Package 'reactablefmtr'

October 14, 2022

Type Package

Title Streamlined Table Styling and Formatting for Reactable

Version 2.0.0

Maintainer Kyle Cuilla <kyle.cuilla@gmail.com>

Description Provides various features to streamline and enhance the styling of interactive reactable tables with easy-to-use and highly-customizable functions and themes. Apply conditional formatting to cells with data bars, color scales, color tiles, and icon sets. Utilize custom table themes inspired by popular websites such and bootstrap themes. Apply sparkline line & bar charts (note this feature requires the 'dataui' package which can be downloaded from <https://github.com/timelyportfolio/dataui>). Increase the portability and reproducibility of reactable tables by embedding images from the web directly into cells. Save the final table output as a static image or interactive file.

URL https://kcuilla.github.io/reactablefmtr/, https://github.com/kcuilla/reactablefmtr

BugReports https://github.com/kcuilla/reactablefmtr/issues

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5.0), reactable (>= 0.2.0)

Imports dplyr, grDevices, htmltools, htmlwidgets (>= 1.5.3), magrittr, purrr, sass (>= 0.4.0), shiny, stats, stringr (>= 1.4.0), tippy (>= 0.1.0), tools, webshot

Suggests dataui (>= 0.0.1), MASS, scales

Additional_repositories https://kcuilla.github.io/drat/

RoxygenNote 7.1.1

Language en-US

NeedsCompilation no

Author Kyle Cuilla [aut, cre, cph],
Greg Lin [ctb],
June Choe [ctb],
Kent Russell [ctb]

Repository CRAN

Date/Publication 2022-03-16 07:50:02 UTC

R topics documented:

`add_icon_legend` *Add an icon legend to a reactable table*

Description

Use ‘add_icon_legend()‘ to place a legend below a reactable table. The legend can be used to display the icon set used within ‘icon_sets()‘. The legend can be aligned to the right or left of the table. Custom labels can be applied to the upper and lower bounds of the legend.

Usage

```
add_icon_legend(
  table = NULL,
  icon_set = NULL,
  show_labels = TRUE,
  labels = NULL,
  align = "right",
  title = NULL,
  footer = NULL,
  margin = NULL
)
```

Arguments

table	A reactable table.
icon_set	The icon set to be displayed in the legend. Options are "ski rating", "medals", and "batteries". Default is NULL.
show_labels	Logical. Show or hide the labels next to the legend. Default is TRUE.
labels	Assign label to each icon in the specified icon set. Number of labels must match the number of icons in the icon set. Default is NULL.
align	The horizontal alignment of the legend. Options are 'left' or 'right'. Default is 'right'.
title	The title above the legend. Default is NULL.
footer	The footer below the legend. Default is NULL.
margin	Use margin() to set the margin around the legend (top, right, bottom, left). Default is NULL.

Value

a function that adds a source below a reactable table.

Examples

```
## Not run:
## Create the reactable table and then pipe in the legend
library(dplyr)
data <- iris[10:29, ]
table <- reactable(data,
columns = list(Petal.Length = colDef(
cell = icon_sets(data, icon_set = "medals"))))

table %>%
  add_icon_legend(icon_set = "medals")

## The legend can be aligned to the left or right of the table
table %>%
  add_icon_legend(icon_set = "medals", align = "left")

## Add custom labels to each icon in the legend
table %>%
 add_icon_legend(icon_set = "medals", labels = c("Shortest Length","Avg Length","Longest Length"))

## Add a title and footer to the legend
table %>%
 add_icon_legend(icon_set = "medals", title = "Icon Legend Title", footer = "Icon Legend Footer")

## End(Not run)
```

add_legend *Add a legend to a reactable table*

Description

Use ‘add_legend()‘ to place a legend below a reactable table. The legend can be used to display the color scale of a color palette used within the table. Supply the name of the dataset used with ‘data‘ and the name of the column you would like to show a legend for with ‘col_name‘. By default, the colors within ‘colors‘ are the default color palette used in ‘color_tiles()‘ and ‘color_scales()‘, but can be modified to match the color palette used in the column of the reactable table. The number of bins for the legend can be changed to any number. By default, label bins are given. The labels under the bins can be formatted with ‘number_fmt‘ or hidden by setting ‘labels‘ to FALSE. Use ‘title‘ to place a title above the legend, and ‘footer‘ to place a footer below the legend. The legend can be aligned to either the bottom-left or bottom-right of the table.

Usage

```
add_legend(
  table,
  data = NULL,
  col_name = NULL,
  bins = 5,
  colors = NULL,
  bias = 1,
  labels = TRUE,
  number_fmt = NULL,
  title = NULL,
  footer = NULL,
  align = "right"
)
```

Arguments

table	A reactable table.
data	Dataset containing at least one numeric column.
col_name	The name of a column containing numeric data within the dataset.
bins	The number of bins for the legend. Default is 5.
colors	The color palette to be displayed in the legend. By default, the colors are shown to match the default colors used in ‘color_tiles()‘ and v‘color_scales()‘.
bias	A positive value that determines the spacing between multiple colors. A higher value spaces out the colors at the higher end more than a lower number. Default is 1.
labels	Logical. Show or hide the labels next to the legend. Default is TRUE.
number_fmt	Optionally format numbers using formats from the scales package. Default is NULL.

title	The title above the legend. Default is NULL.
footer	The footer below the legend. Default is NULL.
align	The horizontal alignment of the legend. Options are 'left' or 'right'. Default is 'right'.

Value

a function that adds a legend below a reactable table.

Examples

```
library(magrittr)
## Create the reactable table and then pipe in the legend
data <- iris[10:29, ]
table <- reactable(data,
columns = list(Sepal.Length = colDef(
cell = color_tiles(data))))

table %>%
  add_legend(data = data,
             col_name = "Sepal.Length")

## The legend can be aligned to either the left or right side
table %>%
  add_legend(data = data,
             col_name = "Sepal.Length",
             align = "left")

## Change the number of bins within the legend
table %>%
  add_legend(data = data,
             col_name = "Sepal.Length",
             bins = 9)

## Add a title and footer to the legend
table %>%
  add_legend(data = data,
             col_name = "Sepal.Length",
             title = "Sepal Length",
             footer = "measured in cm")

## If custom colors are used in the table, you can assign those to the legend as well
table <- reactable(data,
columns = list(Sepal.Length = colDef(
style = color_scales(data, colors = c("red","white","blue")))))

table %>%
  add_legend(data = data,
             col_name = "Sepal.Length",
             colors = c("red","white","blue"))
```

add_source *Add a source below a reactable table*

Description

Use ‘add_source()‘ to place a source below a reactable or reactablefmtr table. The same options that are present in ‘add_title()‘ and ‘add_subtitle()‘ are also available in ‘add_source()‘. The source can be aligned to the left, right, or center with the align option. The text properties of the source, such as the font size and font style can be customized. The background color of the source can also be adjusted as well as the margin around the source.

Usage

```
add_source(
  table = NULL,
  source = NULL,
  align = "left",
  font_color = "#000",
  font_size = 16,
  font_style = "normal",
  font_weight = "normal",
  text_decoration = NULL,
  text_transform = NULL,
  letter_spacing = NULL,
  word_spacing = NULL,
  text_shadow = NULL,
  background_color = "#FFFFFF",
  margin = NULL
)
```

Arguments

table	A reactable table.
source	A string to be displayed as the source.
align	The alignment of the source. Options are "left", "right", "center". Default is "left".
font_color	Color of the source text. Default is #000.
font_size	Numeric value representing the size of the font of the source (in px). Default is 16.
font_style	Style of the source font. Options are "normal" or "italic". Default is "normal".
font_weight	The font weight of the source. Options are "bold" or "normal". Default is "normal".
text_decoration	Add an underline, overline, or line-through source. Options are "underline", "overline", "underline overline", or "line-through". Default is NULL.

text_transform Specify how to capitalize the title. Options are "uppercase", "lowercase", or "capitalize". Default is NULL.

letter_spacing Numeric value that adjusts the horizontal spacing between letters. A number above 0 adds more spacing between letters, a number below 0 decreases the spacing. Default is NULL.

word_spacing Numeric value that adjusts the horizontal spacing between words. A number above 0 adds more spacing between words, a number below 0 decreases the spacing. Default is NULL.

text_shadow Apply a shadow around the title. See <https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow> for options. Default is NULL.

background_color Color of the source background. Default is #FFFFFF.

margin Use margin() to set the margin around the text (top, right, bottom, left). Default is NULL.

Value

a function that adds a source below a reactable table.

Examples

```
## Not run:
## Create the reactable table and then pipe in the source
table <- reactable(iris[10:29, ])

table %>%
  add_source("This is a source")

## Use options to adjust the style and position of the source
table %>%
  add_source("This is a source", font_style = "italic", font_color = "grey")

## End(Not run)
```

---

add_subtitle *Add a subtitle above a reactable table*

Description

Use ‘add_subtitle()‘ to place a subtitle above a reactable or reactablefmtr table. The same options that are present in ‘add_title()‘ and ‘add_source()‘ are also available in ‘add_subtitle()‘. The subtitle can be aligned to the left, right, or center with the align option. The text properties of the subtitle, such as the font size and font style can be customized. The background color of the subtitle can also be adjusted as well as the margin around the subtitle.

Usage

```
add_subtitle(
  table = NULL,
  subtitle = NULL,
  align = "left",
  font_color = "#000",
  font_size = 24,
  font_style = "normal",
  font_weight = "bold",
  text_decoration = NULL,
  text_transform = NULL,
  letter_spacing = NULL,
  word_spacing = NULL,
  text_shadow = NULL,
  background_color = "#FFFFFF",
  margin = NULL
)
```

Arguments

table	A reactable table.
subtitle	A string to be displayed as the subtitle.
align	The alignment of the subtitle. Options are "left", "right", "center". Default is "left".
font_color	Color of the subtitle text. Default is #000.
font_size	Numeric value representing the size of the font of the subtitle (in px). Default is 24.
font_style	Style of the subtitle font. Options are "normal" or "italic". Default is "normal".
font_weight	The font weight of the subtitle. Options are "bold" or "normal". Default is "bold".
text_decoration	Add an underline, overline, or line-through subtitle. Options are "underline", "overline", "underline overline", or "line-through". Default is NULL.
text_transform	Specify how to capitalize the title. Options are "uppercase", "lowercase", or "capitalize". Default is NULL.
letter_spacing	Numeric value that adjusts the horizontal spacing between letters. A number above 0 adds more spacing between letters, a number below 0 decreases the spacing. Default is NULL.
word_spacing	Numeric value that adjusts the horizontal spacing between words. A number above 0 adds more spacing between words, a number below 0 decreases the spacing. Default is NULL.
text_shadow	Apply a shadow around the title. See <https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow> for options. Default is NULL.
background_color	Color of the subtitle background. Default is #FFFFFF.

margin	Use margin() to set the margin around the text (top, right, bottom, left). Default is NULL.

Value

a function that adds a subtitle above a reactable table.

Examples

```
## Not run:
## Create the reactable table and then pipe in the subtitle
table <- reactable(iris[10:29, ])

table %>%
  add_subtitle("This is a subtitle")

## If a title proceeds a subtitle, the subtite will be placed below the title
table %>%
  add_title("This is a title") %>%
  add_subtitle("This is a subtitle")

## Use options to adjust the style and position of the subtitle
table %>%
  add_subtitle("This is a subtitle", align = "center", font_color = "red")

## End(Not run)
```

---

add_title *Add a title above a reactable table*

Description

Use ‘add_title()‘ to place a title above a reactable or reactablefmtr table. The title can be aligned to the left, right, or center with the align option. The text properties of the title, such as the font size and font style can be customized. The background color of the title can also be adjusted as well as the margin around the title.

Usage

```
add_title(
  table = NULL,
  title = NULL,
  align = "left",
  font_color = "#000",
  font_size = 32,
  font_style = "normal",
  font_weight = "bold",
  text_decoration = NULL,
  text_transform = NULL,
```

```
  letter_spacing = NULL,
  word_spacing = NULL,
  text_shadow = NULL,
  background_color = "#FFFFFF",
  margin = NULL
)
```

Arguments

table	A reactable table.
title	A string to be displayed as the title.
align	The alignment of the table. Options are "left", "right", "center". Default is "left".
font_color	Color of the title text. Default is #000.
font_size	Numeric value representing the size of the font of the title (in px). Default is 32.
font_style	Style of the title font. Options are "normal" or "italic". Default is "normal".
font_weight	The font weight of the title. Options are "bold" or "normal". Default is "bold".
text_decoration	Add an underline, overline, or line-through title. Default is NULL.
text_transform	Specify how to capitalize the title. Options are "uppercase", "lowercase", or "capitalize". Default is NULL.
letter_spacing	Numeric value that adjusts the horizontal spacing between letters. A number above 0 adds more spacing between letters, a number below 0 decreases the spacing. Default is NULL.
word_spacing	Numeric value that adjusts the horizontal spacing between words. A number above 0 adds more spacing between words, a number below 0 decreases the spacing. Default is NULL.
text_shadow	Apply a shadow around the title. See <https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow> for options. Default is NULL.
background_color	Color of the title background. Default is #FFFFFF.
margin	Use margin() to set the margin around the text (top, right, bottom, left). Default is NULL.

Value

a function that adds a title above a reactable table.

Examples

```
## Not run:
## Create the reactable table and then pipe in the title
table <- reactable(iris[10:29, ])

table %>%
  add_title("This is a title")
```

```
## Use options to adjust the style and position of the title
table %>%
  add_title("This is a title", align = "center", font_color = "red")

## End(Not run)
```

---

background_img *Embed a background image from web to rows of a reactable table*

Description

The ‘background_img()‘ function fills the background of a cell with an image from the web. The image must be provided with a http or https address. The difference between ‘background_img()‘ and ‘embed_img()‘ is the image in ‘background_img()‘ takes up the entire contents of a cell whereas ‘embed_img()‘ does not. ‘background_img()‘ can be used in conjunction with ‘embed_img()‘ and the image will be placed behind the image from ‘embed_img()‘. ‘background_img()‘ will also directly accept a singular image URL within ‘img‘ that does not come from the column itself. Additionally, images can be assigned from another column by providing the name of the column containing the img URL’s in quotes within ‘img_ref‘. ‘background_img()‘ should be placed within the style argument of reactable::colDef.

