Package ‘apexcharter’

March 31, 2020

Version 0.1.4
Title Create Interactive Chart with the JavaScript 'ApexCharts'
Library
Description Provides an 'htmlwidgets' interface to 'apexcharts.js'.
'Apexcharts' is a modern JavaScript charting library to build interactive charts and visualizations with simple API.
'Apexcharts' examples and documentation are available here: <https://apexcharts.com/>.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
ByteCompile true
Depends R (>= 2.10)
Imports htmlwidgets, magrittr, rlang, ggplot2, jsonlite, shiny (>= 1.1.0)
Suggests testthat, dplyr, knitr, scales, rmarkdown, gapminder, highcharter
RoxygenNote 7.1.0
URL https://github.com/dreamRs/apexcharter,
https://dreamrs.github.io/apexcharter
BugReports https://github.com/dreamRs/apexcharter/issues
VignetteBuilder knitr
NeedsCompilation no
Author Victor Perrier [aut, cre],
Fanny Meyer [aut],
Juned Chhipa [cph] (apexcharts.js library),
Mike Bostock [cph] (d3.format library)
Maintainer Victor Perrier <victor.perrier@dreamrs.fr>
Repository CRAN
Date/Publication 2020-03-31 09:50:10 UTC
## R topics documented:

- `apexcharter-package` .............................................. 3
- `apex` .................................................................. 3
- `apexchart` ........................................................... 5
- `apexcharter-exports` .............................................. 6
- `apexcharter-shiny` ................................................. 6
- `apexchartProxy` ..................................................... 8
- `ax-series` ............................................................ 8
- `ax_annotations` ..................................................... 9
- `ax_chart` ............................................................ 12
- `ax_colors` ........................................................... 15
- `ax_dataLabels` ....................................................... 16
- `ax_fill` ............................................................... 17
- `ax_grid` .............................................................. 19
- `ax_labels` ............................................................ 21
- `ax_labs` ............................................................... 22
- `ax_legend` ........................................................... 22
- `ax_markers` .......................................................... 24
- `ax_plotOptions` ...................................................... 26
- `ax_proxy_options` .................................................. 27
- `ax_proxy_series` ................................................... 29
- `ax_responsive` ...................................................... 30
- `ax_states` ............................................................ 31
- `ax_stroke` ............................................................ 32
- `ax_subtitle` ........................................................... 34
- `ax_theme` ............................................................. 35
- `ax_title` ............................................................. 36
- `ax_tooltip` ........................................................... 38
- `ax_xaxis` ............................................................. 40
- `ax_yaxis` ............................................................. 42
- `ax_yaxis2` ............................................................. 44
- `bar_opts` .............................................................. 45
- `config_update` ....................................................... 46
- `event_opts` .......................................................... 47
- `format_date` ........................................................ 47
- `format_num` ........................................................ 49
- `heatmap_opts` ........................................................ 51
- `parse_df` .............................................................. 52
- `pie_opts` ............................................................. 53
- `radialBar_opts` ...................................................... 54
- `run_input_demo` ..................................................... 56
- `set_input_click` ..................................................... 56
- `set_input_selection` ............................................... 57
- `set_input_zoom` ..................................................... 59
- `unhcr_popstats_2017` ............................................. 60
- `unhcr_ts` ............................................................ 60

**Index**

62
**Description**

This package allows you to use ApexCharts.js ([https://apexcharts.com/](https://apexcharts.com/)), to create interactive and modern SVG charts.

**Author(s)**

Victor Perrier (@dreamRs_fr)

---

**Quick ApexChart**

**Description**

Initialize a chart with three main parameters: data, mapping, and type of chart.

**Usage**

```r
apex(
  data,
  mapping,
  type = "column",
  ..., 
  auto_update = TRUE,
  serie_name = NULL,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

**Arguments**

- **data**
  Default dataset to use for chart. If not already a `data.frame`, it will be coerced to with `as.data.frame`.

- **mapping**
  Default list of aesthetic mappings to use for chart

- **type**

- **...**
  Other arguments passed on to methods. Not currently used.

- **auto_update**
  In Shiny application, update existing chart rather than generating new one. Can be TRUE/FALSE or use `config_update` for more control.
serie_name Name for the serie displayed in tooltip, only used for single serie.
width A numeric input in pixels.
height A numeric input in pixels.
elemntId Use an explicit element ID for the widget.

Value
A apexcharts htmlwidget object.

Examples
library(ggplot2)
library(dplyr)
library(apexcharter)

# make a barchart with a frequency table
data("mpg", package = "ggplot2")
apex(
  data = count(mpg, manufacturer),
  mapping = aes(x = manufacturer, y = n),
  type = "bar"
)

# timeseries
data("economics", package = "ggplot2")
apex(
  data = economics,
  mapping = aes(x = date, y = uempmed),
  type = "line"
)

# you can add option to apex result :
apex(
  data = economics,
  mapping = aes(x = date, y = uempmed),
  type = "line"
) %>%
  ax_stroke(width = 1)

# with group variable
data("economics_long", package = "ggplot2")
apex(
  data = economics_long,
  mapping = aes(x = date, y = value01, group = variable),
  type = "line"
)
apexchart Create an apexcharts.js widget

Description

Create an apexcharts.js widget

Usage

apexchart(
  ax_opts = list(),
  auto_update = TRUE,
  width = NULL,
  height = NULL,
  elementId = NULL
)

Arguments

ax_opts A list in JSON format with chart parameters.
auto_update In Shiny application, update existing chart rather than generating new one. Can be TRUE/FALSE or use config_update for more control.
width A numeric input in pixels.
high height A numeric input in pixels.
lelemen elementId Use an explicit element ID for the widget.

Value

A apexcharts htmlwidget object.

Examples

library(apexchart)

# Use raw API by passing a list of
# parameters to the function

apexchart(ax_opts = list(
  chart = list(
    type = "bar"
  ),
  series = list(list(
    name = "Example",
    data = sample(1:100, 5)
  )),
  xaxis = list(
    categories = LETTERS[1:5]
  )
)),
# Or use apexchart() to initialize the chart
# before passing parameters

```r
apexchart() %>%
  ax_chart(type = "bar") %>%
  ax_series(
    list(
      name = "Example",
      data = sample(1:100, 5)
    )
  ) %>%
  ax_xaxis(
    categories = LETTERS[1:5]
  )
```

---

**apexcharter-exports**  
*apexcharter exported operators and S3 methods*

**Description**

The following functions are imported and then re-exported from the apexcharter package to avoid listing the magrittr as Depends of apexcharter.

---

**apexcharter-shiny**  
*Shiny bindings for apexcharter*

**Description**

Output and render functions for using apexcharter within Shiny applications and interactive Rmd documents.

**Usage**

```r
apexchartOutput(outputId, width = "100\%", height = "400px")
renderApexchart(expr, env = parent.frame(), quoted = FALSE)
```
Arguments

outputId output variable to read from
width, height Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr An expression that generates a apexchart
env The environment in which to evaluate expr.
quoted Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

An Apexchart output that can be included in the application UI.

Examples

