", bookmark = "ROWNAME2" )
```

```
doc = addParagraph( doc, value = c( "Hello World" )
, stylename = "DATASTYLE", bookmark = "ANYDATA" )

writeDoc( doc, docx.file )
```

DropDownMenu *Create a bootstrap DropDownMenu*

Description

Create a DropDownMenu object. This object is to be used with BootstrapMenu to define menu links.

Usage

```
DropDownMenu(label)
```

Arguments

label	"character" value: label of the DropDownMenu.

Value

an object of class DropDownMenu.

See Also

bsdoc, addLinkItem, addBootstrapMenu

Examples

```
mydd = DropDownMenu( label = "My menu" )
mydd = addLinkItem( mydd, label = "GitHub", "http://github.com/", active = TRUE)
mydd = addLinkItem( mydd, separator.after = TRUE)
mydd = addLinkItem( mydd, label = "Wikipedia", "http://www.wikipedia.fr")
```

FlexCell *Cell object for FlexTable*

Description

Create a representation of a cell that can be inserted in a FlexRow. For internal usage.

Usage

```
FlexCell(value, colspan = 1, par.properties = parProperties(),
  cell.properties = cellProperties())
```

Arguments

value	a content value - a value of type character or pot or set_of_paragraphs.
colspan	defines the number of columns the cell should span
par.properties	parProperties to apply to content
cell.properties	cellProperties to apply to content

See Also

addFlexTable, addHeaderRow, addFooterRow

Examples

```
FlexCell( value = "Hello" )
FlexCell( value = "Hello", colspan = 3)
FlexCell( "Column 1", cell.properties = cellProperties(background.color="#527578")  )


# define a complex formatted text
mytext = pot("Hello", format = textProperties(color = "blue")
  ) + " " + pot( "world", format = textProperties(font.size = 9)
)
Fcell = FlexCell( mytext, colspan = 4 )

# define two paragraph and put them in a FlexCell
mytext1 = pot("Hello", format = textProperties(color = "blue") )
mytext2 = pot( "world", format = textProperties(font.size = 9) )
Fcell = FlexCell( set_of_paragraphs( mytext1, mytext2 ) )
```

FlexRow *Row object for FlexTable*

Description

Create a representation of a row that can be inserted in a FlexTable. For internal usage.

Usage

```
FlexRow(values, colspan, text.properties = textProperties(),
  par.properties = parProperties(), cell.properties = cellProperties())
```

Arguments

values	Optional. a character vector to use as text content, the row will contain as many cells as there are in values.
text.properties	Optional. textProperties to apply to each cell. Used only if values are not missing.
par.properties	Optional. parProperties to apply to each cell. Used only if values are not missing.
cell.properties	Optional. cellProperties to apply to each cell. Used only if values are not missing.
colspan	integer Optional. vector specifying for each element the number of columns to span for each corresponding value (in values).

See Also

FlexTable, alterFlexRow , addHeaderRow, addFooterRow

Examples

```
## example with characters
headerRow = FlexRow( c("Column 1", "Column 2")
  , cell.properties = cellProperties(background.color="#527578") )
## example with FlexCell
headerRow = FlexRow()
headerRow[1] = FlexCell( "Column 1"
  , cell.properties = cellProperties(background.color="#527578")  )
headerRow[2] = FlexCell( "Column 2"
  , cell.properties = cellProperties(background.color="#527578")  )
```

FlexTable *FlexTable creation*

Description

Create an object of class FlexTable.

FlexTable can be manipulated so that almost any formatting can be specified.

An API is available to let you manipulate (format, add text, merge cells, etc.) your FlexTable. A FlexTable is made of 3 parts: header, body and footer. To insert headers and footers rows with eventually merged cells, see addHeaderRow and addFooterRow.

Formating can be done on cells, paragraphs and text (borders, colors, fonts, etc.) , see alterFlexTable.

Usage

```
FlexTable(data, numrow, numcol, header.columns = TRUE, add.rownames = FALSE,
  body.cell.props = cellProperties(padding = 1),
  body.par.props = parProperties(padding = 0),
  body.text.props = textProperties(),
  header.cell.props = cellProperties(padding = 1),
  header.par.props = parProperties(padding = 0),
  header.text.props = textProperties(font.weight = "bold"))
```

Arguments

data	(a data.frame or matrix object) to add
numrow	number of row in the table body. Mandatory if data is missing.
numcol	number of col in the table body. Mandatory if data is missing.
header.columns	logical value - should the colnames be included in the table as table headers. If FALSE, no headers will be printed unless you use addHeaderRow.
add.rownames	logical value - should the row.names be included in the table.
body.cell.props	default cells formatting properties for table body
body.par.props	default paragraphs formatting properties for table body
body.text.props	default text formatting properties for table body
header.cell.props	default cells formatting properties for table headers
header.par.props	default paragraphs formatting properties for table headers
header.text.props	default text formatting properties for table headers

Details

The classical workflow would be to create a FlexTable, to add headers rows (see addHeaderRow) and eventually footers rows (see addFooterRow).

A FlexTable lets you add text in cells and modify cells, paragraphs and text properties. Text can be added with operator [<-. Text, paragraphs and cells properties can be also modified with operator [<-. (see alterFlexTable).

Below list of functions to use with FlexTable objects:

Text formatting

Apply a textProperties object to a subset of the FlexTable. Use the operator [<-. The textProperties object will be used to format all text from selected cells. See alterFlexTable.

Text adding

Add text with operator [<-. Text can be added just after the last text in the cell or as a new paragraph. Format can also be specified. Text can also be a pot object if the text format is complex.

Paragraph formatting

Apply a parProperties object to a subset of the FlexTable. Use the operator [<-. The parProperties object will be used to format all paragraphs from selected cells. See alterFlexTable.

Cell formatting

Apply a cellProperties object to a subset of the FlexTable. Use the operator [<-. The cellProperties object will be used to format selected cells. See alterFlexTable.

Borders

Apply borders scheme to a FlexTable with function setFlexTableBorders.

Set a border to a selection in a FlexTable with the operator [<- and an object of class borderProperties. Don't forget to specify argument side. See alterFlexTable.

Cell background colors

Applies background colors to cells. See setFlexTableBackgroundColors.

Alternate row colors (zebra striping) with function setZebraStyle.

Applies background colors to rows with function setRowsColors.

Applies background colors to columns with function setColumnsColors.

Cell merge

Span rows within columns with function spanFlexTableRows.

Span columns within rows with function spanFlexTableColumns.

Columns widths

Set columns widths with function setFlexTableWidths.

See Also

addHeaderRow, addFooterRow, setFlexTableWidths , alterFlexTable, setFlexTableBorders , spanFlexTableRows, spanFlexTableColumns , setRowsColors, setColumnsColors, setZebraStyle , setFlexTableBackgroundColors, pot , addFlexTable.docx , addFlexTable.pptx, addFlexTable.bsdoc

Examples