Usage

```
background_img(
  data,
  height = "100%",
  width = "100%",
  position = "center",
  img = NULL,
  img_ref = NULL
)
```

Arguments

data	Dataset containing URL’s to images
height	A value given for the height of the image. Can provide a percentage of the cell height or a value in px. Default height is 100 percent.
width	A value given for the width of the image. Can provide a percentage of the cell width or a value in px. Default width is 100 percent.
position	The alignment of the image within a cell. Options are "center", "top", "bottom", "left", "right". Default is "center".
img	Optionally provide a direct link to an image URL. Only one image can be provided using this option. Default is NULL.
img_ref	Optionally assign images from another column by referencing the name of the column in quotes. Default is NULL.

Value

a function that renders a background image to a column containing a valid web link.

Examples

```
## If no image links are in the original dataset, you need to assign them like so:
library(dplyr)
data <- iris %>%
 mutate(
 img = case_when(
 Species == "setosa" ~
 "https://upload.wikimedia.org/wikipedia/commons/d/d9/Wild_iris_flower_iris_setosa.jpg",
 Species == "versicolor" ~
 "https://upload.wikimedia.org/wikipedia/commons/7/7a/Iris_versicolor.jpg",
 Species == "virginica" ~
 "https://upload.wikimedia.org/wikipedia/commons/9/9f/Iris_virginica.jpg",
 TRUE ~ "NA"))

reactable(data,
 columns = list(
  img = colDef(style = background_img())))

## By default, images are given a size of 100% to fill the entire cell,
## but you can adjust the size using height and width:
reactable(data,
 columns = list(
  img = colDef(style = background_img(height = "50%", width = "50%"))))

## An image can be applied directly over an existing column using img:
reactable(data,
 columns = list(
  Species = colDef(
  style = background_img(
  img = "https://upload.wikimedia.org/wikipedia/commons/2/26/Colored_flowers_e.jpg"))))

## Conditionally assigned images can be applied directly over an existing column using img_ref:
reactable(data,
 columns = list(
  Species = colDef(style = background_img(data, img_ref = "img"))))
```

---

bubble_grid *Build a customizable bubble grid chart*

Description

The 'bubble_grid()' function creates a customizable bubble grid chart within a reactable table. The size of the bubbles are in relation to the values within each column - the bigger the value, the bigger the size of the bubble. There are two shapes available for the bubble grid: circles and squares,

which can be specified with ‘shape‘. The colors can be provided within a vector in ‘colors‘ or via another column in the dataset by referencing the column by name with ‘color_ref‘. If more than one color is provided in ‘colors‘, the colors will be assigned to the values from low to high within the palette. This is the default setting of ‘bubble_grid()‘, which applies a blue-to-orange color palette to the bubbles. However, a singular color can be provided instead if desired. ‘bubble_grid()‘ can be applied to columns containing character data by referencing another column with numeric values in it with ‘color_by‘. The opacity of the colors provided can be adjusted by providing a value between 0 and 1 in ‘opacity‘. ‘text_color‘ can be used to change the color of the values displayed within the bubbles. If values are displayed within a dark-colored background, ‘brighten_text‘ will display the values in white text so they are more visible. For smaller values with a dark-colored background, the values may not be visible. If you would like these numbers to be visible, you could do so by either: A) setting ‘brighten_text‘ to FALSE and assigning a universal color to the text within ‘text_color‘. B) leaving ‘brighten_text‘ as TRUE and setting ‘brighten_text_color‘ to a darker color other than the default white color. If the user wants to assign colors row-wise instead of column-wise, set ‘span‘ equal to TRUE to apply across all columns. Or can provide the names of the columns by either column name or column position number to apply to only a subset of the columns. The format of the numbers within the bubbles can be changed by defining the format from a package such as the scales package within ‘number_fmt‘. ‘bubble_grid()‘ needs to placed within the cell argument in reactable::colDef.

Usage

```
bubble_grid(
  data,
  shape = "circles",
  colors = c("#15607A", "#FFFFFF", "#FA8C00"),
  color_ref = NULL,
  color_by = NULL,
  min_value = NULL,
  max_value = NULL,
  opacity = 1,
  bias = 1,
  number_fmt = NULL,
  text_size = NULL,
  text_color = "black",
  text_color_ref = NULL,
  show_text = TRUE,
  brighten_text = TRUE,
  brighten_text_color = "white",
  bold_text = FALSE,
  span = FALSE,
  box_shadow = FALSE,
  tooltip = FALSE,
  animation = "background 1s ease"
)
```

Arguments

data	Dataset containing at least one numeric column.

shape	The shape of the bubbles. Options are 'circles' or 'squares'. Default is 'circles'.
colors	A vector of colors to color the bubbles. Colors should be given in order from low values to high values. Default colors provided are blue-white-orange: c("#15607A", "#FFFFFF", "#FA8C00"). Can use R's built-in colors or other color packages.
color_ref	Optionally assign colors to from another column by providing the name of the column containing the colors in quotes. Only one color can be provided per row. Default is NULL.
color_by	Assign colors to a column based on the values of another column. The column in reference must contain numeric data. The column in which the colors are being assigned to can be either numerical or character. Default is NULL.
min_value	A value to use as the minimum value for the size of the bubbles. Default is NULL.
max_value	A value to use as the maximum value for the size of the bubbles. The default maximum value is the maximum value in the column. Default is NULL.
opacity	A value between 0 and 1 that adjusts the opacity in colors. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
bias	A positive value that determines the spacing between multiple colors. A higher value spaces out the colors at the higher end more than a lower number. Default is 1.
number_fmt	Optionally format numbers using formats from the scales package. Default is NULL.
text_size	Numeric value representing the size of the text labels. Default is NULL.
text_color	Assigns text color to values. Default is black.
text_color_ref	Optionally assign text color from another column by providing the name of the column containing the text colors in quotes. Only one color can be provided per cell. Default is NULL.
show_text	Logical: show text or hide text. Default is TRUE.
brighten_text	Logical: automatically assign color to text based on background color of cell. Text within dark-colored backgrounds will turn white, text within light-colored backgrounds will be black. Default is TRUE.
brighten_text_color	Assigns text color to values if values are within a dark-colored backgrounds. Default is white.
bold_text	Logical: bold text. Default is FALSE.
span	Optionally apply colors to values across multiple columns instead of by each column. To apply across all columns set to TRUE. If applying to a set of columns, can provide either column names or column positions. Default is FALSE.
box_shadow	Logical: add a box shadow to the tiles. Default is FALSE.
tooltip	Logical: hover tooltip. Default is FALSE.
animation	Control the duration and timing function of the animation when sorting/updating values shown on a page. See [CSS transitions](https://developer.mozilla.org/en-US/docs/Web/CSS/transition) for available timing functions and examples. Animation can be turned off by setting to "none". Animation can be applied to the size of the bubbles by setting it to "all 1s ease". Default is "background 1s ease".

Value

a function that builds a bubble grid chart to a column of values.

Examples

```
data <- iris[10:29, ]

## By default, the bubble_grid() function uses a blue-white-orange three-color pattern:
reactable(
  data,
  columns = list(
    Petal.Length = colDef(
      align = "center",
      cell = bubble_grid(data))))

## You can specify your own color palette or a single color across all values with `colors`;
reactable(
  data,
  columns = list(
    Petal.Length = colDef(
      align = "center",
      cell = bubble_grid(data,
                         colors = c("orange")))))

## Use squares instead of circles:
reactable(
  data,
  columns = list(
    Petal.Length = colDef(
      align = "center",
      cell = bubble_grid(data,
                         shape = "squares"))))

## Hide text and show on hover by enabling the tooltip:
reactable(
  data,
  columns = list(
    Petal.Length = colDef(
      align = "center",
      cell = bubble_grid(data,
                         show_text = FALSE,
                         tooltip = TRUE))))

## Control the scale of the circles by adjusting the min and max values:
reactable(
  data,
  columns = list(
    Petal.Length = colDef(
      align = "center",
      cell = bubble_grid(data,
                         min_value = 1,
                         max_value = 2))))
```

```
## Use span to apply bubbles to values in relation to the entire data set:
reactable(
  data,
  defaultColDef = colDef(
    cell = bubble_grid(data,
                       span = TRUE)))

## Use number_fmt to format numbers using the scales package:
car_prices <- MASS::Cars93[20:49, c("Make", "Price")]

reactable(car_prices,
  defaultColDef = colDef(
    align = "center",
    cell = bubble_grid(car_prices,
                       number_fmt = scales::dollar)))
```

`cell_style` *Add custom styles to cells*

Description

Use 'cell_style()' to customize the appearance of certain cells in a reactable or reactablefmtr table. Assign custom styles by either row number(s) or by values within a particular column. The font color, font size, font style, and font weight can all be modified. Borders can also be placed around cells and customized by style, width, and color. By default, animation is applied to the cells that are styled, but can be turned off by setting to 'none'. Some options within 'cell_style()' will work with other reactablefmtr formatters (such as data_bars() and color_tiles()), but it is not fully supported and should be used separately, not together. 'cell_style()' needs to be placed within the style argument of reactable::colDef.

Usage

```
cell_style(
  data,
  rows = NULL,
  values = NULL,
  font_color = NULL,
  font_size = NULL,
  font_style = "normal",
  font_weight = "normal",
  horizontal_align = "right",
  vertical_align = "top",
  text_decoration = NULL,
  border_width = NULL,
  border_style = NULL,
  border_color = NULL,
  background_color = NULL,
```

```
  animation = "1s ease"
)
```

Arguments

data	A dataset to be displayed within a reactable table.
rows	Numeric value representing the row number to apply the custom style. Can provide a vector of rows if applying to more than one row. If no rows are provided, styles are applied to all rows/values.
values	A value, either numeric or character, that is present within a column. Can provide a vector of values if applying to more than one value. If no values are provided, styles are applied to all rows/values.
font_color	Color of the text.
font_size	Numeric value representing the size of the font of the text (in px). Default is 16.
font_style	Style of the text font. Options are "normal" or "italic". Default is "normal".
font_weight	The font weight of the text Options are "normal", "bold", "bolder", "lighter" or a value between 100 and 900. Default is "normal".
horizontal_align	The horizontal alignment of the text within a cell. Options are "left", "right", or "center". Default is "right".
vertical_align	The vertical alignment of the text within a cell. Options are "top", "bottom", or "center". Default is "top".
text_decoration	Optionally add an underline, overline, or line-through to the text Options are "underline", "overline", "underline overline", or "line-through". Default is NULL.
border_width	The width of the border around the cell. Options are "thin", "medium", "thick", or a numeric value such as "2px". May be specified using one, two, three, or four values. See [CSS border-width](https://developer.mozilla.org/en-US/docs/Web/CSS/border-width) for more options.
border_style	The style of the border around the cell. Options are "solid", "dashed", "dotted", "double", "groove", "ridge", "inset", "outset", "none", or "hidden". May be specified using one, two, three, or four values. See [CSS border-style](https://developer.mozilla.org/en-US/docs/Web/CSS/border-style) for more options.
border_color	The color of the border around the cell. May be specified using one, two, three, or four values. See [CSS border-color](https://developer.mozilla.org/en-US/docs/Web/CSS/border-color) for more options.
background_color	Color of the background of the cell.
animation	Control the duration and timing function of the animation when sorting/updating values shown on a page. See [CSS transitions](https://developer.mozilla.org/en-US/docs/Web/CSS/transition) for available timing functions and examples. Animation can be turned off by setting to "none". Default is "1s ease".

Value

a function that adds a custom style to a row or rows in a reactable table.

Examples

```
## Not run:
## Add a dotted blue border around the third row in Sepal.Length
data <- iris[10:29, ]
reactable(data,
         columns = list(
           Sepal.Length = colDef(
             style = cell_style(data,
                                rows = 3,
                                border_width = "thick",
                                border_color = "blue",
                                border_style = "dotted"))))

## For all setosa species, highlight cell yellow and assign red font color
data <- iris[10:100, ]
reactable(data,
         columns = list(
         Species = colDef(
             style = cell_style(data,
                                values = "setosa",
                                font_color = "red",
                                background_color = "yellow"))))

## End(Not run)
```

`cerulean` *Theme cerulean*

Description

Bootstrap-inspired cerulean theme

Usage

```
cerulean(
  font_size = 14,
  font_color = "#141415",
  header_font_size = 15,
  header_font_color = "#cfe9f7",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

`font_size`	Numeric value representing the size of the font within the table (in px). Default is 14.
`font_color`	Color of the font for the text within the table and the group headers. Default is #141415.

header_font_size
: Numeric value representing the size of the font within the table (in px). Default is 15.

header_font_color
: Color of the font for the header text. Default is #cfe9f7.

cell_padding
: Numeric value representing the padding size between cells (in px). Default is 6.

centered
: Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard cerulean theme
reactable(data,
          theme = cerulean())

## Additional options applied
reactable(data,
          theme = cerulean(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

clean *Theme clean*

Description

Simple clean-look theme

Usage

```
clean(
  font_size = 14,
  font_color = "#222222",
  header_font_size = 15,
  header_font_color = "#222222",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table. Default is #222222.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #222222.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard clean theme
reactable(data,
          theme = clean())

## Additional options applied
reactable(data,
          theme = clean(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

color_scales *Add color scales to cells in a column*

Description

The ‘color_scales()‘ function conditionally colors each cell of a column depending on their value in relation to other values in that particular column. The colors can be provided within a vector in ‘colors‘ or via another column in the dataset by referencing the column by name with ‘color_ref‘. The opacity of the colors provided can be adjusted by providing a value between 0 and 1 in ‘opacity‘. ‘text_color‘ can be used to change the color of the values. If values are displayed within a dark-colored background, ‘brighten_text‘ will display the values in white text so they are more visible. The color of ‘brighten_text_color‘ can be changed to a color other than white if desired. If the user wants to assign colors row-wise instead of column-wise, set ‘span‘ equal to TRUE to apply across all columns. Or can provide the names of the columns by either column name or column position number to apply to only a subset of the columns. ‘color_scales()‘ should be placed within the style argument in reactable::colDef.