```r
if (interactive()) {
  library(shiny)
  library(apexcharter)

  ui <- fluidPage(
    fluidRow(
      column(
        width = 8, offset = 2,
        h2("Apexchart in Shiny"),
        drawButton("redraw", "Redraw chart"),
        apexchartOutput("chart")
      )
    )
  )

  server <- function(input, output, session) {
    output$chart <- renderApexchart({
      input$redraw
      apexchart() %>%
      ax_chart(type = "bar") %>%
      ax_series(
        list(
          name = "Example",
          data = sample(1:100, 5)
        )
      ) %>%
      ax_xaxis(
        categories = LETTERS[1:5]
      )
    })
  }

  shinyApp(ui, server)
}
```
apexchartProxy  
Proxy for apexchart

Description
Allow to update a chart in Shiny application.

Usage
apexchartProxy(shinyId, session = shiny::getDefaultReactiveDomain())

Arguments
- shinyId: single-element character vector indicating the output ID of the chart to modify (if invoked from a Shiny module, the namespace will be added automatically)
- session: the Shiny session object to which the chart belongs; usually the default value will suffice

ax-series  
Add data to a chart

Description
Add data to a chart

Usage
ax_series(ax, ...)
ax_series2(ax, l)

Arguments
- ax: A apexcharts htmlwidget object.
- ...: Lists containing data to plot, typically list with two items: name and data.
- l: A list.

Value
A apexcharts htmlwidget object.
Examples

# One serie
```
apexchart() %>%
  ax_series(list(
    name = "rnorm",
    data = rnorm(10)
  ))
```

# Two series
```
apexchart() %>%
  ax_series(
    list(
      name = "rnorm 1",
      data = rnorm(10)
    ),
    list(
      name = "rnorm 2",
      data = rnorm(10)
    )
  )
```

---

**ax_annotations**  
Annotations properties

**Description**
Annotations properties

**Usage**
```
ax_annotations(
  ax,  
  position = NULL, 
  yaxis = NULL, 
  xaxis = NULL, 
  points = NULL, 
  ...  
)
```

**Arguments**
- **ax**  
  A apexcharts htmlwidget object.
- **position**  
  Whether to put the annotations behind the charts or in front of it. Available Options: "front" or "back".
- **yaxis**  
  List of lists.
- **xaxis**  
  List of lists.
- **points**  
  List of lists.
- **...**  
  Additional parameters.
Value

A apexcharts htmlwidget object.

Note

See https://apexcharts.com/docs/options/annotations/.

Examples

data("economics", package = "ggplot2")

# Horizontal line
apex(
  data = tail(economics, 200),
  mapping = aes(x = date, y = uempmed),
  type = "line"
) %>%
ax_annotations(
  yaxis = list(list(
    y = 11.897,
    borderColor = "firebrick",
    opacity = 1,
    label = list(
      text = "Mean uempmed",
      position = "left",
      textAnchor = "start"
    )
  )))

# Vertical line
apex(
  data = tail(economics, 200),
  mapping = aes(x = date, y = uempmed),
  type = "line"
) %>%
ax_annotations(
  xaxis = list(list(
    x = htmlwidgets::JS("new Date('1 Mar 2007').getTime()"),
    strokeDashArray = 0,
    borderColor = "#775DD0",
    label = list(
      text = "A label",
      borderColor = "#775DD0",
      style = list(
        color = "#fff",
        background = "#775DD0"
      )
    )
  )))
# Vertical range

```r
apex(
  data = tail(economics, 200),
  mapping = aes(x = date, y = uempmed),
  type = "line"
) %>%
  ax_annotations(
    xaxis = list(list(
      x = htmlwidgets::JS("new Date('1 Jan 2009').getTime()"),
      x2 = htmlwidgets::JS("new Date('1 Feb 2010').getTime()"),
      fillColor = "#B3F7CA",
      opacity = 0.4,
      label = list(
        text = "A label",
        borderColor = "#B3F7CA",
        style = list(
          color = "#fff",
          background = "#B3F7CA"
        )
      )
    )
  )
)
```

# Point annotation

```r
apex(
  data = tail(economics, 200),
  mapping = aes(x = date, y = uempmed),
  type = "line"
) %>%
  ax_annotations(
    points = list(list(
      x = htmlwidgets::JS("new Date('1 Jun 2010').getTime()"),
      y = 25.2,
      marker = list(
        size = 8,
        fillColor = "#fff",
        strokeColor = "red",
        radius = 2
      ),
      label = list(
        text = "Highest",
        offsetY = 0,
        borderColor = "#FF4560",
        style = list(
          color = "#fff",
          background = "#FF4560"
        )
      )
    ))
)
```
**Description**

Chart parameters

**Usage**

```r
ax_chart(
  ax,
  type = NULL,
  stacked = NULL,
  stackType = NULL,
  defaultLocale = NULL,
  locales = NULL,
  animations = NULL,
  background = NULL,
  foreColor = NULL,
  dropShadow = NULL,
  events = NULL,
  offsetX = NULL,
  offsetY = NULL,
  selection = NULL,
  sparkline = NULL,
  toolbar = NULL,
  zoom = NULL,
  width = NULL,
  height = NULL,
  ...  
)
```

**Arguments**

- `ax` A `apexcharts htmlwidget` object.
- `type` Specify the chart type. Available Options: "bar", "column", "line", "pie", "donut", "radialBar", "scatter", "bubble", "heatmap".
- `stacked` Logical. Enables stacked option for axis charts.
- `stackType` When stacked, should the stacking be percentage based or normal stacking. Available options: "normal" or "100%".
- `locales` Array of custom locales parameters.
- `animations` A list of parameters.
**background**  
Background color for the chart area. If you want to set background with css, use .apexcharts-canvas to set it.

**foreColor**  
Sets the text color for the chart. Defaults to #373d3f.

**dropShadow**  
A list of parameters. See https://apexcharts.com/docs/options/chart/dropshadow/.

**events**  
See events_OPTS.

**offsetX**  
Sets the left offset for chart.

**offsetY**  
Sets the top offset for chart.

**selection**  
A list of parameters.

**sparkline**  
List. Sparkline hides all the elements of the charts other than the primary paths. Helps to visualize data in small areas.

**toolbar**  
A list of parameters. See https://apexcharts.com/docs/options/chart/toolbar/.

**zoom**  
A list of parameters. See https://apexcharts.com/docs/options/chart/zoom/.

**width**  
Width of the chart.

**height**  
Height of the chart.

...  
Additional parameters.

**Value**

A apexcharts htmlwidget object.

**Examples**

```r
library(dplyr)
data("diamonds", package = "ggplot2")

## Stack bar type
# default is dodge
apex(
  data = count(diamonds, cut, color),
  mapping = aes(x = cut, y = n, fill = color)
)

# stack
apex(
  data = count(diamonds, cut, color),
  mapping = aes(x = cut, y = n, fill = color)
) %>%
  ax_chart(stacked = TRUE)