```
#####################################################################

# Create a FlexTable with data.frame mtcars, display rownames
# use different formatting properties for header and body
MyFTable = FlexTable( data = mtcars, add.rownames = TRUE,
  header.cell.props = cellProperties( background.color = "#00557F" ),
  header.text.props = textProperties( color = "white",
    font.size = 11, font.weight = "bold" ),
  body.text.props = textProperties( font.size = 10 )
)
# zebra stripes - alternate colored backgrounds on table rows
MyFTable = setZebraStyle( MyFTable, odd = "#E1EEf4", even = "white" )

# applies a border grid on table
MyFTable = setFlexTableBorders(MyFTable,
  inner.vertical = borderProperties( color="#0070A8", style="solid" ),
  inner.horizontal = borderNone(),
  outer.vertical = borderProperties( color = "#006699", style = "solid", width = 2 ),
  outer.horizontal = borderProperties( color = "#006699", style = "solid", width = 2 )
)
#####################################################################

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

# a summary of mtcars
dataset = aggregate( mtcars[, c("disp", "mpg", "wt")]
  , by = mtcars[, c("cyl", "gear", "carb")]
  , FUN = mean )
dataset = dataset[ order(dataset$cyl, dataset$gear, dataset$carb), ]


# set cell padding defaut to 2
baseCellProp = cellProperties( padding = 2 )

# Create a FlexTable with data.frame dataset
MyFTable = FlexTable( data = dataset
  , body.cell.props = baseCellProp
  , header.cell.props = baseCellProp
  , header.par.props = parProperties(text.align = "right" )
)

# set columns widths (in inches)
MyFTable = setFlexTableWidths( MyFTable, widths = c(0.5, 0.5, 0.5, 0.7, 0.7, 0.7) )

# span successive identical cells within column 1, 2 and 3
MyFTable = spanFlexTableRows( MyFTable, j = 1, runs = as.character( dataset$cyl ) )
MyFTable = spanFlexTableRows( MyFTable, j = 2, runs = as.character( dataset$gear ) )
MyFTable = spanFlexTableRows( MyFTable, j = 3, runs = as.character( dataset$carb ) )
```

```
# overwrites some text formatting properties
MyFTable[dataset$wt < 3, 6] = textProperties( color="#003366")
MyFTable[dataset$mpg < 20, 5] = textProperties( color="#993300")

# overwrites some paragraph formatting properties
MyFTable[, 1:3] = parProperties(text.align = "center")
MyFTable[, 4:6] = parProperties(text.align = "right")


Footnote1 = Footnote(  )

par1 = pot("About this reference", textBold( ) )
par2 = pot("Omni ab coalitos pro malivolus obsecrans graviter
cum perquisitor perquisitor pericula saepeque inmunibus coalitos ut.",
  textItalic(font.size = 8) )
Footnote1 = addParagraph( Footnote1, set_of_paragraphs( par1, par2 ),
  parProperties(text.align = "justify"))

Footnote1 = addParagraph( Footnote1, set_of_paragraphs( "list item 1", "list item 2" ),
  parProperties(text.align = "left", list.style = "ordered"))

an_rscript = RScript( text = "ls()
x = rnorm(10)" )
Footnote1 = addParagraph( Footnote1, an_rscript )

MyFTable[1, 1, newpar = TRUE] = pot("a note",
  footnote = Footnote1, format = textBold(color="gray") )

# applies a border grid on table
MyFTable = setFlexTableBorders( MyFTable, footer=TRUE
  , inner.vertical = borderProperties( color = "#666666" )
  , inner.horizontal = borderProperties( color = "#666666" )
  , outer.vertical = borderProperties( width = 2, color = "#666666" )
  , outer.horizontal = borderProperties( width = 2, color = "#666666" )
)
data = cor( cor(mtcars) )

pal = c( "#D73027", "#F46D43", "#FDAE61", "#FEE08B",
"#D9EF8B", "#A6D96A", "#66BD63", "#1A9850" )
mycut = cut( data,
breaks = c(-1,-0.75,-0.5,-0.25,0,0.25,0.5,0.75,1),
include.lowest = TRUE, label = FALSE )
mycolors = pal[ mycut ]

MyFTable = FlexTable( round(data, 3), add.rownames = TRUE )

# set computed colors
MyFTable = setFlexTableBackgroundColors( MyFTable,
j = seq_len(ncol(data)) + 1,
colors = mycolors )

# cosmetics
MyFTable = setFlexTableBackgroundColors( MyFTable, i = 1,
```

```
colors = "gray", to = "header" )
MyFTable[1, , to = "header"] = textBold(color="white")

MyFTable = setFlexTableBackgroundColors( MyFTable, j = 1, colors = "gray" )
MyFTable[,1] = textBold(color="white")

MyFTable = setFlexTableBorders( MyFTable
, inner.vertical = borderProperties( style = "dashed", color = "white" )
, inner.horizontal = borderProperties( style = "dashed", color = "white"  )
, outer.vertical = borderProperties( width = 2, color = "white"  )
, outer.horizontal = borderProperties( width = 2, color = "white"  )
)



data( iris )
iris = head( iris[, c(5, 1:4)] )

default_text = textProperties( font.size = 11 )
note_text = chprop(default_text, vertical.align = "superscript", color = "blue")

iris_ft = FlexTable( data = iris, header.columns = FALSE )
iris_ft = addHeaderRow( iris_ft, value = c("", "Measures" ), colspan = c( 4, 1 ) )
iris_ft = addHeaderRow( iris_ft, value = gsub( "\\.", " ", names( iris ) ) )
iris_ft[2, 2, newpar = TRUE ] = "Hi there"
iris_ft[2, 1, to="header"] = pot("* this is a note", note_text )


iris_ft = spanFlexTableRows( iris_ft, j = "Species", runs = as.character( iris$Species ) )
iris_ft = setFlexTableBorders( iris_ft,
  inner.vertical = borderProperties( style = "none" ),
  inner.horizontal = borderProperties( width = 1 ),
  outer.vertical = borderProperties( width = 0 ),
  outer.horizontal = borderProperties( width = 2 ),
  footer = TRUE
)
```

FontMetric *Font metric*

Description

get font metric from a font name and a size

Usage

```
FontMetric(fontfamily, fontsize)
```

Arguments

fontfamily	font name
fontsize	font size

`Footnote` *Create a Footnote*

Description

A footnote is a a set of paragraphs placed at the bottom of a page if document object is a docx object or used as a tooltip if document object is an bsdoc object.

If in a docx object, footnote will be flagged by a number immediately following the portion of the text the note is in reference to.

Usage

```
Footnote(index.text.properties = textProperties(vertical.align =
  "superscript"))
```

Arguments

index.text.properties	textProperties to apply to note index symbol (only for docx object).

Value

an object of class Footnote.

See Also

docx, bsdoc, pot

Examples