Usage

```
color_scales(
  data,
  colors = c("#15607A", "#FFFFFF", "#FA8C00"),
  color_ref = NULL,
  color_by = NULL,
  opacity = 1,
  bias = 1,
  text_size = NULL,
  text_color = "black",
  text_color_ref = NULL,
  show_text = TRUE,
  brighten_text = TRUE,
  brighten_text_color = "white",
  bold_text = FALSE,
  span = FALSE,
  animation = "background 1s ease"
)
```

Arguments

data	Dataset containing at least one numeric column.
colors	A vector of colors to color the cells. Colors should be given in order from low values to high values. Default colors provided are blue-white-orange: c("#15607A", "#FFFFFF", "#FA8C00"). Can use R's built-in colors or other color packages.
color_ref	Assign colors from another column that contains the colors for each row. Only one color can be provided per row. Default is NULL.
color_by	Assign colors to a column based on the values of another column. The column in reference must contain numeric data. The column in which the colors are being assigned to can be either numerical or character. Default is NULL.
opacity	A value between 0 and 1 that adjusts the opacity in colors. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
bias	A positive value that determines the spacing between multiple colors. A higher value spaces out the colors at the higher end more than a lower number. Default is 1.
text_size	Numeric value representing the size of the text labels. Default is NULL.
text_color	Assigns text color to values. Default is black.
text_color_ref	Assign text color from another column by providing the name of the column containing the text colors in quotes. Only one color can be provided per cell. Default is NULL.
show_text	Logical: show text or hide text. Default is TRUE.
brighten_text	Logical: automatically assign color to text based on background color of cell. Text within dark-colored backgrounds will turn white, text within light-colored backgrounds will be black. Default is TRUE.

brighten_text_color
: Assigns text color to values if values are within a dark-colored backgrounds. Default is white.

bold_text
: Logical: bold text. Default is FALSE.

span
: Optionally apply colors to values across multiple columns instead of by each column. To apply across all columns set to TRUE. If applying to a set of columns, can provide either column names or column positions. Default is set to FALSE.

animation
: Control the duration and timing function of the animation when sorting/updating values shown on a page. See [CSS transitions](https://developer.mozilla.org/en-US/docs/Web/CSS/transition) for available timing functions and examples. Animation can be turned off by setting to "none". Default is "background 1s ease".

Value

a function that applies conditional colors to a column of numeric values.

Examples

```
data <- iris[10:29, ]

## By default, the colors_scales() function uses a blue-white-orange three-color pattern
reactable(data,
 columns = list(
 Petal.Length = colDef(style = color_scales(data))))

## If only two colors are desired,
## you can specify them with colors = 'c(color1, color2)';
reactable(data,
 columns = list(
 Petal.Length = colDef(style = color_scales(data,
 colors = c("red", "green")))))

## Apply color_scales() across all numeric columns using reactable::defaultColDef
reactable(data,
defaultColDef = colDef(style = color_scales(data)))

## Use span to apply colors to values in relation to the entire dataset
reactable(data,
defaultColDef = colDef(style = color_scales(data, span = TRUE)))

## Span can take column names
reactable(data,
defaultColDef = colDef(style = color_scales(data, span = c("Sepal.Length", "Sepal.Width"))))

## Or it can also take column positions instead
reactable(data,
defaultColDef = colDef(style = color_scales(data, span = 1:2)))
```

color_tiles *Add color tiles to cells in a column*

Description

The ‘color_tiles()‘ function conditionally colors the background of each cell similarly to color_scales(). The difference is that color_tiles() uses round colored tiles around values instead of the entire background of the cell. Another difference is color_tiles() allows number formatting with number_fmt whereas color_scales() does not. The colors can be provided within a vector in ‘colors‘ or via another column in the dataset by referencing the column by name with ‘color_ref‘. The opacity of the colors provided can be adjusted by providing a value between 0 and 1 in ‘opacity‘. ‘text_color‘ can be used to change the color of the values. If values are displayed within a dark-colored background, ‘brighten_text‘ will display the values in white text so they are more visible. The color of ‘brighten_text_color‘ can be changed to a color other than white if desired. If the user wants to assign colors row-wise instead of column-wise, set ‘span‘ equal to TRUE to apply across all columns. Or can provide the names of the columns by either column name or column position number to apply to only a subset of the columns. ‘color_tiles()‘ needs to placed within the cell argument in reactable::colDef.

Usage

```
color_tiles(
  data,
  colors = c("#15607A", "#FFFFFF", "#FA8C00"),
  color_ref = NULL,
  color_by = NULL,
  opacity = 1,
  bias = 1,
  number_fmt = NULL,
  text_size = NULL,
  text_color = "black",
  text_color_ref = NULL,
  show_text = TRUE,
  brighten_text = TRUE,
  brighten_text_color = "white",
  bold_text = FALSE,
  span = FALSE,
  box_shadow = FALSE,
  tooltip = FALSE,
  animation = "background 1s ease"
)
```

Arguments

data	Dataset containing at least one numeric column.

colors	A vector of colors to color the cells. Colors should be given in order from low values to high values. Default colors provided are blue-white-orange: c("#15607A", "#FFFFFF", "#FA8C00"). Can use R's built-in colors or other color packages.
color_ref	Optionally assign colors to from another column by providing the name of the column containing the colors in quotes. Only one color can be provided per row. Default is NULL.
color_by	Assign colors to a column based on the values of another column. The column in reference must contain numeric data. The column in which the colors are being assigned to can be either numerical or character. Default is NULL.
opacity	A value between 0 and 1 that adjusts the opacity in colors. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
bias	A positive value that determines the spacing between multiple colors. A higher value spaces out the colors at the higher end more than a lower number. Default is 1.
number_fmt	Optionally format numbers using formats from the scales package. Default is NULL.
text_size	Numeric value representing the size of the text labels. Default is NULL.
text_color	Assigns text color to values. Default is black.
text_color_ref	Optionally assign text color from another column by providing the name of the column containing the text colors in quotes. Only one color can be provided per cell. Default is NULL.
show_text	Logical: show text or hide text. Default is TRUE.
brighten_text	Logical: automatically assign color to text based on background color of cell. Text within dark-colored backgrounds will turn white, text within light-colored backgrounds will be black. Default is TRUE.
brighten_text_color	Assigns text color to values if values are within a dark-colored backgrounds. Default is white.
bold_text	Logical: bold text. Default is FALSE.
span	Optionally apply colors to values across multiple columns instead of by each column. To apply across all columns set to TRUE. If applying to a set of columns, can provide either column names or column positions. Default is FALSE.
box_shadow	Logical: add a box shadow to the tiles. Default is FALSE.
tooltip	Logical: hover tooltip. Default is FALSE.
animation	Control the duration and timing function of the animation when sorting/updating values shown on a page. See [CSS transitions](https://developer.mozilla.org/en-US/docs/Web/CSS/transition) for available timing functions and examples. Animation can be turned off by setting to "none". Default is "background 1s ease".

Value

a function that applies conditional color tiles to a column of numeric values.

Examples

```
data <- iris[10:29, ]

## By default, the colors_tiles() function uses a blue-white-orange three-color pattern
reactable(data,
 columns = list(
 Petal.Length = colDef(cell = color_tiles(data))))

## If only two colors are desired,
## you can specify them with colors = 'c(color1, color2)';
reactable(data,
 columns = list(
 Petal.Length = colDef(cell = color_tiles(data,
 colors = c("red", "green")))))

## Use span to apply colors to values in relation to the entire dataset
reactable(data,
defaultColDef = colDef(cell = color_tiles(data, span = TRUE)))

## Span can take column names
reactable(data,
defaultColDef = colDef(cell = color_tiles(data, span = c("Sepal.Length", "Sepal.Width"))))

## Or it can also take column positions instead
reactable(data,
defaultColDef = colDef(cell = color_tiles(data, span = 1:2)))

## Use number_fmt to format numbers using the scales package
car_prices <- MASS::Cars93[20:49, c("Make", "Price")]

reactable(car_prices,
defaultColDef = colDef(cell = color_tiles(car_prices,
number_fmt = scales::dollar)))
```

`cosmo` *Theme cosmo*

Description

Bootstrap-inspired cosmo theme

Usage

```
cosmo(
  font_size = 14,
  font_color = "#141415",
  header_font_size = 15,
  header_font_color = "#ffffff",
  cell_padding = 6,
```

```
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table and the group headers. Default is #141415.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #ffffff.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard cosmo theme
reactable(data,
          theme = cosmo())

## Additional options applied
reactable(data,
          theme = cosmo(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

cyborg *Theme cyborg*

Description

Bootstrap-inspired cyborg theme

Usage

```
cyborg(
  font_size = 14,
  font_color = "#888888",
  header_font_size = 15,
```

```
  header_font_color = "#7b7b7b",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table and the group headers. Default is #888888.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #7b7b7b.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard cyborg theme
reactable(data,
          theme = cyborg())

## Additional options applied
reactable(data,
          theme = cyborg(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

dark *Theme dark*

Description

dark table theme

Usage

```
dark(
  font_size = 15,
  font_color = "#FFFFFF",
  header_font_size = 16,
  header_font_color = "#FFFFFF",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
font_color	Color of the font for the text within the table and the group headers. Default is #FFFFFF.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 16.
header_font_color	Color of the font for the header text. Default is #FFFFFF.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard dark theme
reactable(data,
          theme = dark())

## Additional options applied
reactable(data,
          theme = dark(font_size = 12, font_color = "red", cell_padding = 3))
```

darkly *Theme darkly*

Description

Bootstrap-inspired darkly theme

Usage

```
darkly(
  font_size = 14,
  font_color = "#ffffff",
  header_font_size = 15,
  header_font_color = "#afbdcc",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table and the group headers. Default is #ffffff.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #afbdcc.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard darkly theme
reactable(data,
          theme = darkly())

## Additional options applied
reactable(data,
          theme = darkly(font_size = 12, font_color = "grey", cell_padding = 3))
```

data_bars *Add horizontal bars to cells in a column*

Description

The ‘data_bars()‘ function adds a horizontal bar to each row of a column. The length of the bars are relative to the value of the row in relation to other values within the same column. The maximum width of the filled bars can be adjusted. Ex. if you are displaying percentages, and the maximum value in your column is 50 you could increase the maximum fill to 100 The values for the bars can be displayed inside or outside the filled bars with the ‘text_position‘ option. By default, the values are displayed on the outside-end of the filled bars. The fill_color of both the fill and the background of the bars can be adjusted. To adjust the fill_color of the filled bar, use ‘fill_color‘. If more than one color is provided, a conditional color palette will be applied to to the values, or if ‘fill_gradient‘ is set to TRUE, a left-to-right gradient fill color will be applied. The fill colors can also be provided via another column in the dataset by referencing the column by name with ‘fill_color_ref‘. ‘text_color‘ can be used to change the color of the text_position. By default, the label color is black. If values are displayed inside the bars and a dark color palette is used to fill the bars, ‘brighten_text‘ will display the values in white text so the values are visible by default. The color of ‘brighten_text_color‘ can be changed to a color other than white if desired. The border around the filled bars can be controlled via three different border options: ‘border_style‘, ‘border_width‘, and ‘border_color‘. An icon or image can be added to the data bars with ‘icon‘ or ‘img‘. Alternatively, icons and images can be assigned from another column with ‘icon_ref‘ and ‘img_ref‘, similar to ‘fill_color_ref‘. The color of the icons can be assigned through either ‘icon_color‘ (a single color) or ‘icon_color_ref‘ (from another column). The size of the images can be adjusted using ‘img_height‘ and ‘img_width‘. The size of the icons can be adjusted using ‘icon_size‘. ‘data_bars()‘ works with columns containing both positive and negative values. It should be placed within the cell argument in reactable::colDef.