# stack filled
apex(
  data = count(diamonds, cut, color),
  mapping = aes(x = cut, y = n, fill = color)
) %>%
```
ax_chart(stacked = TRUE, stackType = "100%")

# Toolbar ----------------------------------------

# Hide the toolbar
apex(
    data = count(diamonds, cut, color),
    mapping = aes(x = cut, y = n, fill = color)
) %>%
    ax_chart(toolbar = list(show = FALSE))

# Hide download buttons
data("economics", package = "ggplot2")
apex(
    data = economics,
    mapping = aes(x = date, y = pce),
    type = "line"
) %>%
    ax_chart(
        toolbar = list(tools = list(download = FALSE))
    )

# Zoom -----------------------------------------

# Disable
apex(
    data = economics,
    mapping = aes(x = date, y = pce),
    type = "line"
) %>%
    ax_chart(
        zoom = list(enabled = FALSE)
    )

# Auto-scale Y axis
apex(
    data = economics,
    mapping = aes(x = date, y = pce),
    type = "line"
) %>%
    ax_chart(
        zoom = list(autoScaleYaxis = TRUE)
    )

# Localization ----------------------------------
# Use included localization config
dat <- data.frame(
  x = Sys.Date() + 1:20,
  y = sample.int(20, 20)
)

# French
apex(dat, aes(x, y), "line") %>%
ax_chart(defaultLocale = "fr")

# Italian
apex(dat, aes(x, y), "line") %>%
ax_chart(defaultLocale = "it")

# Custom config
apex(dat, aes(x, y), "line") %>%
ax_chart(locales = list(
  list(
    name = "en", # override 'en' locale
    options = list(
      toolbar = list(
        exportToSVG = "GET SVG",
        exportToPNG = "GET PNG"
      )
    )
  )
))

## ax_colors

<table>
<thead>
<tr>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Colors</td>
</tr>
</tbody>
</table>

Usage

`ax_colors(ax, ...)`

Arguments

- `ax` A `apexcharts htmlwidget` object.
- `...` Colors for the chart’s series. When all colors are used, it starts from the beginning.
Value

A apexcharts htmlwidget object.

Note

See https://apexcharts.com/docs/options/colors/

Examples

```r
library(dplyr)
data("diamonds", package = "ggplot2")

# Change default color(s)
apex(
  data = count(diamonds, cut),
  mapping = aes(x = cut, y = n)
) %>%
  ax_colors("#F7D358")

library(scales)
apex(
  data = count(diamonds, cut, color),
  mapping = aes(x = cut, y = n, fill = color)
) %>%
  ax_colors(brewer_pal(palette = "Set2")(7))
```

<table>
<thead>
<tr>
<th>ax_dataLabels</th>
<th>Labels on data</th>
</tr>
</thead>
</table>

Description

Labels on data

Usage

```r
ax_dataLabels(
  ax,
  enabled = NULL,
  textAnchor = NULL,
  offsetX = NULL,
  offsetY = NULL,
  style = NULL,
  dropShadow = NULL,
  formatter = NULL,
  ...
)
```
Arguments

ax A apexcharts htmlwidget object.
enabled To determine whether to show dataLabels or not.
textAnchor The alignment of text relative to dataLabel’s drawing position. Accepted values "start", "middle" or "end".
offsetX Sets the left offset for dataLabels.
offsetY Sets the top offset for dataLabels.
style A list of parameters.
dropShadow A list of parameters.
formatter The formatter function takes in a single value and allows you to format the value before displaying
... Additional parameters.

Value

A apexcharts htmlwidget object.

Note

See https://apexcharts.com/docs/options/datalabels/

Examples

library(dplyr)
data("diamonds", package = "ggplot2")

# Add data labels
apex(
data = count(diamonds, cut),
mapping = aes(x = cut, y = n)
) %>%
  ax_dataLabels(enabled = TRUE)
Usage

ax_fill(
  ax,
  type = NULL,
  colors = NULL,
  opacity = NULL,
  gradient = NULL,
  image = NULL,
  pattern = NULL,
  ...
)

Arguments

  ax       A apexcharts htmlwidget object.
  type     Whether to fill the paths with solid colors or gradient. Available options: "solid", "gradient", "pattern" or "image".
  colors   Colors to fill the svg paths.
  opacity  Opacity of the fill attribute.
  gradient A list of parameters.
  image    A list of parameters.
  pattern  A list of parameters.
  ...      Additional parameters.

Value

  A apexcharts htmlwidget object.

Note

  See https://apexcharts.com/docs/options/fill/

Examples

library(dplyr)
data("diamonds", package = "ggplot2")

  # Use a pattern to fill bars
  apex(
    data = count(diamonds, cut, color),
    mapping = aes(x = color, y = n, fill = cut)
  )

  ax_fill(
    type = "pattern",
    opacity = 1,
    pattern = list(
      style = c("circles", "slantedLines", "verticalLines", "horizontalLines", "squares")
    )
  )
### ax_grid

Add grids on chart

**Description**

Add grids on chart

**Usage**

```r
ax_grid(
  ax,  
  show = NULL,  
  borderColor = NULL,  
  strokeDashArray = NULL,  
  position = NULL,  
  xaxis = NULL,  
  yaxis = NULL,  
  row = NULL,  
  column = NULL,  
  padding = NULL,  
  ...  
)
```

**Arguments**

- **ax**
  A apexcharts htmlwidget object.

- **show**
  Logical. To show or hide grid area (including xaxis / yaxis)

- **borderColor**
  Colors of grid borders / lines.
strokeDashArray  Creates dashes in borders of svg path. Higher number creates more space between dashes in the border.

position  Whether to place grid behind chart paths of in front. Available options for position: "front" or "back"

xaxis  A list of parameters.

yaxis  A list of parameters.

row  A list of parameters.

column  A list of parameters.

padding  A list of parameters.

...  Additional parameters.

Value

A apexcharts htmlwidget object.

Note

See https://apexcharts.com/docs/options/grid/

Examples

library(dplyr)
data("mpg", package = "ggplot2")

# Hide Y-axis and gridelines
apex(
  data = count(mpg, manufacturer),
  mapping = aes(x = manufacturer, y = n)
)

# just grid lines
apex(
  data = count(mpg, manufacturer),
  mapping = aes(x = manufacturer, y = n)
)

# both x & y
data("economics", package = "ggplot2")
apex(
  data = economics,
  mapping = aes(x = date, y = psavert),
  type = "line"
)

ax_grid(
  yaxis = list(lines = list(show = TRUE)),
  xaxis = list(lines = list(show = TRUE))
)
**ax_labels**

**Alternative axis labels**

**Description**

Alternative axis labels

**Usage**

```r
ax_labels(ax, ...)  
ax_labels2(ax, labels)
```

**Arguments**

- `ax` A `apexcharts htmlwidget` object.
- `...` Vector. In Axis Charts (line / column), labels can be set instead of setting xaxis categories option. While, in pie/donut charts, each label corresponds to value in series array.
- `labels` A vector to use as labels.

**Value**

A `apexcharts htmlwidget` object.

**Note**

See [https://apexcharts.com/docs/options/labels/](https://apexcharts.com/docs/options/labels/)

**Examples**

```r
apexchart() %>%  
  ax_chart(type = "pie") %>%  
  ax_series(23, 45, 56) %>%  
  ax_labels("A", "B", "C")