```
## docx example
doc = docx( )

par1 = pot("About this reference", textBold( ) )
par2 = pot("Omni ab coalitos pro malivolus obsecrans graviter
cum perquisitor perquisitor pericula saepeque inmunibus coalitos ut.",
textItalic(font.size = 8) )

Footnote1 = Footnote( )
Footnote1 = addParagraph( Footnote1, set_of_paragraphs( par1, par2 ),
parProperties(text.align = "justify"))
Footnote1 = addParagraph( Footnote1, set_of_paragraphs( "list item 1", "list item 2" ),
parProperties(text.align = "left", list.style = "ordered"))
```

```
an_rscript = RScript( par.properties = parProperties(shading.color = "gray90"),
text = "ls()
x = rnorm(10)" )
Footnote1 = addParagraph( Footnote1, an_rscript,
parProperties(text.align = "left"))


Footnote2 = Footnote(  )
Footnote2 = addParagraph( Footnote2, pot("This is another reference" ),
par.properties = parProperties(text.align = "center"))

doc = addTitle( doc, "Title example 1", level = 1 )

pot1 = "Hae duae provinciae " + pot("bello",
footnote = Footnote1 ) + " quondam piratico catervis mixtae
praedonum a Servilio pro consule missae sub
iugum factae sunt vectigales. et hae quidem regiones velut in prominenti terrarum
lingua positae ob orbe eoo monte Amano disparantur."

pot2 = pot("Latius iam disseminata licentia onerosus bonis omnibus Caesar nullum
post haec adhibens modum orientis latera cuncta vexabat nec honoratis parcens
nec urbium primatibus nec plebeiis." ) + pot(" Here is another note.", footnote = Footnote2)


# Add my.pars into the document doc
doc = addParagraph(doc, set_of_paragraphs( pot1, pot2 ) )

docx.file = "footnote.docx"

writeDoc( doc, file = docx.file )
## bsdoc example
doc = bsdoc( title = "Footnote example" )

par1 = pot("About this reference", textBold( ) )
par2 = pot("Omni ab coalitos pro malivolus obsecrans graviter
cum perquisitor perquisitor pericula saepeque inmunibus coalitos ut.",
textItalic(font.size = 8) )

Footnote1 = Footnote( )
Footnote1 = addParagraph( Footnote1, set_of_paragraphs( par1, par2 ),
parProperties(text.align = "justify"))
Footnote1 = addParagraph( Footnote1, set_of_paragraphs( "list item 1", "list item 2" ),
parProperties(text.align = "left", list.style = "ordered"))
an_rscript = RScript( par.properties = parProperties(shading.color = "gray90"),
text = "ls()
x = rnorm(10)" )
Footnote1 = addParagraph( Footnote1, an_rscript,
parProperties(text.align = "left"))


Footnote2 = Footnote(  )
Footnote2 = addParagraph( Footnote2, pot("This is another reference" ),
par.properties = parProperties(text.align = "center"))
```

```
doc = addTitle( doc, "Title example 1", level = 1 )

pot1 = "Hae duae provinciae " + pot("bello",
footnote = Footnote1 ) + " quondam piratico catervis mixtae
praedonum a Servilio pro consule missae sub
iugum factae sunt vectigales. et hae quidem regiones velut in prominenti terrarum
lingua positae ob orbe eoo monte Amano disparantur."

pot2 = pot("Latius iam disseminata licentia onerosus bonis omnibus Caesar nullum
post haec adhibens modum orientis latera cuncta vexabat nec honoratis parcens
nec urbium primatibus nec plebeiis." ) + pot(" Here is another note.", footnote = Footnote2)


# Add my.pars into the document doc
doc = addParagraph(doc, set_of_paragraphs( pot1, pot2 ) )

writeDoc( doc, file = "Footnote/example.html" )
```

`is.color` *color checking*

Description

Check if character string is a valid color representation

Usage

```
is.color(x)
```

Arguments

x	value(s) to be tested

Details

see http://stackoverflow.com/questions/13289009/check-if-character-string-is-a-valid-color-represent
13290832#13290832

See Also

pptx, docx

Examples

```
is.color( c(NA, "black", "blackk", "1", "#00", "#000000") )
```

light.table *get a simple FlexTable from a dataset*

Description

get a simple FlexTable from a dataset

Usage

```
light.table(dataset, double.format = "%0.3f")
```

Arguments

dataset	the data to use
double.format	format string for double column to format in the dataset. See argument fmt of sprintf.

See Also

FlexTable

Examples

```
light.table( iris)
```

parCenter *shortcut for centered alignment*

Description

shortcut for center alignment parProperties()

Usage

```
parCenter(...)
```

Arguments

...	arguments passed to parProperties

See Also

textNormal , textBold , textItalic , textBoldItalic , parRight , parLeft , parJustify , borderDotted , borderDashed , borderNone , borderSolid

Examples

```
parLeft()
```

---

parJustify *shortcut for justified alignment*

Description

shortcut for center alignment parProperties()

Usage

```
parJustify(...)
```

Arguments

...	arguments passed to parProperties

See Also

textNormal , textBold , textItalic , textBoldItalic , parRight , parLeft , parCenter , borderDotted , borderDashed , borderNone , borderSolid

Examples

```
parLeft()
```

---

parLeft *shortcut for left alignment*

Description

shortcut for left alignment parProperties()

Usage

```
parLeft(...)
```

Arguments

...	arguments passed to parProperties

See Also

textNormal , textBold , textItalic , textBoldItalic , parRight , parCenter , parJustify , borderDotted , borderDashed , borderNone , borderSolid

Examples

```
parLeft()
```

`parProperties` *Paragraph formatting properties*

Description

Create a `parProperties` object that describes paragraph formatting properties.

Usage

```
parProperties(text.align = "left", padding.bottom = 1, padding.top = 1,
  padding.left = 1, padding.right = 1, padding, list.style = "none",
  level = 1, border.bottom = borderNone(), border.left = borderNone(),
  border.top = borderNone(), border.right = borderNone(), shading.color)
```

Arguments

`text.align`	text alignment - a single character value, expected value is one of 'left', 'right', 'center', 'justify'.
`padding.bottom`	paragraph bottom padding - 0 or positive integer value.
`padding.top`	paragraph top padding - 0 or positive integer value.
`padding.left`	paragraph left padding - 0 or positive integer value.
`padding.right`	paragraph right padding - 0 or positive integer value.
`padding`	paragraph padding - 0 or positive integer value. Argument `padding` overwrites arguments `padding.bottom`, `padding.top`, `padding.left`, `padding.right`.
`list.style`	list style - a single character value, expected value is one of 'none' (default), 'unordered', 'ordered', 'blockquote'. This will not have any effect if used in a FlexTable.
`level`	list level if argument `list` is not 'none'. This will not have any effect if used in a FlexTable.
`border.bottom`	`borderProperties` for bottom border. overwrite all border.bottom.* if specified.
`border.left`	`borderProperties` for left border. overwrite all border.left.* if specified.
`border.top`	`borderProperties` for top border. overwrite all border.top.* if specified.
`border.right`	`borderProperties` for right border. overwrite all border.right.* if specified.
`shading.color`	shading color - a single character value specifying a valid color (e.g. "#000000" or "black").

Details

parProperties is used to control paragraph properties. It is used when adding plots or when adding content in a FlexTable.

Default values are:

- `text.align` "left"
- `padding.bottom` 1
- `padding.top` 1
- `padding.left` 1
- `padding.right` 1
- `list.style` 'none'
- `level` 1

Value

a `parProperties` object

See Also

`chprop.parProperties`, `alterFlexTable` , `addParagraph`

Examples