Usage

```
data_bars(
  data,
  text_position = "inside-end",
  fill_color = "#15607A",
  fill_color_ref = NULL,
  fill_by = NULL,
  fill_opacity = 1,
  fill_gradient = FALSE,
  background = "#EEEEEE",
  number_fmt = NULL,
  bias = 1,
  min_value = NULL,
  max_value = NULL,
  align_bars = "left",
  bar_height = NULL,
  force_outside = NULL,
  text_color = "black",
  text_color_ref = NULL,
```

```
  text_size = NULL,
  brighten_text = TRUE,
  brighten_text_color = "white",
  bold_text = FALSE,
  border_width = NULL,
  border_style = NULL,
  border_color = NULL,
  icon = NULL,
  icon_ref = NULL,
  icon_size = 20,
  icon_color = NULL,
  icon_color_ref = NULL,
  img = NULL,
  img_ref = NULL,
  img_height = 20,
  img_width = 20,
  box_shadow = FALSE,
  round_edges = FALSE,
  tooltip = FALSE,
  animation = "width 1s ease"
)
```

Arguments

data	Dataset containing at least one numeric column.
text_position	Choose where to display the text labels relative to the filled data bars. Text labels can be displayed within the filled bars ("inside-end" or "inside-base"), outside of the filled bars ("outside-end" or "outside-base"), within the center of the filled bars ("center"), above the filled bars ("above"), or not displayed at all ("none"). Default is inside-end.
fill_color	A single color or a vector of fill_color for the fill of the data bars. fill_color should be given in order from low values to high values. Can use R's built-in fill_color or other color packages. Default is #15607A.
fill_color_ref	Optionally assign fill_color to from another column by providing the name of the column containing the fill colors in quotes. Only one color can be provided per row, and therefore will not work with fill_gradient. Default is NULL.
fill_by	Assign data bars to a column based on the values of another column. The column in reference must contain numeric data. The column in which the colors are being assigned to can be either numerical or character. Default is NULL.
fill_opacity	A value between 0 and 1 that adjusts the opacity in fill_color. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
fill_gradient	Logical: if two or more colors are provided in fill_color, the colors in the fill of the bars are converted to a left-to-right gradient. Default is FALSE.
background	The color for the background of the data bars. Default is #EEEEEE.
number_fmt	Optionally format numbers using formats from the scales package. Default is NULL.

bias — A positive value that determines the spacing between multiple colors. A higher value spaces out the colors at the higher end more than a lower number. Default is 1.

min_value — A value to use as the minimum value for the width of the filled bars. Default is NULL.

max_value — A value to use as the maximum value for the width of the filled bars. The default maximum value is the maximum value in the column. Default is NULL.

align_bars — Display filled bars from left-to-right or right-to-left. Options are "left" or "right". Default is left.

bar_height — Numeric height of the data bars in px. Default is NULL.

force_outside — Optionally force a range of values to display their text labels on the outside-end of the filled bars when the text_position is set to either "inside-end", "inside-base", or "center". Must provide a start and a stop number for the range of values to be forced to outside-end. Ex. c(0, 100). Default is NULL.

text_color — The color of the text labels. Default is black.

text_color_ref — Optionally assign text color from another column by providing the name of the column containing the text colors in quotes. Only one color can be provided per cell. Default is NULL.

text_size — Numeric value representing the size of the text labels. Default is NULL.

brighten_text — Logical: automatically assign color to text labels based on filled color when the text labels are positioned within the filled bars. Text within dark-colored filled bars will turn white, text within light-colored bars will be black. Default is TRUE.

brighten_text_color — Assigns color to text labels if values are within a dark-colored filled bar. Default is white.

bold_text — Logical: display the text labels in bold. Default is FALSE.

border_width — The width of the border around the filled data bars Options are "thin", "medium", "thick", or a numeric value with the units included such as "2px" or "2mm". May be specified using one, two, three, or four values. See [CSS border-width](https://developer.mozilla.org/en US/docs/Web/CSS/border-width) for more options. Default is NULL.

border_style — The style of the border around the filled data bars Options are "solid", "dashed", "dotted", "double", "groove", "ridge", "inset", "outset", "none", or "hidden". May be specified using one, two, three, or four values. See [CSS border-style](https://developer.mozilla.or US/docs/Web/CSS/border-style) for more options. Default is NULL.

border_color — The color of the border around the filled data bars May be specified using one, two, three, or four values. See [CSS border-color](https://developer.mozilla.org/en-US/docs/Web/CSS/border-color) for more options. Default is NULL.

icon — Assign an icon label from the Font Awesome library (via shiny). If an icon is provided, it will be positioned so that it does not overlap the text for the data bars. Default is NULL.

icon_ref — Optionally assign icons from another column by providing the name of the column containing the icons in quotes. Only one icon can be provided per cell. Default is NULL.

icon_size	A value representing the size of the icon in px. Default is 20.
icon_color	The color for the icon. If no color is provided, default is set to the color of the filled bars. Default is NULL.
icon_color_ref	Optionally assign color to the icons from another column by providing the name of the column containing the icon colors in quotes. Only one color can be provided per cell. Default is NULL.
img	Optionally assign an image label via a valid URL.
img_ref	Optionally assign images from another column by providing the name of the column containing the image URLs in quotes. Only one image can be provided per cell. Default is NULL.
img_height	A value for the height of the image in px. Default is 20.
img_width	A value for the width of the image in px. Default is 20.
box_shadow	Logical: add a box shadow to the bars. Default is FALSE.
round_edges	Logical: round the edges around the data bars. Default is FALSE.
tooltip	Logical: hover tooltip. Default is FALSE.
animation	Control the duration and timing function of the animation when sorting/updating values shown on a page. See [CSS transitions](https://developer.mozilla.org/en-US/docs/Web/CSS/transition) for available timing functions and examples. Animation can be turned off by setting to "none". Default is "width 1s ease".

Value

a function that applies data bars to a column of numeric values.

Examples

```
data <- MASS::Cars93[20:49, c("Make", "MPG.city", "MPG.highway")]

## By default, data bars are aligned left and text_position are placed on the inside end
reactable(data,
          defaultColDef = colDef(
            cell = data_bars(data)))

## Align the bars to the right
reactable(data,
          defaultColDef = colDef(
            cell = data_bars(data,
                             align_bars = "right")))

## Move the text labels outside of the filled bars
reactable(data,
          defaultColDef = colDef(
            cell = data_bars(data,
                             text_position = "outside-end")))

## Apply multiple fill_color to the filled bars
reactable(data,
          defaultColDef = colDef(
```

```
            cell = data_bars(data,
                             fill_color = c("lightblue","royalblue","navy"))))

## Apply a fill_gradient pattern to the filled bars
reactable(data,
          defaultColDef = colDef(
            cell = data_bars(data,
                             fill_color = c("lightblue","royalblue","navy"),
                             fill_gradient = TRUE,
                             text_position = "outside-end")))
```

`data_bars_gradient` *Add horizontal gradient bars to rows in a column*

Description

The ‘data_bars_gradient()‘ function is depreciated. The new version of ‘data_bars()‘ can convert colors into gradients with ‘gradient = TRUE‘. Please use ‘data_bars()‘ instead.

Usage

```
data_bars_gradient(
  data,
  colors = c("#1efffd", "#1e20ff"),
  background = "white",
  number_fmt = NULL
)
```

Arguments

`data`	Dataset containing at least one numeric column.
`colors`	A vector of colors of at least two colors. Colors should be given in order from left to right as shown on the data bar. Default colors are c("#1efffd", "#1e20ff").
`background`	Optionally assign a color to use as the background for cells. Default is set to white.
`number_fmt`	Optionally format numbers using formats from the scales package. Default is set to NULL.

Value

a function that applies data bars to a column of numeric values.

Examples

```
data <- MASS::Cars93[20:49, c("Make", "MPG.city", "MPG.highway")]

## By default, colors are provided
reactable(data,
defaultColDef = colDef(
align = "left",
cell = data_bars(data,
fill_color = c("#1efffd", "#1e20ff"),
fill_gradient = TRUE)))
```

`data_bars_pos_neg` *Add horizontal bars to rows in a column containing positive and negative values*

Description

The ‘data_bars_pos_neg()‘ function is depreciated. The new version of ‘data_bars()‘ can handle both positive and negative values now. Please use ‘data_bars()‘ instead.

Usage

```
data_bars_pos_neg(data, colors = c("red", "green"), number_fmt = NULL)
```

Arguments

`data`	Dataset containing at least one numeric column.
`colors`	A minimum of two colors or a vector of colors. Colors should be given in order from negative values to positive values. Can use R's built-in colors or other color packages.
`number_fmt`	Optionally format numbers using formats from the scales package. Default is set to NULL.

Value

a function that applies positive and negative data bars to a column of numeric values.

Examples

```
data <- data.frame(
company = sprintf("Company%02d", 1:10),
profit_chg = c(0.2, 0.685, 0.917, 0.284, 0.105, -0.701, -0.528, -0.808, -0.957, -0.11))

## By default, the negative values are assigned a red bar,
## and the positive values are assigned a green bar
reactable(data,
bordered = TRUE,
```

```
columns = list(
 company = colDef(name = "Company",
 minWidth = 100),
 profit_chg = colDef(
   name = "Change in Profit",
   defaultSortOrder = "desc",
   align = "center",
   minWidth = 400,
   cell = data_bars(data))))

## You can apply a relative color scale to the bars by assigning three or more colors
reactable(data,
bordered = TRUE,
columns = list(
  company = colDef(name = "Company",
  minWidth = 100),
  profit_chg = colDef(
  name = "Change in Profit",
  defaultSortOrder = "desc",
  align = "center",
  minWidth = 400,
  cell = data_bars(data,
  fill_color = c("#ff3030", "#ffffff", "#1e90ff")))))
```

`default` *Theme default*

Description

Reactable-inspired default theme

Usage

```
default(
  font_size = 15,
  font_color = "#333333",
  header_font_size = 15,
  header_font_color = "#333333",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

`font_size`	Numeric value representing the size of the font within the table (in px). Default is 15.
`font_color`	Color of the font for the text within the table and the group headers. Default is #333333.

header_font_size
: Numeric value representing the size of the font within the table (in px). Default is 15.

header_font_color
: Color of the font for the header text. Default is #333333.

cell_padding
: Numeric value representing the padding size between cells (in px). Default is 6.

centered
: Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard default theme
reactable(data,
          theme = default())

## Additional options applied
reactable(data,
          theme = default(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

embed_img *Embed image from web to cells in a column*

Description

The ‘embed_img()‘ function adds images obtained from the web to a column within reactable. It should be placed within the cell argument in reactable::colDef.

Usage

```
embed_img(
  data,
  height = 24,
  width = 24,
  horizontal_align = "center",
  label = NULL,
  label_position = "right"
)
```

Arguments

data	Dataset containing URL's to images
height	A value given for the height of the image in px. Default height is 24px.
width	A value given for the width of the image in px. Default width is 24px.
horizontal_align	The horizontal alignment of the image within a cell. Options are "left", "right", or "center". Default is "center".
label	Optionally assign a label to the image from another column. Default is set to NULL or no label.
label_position	Position of label relative to image. Options are "right", "left", "below", or "above". Default is right.

Value

a function that renders an image to a column containing a valid web link.

Examples

```
## If no image links are in the original dataset, you need to assign them like so:
library(dplyr)
data <- iris %>%
 mutate(
 img = case_when(
 Species == "setosa" ~
 "https://upload.wikimedia.org/wikipedia/commons/d/d9/Wild_iris_flower_iris_setosa.jpg",
 Species == "versicolor" ~
 "https://upload.wikimedia.org/wikipedia/commons/7/7a/Iris_versicolor.jpg",
 Species == "virginica" ~
 "https://upload.wikimedia.org/wikipedia/commons/9/9f/Iris_virginica.jpg",
 TRUE ~ "NA"))

## Then use embed_img() to display images
reactable(data,
columns = list(
 img = colDef(cell = embed_img())))

## By default, images are given a size of 24px by 24px,
## but you can adjust the size using height and width:
reactable(data,
columns = list(
 img = colDef(cell = embed_img(height = 50, width = 45))))

## Optionally assign a label to the image from another column
reactable(data,
columns = list(
 img = colDef(cell = embed_img(data, label = "Species"))))
```

espn *Theme espn*

Description

ESPN-inspired table theme

Usage

```
espn(
  font_size = 12,
  font_color = "#6C6D6F",
  header_font_size = 11,
  header_font_color = "#48494a",
  cell_padding = 7,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 12.
font_color	Color of the font for the text within the table and the group headers. Default is #6C6D6F.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 11.
header_font_color	Color of the font for the header text. Default is #48494a.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 7.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard espn theme
reactable(data,
          theme = espn())

## Additional options applied
reactable(data,
          theme = espn(font_size = 12, font_color = "grey", cell_padding = 3))
```

`fivethirtyeight` *Theme fivethirtyeight*

Description

538-inspired table theme

Usage

```
fivethirtyeight(
  font_size = 14,
  font_color = "#222222",
  header_font_size = 12,
  header_font_color = "#000000",
  cell_padding = 5,
  centered = FALSE
)
```

Arguments

`font_size`	Numeric value representing the size of the font within the table (in px). Default is 14.
`font_color`	Color of the font for the text within the table and the group headers. Default is #222222.
`header_font_size`	Numeric value representing the size of the font within the table (in px). Default is 12.
`header_font_color`	Color of the font for the header text. Default is #000000.
`cell_padding`	Numeric value representing the padding size between cells (in px). Default is 5.
`centered`	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard fivethirtyeight theme
reactable(data,
          theme = fivethirtyeight())

## Additional options applied
reactable(data,
         theme = fivethirtyeight(font_size = 12, font_color = "grey", cell_padding = 3))
```

`flatly` *Theme flatly*

Description

Bootstrap-inspired flatly theme

Usage

```
flatly(
  font_size = 14,
  font_color = "#212529",
  header_font_size = 15,
  header_font_color = "#ffffff",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

`font_size`	Numeric value representing the size of the font within the table (in px). Default is 14.
`font_color`	Color of the font for the text within the table and the group headers. Default is #212529.
`header_font_size`	Numeric value representing the size of the font within the table (in px). Default is 15.
`header_font_color`	Color of the font for the header text. Default is #ffffff.
`cell_padding`	Numeric value representing the padding size between cells (in px). Default is 6.
`centered`	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard flatly theme
reactable(data,
          theme = flatly())

## Additional options applied
reactable(data,
          theme = flatly(font_size = 12, font_color = "grey", cell_padding = 3))
```

gauge_chart *Display numeric values in a gauge chart*

Description

The ‘gauge_chart()‘ function displays numeric values in a column in a gauge or aka a speedometer chart. The minimum and maximum values that are present in the column can be added to the gauge chart by setting ‘show_min_max‘ to TRUE. By default, the minimum and maximum bounds of the gauge chart are the min/max of the column, but can be changed with ‘min_value‘ and ‘max_value‘. The format of the min/max values and the values shown within the gauge chart can be changed with ‘number_fmt‘. There are two sizes available for the gauge. The smaller default size 1 and the bigger size 2. The size can be specified within ‘size‘. Many options that are available in ‘data_bars()‘ are also available in ‘gauge_chart()‘. There are a few different ways to color the fill of the gauge. One way would be to apply either a single or multiple colors within ‘fill_color‘. Colors may be assigned via another column if referenced within ‘fill_color_ref‘. The opacity of the fill color can be controlled with ‘opacity‘. If multiple colors are used within ‘fill_color‘, the bias of the color normalization can be controlled with ‘bias‘. The empty fill of the gauge can be colored with ‘background‘. The color of the values within the gauge can be changed using ‘text_color‘. Or they can be assigned via another column with ‘text_color_ref‘. ‘gauge_chart()‘ needs to placed within the cell argument in reactable::colDef.