# same as
apexchart() %>%  
  ax_chart(type = "pie") %>%  
  ax_series2(c(23, 45, 56)) %>%  
  ax_labels2(c("A", "B", "C"))
```
ax_labs

Modify axis, legend, and chart labels

Description

Modify axis, legend, and chart labels

Usage

ax_labs(ax, title = NULL, subtitle = NULL, x = NULL, y = NULL)

Arguments

ax
A apexcharts htmlwidget object.
title
Text for the title.
subtitle
Text for the subtitle.
x
Text for the x-axis label.
y
Text for the y-axis label.

Examples

meteo_paris <- data.frame(
  month = month.name,
  tmax = c(7, 8, 12, 15, 19, 23, 25, 25, 21, 16, 11, 8),
  tmin = c(3, 3, 5, 7, 11, 14, 16, 16, 13, 10, 6, 3)
)

apex(meteo_paris, type = "column", aes(x = month, y = tmin)) %>%
  ax_labs(
    title = "Average minimal temperature in Paris",
    subtitle = "Data from NOAA",
    x = "Month",
    y = "Temperature (°C)"
  )

ax_legend

Legend properties

Description

Legend properties
Usage

ax_legend(
  ax,
  show = NULL,
  position = NULL,
  showForSingleSeries = NULL,
  showForNullSeries = NULL,
  showForZeroSeries = NULL,
  horizontalAlign = NULL,
  fontSize = NULL,
  textAnchor = NULL,
  offsetY = NULL,
  offsetX = NULL,
  formatter = NULL,
  labels = NULL,
  markers = NULL,
  itemMargin = NULL,
  containerMargin = NULL,
  onItemClick = NULL,
  onItemHover = NULL,
  floating = NULL,
  ...
)

Arguments

ax A apexcharts htmlwidget object.
show Logical. Whether to show or hide the legend container.
position Available position options for legend: "top", "right", "bottom", "left".
showForSingleSeries Show legend even if there is just 1 series.
showForNullSeries Allows you to hide a particular legend if it’s series contains all null values.
showForZeroSeries Allows you to hide a particular legend if it’s series contains all 0 values.
horizontalAlign Available options for horizontal alignment: "right", "center", "left".
fontSize Sets the fontSize of legend text elements
textAnchor The alignment of text relative to legend’s drawing position
offsetY Sets the top offset for legend container.
offsetX Sets the left offset for legend container.
formatter JS function. A custom formatter function to append additional text to the legend series names.
labels List with two items "foreColor" (Custom text color for legend labels) and "useSeriesColors" (Logical, whether to use primary colors or not)
markers List.
itemMargin List with two items "horizontal" (Horizontal margin for individual legend item) and "vertical" (Vertical margin for individual legend item).
containerMargin List with two items "top" (Top margin for the whole legend container) and "left" (Left margin for the whole legend container).
onItemClick List with item "toggleDataSeries", logical, when clicked on legend item, it will toggle the visibility of the series in chart.
onItemHover List with item "highlightDataSeries", logical, when hovered on legend item, it will highlight the paths of the hovered series in chart.
floating Logical. The floating option will take out the legend from the chart area and make it float above the chart.
... Additional parameters.

Value
A apexcharts htmlwidget object.

Note
See https://apexcharts.com/docs/options/legend/

Examples
library(dplyr)
data("mpg", package = "ggplot2")

# Legend position
apex(
  data = count(mpg, manufacturer, year),
  mapping = aes(x = manufacturer, y = n, fill = year)
) %>%
  ax_legend(position = "right")

# hide legend
apex(
  data = count(mpg, manufacturer, year),
  mapping = aes(x = manufacturer, y = n, fill = year)
) %>%
  ax_legend(show = FALSE)

ax_markers

Markers properties
### ax_markers

#### Usage

```r
ax_markers(
    ax,
    size = NULL,
    colors = NULL,
    strokeColor = NULL,
    strokeWidth = NULL,
    strokeOpacity = NULL,
    fillOpacity = NULL,
    shape = NULL,
    radius = NULL,
    offsetX = NULL,
    offsetY = NULL,
    hover = NULL,
    ...
)
```

#### Arguments

- **ax**: A `apexcharts htmlwidget` object.
- **size**: Numeric. Size of the marker point.
- **colors**: Sets the fill color(s) of the marker point.
- **strokeColor**: Stroke Color of the marker.
- **strokeWidth**: Stroke Size of the marker.
- **strokeOpacity**: Opacity of the border around marker.
- **fillOpacity**: Opacity of the marker fill color.
- **shape**: Shape of the marker. Available Options for shape: "square" or "circle".
- **radius**: Numeric. Radius of the marker (applies to square shape)
- **offsetX**: Numeric. Sets the left offset of the marker.
- **offsetY**: Numeric. Sets the top offset of the marker.
- **hover**: List with item size (Size of the marker when it is active).
- **...**: Additional parameters.

#### Value

A `apexcharts htmlwidget` object.

#### Note

See [https://apexcharts.com/docs/options/markers/](https://apexcharts.com/docs/options/markers/)
Examples

data("economics", package = "ggplot2")

# show points
apex(
  data = tail(economics, 20),
  type = "line",
  mapping = aes(x = date, y = uempmed)
) %>%
  ax_markers(size = 6)

---

**ax_plotOptions**

Specific options for chart

Description

Specific options for chart

Usage

```r
ax_plotOptions(
  ax,
  bar = NULL,
  heatmap = NULL,
  radialBar = NULL,
  pie = NULL,
  ...
)
```

Arguments

- **ax** A apexcharts htmlwidget object.
- **bar** See `bar_opts`.
- **heatmap** See `heatmap_opts`.
- **radialBar** See `radialBar_opts`.
- **pie** See `pie_opts`.
- **...** Additional parameters.

Value

A apexcharts htmlwidget object.
ax_proxy_options

Examples

```r
library(dplyr)
data("diamonds", package = "ggplot2")

# Stack bar type
apex(
data = count(diamonds, cut),
mapping = aes(x = cut, y = n)
) %>%
  ax_plotOptions(
    bar = bar_opts(endingShape = "rounded", columnWidth = "10%")
  )

# Pie
apex(
data = count(diamonds, cut),
mapping = aes(x = cut, y = n),
type = "pie"
) %>%
  ax_plotOptions(
    pie = pie_opts(customScale = 0.5)
  )

# Radial
apexchart() %>%
  ax_chart(type = "radialBar") %>%
  ax_plotOptions(
    radialBar = radialBar_opts(
      hollow = list(size = "70%")
    )
  ) %>%
  ax_series(70) %>%
  ax_labels("Indicator")
```

---

**ax_proxy_options**  
Proxy for updating options

**Description**

Allows you to update the configuration object.

**Usage**

`ax_proxy_options(proxy, options)`

**Arguments**

- `proxy` A apexchartProxy htmlwidget object.
- `options` New options to set.
Examples

```r
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    fluidRow(
      column(
        width = 8, offset = 2,
        tags$h2("Update options"),
        apexchartOutput(outputId = "chart"),
        checkboxInput(
          inputId = "show_label_xaxis",
          label = "Show x-axis labels"
        ),
        textInput(
          inputId = "yaxis_title",
          label = "Y-axis title"
        )
      )
    )
  )

  server <- function(input, output, session) {
    output$chart <- renderApexchart({
      apexchart() %>%
        ax_chart(type = "bar") %>%
        ax_series(list(
          name = "Example",
          data = c(23, 43, 76, 31)
        )) %>%
        ax_xaxis(
          categories = c("Label A", "Label B",
                         "Label C", "Label D")
        )
    })

    observe({
      apexchartProxy("chart") %>%
        ax_proxy_options(list(
          xaxis = list(
            labels = list(show = input$show_label_xaxis)
          ),
          yaxis = list(
            title = list(text = input$yaxis_title)
          )
        )
    })
  }

  shinyApp(ui, server)
```
**ax_proxy_series**

Proxy for updating series.

**Description**

Allows you to update the series array overriding the existing one.

**Usage**