```
parProperties( text.align = "center", padding = 5)

parProperties( text.align = "center",
  padding.top = 5,
  padding.bottom = 0,
  padding.left = 2,
  padding.right = 0
)

parProperties( list.style = "ordered", level = 2)

parProperties( list.style = "unordered", level = 2)
```

---

parRight *shortcut for right alignment*

Description

shortcut for right alignment parProperties()

Usage

```
parRight(...)
```

Arguments

...	arguments passed to parProperties

See Also

textNormal , textBold , textItalic , textBoldItalic , parLeft , parCenter , parJustify , borderDotted , borderDashed , borderNone , borderSolid

Examples

```
parRight()
```

---

pbc_summary *pbc summary*

Description

pbc summary

Usage

```
data(pbc_summary)
```

Format

A data frame

---

pot *Piece of Text (formated text)*

Description

Create an object with a text to display and its formatting properties.

Usage

```
pot(value = "", format = textProperties(), hyperlink, footnote)
```

Arguments

value	text value or a value that has a format method returning character value.
format	formatting properties (an object of class textProperties).
hyperlink	a valid url to use as hyperlink when clicking on value.
footnote	a Footnote object.

Details

a pot (piece of text) is a convenient way to define a paragraph of text where some text are not all formated the same.

A pot can be associated with an hyperlink.

A pot can be associated with a Footnote. Note that footnotes can not be inserted in a pptx object.

See Also

addParagraph.docx, addParagraph.pptx, addParagraph.bsdoc, Footnote , +.pot

Examples

```
# "My tailor is rich" with formatting on some words
pot1 = pot("My tailor", textProperties(color = "red" )
  ) + " is " + pot("rich", textProperties(shading.color = "red", font.weight = "bold" ) )

# "Cats and dogs" with formatting on some words
pot2 = pot("Cats", textProperties(color = "red" ) ) +
  " and " +
  pot("dogs", textProperties( color = "blue" ),
    hyperlink = "http://www.wikipedia.org/" )
```

---

pptx *Create Microsoft PowerPoint document object representation*

Description

Create a pptx object

Usage

```
pptx(title, template,
  list.definition = getOption("ReporteRs-list-definition"))
```

Arguments

title	"character" value: title of the document (in the doc properties).
template	"character" value, it represents the filename of the pptx file used as a template.
list.definition	a list definition to specify how ordered and unordered lists have to be formated. See list.settings. Default to getOption("ReporteRs-list-definition").

Details

To send R output in a pptx document, a slide (see addSlide.pptx have to be added to the object first (because output is beeing written in slides).

Several methods can used to send R output into an object of class pptx.

- addTitle.pptx add titles
- addParagraph.pptx add text
- addPlot.pptx add plots
- addMarkdown.pptx add markdown
- addFlexTable.pptx add FlexTable
- addDate.pptx add a date (most often in the bottom left area of the slide)
- addFooter.pptx add a comment in the footer (most often in the bottom center area of the slide)
- addPageNumber.pptx add a page number (most often in the bottom right area of the slide)
- addImage.pptx add external images

Once object has content, user can write the pptx into a ".pptx" file, see writeDoc.

Value

an object of class pptx.

Note

Power Point 2007-2013 (*.pptx) file formats are the only supported files.

Document are manipulated in-memory ; a pptx's document is not written to the disk unless the writeDoc method has been called on the object.

References

Wikipedia: Office Open XML
http://en.wikipedia.org/wiki/Office_Open_XML

See Also

docx, bsdoc

Examples

```
require( ggplot2 )

# Word document to write
pptx.file = "presentation_example.pptx"

# set default font size to 26
options( "ReporteRs-fontsize" = 26 )

# Create a new document
doc = pptx( title = "title" )

# display layouts names
slide.layouts( doc )

# add a slide with layout "Title Slide"
doc = addSlide( doc, slide.layout = "Title Slide" )

doc = addTitle( doc, "Presentation title" ) #set the main title
doc = addSubtitle( doc , "This document is generated with ReporteRs.")#set the sub-title


################ TEXT DEMO ################

# add a slide with layout "Title and Content" then add content
doc = addSlide( doc, slide.layout = "Two Content" )

# add a title
doc = addTitle( doc, "Text demo" )
sometext = c( "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
, "In sit amet ipsum tellus. Vivamus dignissim arcu sit amet faucibus auctor."
, "Quisque dictum tristique ligula."
)

# add simple text
doc = addParagraph( doc, value = sometext )

# Add "My tailor is rich" and "Cats and Dogs"
# format some of the pieces of text
pot1 = pot("My tailor"
, textProperties(color="red" ) ) + " is " + pot("rich"
, textProperties(font.weight="bold") )
pot2 = pot("Cats"
, textProperties(color="red" )
) + " and " + pot("Dogs"
, textProperties(color="blue" ) )
doc = addParagraph(doc, set_of_paragraphs( pot1, pot2 ) )

################ PLOT DEMO ################
doc = addSlide( doc, slide.layout = "Title and Content" )
doc = addTitle( doc, "Plot examples" )
```

```
myplot = qplot(Sepal.Length, Petal.Length
, data = iris, color = Species
, size = Petal.Width, alpha = I(0.7)
)
# Add titles and then 'myplot'
doc = addPlot( doc, function( ) print( myplot ) )

################ FLEXTABLE DEMO ################
doc = addSlide( doc, slide.layout = "Title and Content" )
doc = addTitle( doc, "FlexTable example" )

# set default font size to 10
options( "ReporteRs-fontsize" = 10 )

# Create a FlexTable with data.frame mtcars, display rownames
# use different formatting properties for header and body
MyFTable = FlexTable( data = mtcars, add.rownames = TRUE,
header.cell.props = cellProperties( background.color = "#00557F" ),
header.text.props = textProperties( color = "white",
font.size = 11, font.weight = "bold" ),
body.text.props = textProperties( font.size = 10 )
)
# zebra stripes - alternate colored backgrounds on table rows
MyFTable = setZebraStyle( MyFTable, odd = "#E1EEf4", even = "white" )