Usage

```
gauge_chart(
  data,
  fill_color = "#15607A",
  background = "#EEEEEE",
  show_min_max = FALSE,
  size = 1,
  min_value = NULL,
  max_value = NULL,
  number_fmt = NULL,
  fill_color_ref = NULL,
  text_color = "black",
  bold_text = FALSE,
  min_text_color = "black",
  max_text_color = "black",
  text_color_ref = NULL,
  text_size = NULL,
  bias = 1,
  opacity = 1,
  tooltip = FALSE,
  animation = "transform 1s ease"
)
```

Arguments

data	Dataset containing at least one numeric column.

fill_color	A single color or a vector of fill_color for the fill of the gauge. fill_color should be given in order from low values to high values. Can use R's built-in fill_color or other color packages. Default is #15607A.
background	The color for the background of the gauge. Default is #EEEEEE.
show_min_max	Show or hide the min and max values on the gauge. Default is FALSE.
size	Size of the gauge. Options are 1 (small gauge) or 2 (large gauge). Default is 1.
min_value	A value to use as the minimum value for the gauge. The default minimum value is 0. Default is NULL.
max_value	A value to use as the maximum value for the gauge. The default maximum value is the maximum value in the column. Default is NULL.
number_fmt	Optionally format numbers using formats from the scales package. Default is NULL.
fill_color_ref	Assign colors to the fill of the gauge via another column by providing the name of the column containing the colors in quotes. Only one color can be provided per row. Default is NULL.
text_color	The color of the value shown within the gauge. Default is black.
bold_text	Logical: bold the text of the value within the gauge. Default is FALSE.
min_text_color	The color of the minimum value shown underneath the gauge. Default is black.
max_text_color	The color of the maximum value shown underneath the gauge. Default is black.
text_color_ref	Assign color to the text within the gauge via another column by providing the name of the column containing the text colors in quotes. Only one color can be provided per cell. Default is NULL.
text_size	Numeric value representing the size of the value within the gauge. Default is NULL.
bias	A positive value that determines the spacing between multiple colors. A higher value spaces out the colors at the higher end more than a lower number. Default is 1.
opacity	A value between 0 and 1 that adjusts the opacity of fill_color. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
tooltip	Logical: hover tooltip. Default is FALSE.
animation	Control the duration and timing function of the animation when sorting/updating values shown on a page. See [CSS transitions](https://developer.mozilla.org/en-US/docs/Web/CSS/transition) for available timing functions and examples. Animation can be turned off by setting to "none". Default is "background 1s ease".

Value

a function that displays values in a column in a gauge chart.

Examples

```
library(dplyr)
data <- iris[45:54, ]
```

```
## Show values within a gauge chart:
reactable(
data,
defaultColDef = colDef(
align = "left",
maxWidth = 150,
cell = gauge_chart(data)))

## Show the min and max below the gauge:
reactable(
data,
defaultColDef = colDef(
align = "left",
maxWidth = 150,
cell = gauge_chart(data, show_min_max = TRUE)))

## Adjust the min and max value of the gauge:
reactable(
data,
defaultColDef = colDef(
align = "left",
maxWidth = 150,
cell = gauge_chart(data, show_min_max = TRUE, min_value = 0, max_value = 7)))

## Increase the size of the gauge chart:
reactable(
data,
defaultColDef = colDef(
align = "left",
maxWidth = 150,
cell = gauge_chart(data, size = 2)))

## Assign multiple colors to create a normalized fill based on value:
reactable(
data,
defaultColDef = colDef(
align = "left",
maxWidth = 150,
cell = gauge_chart(data, fill_color = c("blue","white","orange"))))

## Conditionally apply colors from another column:
data %>%
mutate(color_assign = case_when(
Species == "setosa" ~ "red",
Species == "versicolor" ~ "forestgreen",
TRUE ~ "grey")) %>%
reactable(
.,
defaultColDef = colDef(
align = "left",
maxWidth = 150,
cell = gauge_chart(., fill_color_ref = "color_assign")))
```

```
## Change the color of the empty fill of the gauge:
reactable(
data,
defaultColDef = colDef(
align = "left",
maxWidth = 150,
cell = gauge_chart(data, background = "transparent")))
```

---

google_font *Apply a font from Google Fonts <https://fonts.google.com/> to the table.*

Description

Apply a font from Google Fonts <https://fonts.google.com/> to the table.

Usage

```
google_font(
  table = NULL,
  font_family = "Poppins",
  font_weight = 400,
  font_style = "normal"
)
```

Arguments

table	Null.
font_family	Color of the font for the text within the table. Default is #222222.
font_weight	The numeric weight of the font. Must be a value between 100 and 900. Note: not every font on Google Fonts has all font weights available. Please check <https://fonts.google.com/> for available weights for desired font family. Default is 400.
font_style	Style of the text font. Options are "normal" or "italic". Default is "normal".

Value

a function that applies a font to a reactable table.

Examples

```
## Not run:
data <- iris[10:29, ]

## Default 'Poppins' font from Google Fonts
reactable(data) %>%
google_font()
```

```
## Apply styles to fonts
reactable(data) %>%
google_font("Roboto Mono", font_weight = 500, font_style = "italic")

## End(Not run)
```

`group_border_sort` *Add a styled border beneath rows of specified groups on sort. Must be placed within reactable::rowStyle(). Credit to Greg Lin, creator of reactable for writing the JS function.*

Description

Add a styled border beneath rows of specified groups on sort. Must be placed within reactable::rowStyle(). Credit to Greg Lin, creator of reactable for writing the JS function.

Usage

```
group_border_sort(
  columns = NULL,
  border_width = "thin",
  border_color = "#777",
  border_style = "solid"
)
```

Arguments

`columns`	Name of the column(s). Can provide up to four column names.
`border_width`	The width of the border. Options are "thin", "medium", "thick", or a numeric value such as "2px". May be specified using one, two, three, or four values. See [CSS border-width](https://developer.mozilla.org/en-US/docs/Web/CSS/border-width) for more options. Default is 'thin'.
`border_color`	The color of the border. May be specified using one, two, three, or four values. See [CSS border-color](https://developer.mozilla.org/en-US/docs/Web/CSS/border-color) for more options. Default is #777.
`border_style`	The style of the border. Options are "solid", "dashed", "dotted", "double", "groove", "ridge", "inset", "outset", "none", or "hidden". May be specified using one, two, three, or four values. See [CSS border-style](https://developer.mozilla.org/en-US/docs/Web/CSS/border-style) for more options. Default is 'solid'.

Value

a function that applies a bottom border to each group in a column of a reactable table.

Examples

```
data <- MASS::Cars93[1:20, c("Manufacturer", "Model", "Type", "MPG.city")]

## Add border beneath each unique group within a column on sort:
reactable(data,
pagination = FALSE,
rowStyle = group_border_sort("Manufacturer")
)

## Can specify up to 4 columns:
reactable(data,
pagination = FALSE,
rowStyle = group_border_sort(columns = c("Manufacturer","Model","Type"))
)

## Apply styles to the border:
reactable(data,
pagination = FALSE,
rowStyle = group_border_sort(columns = c("Manufacturer","Model","Type"),
                             border_color = "red",
                             border_style = "dashed",
                             border_width = "3px")
)
```

`group_merge_sort` *Hide rows containing duplicate values on sort. Must be placed within reactable::colDef(style). Credit to Greg Lin, creator of reactable for writing the JS function.*

Description

Hide rows containing duplicate values on sort. Must be placed within reactable::colDef(style). Credit to Greg Lin, creator of reactable for writing the JS function.

Usage

```
group_merge_sort(col_name = NULL)
```

Arguments

`col_name`	Name of the column.

Value

a function that hides duplicate values on sort in a reactable table.

Examples

```
data <- MASS::Cars93[1:20, c("Manufacturer", "Model", "Type", "MPG.city")]

## Merge unique groups in a column:
reactable(data,
pagination = FALSE,
columns = list(Manufacturer = colDef(
style = group_merge_sort("Manufacturer")
))
)

## Works with columns containing numeric data as well:
reactable(data,
pagination = FALSE,
columns = list(MPG.city = colDef(
style = group_merge_sort("MPG.city")
))
)
```

`highlight_bars` *Color of highlight used in 'react_sparkbar'.*

Description

Color of highlight used in 'react_sparkbar'.

Usage

```
highlight_bars(
  first = "transparent",
  last = "transparent",
  min = "transparent",
  max = "transparent"
)
```

Arguments

`first, last, min, max`
: The colors of first, last, min, and max bars

Value

a function that provides colors for specific bars.

highlight_max *Highlights the maximum value in a column*

Description

The 'highlight_max()' function assigns a font color and/or background color to the maximum value in a column. It should be placed within the style argument in reactable::colDef.

Usage

```
highlight_max(data, font_color = "green", highlighter = NULL)
```

Arguments

data	Dataset containing at least one numeric column.
font_color	color to assign to maximum value in a column. Default color is green.
highlighter	color to assign the background of a cell containing maximum value in a column.

Value

a function that applies a color to the maximum value in a column of numeric values.

Examples

```
data <- MASS::road[11:17, ]

## By default, the maximum value is bold with a green font color
reactable(data,
defaultColDef = colDef(
    style = highlight_max(data)))

## Assign a different font color
reactable(data,
defaultColDef = colDef(
    style = highlight_max(data,
    font_color = "red")))

## Highlight the background of the cell for the maximum value in each column
reactable(data,
defaultColDef = colDef(
    style = highlight_max(data,
    highlighter = "yellow")))
```

highlight_min *Highlights the minimum value in a column*

Description

The 'highlight_min()' function assigns a font color and/or background color to the minimum value in a column. It should be placed within the style argument in reactable::colDef.

Usage

```
highlight_min(data, font_color = "red", highlighter = NULL)
```

Arguments

data	Dataset containing at least one numeric column.
font_color	color to assign to minimum value in a column. Default color is red.
highlighter	color to assign the background of a cell containing minimum value in a column.

Value

a function that applies a color to the minimum value in a column of numeric values.

Examples

```
data <- MASS::road[11:17, ]

## By default, the minimum value is bold with a red font color
reactable(data,
defaultColDef = colDef(
    style = highlight_min(data)))

## Assign a different font color
reactable(data,
defaultColDef = colDef(
    style = highlight_min(data,
    font_color = "green")))

## Highlight the background of the cell for the minimum value in each column
reactable(data,
defaultColDef = colDef(
    style = highlight_min(data,
    highlighter = "yellow")))
```

highlight_min_max *Highlights the minimum and maximum value in a column*

Description

The 'highlight_min_max()' function assigns a font color and/or background color to both the minimum and maximum values in a column. It should be placed within the style argument in reactable::colDef.

Usage

```
highlight_min_max(
  data,
  min_font_color = "red",
  max_font_color = "green",
  min_highlighter = NULL,
  max_highlighter = NULL
)
```

Arguments

data	Dataset containing at least one numeric column.
min_font_color	color to assign to minimum value in a column. Default color is red.
max_font_color	color to assign to maximum value in a column. Default color is green.
min_highlighter	color to assign the background of a cell containing minimum value in a column.
max_highlighter	color to assign the background of a cell containing maximum value in a column.

Value

a function that applies a color to the minimum and maximum values in a column of numeric values.

Examples

```
data <- MASS::road[11:17, ]

## By default, the minimum and maximum values are bold with a red and green font color respectively
reactable(data,
defaultColDef = colDef(
    style = highlight_min_max(data)))

## Assign a different font color to the min and max values
reactable(data,
defaultColDef = colDef(
    style = highlight_min_max(data,
    min_font_color = "orange",
```

```
    max_font_color = "blue")))

## Highlight the background of the cell for the min and max values in each column
reactable(data,
defaultColDef = colDef(
    style = highlight_min_max(data,
    min_highlighter = "salmon",
    max_highlighter = "skyblue")))
```

`highlight_points` *Color of points used in 'react_sparkline'.*

Description

Color of points used in 'react_sparkline'.

Usage

```
highlight_points(
  all = "transparent",
  first = "transparent",
  last = "transparent",
  min = "transparent",
  max = "transparent"
)
```

Arguments

`all`, `first`, `last`, `min`, `max`
: The colors of all, first, last, min, and max points.

Value

a function that provides colors for specific points.

`hoverdark` *Theme hoverdark*

Description

Changes from light-themed to dark-themed on hover

Usage

```
hoverdark(
  font_size = 15,
  font_color = "#222222",
  header_font_size = 15,
  cell_padding = 4,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
font_color	Color of the font for the text within the table. Default is #222222.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 4.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard hoverdark theme
reactable(data,
          theme = hoverdark())

## Additional options applied
reactable(data,
          theme = hoverdark(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

hoverlight *Theme hoverlight*

Description

Changes from dark-themed to light-themed on hover

Usage

```
hoverlight(
  font_size = 15,
  font_color = "#ffffff",
  header_font_size = 15,
  cell_padding = 4,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
font_color	Color of the font for the text within the table. Default is #ffffff.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 4.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard hoverlight theme
reactable(data,
          theme = hoverlight())

## Additional options applied
reactable(data,
          theme = hoverlight(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

html *Apply HTML attributes to title, subtitle, and source text.*

Description

Use ‘html()‘ to apply HTML attributes to text within ‘add_title()‘, ‘add_subtitle()‘, and ‘add_source()‘.

Usage

```
html(text, ...)
```

Arguments

text, ...	The text provided within the title, subtitle or source with HTML attributes applied.

Value

an object of class HTML.

Examples

```
## Not run:
## Change the title color to blue
data <- iris[10:29, ]
reactable(data) %>%
add_title(html("Normal title. <span style='color:DodgerBlue;'>Blue title.</span>"))

## Add emojis to the source
data <- iris[10:100, ]
reactable(data) %>%
add_source(html("<p>Made with &#128151; by: John Doe &#128512;</p>"))

## End(Not run)
```

---

icon_assign *Assign icons to cells in a column*

Description

The ‘icon_assign()‘ function assigns icons from the Font Awesome library (via shiny) to each cell of a numeric column depending on the value in each row. By default, the number of icons assigned will be equal to the value in that cell. If the value is less than the max, it will receive empty icons. Both the icon shape, size, and color of the filled and empty icons can be modified through the parameters. Values can optionally be shown with the icons if desired. It should be placed within the cell argument in reactable::colDef.