```r
ax_proxy_series(proxy, newSeries, animate = TRUE)
```

**Arguments**

- **proxy**: A `apexchartProxy htmlwidget` object.
- **newSeries**: The series array to override the existing one.
- **animate**: Should the chart animate on re-rendering.

**Examples**

```r
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    fluidRow(
      column(
        width = 8, offset = 2,
        tags$h2("Real time chart"),
        apexchartOutput(outputId = "chart")
      )
    )
  )
  server <- function(input, output, session) {
    rv <- reactiveValues()
    rv$df <- data.frame(  
      date = Sys.Date() + 1:20,
      values = sample(10:90, 20, TRUE)
    )
    observe({
      invalidateLater(1000, session)
      df <- isolate(rv$df)
      # Append new line of data
      df <- rbind(
        df, data.frame(date = Sys.Date() + 1, values = sample(10:90, 10, TRUE))
      )
      # Update the chart with new data
      output$chart <- renderApexchart(
        data = df,
        options = list(
          chartType = "bar",
          chartHeight = 500,
          chartWidth = 1000,
          chartData = "chartData",
          chartLegend = TRUE
        ),
        animate = FALSE
      )
    })
  }
}
```
df, data.frame(
    date = df$date[length(df$date)] + 1,
    values = sample(10:90, 1, TRUE)
  )
)
rv$df <- df
})

output$chart <- renderApexchart({
  # Generate chart once
  apex(isolate(rv$df), aes(date, values), "spline") %>%
    ax_xaxis(
      range = 10 * 24 * 60 * 60 * 1000
      # Fixed range for x-axis: 10 days
      # days*hours*minutes*seconds*milliseconds
    )
  )
})

observe({
  # Update chart to add new data
  apexchartProxy("chart") %>%
    ax_proxy_series(
      parse_df(rv$df),
      T
    )
  )
})

shinyApp(ui, server)

---

**ax_responsive**

**Responsive options**

**Description**

Responsive options

**Usage**

`ax_responsive(ax, ...)`

**Arguments**

- `ax` A apexcharts htmlwidget object.
- `...` Additional parameters.
Values

A `apexcharts htmlwidget` object.

Note

See https://apexcharts.com/docs/options/responsive/

Examples

```r
library(dplyr)
data("mpg", package = "ggplot2")

# Open in browser and resize window
apex(
  data = count(mpg, manufacturer, year),
  mapping = aes(x = manufacturer, y = n, fill = year),
  type = "bar"
)

ax_legend(position = "right")

ax_responsive(
  list(
    breakpoint = 1000,
    options = list(
      plotOptions = list(
        bar = list(
          horizontal = FALSE
        ),
      ),
      legend = list(
        position = "bottom"
      )
    )
  )
)
```

---

**ax_states**

Charts’ states

### Description

Charts’ states

### Usage

`ax_states(ax, normal = NULL, hover = NULL, active = NULL, ...)`
Arguments

ax  A `apexcharts` htmlwidget object.
normal  A list of parameters.
hover  A list of parameters.
active  A list of parameters.
...  Additional parameters.

Value

A `apexcharts` htmlwidget object.

Note

See [https://apexcharts.com/docs/options/states/](https://apexcharts.com/docs/options/states/)

Examples

```r
library(dplyr)
data("mpg", package = "ggplot2")

# Inverse effect on hover
apex(
  data = count(mpg, manufacturer),
  mapping = aes(x = manufacturer, y = n),
  type = "bar"
) %>%
ax_states(
  hover = list(
    filter = list(
      type = "darken"
    )
  )
)
```

---

| **ax_stroke** | **Stroke properties** |

Description

Stroke properties
ax_stroke

Usage

ax_stroke(
  ax,
  show = NULL,
  curve = NULL,
  lineCap = NULL,
  width = NULL,
  colors = NULL,
  dashArray = NULL,
  ...
)

Arguments

ax      A apexcharts htmlwidget object.
show    Logical. To show or hide path-stroke / line
curve   In line / area charts, whether to draw smooth lines or straight lines. Available Options: "smooth" (connects the points in a curve fashion. Also known as spline) and "straight" (connect the points in straight lines.).
lineCap For setting the starting and ending points of stroke. Available Options: "butt" (ends the stroke with a 90-degree angle), "square" (similar to butt except that it extends the stroke beyond the length of the path) and "round" (ends the path-stroke with a radius that smooths out the start and end points)
width   Sets the width of border for svg path.
colors  Colors to fill the border for paths.
dashArray Creates dashes in borders of svg path. Higher number creates more space between dashes in the border.
...     Additional parameters.

Value

A apexcharts htmlwidget object.

Note

See https://apexcharts.com/docs/options/stroke/

Examples

data("economics", package = "ggplot2")
apex(
  data = economics,
  mapping = aes(x = date, y = uempmed),
  type = "line"
) %>%
ax_stroke(
  width = 1,
ax_subtitle

```
dashArray = 4

data("economics_long", package = "ggplot2")

apex(
  data = economics_long,
  mapping = aes(x = date, y = value01, group = variable),
  type = "line"
) %>%
  ax_stroke(
    width = c(1, 2, 3, 4, 5),
    dashArray = c(1, 2, 3, 4, 5)
  )
```

---

**Description**

Chart’s subtitle

**Usage**

```
ax_subtitle(
  ax,
  text = NULL,
  align = NULL,
  margin = NULL,
  offsetX = NULL,
  offsetY = NULL,
  floating = NULL,
  style = NULL,
  ...
)
```

**Arguments**

- **ax**: A apexcharts htmlwidget object.
- **text**: Text to display as a subtitle of chart.
- **align**: Alignment of subtitle relative to chart area. Possible Options: "left", "center" and "right".
- **margin**: Numeric. Vertical spacing around the subtitle text.
- **offsetX**: Numeric. Sets the left offset for subtitle text.
- **offsetY**: Numeric. Sets the top offset for subtitle text.
- **floating**: Logical. The floating option will take out the subtitle text from the chart area and make it float on top of the chart.
**ax_theme**

**style**

List with two items: `fontSize` (Font Size of the subtitle text) and `color` (Foreground color of the subtitle text).

... Additional parameters.

**Value**

A `apexcharts htmlwidget` object.

**Note**

See https://apexcharts.com/docs/options/subtitle/

**Examples**

```r
data("economics", package = "ggplot2")
apex(
  data = economics,
  mapping = aes(x = date, y = uempmed),
  type = "line"
) %>%
  ax_title(
    text = "Median duration of unemployment"
  ) %>%
  ax_subtitle(
    text = "in weeks"
  )
```

---

**Description**

Theme for charts

**Usage**

```r
ax_theme(ax, mode = c("light", "dark"), palette = NULL, monochrome = NULL, ...)
```

**Arguments**

- `ax` A `apexcharts htmlwidget` object.
- `mode` use light or dark theme.
- `palette` Character. Available palettes: "palette1" to "palette10".
- `monochrome` A list of parameters.
- ... Additional parameters.
Value

A apexcharts htmlwidget object.

Note

See https://apexcharts.com/docs/options/theme/

Examples