# applies a border grid on table
MyFTable = setFlexTableBorders(MyFTable,
inner.vertical = borderProperties( color="#0070A8", style="solid" ),
inner.horizontal = borderNone(),
outer.vertical = borderProperties( color = "#006699", style = "solid", width = 2 ),
outer.horizontal = borderProperties( color = "#006699", style = "solid", width = 2 )
)

# add MyFTable into document
doc = addFlexTable( doc, MyFTable )

# write the doc
writeDoc( doc, file = pptx.file )
```

`print.bsdoc` *print informations about an object of class* bsdoc.

Description

print informations about an object of class bsdoc.

Usage

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

Arguments

x	an object of class bsdoc
...	further arguments, not used.

See Also

bsdoc, print

Examples

```
# Create a new document
doc = bsdoc( )
print( doc )
```

`print.docx` *print informations about an object of class* docx.

Description

print informations about an object of class docx.

Usage

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

Arguments

x	an object of class docx
...	further arguments, not used.

See Also

docx, print

Examples

```
# Create a new document
doc = docx( title = "title" )
print( doc )
```

print.pptx *print informations about an object of class pptx.*

Description

print informations about an object of class pptx.

Usage

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

Arguments

x	an object of class pptx
...	further arguments, not used.

See Also

pptx, print

Examples

```
# Create a new document
doc = pptx( title = "title" )
print( doc )
```

print.textProperties *print formatting properties*

Description

print text formatting properties (an object of class "textProperties").

Usage

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

Arguments

x	an object of class "textProperties"
...	further arguments, not used.

See Also

textProperties

Examples

```
print( textProperties (color="red", font.size = 12) )
```

---

raphael.html *get HTML code from a plot*

Description

get HTML code from a plot

Usage

```
raphael.html(fun, pointsize = getOption("ReporteRs-fontsize"), width = 6,
  height = 6, fontname = getOption("ReporteRs-default-font"),
  canvas_id = 0, par.properties = parCenter(padding = 5), ...)
```

Arguments

fun	plot function
width	plot width in inches (default value is 6).
height	plot height in inches (default value is 6).
pointsize	the default pointsize of plotted text in points, default to 12.
fontname	the default font family to use, default to getOption("ReporteRs-default-font").
canvas_id	canvas id - an integer - unique id in the web page
par.properties	paragraph formatting properties of the paragraph that contains images. An object of class parProperties
...	arguments for fun.

Value

an html string.

See Also

bsdoc, addPlot, add.plot.interactivity , addPlot.bsdoc

Examples

```
# load ggplot2
require( ggplot2 )

# create a ggplot2 plot
myplot = qplot(Sepal.Length, Petal.Length, data = iris
, color = Species, size = Petal.Width, alpha = I(0.7) )

raphael.html( fun = function( ){
plot( x = rnorm( 100 ), y = rnorm (100 ), main = "base plot main title" )
print( myplot )
}
, width = 5, height = 7
)
```

`registerRaphaelGraph` *register Raphael plots*

Description

register Raphael plots - internal use only

Usage

```
registerRaphaelGraph(plot_attributes, env)
```

Arguments

`plot_attributes`	plot attributes
`env`	environment

`RScript` *RScript object*

Description

Colored RScript object

Usage

```
RScript(file, text, comment.properties = textProperties(color = "#A7947D"),
  roxygencomment.properties = textProperties(color = "#5FB0B8"),
  symbol.properties = textProperties(color = "black"),
  operators.properties = textProperties(color = "black"),
  keyword.properties = textProperties(color = "#4A444D"),
  string.properties = textProperties(color = "#008B8B", font.style =
  "italic"), number.properties = textProperties(color = "blue"),
  functioncall.properties = textProperties(color = "blue"),
  argument.properties = textProperties(color = "#666666"),
  package.properties = textProperties(color = "green"),
  formalargs.properties = textProperties(color = "#424242"),
  eqformalargs.properties = textProperties(color = "#424242"),
  assignement.properties = textProperties(color = "black"),
  slot.properties = textProperties(color = "#F25774"),
  default.properties = textProperties(color = "black"),
  par.properties = parProperties())
```

Arguments

file	R script file. Not used if text is provided.
text	character vector. The text to parse. Not used if file is provided.
comment.properties	comment txtProperties object
roxygencomment.properties	roxygencomment txtProperties object
operators.properties	operators txtProperties object
keyword.properties	keyword txtProperties object
string.properties	string txtProperties object
number.properties	number txtProperties object
functioncall.properties	functioncall txtProperties object
argument.properties	argument txtProperties object
package.properties	package txtProperties object
formalargs.properties	formalargs txtProperties object
eqformalargs.properties	eqformalargs txtProperties object
assignement.properties	assignement txtProperties object

`symbol.properties`
: symbol txtProperties object

`slot.properties`
: slot txtProperties object

`default.properties`
: default txtProperties object

`par.properties`
: a parProperties object

See Also

addRScript

Examples

```
an_rscript = RScript( text = "ls()
x = rnorm(10)" )
```

---

setColumnsColors *applies background colors to columns of a FlexTable*

Description

applies background colors to columns of a FlexTable

Usage

```
setColumnsColors(object, j, colors)
```

Arguments

`object`	a `FlexTable` object
`j`	vector (integer index, col.names values or boolean vector) for columns selection.
`colors`	background colors to apply (e.g. "#000000" or "black")

See Also

setRowsColors, FlexTable, setZebraStyle

Examples

```
# Create a FlexTable object with first 10 lines of data.frame mtcars
# add row.names as first column
MyFTable = FlexTable( data = mtcars[1:10, ]
  , add.rownames=TRUE
)
MyFTable = setColumnsColors( MyFTable, j=3:4, colors = "red" )
```

setFlexTableBackgroundColors *applies background colors to cells of a FlexTable*

Description

applies background colors to cells of a FlexTable

Usage

```
setFlexTableBackgroundColors(object, i, j, colors, to = "body")
```

Arguments

object	a FlexTable object
i	vector (integer index, row.names values or boolean vector) for rows selection.
j	vector (integer index, col.names values or boolean vector) for columns selection.
colors	background colors to apply (e.g. "#000000" or "black"). a character vector of colors with as many elements as defined by the selection.
to	specify on which part of the FlexTable to apply colors, must be one of the following values "body" (default) or "header" or "footer"

See Also

FlexTable, is.color

Examples

```
data = cor( cor(mtcars) )

pal = c( "#D73027", "#F46D43", "#FDAE61", "#FEE08B",
"#D9EF8B", "#A6D96A", "#66BD63", "#1A9850" )
mycut = cut( data,
breaks = c(-1,-0.75,-0.5,-0.25,0,0.25,0.5,0.75,1),
include.lowest = TRUE, label = FALSE )
mycolors = pal[ mycut ]

MyFTable = FlexTable( round(data, 3), add.rownames = TRUE )

# set computed colors
MyFTable = setFlexTableBackgroundColors( MyFTable,
j = seq_len(ncol(data)) + 1,
colors = mycolors )