Usage

```
icon_assign(
  data,
  icon = "circle",
  fill_color = "#67a9cf",
  empty_color = "lightgrey",
  fill_opacity = 1,
  empty_opacity = 1,
  align_icons = "left",
  icon_size = 16,
  buckets = NULL,
  number_fmt = NULL,
```

```
  seq_by = 1,
  show_values = "none",
  animation = "1s ease"
)
```

Arguments

data	Dataset containing at least one numeric column.
icon	A single icon from the Font Awesome library (via shiny). Default icon is a circle.
fill_color	A single color for the filled icons. Default color is #1e90ff.
empty_color	A single color for the empty icons. Default color is lightgrey.
fill_opacity	A value between 0 and 1 that adjusts the opacity in fill_color. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
empty_opacity	A value between 0 and 1 that adjusts the opacity in empty_color. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
align_icons	Choose how to align the icons in a column. Options are left, right, or center. Default is left.
icon_size	A value representing the size of the icon in px. Default is 16.
buckets	Optionally divide values in a column into buckets by providing a numeric value. Icons are then assigned by rank from lowest to highest. Default is set to NULL.
number_fmt	Optionally format numbers using formats from the scales package. Default is set to NULL.
seq_by	A numerical input that determines what number each icon represents. Ex. instead of displaying 100 icons for the number 100, can set seq_by = 10 to show only 10 icons. Default value is set to 1.
show_values	Optionally display values next to icons. Options are "left", "right", above", "below", or "none". Default is none.
animation	Control the duration and timing function of the animation when sorting/updating values shown on a page. See [CSS transitions](https://developer.mozilla.org/en-US/docs/Web/CSS/transition) for available timing functions and examples. Animation can be turned off by setting to "none". Default is "1s ease".

Value

a function that applies colored icons to a column of numeric values.

Examples

```
data <- iris[10:29, ]
## By default, icon_assign() assigns a cirlce icon for each value up to the maximum value.
## If a value is 5 and the maximum value in the column is 6,
## It will assign 5 blue icons and 1 grey icon.
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data))))
```

```
## Assign colors to filled icons and empty icons
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data,
fill_color = "red",
empty_color = "white"))))

## Assign any icon from the Font Awesome Library
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data,
icon = "fan"))))

## Optionally divide values into buckets and assign icons based on rank.
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data,
buckets = 3))))

## Optionally display values next to icons.
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data,
show_values = "right"))))

## Change the alignment of the icons within a column.
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data,
align_icons = "center"))))
```

`icon_sets` *Add colored icons to cells in a column*

Description

The ‘icon_sets()‘ function conditionally adds an icon from the Font Awesome library (via shiny) to each cell of a column and assigns a color depending on their value in relation to other values in that particular column. Any number of icons and any number of colors can be used. The number of icons and colors determines how the values are shown from low values to high values. The icons can be positioned over, above, below, or to the right or left of the values. The size of the icon can be adjusted. Icons and icon colors can be provided via another reference column in the dataset which is useful when assigning icons/colors to particular occurrences. It should be placed within the cell argument in reactable::colDef.

Usage

```
icon_sets(
```

```
  data,
  icons = c("circle"),
  icon_set = NULL,
  colors = c("#15607A", "#B0B0B0", "#FA8C00"),
  opacity = 1,
  icon_position = "right",
  icon_ref = NULL,
  icon_size = 16,
  icon_color_ref = NULL,
  number_fmt = NULL,
  tooltip = FALSE,
  animation = "1s ease"
)
```

Arguments

data	Dataset containing at least one numeric column.
icons	A vector of three icons from the Font Awesome library (via shiny). Icons should be given in order from low values to high values. Default icons are circles.
icon_set	Apply a pre-selected set of icons to values. Options are "ski rating", "medals", and "batteries". Default is NULL.
colors	The color(s) to assign to the icons. Colors should be given in order from low values to high values. Default colors provided are blue-grey-orange: c("#15607A", "#B0B0B0", "#FA8C00"). Can use R's built-in colors or other color packages.
opacity	A value between 0 and 1 that adjusts the opacity in colors. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
icon_position	Position of icon relative to numbers. Options are "left", "right", above", "below", or "over". Default is right.
icon_ref	Optionally assign icons from another column by providing the name of the column containing the icons in quotes. Only one icon can be provided per cell. Default is NULL.
icon_size	A value representing the size of the icon in px. Default is 16.
icon_color_ref	Optionally assign color to the icons from another column by providing the name of the column containing the icon colors in quotes. Only one color can be provided per cell. Default is NULL.
number_fmt	Optionally format numbers using formats from the scales package. Default is set to NULL.
tooltip	Logical: hover tooltip. Default is FALSE.
animation	Control the duration and timing function of the animation when sorting/updating values shown on a page. See [CSS transitions](https://developer.mozilla.org/en-US/docs/Web/CSS/transition) for available timing functions and examples. Animation can be turned off by setting to "none". Default is "1s ease".

Value

a function that applies an icon to a column of numeric values.

Examples

```
data <- MASS::Cars93[20:49, c("Make", "MPG.city", "MPG.highway")]

## By default, icon_sets() assigns blue circles to the lowest-third values,
## grey circles to the middle-third values,
## and orange to the top-third values
reactable(data,
defaultColDef = colDef(cell = icon_sets(data)))

## Apply pre-set icon sets with icon_set:
reactable(data,
defaultColDef = colDef(cell = icon_sets(data,
icon_set = 'ski rating')))

## Assign custom colors
reactable(data,
defaultColDef = colDef(cell = icon_sets(data,
colors = c("tomato", "grey", "dodgerblue"))))

## Assign icons from Font Awesome's icon library
reactable(data,
defaultColDef = colDef(cell = icon_sets(data,
icons = c("arrow-down","minus","arrow-up"))))

## Use number_fmt to format numbers using the scales package
car_prices <- MASS::Cars93[20:49, c("Make", "Price")]

reactable(car_prices,
defaultColDef = colDef(cell = icon_sets(car_prices,
number_fmt = scales::dollar)))

## Position icons relative to the numbers. Options are to the left, right, above, below, or over.
reactable(car_prices,
defaultColDef = colDef(cell = icon_sets(car_prices,
icon_position = "above")))
```

`journal` *Theme journal*

Description

Bootstrap-inspired journal theme

Usage

```
journal(
  font_size = 14,
  font_color = "#222222",
```

```
  header_font_size = 15,
  header_font_color = "#fad9d8",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table and the group headers. Default is #222222.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #fad9d8.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard journal theme
reactable(data,
          theme = journal())

## Additional options applied
reactable(data,
          theme = journal(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

lux *Theme lux*

Description

Bootstrap-inspired lux theme

Usage

```
lux(
  font_size = 14,
  font_color = "#8c8c8c",
  header_font_size = 15,
  header_font_color = "#7f7f7f",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table and the group headers. Default is #8c8c8c.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #7f7f7f.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard lux theme
reactable(data,
          theme = lux())

## Additional options applied
reactable(data,
          theme = lux(font_size = 12, font_color = "grey", cell_padding = 3))
```

margin *Margin dimensions.*

Description

Margin dimensions.

Usage

```
margin(t = 0, r = 0, b = 0, l = 0)
```

Arguments

t, r, b, l	The dimensions of the top, right, bottom, and left margins.

Value

a function that provides margin dimensions.

merge_column *Merge two columns together in a specified arrangement.*

Description

The ‘merge_column()‘ function can be used to merge and style values from two columns within a reactable table. ‘merge_column()‘ works with both numeric and non-numeric columns. The style/format of both the current column and merged column can be controlled via size, color, weight, and text decoration. Any style parameters that start with "merged_" will control the column that is being merged and specified by ‘merged_name‘. Any style parameters that do not start with "merged_" control the current column you are within. The position of the column to be merged relative to the current column can be controlled with ‘merged_position‘. The position options for the merged column are above, below, left, or right to the current column. The spacing between the current column and the merged column can be controlled with ‘spacing‘. ‘merge_column()‘ needs to placed within the cell argument in reactable::colDef.

Usage

```
merge_column(
  data,
  merged_name = NULL,
  merged_position = "right",
  merged_size = 12,
  merged_color = "#777",
  merged_weight = "normal",
  merged_style = "normal",
  merged_decoration = "normal",
```

```
  size = 14,
  color = NULL,
  weight = "bold",
  style = "normal",
  decoration = NULL,
  spacing = 0
)
```

Arguments

data	Dataset containing either a text or numeric column.
merged_name	The name of the column to merge with the existing column. The column can contain either numeric or non-numeric data. Only a single column can be provided. Default is NULL.
merged_position	The position of the merged column in relation to the existing column. Options are 'right', 'left', 'above', or 'below'. Default is right.
merged_size	The size of the text of the merged column. Default is 12.
merged_color	The color of the text of the merged column. Default is #777.
merged_weight	The font weight of the text of the merged column. Options are "bold" or "normal". Default is "normal".
merged_style	The style of the text of the merged column. Options are "normal" or "italic". Default is "normal".
merged_decoration	The decoration of the text of the merged column. For example, add an underline, overline, or line-through to the text. Default is NULL.
size	The size of the text displayed in the current column. Default is 12.
color	The color of the text displayed in the current column. Default is NULL.
weight	The font weight of the text displayed in the current column. Options are "bold" or "normal". Default is "normal".
style	The style of the text displayed in the current column. Options are "normal" or "italic". Default is "normal".
decoration	The decoration of the text displayed in the current column. For example, add an underline, overline, or line-through to the text. Default is NULL.
spacing	The spacing between the merged and existing columns. A value greater than 0 creates more space between, a value less than 0 creates less space. Default is 0.

Value

a function that merges two columns together.

Examples

```
data <- MASS::Cars93[20:49, c("Manufacturer", "Model", "MPG.city", "MPG.highway")]

## Stack text from one column with another column:
```

```
reactable(
data,
columns = list(
Manufacturer = colDef(name = "Manufacturer/Model",
                      cell = merge_column(data, merged_name = "Model"
                      )),
Model = colDef(show = FALSE)))

## Control the appearance of both the current and merged columns:
reactable(
data,
columns = list(
Manufacturer = colDef(name = "Manufacturer/Model",
                      cell = merge_column(data,
                                          merged_name = "Model",
                                          merged_size = 16,
                                          merged_color = "blue",
                                          merged_style = "italic",
                                          size = 18,
                                          color = "red"
                                          )),
Model = colDef(show = FALSE)))

## Combine both numeric and non-numeric columns together:
reactable(
data,
columns = list(
Model = colDef(name = "Model/MPG Highway",
                  cell = merge_column(data,
                                      merged_name = "MPG.highway",
                                      merged_position = "below",
                                      merged_size = 20,
                                      merged_color = "green"
)),
MPG.highway = colDef(show = FALSE),
MPG.city = colDef(show = FALSE)))
```

`midnight` *Theme midnight*

Description

midnight table theme

Usage

```
midnight(
  font_size = 15,
  font_color = "#727272",
  header_font_size = 15,
```

```
  header_font_color = "#666666",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
font_color	Color of the font for the text within the table and the group headers. Default is #727272.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #666666.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard midnight theme
reactable(data,
          theme = midnight())

## Additional options applied
reactable(data,
          theme = midnight(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

midnightblue *Theme midnightblue*

Description

midnightblue table theme

Usage

```
midnightblue(
  font_size = 15,
  font_color = "#bababa",
  header_font_size = 15,
  header_font_color = "lightgrey",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
font_color	Color of the font for the text within the table and the group headers. Default is #bababa.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is lightgrey.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard midnightblue theme
reactable(data,
          theme = midnightblue())

## Additional options applied
reactable(data,
          theme = midnightblue(font_size = 12, font_color = "grey", cell_padding = 3))
```

`minty` *Theme minty*

Description

Bootstrap-inspired minty theme

Usage

```
minty(
  font_size = 15,
  font_color = "#9a9a9a",
  header_font_size = 16,
  header_font_color = "#c9e7de",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

`font_size`	Numeric value representing the size of the font within the table (in px). Default is 15.
`font_color`	Color of the font for the text within the table and the group headers. Default is #9a9a9a.
`header_font_size`	Numeric value representing the size of the font within the table (in px). Default is 16.
`header_font_color`	Color of the font for the header text. Default is #c9e7de.
`cell_padding`	Numeric value representing the padding size between cells (in px). Default is 6.
`centered`	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard minty theme
reactable(data,
          theme = minty())

## Additional options applied
reactable(data,
          theme = minty(font_size = 12, font_color = "grey", cell_padding = 3))
```

`no_lines` *Theme no_lines*

Description

A table style with no lines or borders

Usage

```
no_lines(
  font_size = 14,
  font_color = "#222222",
  header_font_size = 15,
  header_font_color = "#222222",
  centered = FALSE,
  cell_padding = 6
)
```

Arguments

`font_size`	Numeric value representing the size of the font within the table (in px). Default is 14.
`font_color`	Color of the font for the text within the table. Default is #222222.
`header_font_size`	Numeric value representing the size of the font within the table (in px). Default is 15.
`header_font_color`	Color of the font for the header text. Default is transparent
`centered`	Logical: vertically center the contents of the table. Default is FALSE.
`cell_padding`	Numeric value representing the padding size between cells (in px). Default is 6.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard no_lines theme
reactable(data,
          theme = no_lines())

## Additional options applied
reactable(data,
          theme = no_lines(font_size = 12, font_color = "grey", cell_padding = 3))
```

nytimes *Theme nytimes*

Description

The New York Times-inspired table theme

Usage

```
nytimes(
  font_size = 13,
  font_color = "#333333",
  header_font_size = 11,
  header_font_color = "#999999",
  cell_padding = 5,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 13.
font_color	Color of the font for the text within the table and the group headers. Default is #333333.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 11.
header_font_color	Color of the font for the header text. Default is #999999.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 5.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard nytimes theme
reactable(data,
          theme = nytimes())