```r
library(dplyr)
data("mpg", package = "ggplot2")
data("diamonds", package = "ggplot2")

# Dark mode
apex(
  data = count(mpg, manufacturer),
  mapping = aes(x = manufacturer, y = n)
) %>%
  ax_theme(mode = "dark")

# Use predefined palette (1 to 10)
apex(
  data = count(diamonds, cut, color),
  mapping = aes(x = color, y = n, fill = cut)
) %>%
  ax_theme(palette = "palette2")

# monochrome palette
apex(
  data = count(diamonds, cut, color),
  mapping = aes(x = color, y = n, fill = cut)
) %>%
  ax_theme(monochrome = list(enabled = TRUE, color = "#0B6121"))
```

#### ax_title

<table>
<thead>
<tr>
<th>Chart’s title</th>
</tr>
</thead>
</table>

Description

Chart’s title

Usage

```r
ax_title(
  ax,
  text = NULL,
  align = NULL,
  margin = NULL,
  offsetX = NULL,
```
offsetY = NULL,
floating = NULL,
style = NULL,
... )

Arguments

ax A apexcharts htmlwidget object.
text Text to display as a title of chart.
align Alignment of subtitle relative to chart area. Possible Options: "left", "center" and "right".
margin Numeric. Vertical spacing around the title text.
offsetX Numeric. Sets the left offset for subtitle text.
offsetY Numeric. Sets the top offset for subtitle text
floating Logical. The floating option will take out the subtitle text from the chart area and make it float on top of the chart.
style List with two items: fontSize (Font Size of the title text) and color (Fore color of the title text).
... Additional parameters.

Value

A apexcharts htmlwidget object.

Note

See https://apexcharts.com/docs/options/title/

Examples

data("economics", package = "ggplot2")
apex(
  data = economics,
  mapping = aes(x = date, y = uempmed),
  type = "line"
) %>%
x_title(
  text = "Median duration of unemployment, in weeks"
)
**ax_tooltip**  
*Tooltip options*

### Description

Tooltip options

### Usage

```r
ax_tooltip(
  ax,
  enabled = NULL,
  shared = NULL,
  followCursor = NULL,
  intersect = NULL,
  inverseOrder = NULL,
  custom = NULL,
  fillSeriesColor = NULL,
  onDatasetHover = NULL,
  theme = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  marker = NULL,
  items = NULL,
  fixed = NULL,
  ...
)
```

### Arguments

- **ax**: A apexcharts htmlwidget object.
- **enabled**: Logical. Show tooltip when user hovers over chart area.
- **shared**: Logical. When having multiple series, show a shared tooltip.
- **followCursor**: Logical. Follow user’s cursor position instead of putting tooltip on actual data points.
- **intersect**: Logical. Show tooltip only when user hovers exactly over datapoint.
- **inverseOrder**: Logical. In multiple series, when having shared tooltip, inverse the order of series (for better comparison in stacked charts).
- **custom**: JS function. Draw a custom html tooltip instead of the default one based on the values provided in the function arguments.
- **fillSeriesColor**: Logical. When enabled, fill the tooltip background with the corresponding series color.
- **onDatasetHover**: A list of parameters.
theme A list of parameters.
x A list of parameters.
y A list of parameters.
z A list of parameters.
marker A list of parameters.
items A list of parameters.
fixed A list of parameters.
... Additional parameters.

Value
A apexcharts htmlwidget object.

Note
See https://apexcharts.com/docs/options/tooltip/

Examples

library(dplyr)
data("mpg", package = "ggplot2")

# Hide tooltip
apex(
data = count(mpg, manufacturer, year),
mapping = aes(x = manufacturer, y = n, fill = year)
) %>%
  ax_tooltip(enabled = FALSE)

# Share between series
apex(
data = count(mpg, manufacturer, year),
mapping = aes(x = manufacturer, y = n, fill = year)
) %>%
  ax_tooltip(shared = TRUE)

# Fixed tooltip
data("economics", package = "ggplot2")
apex(
data = economics,
mapping = aes(x = date, y = psavert),
type = "line"
) %>%
  ax_tooltip(
    fixed = list(enabled = TRUE, position = "topLeft")
  )
### ax_xaxis

#### Description

X-axis options

#### Usage

```r
ax_xaxis(
  ax,
  type = NULL,
  categories = NULL,
  labels = NULL,
  axisBorder = NULL,
  axisTicks = NULL,
  tickAmount = NULL,
  min = NULL,
  max = NULL,
  range = NULL,
  floating = NULL,
  position = NULL,
  title = NULL,
  crosshairs = NULL,
  tooltip = NULL,
  ...
)
```

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ax</td>
<td>A <code>apexcharts</code> htmlwidget object.</td>
</tr>
<tr>
<td>type</td>
<td>Character. Available Options : &quot;categories&quot; and &quot;datetime&quot;.</td>
</tr>
<tr>
<td>categories</td>
<td>Categories are labels which are displayed on the x-axis.</td>
</tr>
<tr>
<td>labels</td>
<td>A list of parameters.</td>
</tr>
<tr>
<td>axisBorder</td>
<td>A list of parameters.</td>
</tr>
<tr>
<td>axisTicks</td>
<td>A list of parameters.</td>
</tr>
<tr>
<td>tickAmount</td>
<td>Number of Tick Intervals to show.</td>
</tr>
<tr>
<td>min</td>
<td>Lowest number to be set for the x-axis. The graph drawing beyond this number will be clipped off.</td>
</tr>
<tr>
<td>max</td>
<td>Highest number to be set for the x-axis. The graph drawing beyond this number will be clipped off.</td>
</tr>
<tr>
<td>range</td>
<td>Range takes the max value of x-axis, subtracts the provided range value and gets the min value based on that. So, technically it helps to keep the same range when min and max values gets updated dynamically.</td>
</tr>
</tbody>
</table>
ax_xaxis

floating Logical. Floating takes x-axis is taken out of normal flow and places x-axis on svg element directly, similar to an absolutely positioned element. Set the offsetX and offsetY then to adjust the position manually.

position Setting this option allows you to change the x-axis position. Available options: "top" and "bottom".

title A list of parameters.
crosshairs A list of parameters.
tooltip A list of parameters.
... Additional parameters.

Value

A apexcharts htmlwidget object.

Note

See https://apexcharts.com/docs/options/xaxis/

Examples

library(dplyr)
data("mpg", package = "ggplot2")

# X axis title
apex(
  data = count(mpg, manufacturer),
  mapping = aes(x = manufacturer, y = n)
) %>%
  ax_xaxis(title = list(text = "Car's manufacturer"))

# force labels to rotate and increase height
apex(
  data = count(mpg, manufacturer),
  mapping = aes(x = manufacturer, y = n)
) %>%
  ax_xaxis(labels = list(rotateAlways = TRUE, maxHeight = 180))

# force to not rotate
apex(
  data = count(mpg, manufacturer),
  mapping = aes(x = manufacturer, y = n)
) %>%
  ax_xaxis(labels = list(rotate = 0, trim = FALSE))

data("economics", package = "ggplot2")

# Custom crosshair
apex(
  data = tail(economics, 50),

mapping = aes(x = date, y = psavert),
type = "line"
) %>%
ax_xaxis(
crosshairs = list(
  opacity = 1,
  width = 2,
  fill = list(color = "red"),
  stroke = list(width = 0)
)
)

# Date format (zoom to see changes)
apex(
data = tail(economics, 150),
mapping = aes(x = date, y = psavert),
type = "line"
) %>%
ax_xaxis(
  labels = list( 
    datetimeFormatter = list(
      year = "yyyy-MM",
      month = "yyyy-MM-dd",
      day = "yyyy-MM-dd HH:mm"
    )
  )
)

<table>
<thead>
<tr>
<th>ax_yaxis</th>
<th>Y-axis options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description**

Y-axis options

**Usage**