# cosmetics
MyFTable = setFlexTableBackgroundColors( MyFTable, i = 1,
colors = "gray", to = "header" )
```

```
MyFTable[1, , to = "header"] = textBold(color="white")

MyFTable = setFlexTableBackgroundColors( MyFTable, j = 1, colors = "gray" )
MyFTable[,1] = textBold(color="white")

MyFTable = setFlexTableBorders( MyFTable
, inner.vertical = borderProperties( style = "dashed", color = "white" )
, inner.horizontal = borderProperties( style = "dashed", color = "white"  )
, outer.vertical = borderProperties( width = 2, color = "white"  )
, outer.horizontal = borderProperties( width = 2, color = "white"  )
)
```

`setFlexTableBorders` *change grid lines of a FlexTable*

Description

apply borders scheme to a FlexTable. A border scheme is a set of 4 different borders: inner vectical and horizontal , outer vectical and horizontal.

Usage

```
setFlexTableBorders(object, inner.vertical = borderProperties(),
  inner.horizontal = borderProperties(),
  outer.vertical = borderProperties(),
  outer.horizontal = borderProperties(), body = TRUE, header = TRUE,
  footer = FALSE)
```

Arguments

`object`	a `FlexTable` object
`inner.horizontal`	a `borderProperties` object
`inner.vertical`	a `borderProperties` object
`outer.horizontal`	a `borderProperties` object
`outer.vertical`	a `borderProperties` object
`body`	a logical value (default to TRUE), specifies to apply scheme to table body
`header`	a logical value (default to TRUE), specifies to apply scheme to table header
`footer`	a logical value (default to FALSE), specifies to apply scheme to table footer

See Also

`FlexTable`

Examples

```
# Create a FlexTable object with first 10 lines of data.frame mtcars
# add row.names as first column
MyFTable = FlexTable( data = mtcars[1:10, ]
  , add.rownames=TRUE
)
MyFTable = setFlexTableBorders( MyFTable
  , inner.vertical = borderProperties( style = "dashed" )
  , inner.horizontal = borderProperties( style = "dashed" )
  , outer.vertical = borderProperties( width = 2 )
  , outer.horizontal = borderProperties( width = 2 )
)
```

---

setFlexTableWidths *set columns widths of a FlexTable*

Description

set columns widths of a FlexTable in inches.

Usage

```
setFlexTableWidths(object, widths)
```

Arguments

object	a FlexTable object
widths	a numeric vector specifying columns widths in inches.

See Also

FlexTable

Examples

```
# Create a FlexTable object with first 10 lines of data.frame iris
MyFTable = FlexTable( data = iris[1:10, ] )
MyFTable = setFlexTableWidths( MyFTable, widths = c(1,1,1,1,3))
```

setRowsColors *applies background colors to rows of a FlexTable*

Description

applies background colors to rows of a FlexTable

Usage

```
setRowsColors(object, i, colors)
```

Arguments

object	a FlexTable object
i	vector (integer index, row.names values or boolean vector) for rows selection.
colors	background colors to apply (e.g. "#000000" or "black")

See Also

FlexTable, setColumnsColors, setZebraStyle

Examples

```
# Create a FlexTable object with first 10 lines of data.frame mtcars
# add row.names as first column
MyFTable = FlexTable( data = mtcars[1:10, ]
  , add.rownames=TRUE
)
MyFTable = setRowsColors( MyFTable, i=1:4, colors = "red" )
```

setZebraStyle *FlexTable rows zebra striping*

Description

applies background color to alternate rows (zebra striping). Set a color if row index is odd and another if row index is even.

Usage

```
setZebraStyle(object, odd, even)
```

Arguments

object	a FlexTable object
odd	background color applied to odd row indexes - single character value (e.g. "#000000" or "black")
even	background color applied to even row indexes - single character value (e.g. "#000000" or "black")

See Also

FlexTable

Examples

```
# Create a FlexTable object with first 10 lines of data.frame mtcars
# add row.names as first column
MyFTable = FlexTable( data = mtcars[1:10, ]
  , add.rownames=TRUE
)
# Zebra striped table
MyFTable = setZebraStyle( MyFTable, odd = "#8A949B", even = "#FAFAFA" )
```

---

set_of_paragraphs *Set of paragraphs of text*

Description

Create a container of paragraphs of text (pot objects).

Usage

```
set_of_paragraphs(...)
```

Arguments

...	pot objects, one per paragraph.

Details

each pot are representing a paragraph. A paragraph consists of one or more pieces of text and ends with an end of line. Objects of class set_of_paragraphs are to be used with addParagraph.

See Also

addParagraph, addParagraph.docx, addParagraph.pptx, addParagraph.bsdoc, pot

Examples

```
pot1 = pot("My tailor", textProperties(color="red") ) + " is " + pot("rich"
, textProperties(font.weight="bold") )
pot2 = pot("Cats", textProperties(color="red") ) + " and " + pot("Dogs"
, textProperties(color="blue") )
my.pars = set_of_paragraphs( pot1, pot2 )
```

---

slide.layouts *Get layout names of a document object*

Description

Get layout names that exist into a document

Usage

```
slide.layouts(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

slide.layouts only works with pptx documents. See slide.layouts.pptx for examples.

See Also

pptx, slide.layouts.pptx, addSlide.pptx

---

slide.layouts.pptx *Get layout names of a pptx document*

Description

Get layout names that exist into the template used when pptx has been created.

Usage

```
## S3 method for class 'pptx'
slide.layouts(doc, layout, ...)
```

Arguments

doc	Object of class pptx to extract layout names from.
layout	optional single string value, one of the layout names
...	further arguments, not used.

Details

Available names are layout names of the template document (e.g. Title and Content , Two Content, etc.). If layout is specified, the layout representation will be produced in a plot. This can be useful to check available shapes.

See Also

pptx, addSlide.pptx, slide.layouts

Examples

```
doc.filename = "addFlexTable_example.pptx"

# set default font size to 24
options( "ReporteRs-fontsize" = 24 )

doc = pptx( title = "title" )
# get layouts names
layouts = slide.layouts(doc)
layouts
# loop over layout names to plot each slide style
for(i in layouts ){
slide.layouts(doc, i )
title(sub = i )
if( interactive() ) readline(prompt = "show next slide layout")
}
```

---

spanFlexTableColumns *Span columns within rows*

Description

Span columns within rows.

Usage

```
spanFlexTableColumns(object, i, from, to, runs)
```

Arguments

object	a FlexTable object
i	vector (integer index, row.names values or boolean vector) for rows selection.
from	index of the first column to span (its content will be the visible one).
to	index of the last column to span.
runs	a vector of size numcol of FlexTable. If provided, successive runs of equal values will indicate to merge corresponding columns.

Note

Overlappings of horizontally merged cells and vertically merged cells are forbidden.