## Additional options applied
reactable(data,
          theme = nytimes(font_size = 12, font_color = "grey", cell_padding = 3))
```

pff *Theme pff*

Description

Pro Football Focus-inspired table theme

Usage

```
pff(
  font_size = 16,
  font_color = "#878e94",
  header_font_size = 12,
  header_font_color = "#ffffff",
  cell_padding = 4,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 16.
font_color	Color of the font for the text within the table and the group headers. Default is #878e94.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 12.
header_font_color	Color of the font for the header text. Default is #ffffff.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 4.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard pff theme
reactable(data,
          theme = pff())

## Additional options applied
reactable(data,
          theme = pff(font_size = 12, font_color = "grey", cell_padding = 3))
```

pill_buttons *Surround text or numeric values in a colored pill button*

Description

The ‘pill_buttons()‘ function surrounds text or numeric values in a column in a colored pill button. If ‘pill_buttons()‘ is applied to a column containing text, the color of the pill button can be provided via a single color can be provided within ‘color‘ or via another column in the dataset by referencing the column name with ‘color_ref‘ (conditionally applying colors to text). If ‘pill_buttons‘ is applied to a numeric column, either a single color or a vector of colors can be provided within ‘color‘ or the colors can also be assigned via ‘color_ref‘. The opacity of the color(s) provided can be adjusted by providing a value between 0 and 1 in ‘opacity‘. The color of the text/values can be changed using ‘text_color‘. If text/values are displayed within a dark-colored background, ‘brighten_text‘ will display the text/values in white so they are more visible. The color of ‘brighten_text_color‘ can be changed to a color other than white if desired. The horizontal alignment of ‘pill_buttons()‘ can be controlled using reactable::colDef(align = "center"). ‘pill_buttons()‘ needs to placed within the cell argument in reactable::colDef.

Usage

```
pill_buttons(
  data,
  colors = "#15607A",
  color_ref = NULL,
  opacity = 1,
  number_fmt = NULL,
  show_text = TRUE,
  text_size = NULL,
  text_color = "black",
  text_color_ref = NULL,
  brighten_text = TRUE,
  brighten_text_color = "white",
  bold_text = FALSE,
  box_shadow = FALSE,
  tooltip = FALSE,
  animation = "background 1s ease"
)
```

Arguments

data	Dataset containing either a text or numeric column.
colors	The background color of the pill button. Only a single color can be provided for columns containing text. Multiple colors can be provided for columns containing values and should be given in order from low values to high values. If multiple colors are provided for columns containing text, the first color in the vector will be assigned to the text. The default color provided is "#15607A". Can use R’s built-in colors or other color packages.

color_ref	Optionally assign colors to from another column by providing the name of the column containing the colors in quotes. Only one color can be provided per row. Default is NULL.
opacity	A value between 0 and 1 that adjusts the opacity in color(s). A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
number_fmt	Optionally format numbers using formats from the scales package. Default is NULL.
show_text	Logical: show text or hide text. Default is TRUE.
text_size	Numeric value representing the size of the text labels. Default is NULL.
text_color	Assigns text color to values. Default is black.
text_color_ref	Optionally assign text color from another column by providing the name of the column containing the text colors in quotes. Only one color can be provided per cell. Default is NULL.
brighten_text	Logical: automatically assign color to text based on background color of the pill button. Text within dark-colored backgrounds will turn white, text within light-colored backgrounds will be black. Default is TRUE.
brighten_text_color	Assigns text color to values if values are within a dark-colored pill button backgrounds. Default is white.
bold_text	Logical: bold text. Default is FALSE.
box_shadow	Logical: add a box shadow to the buttons. Default is FALSE.
tooltip	Logical: hover tooltip. Default is FALSE.
animation	Control the duration and timing function of the animation when sorting/updating values shown on a page. See [CSS transitions](https://developer.mozilla.org/en-US/docs/Web/CSS/transition) for available timing functions and examples. Animation can be turned off by setting to "none". Default is "background 1s ease".

Value

a function that surrounds text/values in a column with a colored pill button.

Examples

```
library(dplyr)
data <- iris[45:54, ]

## Surround text with pill buttons:
reactable(
data,
columns = list(
Species = colDef(cell = pill_buttons(data))))

## Conditionally apply colors from another column:
data %>%
mutate(color_assign = case_when(
Species == "setosa" ~ "red",
```

```
Species == "versicolor" ~ "forestgreen",
TRUE ~ "grey")) %>%
reactable(.,
columns = list(
Species = colDef(cell = pill_buttons(., color_ref = "color_assign"))))

## Surround numeric values with pill buttons:
reactable(
data,
columns = list(
Petal.Width = colDef(cell = pill_buttons(data))))

## Apply multiple colors to numeric values:
reactable(
data,
columns = list(
Petal.Width = colDef(
cell = pill_buttons(data,
                    colors = c("lightpink","white","lightgreen"),
                    opacity = 0.3))))

## Apply a box shadow to the buttons to give a 3-D effect:
reactable(
data,
columns = list(
Petal.Width = colDef(
cell = pill_buttons(data,
                    box_shadow = TRUE,
                    colors = c("lightpink","white","lightgreen"),
                    opacity = 0.3))))
```

`pos_neg_colors` *Assign colors to negative and positive values*

Description

The 'pos_neg_colors()' function assigns a color to all negative values and a color to all positive values. It should be placed within the style argument in reactable::colDef.

Usage

```
pos_neg_colors(neg_col, pos_col, bold = NULL)
```

Arguments

`neg_col`	color to assign to negative values.
`pos_col`	color to assign to positive values.
`bold`	optional argument to bold values. Default is set to NULL or not bold.

Value

a function that applies a color to the positive and negative values of numeric column.

Examples

```
data <- data.frame(
Symbol = c("GOOG", "FB", "AMZN", "NFLX", "TSLA"),
Price = c(1265.13, 187.89, 1761.33, 276.82, 328.13),
Change = c(4.14, 1.51, -19.45, 5.32, -12.45))

## Assign the color red to negative values and green to positive values
reactable(data,
columns = list(
Change = colDef(
style = pos_neg_colors("red", "green"))))

## Bold values
reactable(data,
columns = list(
Change = colDef(
style = pos_neg_colors("red", "green", bold = TRUE))))
```

---

react_sparkbar *Add a sparkline bar chart to a reactable table*

Description

The ‘react_sparkbar()‘ function utilizes the dataui package <https://github.com/timelyportfolio/dataui> to create an interactive sparkline bar chart. The data provided must be in a list format. The vertical height of the sparkbar can be adjusted with ‘height‘. By default, the height is matched to the height of a cell in a reactable table. However, the height can be increased to better see the patterns in the data. The four-sided margin around the sparkbar can be controlled with ‘margin()‘. When labels are added to the sparkbars, the margin will auto-adjust (in most instances) to be able to display those labels. If the labels contain large values or values with many digits, the left and right margins may need to be increased slightly for the full numbers to be visible. The fill color and fill width can be controlled with ‘fill_color‘, ‘fill_color_ref‘, and ‘fill_opacity‘. The outline color and width of the filled bars can be controlled with ‘outline_color‘, ‘outline_color_ref‘, and ‘outline_width‘. ‘statline‘ can be used to show a horizontal dotted line that represents either the mean, median, min, or max (your choice). The appearance of the statline and statline labels can be controlled with ‘statline_color‘ and ‘statline_label_size‘. A bandline can be added by using ‘bandline‘. The options are innerquartiles which highlights the innerquartiles of the data or range which highlights the full range of the data. By default, ‘react_sparkbar()‘ is interactive and data points will be shown when hovering over the sparkbars. This can be turned off by setting ‘tooltip‘ to FALSE. There are two tooltip types available within ‘tooltip_type‘. The size and color of the tooltip labels can be adjusted with ‘tooltip_size‘ and ‘tooltip_color‘. Also by default, there are no labels on the bars themselves. However, one could add labels to the first, last, min, max, or all values within ‘labels‘. The labels that are shown on the sparkbar and in the tooltip are automatically rounded to the nearest whole

integer. But decimals can be shown by providing the number of decimal places in ‘decimals‘. The minimum value of a data series is the minimum value shown for a sparkbar, but this can be adjusted with ‘min_value‘ and the max can be adjusted with ‘max_value‘. ‘react_sparkline()‘ should be placed within the cell argument in reactable::colDef.

Usage

```
react_sparkbar(
  data,
  height = 22,
  fill_color = "slategray",
  fill_color_ref = NULL,
  fill_opacity = 1,
  outline_color = "transparent",
  outline_color_ref = NULL,
  outline_width = 1,
  highlight_bars = NULL,
  labels = "none",
  label_size = "0.8em",
  decimals = 0,
  max_value = NULL,
  min_value = NULL,
  statline = NULL,
  statline_color = "red",
  statline_label_size = "0.8em",
  bandline = NULL,
  bandline_color = "red",
  bandline_opacity = 0.2,
  tooltip = TRUE,
  tooltip_type = 1,
  tooltip_color = NULL,
  tooltip_size = "1.1em",
  margin = NULL
)
```

Arguments

data	Dataset containing a column with numeric values in a list format.
height	Height of the sparkbar. Default is 22.
fill_color	The color of the bar fill. Default is slategray.
fill_color_ref	Optionally assign fill colors from another column by providing the name of the column containing the colors in quotes. Only one color can be provided per row. Default is NULL.
fill_opacity	A value between 0 and 1 that adjusts the opacity. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1.
outline_color	The color of the outline around the filled bars. Default is transparent.

`outline_color_ref`
: Optionally assign outline colors from another column by providing the name of the column containing the colors in quotes. Only one color can be provided per row. Default is NULL.

`outline_width`
: Width of the outline around the filled bars. Default is 1.

`highlight_bars`
: Use 'highlight_bars()' to assign colors to particular bars. Colors can be assigned to all, min, max, first, or last bars. By default, transparent colors are assigned to each bars.

`labels`
: Show labels for points of interest. Options are 'min', 'max', 'first', 'last', 'all', or 'none'. Default is 'none'.

`label_size`
: The size of the labels. Default is 0.8em.

`decimals`
: Numeric: The number of decimals displayed in the labels and tooltip. Default is 0.

`max_value`
: Numeric: the maximum value of the sparkline range. Default is NULL (automatically the maximum value of each sparkline series).

`min_value`
: Numeric: the minimum value of the sparkline range. Default is NULL (automatically the minimum value of each sparkline series).

`statline`
: Inserts a horizontal dotted line representing a statistic, and places the value of that statistic to the right of the line. Options are 'mean', 'median', 'min', or 'max'. Default is NULL.

`statline_color`
: The color of the horizontal dotted statline. Default is red.

`statline_label_size`
: The size of the label to the right of the statline. Default is 0.8em.

`bandline`
: Inserts a horizontal bandline to render ranges of interest. Options are 'innerquartiles' or 'range' (min to max). Default is NULL.

`bandline_color`
: The color of the bandline. Default is red.

`bandline_opacity`
: A value between 0 and 1 that adjusts the opacity. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 0.2.

`tooltip`
: Logical: turn the tooltip on or off. Default is TRUE.

`tooltip_type`
: The tooltip type. Options are 1 or 2. Default is 1.

`tooltip_color`
: The color of the tooltip labels. Default is NULL.

`tooltip_size`
: The size of the tooltip labels. Default is '1.1em'.

`margin`
: The four-sided margin around the sparkbar. Use margin() to assign the top, right, bottom, and left margins.

Value

a function that creates a sparkline bar chart from a column containing a list of values.

Examples

```
## Not run:
library(dplyr)
## Default sparkline bar chart
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkbar(.))))

## Highlight particular bars
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkbar(.,
 decimals = 1,
 min_value = 0,
 highlight_bars = highlight_bars(min="red",max="blue")))))

## Conditionally assign fill colors to groups
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 mutate(flower_cols = case_when(
   Species == "setosa" ~ "purple",
   Species == "versicolor" ~ "darkgreen",
   Species == "virginica" ~ "orange",
   TRUE ~ "grey"
 )) %>%
 reactable(.,
 columns = list(flower_cols = colDef(show=FALSE),
 sepal_length = colDef(cell = react_sparkbar(.,
 height = 80,
 fill_color_ref = "flower_cols"))))

## Add labels to particular bars
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkbar(.,
 height = 80,
 decimals = 1,
 highlight_bars = highlight_bars(first="blue",last="red"),
 labels = c("first","last")))))

## Add statline to show the mean for each sparkbar
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
```

```
 columns = list(sepal_length = colDef(cell = react_sparkbar(.,
 height = 80,
 decimals = 1,
 statline = "mean"))))

## Combine multiple elements together
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkbar(.,
 height = 80,
 decimals = 1,
 statline = "mean",
 bandline = "innerquartiles"))))

## End(Not run)
```

`react_sparkline` *Add a sparkline line chart a reactable table*

Description

The 'react_sparkline()' function utilizes the dataui package <https://github.com/timelyportfolio/dataui> to create an interactive sparkline line chart. The data provided must be in a list format. The vertical height of the sparkline can be adjusted with 'height'. By default, the height is matched to the height of a cell in a reactable table. However, when min/max/all labels are applied, the height is auto-increased to better show the labels. Further adjustment of the height may be needed to better see the patterns in the data. The four-sided margin around the sparkline can be controlled with 'margin()'. When labels are added to the sparklines, the margin will auto-adjust (in most instances) to be able to display those labels. If the labels contain large values or values with many digits, the left and right margins may need to be increased slightly for the full numbers to be visible. By default, the sparkline line (the line that connects the data points) is shown but can be hidden by setting 'show_line' to FALSE. The line color, line width, and line curve can be controlled with 'line_color', 'line_width', and 'line_curve' respectively. The filled area beneath the line can be shown by setting 'show_area' to TRUE. When the area is shown, the area color can be controlled with 'area_color' or 'area_color_ref' and opacity can be controlled with 'area_opacity'. 'statline' can be used to show a horizontal dotted line that represents either the mean, median, min, or max (your choice). The appearance of the statline and statline labels can be controlled with 'statline_color' and 'statline_label_size'. A bandline can be added by using 'bandline'. The options are innerquartiles which highlights the innerquartiles of the data or range which highlights the full range of the data. By default, 'react_sparkline()' is interactive and data points will be shown when hovering over the sparklines. This can be turned off by setting 'tooltip' to FALSE. There are two tooltip types available within 'tooltip_type'. The size and color of the tooltip labels can be adjusted with 'tooltip_size' and 'tooltip_color'. Also by default, there are no labels on the line itself. However, one could add labels to the first, last, min, max, or all values within 'labels'. The labels that are shown on the sparkline and in the tooltip are automatically rounded to the nearest whole integer. But decimals can be shown by providing the number of decimal places in 'decimals'. The minimum value of a

data series is the minimum value shown for a sparkline, but this can be adjusted with ‘min_value‘ and the max can be adjusted with ‘max_value‘. ‘react_sparkline()‘ should be placed within the cell argument in reactable::colDef.

Usage