```r
ax_yaxis(
  ax,
  opposite = NULL,
  tickAmount = NULL,
  max = NULL,
  min = NULL,
  floating = NULL,
  labels = NULL,
  axisBorder = NULL,
  axisTicks = NULL,
  title = NULL,
```
Arguments

ax A apexcharts htmlwidget object.
opposite Logical. When enabled, will draw the yaxis on the right side of the chart.
tickAmount Number of Tick Intervals to show.
max Lowest number to be set for the y-axis. The graph drawing beyond this number will be clipped off.
min Highest number to be set for the y-axis. The graph drawing beyond this number will be clipped off.
floating Logical. Floating takes y-axis is taken out of normal flow and places y-axis on svg element directly, similar to an absolutely positioned element. Set the offsetX and offsetY then to adjust the position manually
labels A list of parameters.
axisBorder A list of parameters.
axisTicks A list of parameters.
title A list of parameters.
tooltip A list of parameters.
crosshairs A list of parameters.
... Additional parameters.

Value

A apexcharts htmlwidget object.

Note

See https://apexcharts.com/docs/options/yaxis/

Examples

data("economics_long", package = "ggplot2")
apex(  
data = economics_long,  
mapping = aes(x = date, y = value01, group = variable),  
type = "line"  
)  

ax_yaxis(  
decimalsInFloat = 2, title = list(text = "Rescaled to [0,1]"))  
)

# Format tick labels
temperature <- data.frame(
month = head(month.name),
    tp = c(4, -2, 2, 7, 11, 14)
  }
apex(temperature, aes(month, tp), "line") %>%
  ax_yaxis(
    labels = list(
      formatter = htmlwidgets::JS("function(value) {return value + '°C';}")
    )
  )
)

ax_yaxis2

Secondary Y-axis options

Description
Secondary Y-axis options

Usage
ax_yaxis2(ax, ...)

Arguments
ax A apexcharts htmlwidget object.
...
See arguments from ax_yaxis.

Value
A apexcharts htmlwidget object.

Examples

library(dplyr)
data("economics_long", package = "ggplot2")

eco <- economics_long %>%
  filter(variable %in% c("pce", "pop")) %>%
  filter(date >= "2000-01-01")

apex(eco, aes(x = date, y = value, color = variable), type = "line") %>%
  ax_yaxis(title = list(text = "Pce")) %>%
  ax_yaxis2(opposite = TRUE, title = list(text = "Pop"))
bar_opts

Description

Use these options in `ax_plotOptions`.

Usage

```r
bar_opts(
  horizontal = NULL,
  endingShape = NULL,
  columnWidth = NULL,
  barHeight = NULL,
  distributed = NULL,
  colors = NULL,
  dataLabels = NULL,
  ...
)
```

Arguments

- `horizontal` Logical. This option will turn a column chart into a horizontal bar chart.
- `endingShape` Available Options: "flat" or "rounded".
- `columnWidth` In column charts, `columnWidth` is the percentage of the available width in the grid-rect.
- `barHeight` In horizontal bar charts, `barHeight` is the percentage of the available height in the grid-rect.
- `distributed` Logical. Turn this option to make the bars discrete. Each value indicates one bar per series.
- `colors` A list of parameters.
- `dataLabels` List with fields `position` (available options: "top", "center" or "bottom")
- `...` Additional parameters.

Value

A list of options that can be used in `ax_plotOptions`.

Note

See [https://apexcharts.com/docs/options/plotoptions/bar/](https://apexcharts.com/docs/options/plotoptions/bar/).
Examples

```r
library(dplyr)
data("mpg", package = "ggplot2")

apex(count(mpg, manufacturer), aes(manufacturer, n)) %>%
  ax_plotOptions(
    bar = bar_opts(
      endingShape = "rounded",
      columnWidth = 100,
      distributed = TRUE
    )
  )
```

---

`config_update`  
*Configuration for auto update*

Description

Configuration for auto update

Usage

```r
config_update(
  series_animate = TRUE,
  update_options = FALSE,
  options_animate = TRUE,
  options_redrawPaths = FALSE
)
```

Arguments

- `series_animate`  Should the chart animate on re-rendering.
- `update_options`  Update or not global options for chart.
- `options_animate`  Should the chart animate on re-rendering.
- `options_redrawPaths`  When the chart is re-rendered, should it draw from the existing paths or completely redraw the chart paths from the beginning. By default, the chart is re-rendered from the existing paths.
events_opts

Description

Events options

Usage

events_opts(
  click = NULL,
  beforeMount = NULL,
  mounted = NULL,
  updated = NULL,
  legendClick = NULL,
  selection = NULL,
  dataPointSelection = NULL,
  dataPointMouseEnter = NULL,
  dataPointMouseLeave = NULL,
  beforeZoom = NULL,
  zoomed = NULL,
  scrolled = NULL,
  ...
)

Arguments

- **click**: Fires when user clicks on any area of the chart.
- **beforeMount**: Fires before the chart has been drawn on screen.
- **mounted**: Fires after the chart has been drawn on screen.
- **updated**: Fires when the chart has been dynamically updated.
- **legendClick**: Fires when user clicks on legend.
- **selection**: Fires when user selects rect using the selection tool.
- **dataPointSelection**: Fires when user clicks on a datapoint (bar/column/marker/bubble/donut-slice).
- **dataPointMouseEnter**: Fires when user’s mouse enter on a datapoint (bar/column/marker/bubble/donut-slice).
- **dataPointMouseLeave**: MouseLeave event for a datapoint (bar/column/marker/bubble/donut-slice).
- **beforeZoom**: This function, if defined, runs just before zooming in/out of the chart allowing you to set a custom range for zooming in/out.
- **zoomed**: Fires when user zooms in/out the chart using either the selection zooming tool or zoom in/out buttons.
- **scrolled**: Fires when user scrolls using the pan tool.
- **...**: Additional parameters.
Value

A list of options that can be used in `ax_chart`.

Note

All arguments should be JavaScript function defined with `htmlwidgets::JS`.
See [https://apexcharts.com/docs/options/chart/events/](https://apexcharts.com/docs/options/chart/events/).

Examples