See Also

spanFlexTableRows, FlexTable

Examples

```
data(pbc_summary)

MyFTable = FlexTable( data = pbc_summary[, 1:4] )
# merge column 2 to 4 in line 3
MyFTable = spanFlexTableColumns( MyFTable, i = 3, from = 2, to = 4 )

# merge cells in rows 1 to 6 when successive values of runs are identical
MyFTable = spanFlexTableColumns( MyFTable, i = 4:6, runs = c( "a", "b", "b", "c") )
```

---

spanFlexTableRows *Span rows within columns*

Description

Span rows within columns.

Usage

```
spanFlexTableRows(object, j, from, to, runs)
```

Arguments

object	a FlexTable object
j	vector (integer index, col.names values or boolean vector) for columns selection.
from	index of the first row to span (its content will be the visible one).
to	index of the last row to span.
runs	a vector of size numrow of FlexTable. If provided, successive runs of equal values will indicate to merge corresponding rows.

Note

Overlappings of horizontally merged cells and vertically merged cells are forbidden.

See Also

FlexTable, spanFlexTableColumns

Examples

```
data(pbc_summary)

MyFTable = FlexTable( data = pbc_summary[, 1:4] )
# merge line 7 to 11 in column 1
MyFTable = spanFlexTableRows( MyFTable, j = 3, from = 5, to = 7 )
# merge cells in column 1 (trt) when successive values of trt are identical
MyFTable = spanFlexTableRows( MyFTable, j=1, runs = as.character( pbc_summary$trt ) )
# merge cells in column 2 (sex) when successive values of sex are identical
MyFTable = spanFlexTableRows( MyFTable, j=2, runs = as.character( pbc_summary$sex ) )
```

---

styles *Get styles names of a document object*

Description

Get styles names that exist into a document

Usage

```
styles(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

styles only works with docx documents.

See styles.docx for examples.

See Also

docx, styles.docx, addParagraph.docx

styles.docx *Get styles names of a docx document*

Description

Get styles names that exist into the template (base document).

Usage

```
## S3 method for class 'docx'
styles(doc, ...)
```

Arguments

doc	Object of class docx to extract style names from.
...	further arguments, not used.

Details

Available styles will be paragraph styles of the base document (e.g. Normal, Title1, etc.). Names of the returned character vector are labels associated with styles names.

See Also

docx, styles

Examples

```
# Create a new document
doc = docx( title = "title" )
styles(doc) #returns available paragraph styles in a character vector
```

textBold *shortcut for bold*

Description

shortcut for bold textProperties()

Usage

```
textBold(...)
```

Arguments

...	arguments passed to textProperties

See Also

textNormal , textItalic , textBoldItalic , parRight , parLeft , parCenter , parJustify , borderDotted , borderDashed , borderNone , borderSolid

Examples

```
textBold()
```

---

textBoldItalic *shortcut for bold italic*

Description

shortcut for bold italic textProperties()

Usage

```
textBoldItalic(...)
```

Arguments

...	arguments passed to textProperties

See Also

textNormal , textBold , textItalic , parRight , parLeft , parCenter , parJustify , borderDotted , borderDashed , borderNone , borderSolid

Examples

```
textBoldItalic()
```

textItalic *shortcut for italic*

Description

shortcut for italic textProperties()

Usage

```
textItalic(...)
```

Arguments

...	arguments passed to textProperties

See Also

textNormal , textBold , textBoldItalic , parRight , parLeft , parCenter , parJustify , borderDotted , borderDashed , borderNone , borderSolid

Examples

```
textItalic()
```

textNormal *shortcut for default textProperties*

Description

shortcut for textProperties(...)

Usage

```
textNormal(...)
```

Arguments

...	arguments passed to textProperties

See Also

textBold , textItalic , textBoldItalic , parRight , parLeft , parCenter , parJustify , borderDotted , borderDashed , borderNone , borderSolid

Examples

```
textNormal()
```

`textProperties` *Text formatting properties*

Description

Create a `textProperties` object that describes text formatting properties.

Usage

```
textProperties(color = "black", font.size = getOption("ReporteRs-fontsize"),
  font.weight = "normal", font.style = "normal", underlined = FALSE,
  font.family = getOption("ReporteRs-default-font"),
  vertical.align = "baseline", 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.
`font.weight`	single character value specifying font weight (expected value is `normal` or `bold`).
`font.style`	single character value specifying font style (expected value is `normal` or `italic`).
`underlined`	single logical value specifying if the font is underlined.
`font.family`	single character value specifying font name (it has to be an existing font in the OS).
`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").

Details

Default values are:

- `color` "black"
- `font.size` getOption("ReporteRs-fontsize")
- `font.weight` "normal"
- `font.style` "normal"
- `underlined` FALSE
- `font.family` getOption("ReporteRs-default-font")
- `vertical.align` "baseline"

Value

a `textProperties` object

See Also

chprop.textProperties, pot , alterFlexTable

Examples

```
textProperties( font.size = 12 )

textProperties(color="red",
  font.weight = "bold",
  font.style = "italic",
  underlined = TRUE,
  font.family = "Courier New"
)

textProperties( shading.color = "red" )
```

---

toc.options *Set TOC options for a document object*

Description

Set custom table of contents options for a document object

Usage

```
toc.options(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

toc.options only works with docx documents.

See toc.options.docx for examples.

Value

a document object

See Also

docx, addTOC.docx

toc.options.docx *Set TOC options*

Description

set options for custom table of contents of a docx object.

Usage

```
## S3 method for class 'docx'
toc.options(doc, list.separator, ...)
```

Arguments

doc	Object of class docx
list.separator	list separator (should be the same than in computer's regional settings)
...	further arguments passed to other methods - not used.

Details

This function is to be used if TOC cannot be built. It is occuring when list separator used when building the TOC is different from the list separator in your computer's regional settings.

see http://support.microsoft.com/kb/302865/EN-US

See Also

docx, addTOC.docx

Examples

```
doc = docx( title = "title" )
doc = toc.options( doc, list.separator = "," )
```

triggerPostCommand *trigger post plot commands*

Description

internal use only

Usage

```
triggerPostCommand(env)
```

Arguments

env	environment

---

vanilla.table *get a simple FlexTable from a dataset*

Description

get a simple FlexTable from a dataset

Usage

```
vanilla.table(dataset, double.format = "%0.3f")
```

Arguments

dataset	the data to use
double.format	format string for double column to format in the dataset. See argument fmt of sprintf.

See Also

FlexTable

Examples

```
vanilla.table( iris)
```

---

writeDoc *Write a document object*

Description

Write a document object into a file

Usage

```
writeDoc(doc, ...)
```

Arguments

doc	document object
...	further arguments passed to other methods

Details

See writeDoc.docx or writeDoc.pptx or writeDoc.bsdoc for examples.

Value

a document object

See Also

docx, writeDoc.docx , pptx, writeDoc.pptx , bsdoc, writeDoc.bsdoc

---

writeDoc.bsdoc *Write a* bsdoc *object in a html file*

Description

Write the bsdoc object in '.html' files located in a specified directory.

bootstrap files will be copied in the directory if directory does not exist.

Usage