```
react_sparkline(
  data,
  height = 22,
  show_line = TRUE,
  line_color = "slategray",
  line_color_ref = NULL,
  line_width = 1,
  line_curve = "cardinal",
  highlight_points = NULL,
  point_size = 1.1,
  labels = "none",
  label_size = "0.8em",
  decimals = 0,
  min_value = NULL,
  max_value = NULL,
  show_area = FALSE,
  area_color = NULL,
  area_color_ref = NULL,
  area_opacity = 0.1,
  statline = NULL,
  statline_color = "red",
  statline_label_size = "0.8em",
  bandline = NULL,
  bandline_color = "red",
  bandline_opacity = 0.2,
  tooltip = TRUE,
  tooltip_type = 1,
  tooltip_color = NULL,
  tooltip_size = "1.1em",
  margin = NULL
)
```

Arguments

data	Dataset containing a column with numeric values in a list format.
height	Height of the sparkline. Default is 22.
show_line	Logical: show or hide the line. Default is TRUE.
line_color	The color of the line. Default is slategray.
line_color_ref	Optionally assign line colors from another column by providing the name of the column containing the colors in quotes. Only one color can be provided per row. Default is NULL.
line_width	Width of the line. Default is 1.

`line_curve` The curvature of the line. Options are 'cardinal', 'linear', 'basis', or 'monotoneX'. Default is 'cardinal'.

`highlight_points` Use 'highlight_points()' to assign colors to particular points. Colors can be assigned to all, min, max, first, or last points. By default, transparent colors are assigned to each point.

`point_size` Size of the points. Must first assigned colors to point(s) using 'highlight_points'. Default is 1.1.

`labels` Show labels for points of interest. Options are 'min', 'max', 'first', 'last', 'all', or 'none'. Default is 'none'.

`label_size` Size of the labels. Default is '0.8em'.

`decimals` The number of decimals displayed in the labels and tooltip. Default is 0.

`min_value` The minimum value of the sparkline range. Default is NULL (automatically the minimum value of each sparkline series).

`max_value` The maximum value of the sparkline range. Default is NULL (automatically the maximum value of each sparkline series).

`show_area` Logical: show or hide area beneath line. Default is FALSE.

`area_color` The color of the area. 'show_area' must be set to TRUE for color to be shown. Default is NULL (inherited from line_color).

`area_color_ref` Optionally assign area colors from another column by providing the name of the column containing the colors in quotes. Only one area color can be provided per row. Default is NULL. Default is FALSE.

`area_opacity` A value between 0 and 1 that adjusts the opacity. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 0.1.

`statline` Inserts a horizontal dotted line representing a statistic, and places the value of that statistic to the right of the line. Options are 'mean', 'median', 'min', or 'max'. Default is NULL.

`statline_color` The color of the horizontal dotted statline. Default is red.

`statline_label_size` The size of the label to the right of the statline. Default is '0.8em'.

`bandline` Inserts a horizontal bandline to render ranges of interest. Options are 'innerquartiles' or 'range' (min to max). Default is NULL.

`bandline_color` The color of the bandline. Default is red.

`bandline_opacity` A value between 0 and 1 that adjusts the opacity. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 0.2.

`tooltip` Logical: turn the tooltip on or off. Default is TRUE.

`tooltip_type` The tooltip type. Options are 1 or 2. Default is 1.

`tooltip_color` The color of the tooltip labels. Default is NULL.

`tooltip_size` The size of the tooltip labels. Default is '1.1em'.

`margin` The four-sided margin around the sparkline. Use margin() to assign the top, right, bottom, and left margins.

Value

a function that creates a sparkline chart from a column containing a list of values.

Examples

```
## Not run:
## Default sparkline line chart
library(dplyr)
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkline(.))))

## Highlight min and max data points
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(
 sepal_length = colDef(cell = react_sparkline(.,
 decimals = 1,
 highlight_points = highlight_points(min="red",max="blue")))))

## Add labels to data points and change curvature of line
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkline(.,
 line_curve = "linear",
 decimals = 1,
 highlight_points = highlight_points(first="orange",last="blue"),
 labels = c("first","last")))))

## Conditionally assign line colors to groups
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 mutate(flower_cols = case_when(
   Species == "setosa" ~ "purple",
   Species == "versicolor" ~ "darkgreen",
   Species == "virginica" ~ "orange",
   TRUE ~ "grey"
 )) %>%
 reactable(.,
 columns = list(flower_cols = colDef(show=FALSE),
 sepal_length = colDef(cell = react_sparkline(.,
 height = 80,
 line_color_ref = "flower_cols"))))

## Show area beneath the line
```

```
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkline(.,
 height = 80,
 line_color = "dodgerblue",
 show_area = TRUE))))

## Conditionally assign colors to the area below the line
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 mutate(flower_cols = case_when(
   Species == "setosa" ~ "purple",
   Species == "versicolor" ~ "darkgreen",
   Species == "virginica" ~ "orange",
   TRUE ~ "grey"
 )) %>%
 reactable(.,
 columns = list(flower_cols = colDef(show=FALSE),
 sepal_length = colDef(cell = react_sparkline(.,
 height = 80,
 show_area = TRUE,
 line_color_ref = "flower_cols",
 area_color_ref = "flower_cols"))))

## Add bandline to show innerquartile range
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkline(.,
 height = 80,
 decimals = 1,
 highlight_points = highlight_points(max="red"),
 labels = c("max"),
 bandline = "innerquartiles",
 bandline_color = "darkgreen"))))

## Add statline to show the mean for each sparkline
iris %>%
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkline(.,
 height = 80,
 decimals = 1,
 statline = "mean",
 statline_color = "red"))))

## Combine multiple elements together
iris %>%
```

```
 group_by(Species) %>%
 summarize(sepal_length = list(Sepal.Length)) %>%
 reactable(.,
 columns = list(sepal_length = colDef(cell = react_sparkline(.,
 height = 80,
 decimals = 1,
 statline = "mean",
 statline_color = "red",
 bandline = "innerquartiles",
 bandline_color = "darkgreen"))))

## End(Not run)
```

`sandstone` *Theme sandstone*

Description

Bootstrap-inspired sandstone theme

Usage

```
sandstone(
  font_size = 15,
  font_color = "#3e3f3a",
  header_font_size = 16,
  header_font_color = "#7c7a78",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

`font_size`	Numeric value representing the size of the font within the table (in px). Default is 15.
`font_color`	Color of the font for the text within the table and the group headers. Default is #3e3f3a.
`header_font_size`	Numeric value representing the size of the font within the table (in px). Default is 16.
`header_font_color`	Color of the font for the header text. Default is #7c7a78.
`cell_padding`	Numeric value representing the padding size between cells (in px). Default is 6.
`centered`	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard sandstone theme
reactable(data,
          theme = sandstone())

## Additional options applied
reactable(data,
          theme = sandstone(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

sanfran *Theme sanfran*

Description

San Francisco Chronicles-inspired table theme

Usage

```
sanfran(
  font_size = 14,
  font_color = "#222222",
  header_font_size = 15,
  header_font_color = "#212121",
  cell_color = "#f5f5f5",
  cell_border_width = 6,
  cell_border_color = "#ffffff",
  cell_padding = 6,
  pagination_color = "#222222",
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table. Default is #222222.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is transparent
cell_color	Color of the background of the cells. Default is #f5f5f5.
cell_border_width	Numeric value representing the border width of the cells. Default is 6.

`cell_border_color`
: Numeric value representing the border color of the cells. Default is #ffffff.

`cell_padding`
: Numeric value representing the padding size between cells (in px). Default is 6.

`pagination_color`
: Color of the pagination below the table. Default is #222222.

`centered`
: Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard void theme
reactable(data,
          theme = sanfran())

## Additional options applied
reactable(data,
          theme = sanfran(font_size = 12, font_color = "grey"))
```

---

save_reactable_test *Save a reactable table as an image or .html file*

Description

The 'save_reactable()' function converts either a reactable table, .html file, or .Rmd file to an image or .html file and saves it in the user's working directory. Table can be saved as either a .png file or .html file. Other file types are not currently supported. If the reactable table is located within an .Rmd file and has additional CSS styles provided, specify the name of the .Rmd file as the input. Alternatively, if the reactable table exists in an .html file, specify the name of the .html file as the input. Additional parameters available within webshot::webshot such as vwidth, vheight, and cliprect can be passed through 'save_reactable()'.

Usage

```
save_reactable_test(input, output, ...)
```

Arguments

`input`
: A reactable table, .html file, or .Rmd file

`output`
: A .png or .html file name for the saved image

`...`
: Optional additional parameters passed from webshot::webshot

Value

a function that converts a reactable table, .html file, or .Rmd file to an .png file or .html file and saves it in the user's working directory.

Examples

```
## Not run:
## Save reactable table as a png file:
iris_table <- reactable(iris)
save_reactable(iris_table, "iris_table.png")

## Also works with a pipe
iris_table %>%
save_reactable("iris_table.png")

## Or save as an html file:
save_reactable(iris_table, "iris_table.html")

## If the reactable table was built in R Markdown with CSS styles applied,
## specify .Rmd file as input and save_reactable will run the file
## and save the output as an image
save_reactable("iris_table.Rmd", "iris_table.png")

## Alternatively, can do the same with an .html file
save_reactable("iris_table.html", "iris_table.png")

## End(Not run)
```

---

slate *Theme slate*

Description

Bootstrap-inspired slate theme

Usage

```
slate(
  font_size = 15,
  font_color = "#aaaaaa",
  header_font_size = 16,
  header_font_color = "#97999b",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
font_color	Color of the font for the text within the table and the group headers. Default is #aaaaaa.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 16.
header_font_color	Color of the font for the header text. Default is #97999b.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard slate theme
reactable(data,
          theme = slate())

## Additional options applied
reactable(data,
          theme = slate(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

spacelab *Theme spacelab*

Description

Bootstrap-inspired spacelab theme

Usage

```
spacelab(
  font_size = 14,
  font_color = "#8e8e8e",
  header_font_size = 15,
  header_font_color = "#8e8e8e",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table and the group headers. Default is #8e8e8e.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #8e8e8e.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard spacelab theme
reactable(data,
          theme = spacelab())

## Additional options applied
reactable(data,
          theme = spacelab(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

sunrise *Theme sunrise*

Description

sunrise table theme

Usage

```
sunrise(
  font_size = 15,
  font_color = "#8069ff",
  header_font_size = 15,
  header_font_color = "#8069ff",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
font_color	Color of the font for the text within the table and the group headers. Default is #8069ff.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #8069ff.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard sunrise theme
reactable(data,
          theme = sunrise())

## Additional options applied
reactable(data,
          theme = sunrise(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

superhero *Theme superhero*

Description

Bootstrap-inspired superhero theme

Usage

```
superhero(
  font_size = 14,
  font_color = "#ebebeb",
  header_font_size = 15,
  header_font_color = "#ebebeb",
  cell_padding = 6,
  centered = FALSE
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table and the group headers. Default is #ebebeb.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.
header_font_color	Color of the font for the header text. Default is #ebebeb.
cell_padding	Numeric value representing the padding size between cells (in px). Default is 6.
centered	Logical: vertically center the contents of the table. Default is FALSE.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard superhero theme
reactable(data,
          theme = superhero())

## Additional options applied
reactable(data,
          theme = superhero(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

tooltip *Apply a tooltip to cells.*

Description

Apply a tooltip to cells.

Usage

```
tooltip(data, number_fmt = NULL)
```

Arguments

data	Null.
number_fmt	Optionally format numbers using formats from the scales package. Default is NULL.

Value

a function that applies a tooltip to a reactable table.

Examples

```
## Not run:
data <- iris[10:29, ]

## Apply a tooltip to color_scales()
reactable(data,
 columns = list(
 Petal.Length = colDef(
 cell = tooltip(),
 style = color_scales(data))
 ))

## End(Not run)
```

---

void *Theme void*

Description

A table style completely void of borders and headers

Usage

```
void(
  font_size = 14,
  font_color = "#222222",
  header_font_size = 15,
  header_font_color = "transparent",
  border_color = "transparent",
  border_width = 0,
  header_border_color = "transparent",
  header_border_width = 0,
  centered = FALSE,
  cell_padding = 6
)
```

Arguments

font_size	Numeric value representing the size of the font within the table (in px). Default is 14.
font_color	Color of the font for the text within the table. Default is #222222.
header_font_size	Numeric value representing the size of the font within the table (in px). Default is 15.

header_font_color
: Color of the font for the header text. Default is transparent

border_color
: Color of the borders between cells. Default is transparent.

border_width
: Numeric value representing the border width between cells (in px). Default is 0.

header_border_color
: Color of the bottom border of the header. Default is transparent.

header_border_width
: Numeric value representing the bottom border width of the header (in px). Default is 0.

centered
: Logical: vertically center the contents of the table. Default is FALSE.

cell_padding
: Numeric value representing the padding size between cells (in px). Default is 6.

Value

an object of class theme that is applied to a reactable table.

Examples

```
data <- iris[10:29, ]

## Standard void theme
reactable(data,
          theme = void())

## Additional options applied
reactable(data,
          theme = void(font_size = 12, font_color = "grey", cell_padding = 3))
```

Index