```
## S3 method for class 'bsdoc'
writeDoc(doc, file, reset.dir = FALSE, ...)
```

Arguments

doc	bsdoc object that has to be written.
file	single character value, name of the html file to write.
reset.dir	logical defaut to FALSE. Used to specify if the directory containing the file to produced should be deleted first (if existing for example). Set to FALSE enable to produce several html files in the same directory.
...	further arguments, not used.

Value

the function a character vector containing generated html documents filenames.

See Also

bsdoc, writeDoc

Examples

```
doc.filename = "writeDoc_bsdoc/example.html"

# set default font size to 11
options( "ReporteRs-fontsize" = 11 )

doc = bsdoc( )

# Write the object
writeDoc( doc, file = doc.filename )
```

`writeDoc.docx` *Write a docx object in a docx file*

Description

Write the docx object in a '.docx' file.

Usage

```
## S3 method for class 'docx'
writeDoc(doc, file, ...)
```

Arguments

`doc`	Object of class docx that has to be written.
`file`	single character value, name of the file to write.
`...`	further arguments, not used.

See Also

docx, writeDoc

Examples

```
# Create a new document
doc = docx( title = "title" )

doc = addParagraph(doc, "Hello Word!", stylename = "Normal")

# Write the object in file "writeDoc_example.docx"
writeDoc( doc, "writeDoc_example.docx" )
```

writeDoc.pptx *Write a pptx object in a pptx file*

Description

Write the pptx object in a '.pptx' file.

Usage

```
## S3 method for class 'pptx'
writeDoc(doc, file, ...)
```

Arguments

doc	pptx object that has to be written.
file	single character value, name of the file to write.
...	further arguments, not used.

See Also

pptx, writeDoc

Examples

```
# Create a new document
doc = pptx( title = "title" )

# add a slide with layout "Title and Content"
doc = addSlide( doc, slide.layout = "Title and Content" )

# add a dummy text in the content shape
doc = addParagraph(doc, "Hello Word!")

# Write the object in file "writeDoc_example.pptx"
writeDoc( doc, "writeDoc_example.pptx" )
```

[<-.FlexRow *modify FlexRow content*

Description

add or replace FlexCell into a FlexRow object

Usage

```
## S3 replacement method for class 'FlexRow'
x[i] <- value
```

Arguments

x	the FlexRow object
i	a single integer value.
value	an object of class FlexCell

See Also

FlexTable, addFlexTable, FlexRow , addHeaderRow, addFooterRow

Examples

```
## example with FlexCell
headerRow = FlexRow()
headerRow[1] = FlexCell( "Column 1"
  , cell.properties = cellProperties(background.color="#527578")  )
headerRow[2] = FlexCell( "Column 2"
  , cell.properties = cellProperties(background.color="#527578")  )
```

[<-.FlexTable *alter FlexTable content and format*

Description

add text or format a FlexTable object.

Usage

```
## S3 replacement method for class 'FlexTable'
 x[i, j, text.properties, newpar = F, byrow = FALSE, to = "body", side = "top"] <- value
```

Arguments

x	the FlexTable object
i	vector (integer index, row.names values or boolean vector) for rows selection.
j	vector (integer index, col.names values or boolean vector) for columns selection. or an object of class textProperties.
text.properties	formatting properties (an object of class textProperties). Used only when value is a data.frame, a maxtrix or a vector. It will be used to format added text.

newpar	logical value specifying wether or not the content should be added as a new paragraph (therefore added on a new line).
byrow	logical. If FALSE (the default) content is added by columns , otherwise content is added by rows.
to	specify on which part of the FlexTable to apply the value, must be one of the following values "body" (default) or "header" or "footer"
side	used only when value is a borderProperties, specify on which side to apply the properties. It must be one of "bottom", "top", "left", "right".
value	see details.

Details

Use `ft_object[1:4, 2:3] <- value` to perform the operation on the body subset of of the FlexTable.

Use `ft_object[] <- value` to perform the operation on the whole part (body, header or footer) of the FlexTable.

Use `ft_object[1, 2, to = "header"] <- value` to perform the operation on the header subset of the FlexTable.

Use `ft_object[1, 2, , to = "footer"] <- value` to perform the operation on the footer subset of the FlexTable.

To **format content**, argument value (the right side of the <-) should be one of the following:

- *for table cells:* an object of class cellProperties
- *for paragraphs contained in table cells:* an object of class parProperties
- *for text contained in table cells:* an object of class textProperties
- *for borders of table cells:* an object of class borderProperties

To **add content**, there are two options:

- *option 1:* value should be a data.frame or a matrix or a vector with as many elements as defined by the selection.
- *option 2:* value is a pot object, its value will be added in all cells defined by the selection.

If dealing with borderProperties objects, use also argument side to specify on which side of cells to apply border properties.

See Also

FlexTable, borderProperties , cellProperties, parProperties , textProperties

Examples

```
# Create a FlexTable object with first 10 lines of data.frame mtcars
# add row.names as first column
MyFTable = FlexTable( data = mtcars[1:10, ]
  , add.rownames=TRUE
```

```
)
# modify the text formatting properties for the row.names column
MyFTable[ , 1] = textProperties( font.style="italic", font.size = 9)
# align text to right for the row.names column
MyFTable[ , 1] = parProperties( text.align = "right" )

# change cell formatting properties for various columns
MyFTable[ c(3,6:9), c( "mpg", "disp"
  , "hp", "drat", "wt", "qsec" ) ] = cellProperties( background.color="#CCCCCC")
# add text to elements of the column cyl
MyFTable[, "cyl", text.properties = textProperties(
  vertical.align="superscript", font.size = 9) ] = " miles/gallon"



data( iris )
iris = head( iris[, c(5, 1:4)] )

default_text = textProperties( font.size = 11 )
note_text = chprop(default_text, vertical.align = "superscript", color = "blue")

iris_ft = FlexTable( data = iris, header.columns = FALSE )
iris_ft = addHeaderRow( iris_ft, value = c("", "Measures" ), colspan = c( 4, 1 ) )
iris_ft = addHeaderRow( iris_ft, value = gsub( "\\.", " ", names( iris ) ) )
iris_ft[2, 2, newpar = TRUE ] = "Hi there"
iris_ft[2, 1, to="header"] = pot("* this is a note", note_text )


iris_ft = spanFlexTableRows( iris_ft, j = "Species", runs = as.character( iris$Species ) )
iris_ft = setFlexTableBorders( iris_ft,
  inner.vertical = borderProperties( style = "none" ),
  inner.horizontal = borderProperties( width = 1 ),
  outer.vertical = borderProperties( width = 0 ),
  outer.horizontal = borderProperties( width = 2 ),
  footer = TRUE
)
```

Index