```r
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    fluidRow(
      column(
        width = 8, offset = 2,
        tags$h2("Apexchart in Shiny"),
        apexchartOutput("chart"),
        verbatimTextOutput(outputId = "res_click")
      )
    )
  )

  server <- function(input, output, session) {
    output$chart <- renderApexchart({
      apexchart() %>%
        ax_chart(
          type = "bar",
          events = events_opts(
            dataPointSelection = JS("function(event, chartContext, config) {
              Shiny.setInputValue('Var click', config.selectedDataPoints)
            }"),
            )
        )%>%
        ax_series(
          list(
            name = "Example",
            data = sample(1:100, 5)
          )
        )%>%
        ax_xaxis(
          categories = LETTERS[1:5]
        )
    })

    output$res_click <- renderPrint({
      input$click
    })
  }
}
```
format_date

Format date in JS

Description
Format date in JS

Usage
format_date(x)

Arguments

x Date to use in JavaScript

Value
a JavaScript string

format_num
Format numbers (with D3)

Description
Format numbers (with D3)

Usage
format_num(format, prefix = "", suffix = "", locale = "en-US")

Arguments

format Format for numbers, currency, percentage, e.g. ".0%" for rounded percentage. See online documentation: https://github.com/d3/d3-format.
prefix Character string to append before formatted value.
suffix Character string to append after formatted value.
locale Localization to use, for example "fr-FR" for French, see possible values here: https://github.com/d3/d3-format/tree/master/locale.
Value

a JS function

Examples

# Use SI prefix

dat <- data.frame(
  labels = c("apex", "charts"),
  values = c(1e4, 2e4)
)

apex(dat, aes(labels, values), "column") %>%
  ax_yaxis(labels = list(
    formatter = format_num("~s")
  ))

apex(dat, aes(labels, values * 100), "column") %>%
  ax_yaxis(labels = list(
    formatter = format_num("~s")
  ))

# Percentage

dat <- data.frame(
  labels = c("apex", "charts"),
  values = c(0.45, 0.55)
)

apex(dat, aes(labels, values), "column") %>%
  ax_yaxis(labels = list(
    formatter = format_num(".0%")
  ))

# Currency

dat <- data.frame(
  labels = c("apex", "charts"),
  values = c(570, 1170)
)

apex(dat, aes(labels, values), "column") %>%
  ax_yaxis(labels = list(
    formatter = format_num("$,.2f")
  ))

# Change locale

apex(dat, aes(labels, values), "column") %>%
  ax_yaxis(labels = list(
    formatter = format_num("$,.2f", locale = "fr-FR")
  ))
# Customize tooltip value
# Use SI prefix
dat <- data.frame(
  labels = c("apex", "charts"),
  values = c(1e4, 2e4)
)

apex(dat, aes(labels, values), "column") %>%
  ax_tooltip(y = list(
    formatter = format_num("", suffix = " GW/h")
  ))

---

heatmap_opts

**Heatmap options**

**Description**

Use these options in `ax_plotOptions`.

**Usage**

```r
heatmap_opts(
  radius = NULL,
  enableShades = NULL,
  shadeIntensity = NULL,
  colorScale = NULL,
  ...
)
```

**Arguments**

- `radius` Numeric. Radius of the rectangle inside heatmap.
- `enableShades` Logical. Enable different shades of color depending on the value.
- `shadeIntensity` Numeric $[0,1]$. The intensity of the shades generated for each value.
- `colorScale` List.
- `...` Additional parameters.

**Value**

A list of options that can be used in `ax_plotOptions`.

**Note**

See [https://apexcharts.com/docs/options/plotoptions/heatmap/](https://apexcharts.com/docs/options/plotoptions/heatmap/).
Examples

df <- expand.grid(
  month = month.name,
  person = c("Obi-Wan", "Luke", "Anakin", "Leia")
)
df$value <- sample(0:1, nrow(df), TRUE)

apex(
  data = df,
  mapping = aes(x = month, y = person, fill = value),
  type = "heatmap"
) %>%
  ax_plotOptions(
    heatmap = heatmap_opts(
      enableShades = FALSE,
      colorScale = list(
        ranges = list(
          list(from = 0, to = 0.5, color = "#FF0000"),
          list(from = 0.5, to = 1, color = "#088A08")
        )
      )
    )
  )
)

---

parse_df  

Convert a data.frame to a list

Description

Convert data to a format suitable for ApexCharts.js

Usage

parse_df(data, add_names = FALSE)

Arguments

data  

A data.frame or an object coercible to data.frame.

add_names  

Use names of columns in output. Can be logical to reuse data names or a character vector of new names.

Value

A list that can be used to specify data in `ax_series` for example.
Examples

# All iris dataset
parse_df(iris)

# Keep variables names
parse_df(iris[, 1:2], add_names = TRUE)

# Use custom names
parse_df(iris[, 1:2], add_names = c("x", "y")

pie_opts

Pie options

Description

Use these options in `ax_plotOptions`.

Usage

```r
pie_opts(
  size = NULL,
  donut = NULL,
  customScale = NULL,
  offsetX = NULL,
  offsetY = NULL,
  dataLabels = NULL,
  ...
)
```

Arguments

- `size` Numeric. Custom size of the pie which will override the default size calculations.
- `donut` List with two fields `size` (Donut / ring size in percentage relative to the total pie area.) and `background` (The background color of the pie).
- `customScale` Numeric. Transform the scale of whole pie/donut overriding the default calculations.
- `offsetX` Numeric. Sets the left offset of the whole pie area.
- `offsetY` Numeric. Sets the top offset of the whole pie area.
- `dataLabels` List with field `offset` (Numeric, Offset by which labels will move outside / inside of the donut area)
- `...` Additional parameters.
radialBar_opts

Value

A list of options that can be used in `ax_plotOptions`.

Note

See https://apexcharts.com/docs/options/plotoptions/pie/.

Examples

```r
library(dplyr)
data("mpg", package = "ggplot2")

apex(count(mpg, cyl), aes(cyl, n), type = "donut") %>%
  ax_plotOptions(
    pie = pie_opts(
      donut = list(size = "90\%", background = "#BABABA")
    )
  )
```

---

**radialBar_opts**  
*Radial bar options*

Description

Use these options in `ax_plotOptions`.

Usage

```r
radialBar_opts(
  size = NULL,
  inverseOrder = NULL,
  startAngle = NULL,
  endAngle = NULL,
  offsetX = NULL,
  offsetY = NULL,
  hollow = NULL,
  track = NULL,
  dataLabels = NULL,
  ...
)
```

Arguments

- `size`  

- `inverseOrder`  
  Logical. Whether to make the first value of series innermost or outermost.
**radialBar_opts**

- **startAngle**: Numeric. Angle from which the radialBars should start.
- **endAngle**: Numeric. Angle to which the radialBars should end. The sum of the startAngle and endAngle should not exceed 360.
- **offsetX**: Numeric. Sets the left offset for radialBars.
- **offsetY**: Numeric. Sets the top offset for radialBars.
- **hollow**: List.
- **track**: List.
- **dataLabels**: List.
- ... Additional parameters.

**Value**

A list of options that can be used in `ax_plotOptions`.

**Note**

See [https://apexcharts.com/docs/options/plotoptions/radialbar/](https://apexcharts.com/docs/options/plotoptions/radialbar/).

**Examples**

```r
apexchart() %>%
  ax_chart(type = "radialBar") %>%
  ax_plotOptions(
    radialBar = radialBar_opts(
      startAngle = -135,
      endAngle = 135,
      dataLabels = list(
        name = list(
          fontSize = "16px",
          # color = undefined,
          offsetY = 120
        ),
        value = list(
          offsetY = 76,
          fontSize = "22px",
          # color = undefined,
          formatter = htmlwidgets::JS("function (val) {return val + '\%'\'}")
        )
      )
    )
  )
  ax_stroke(dashArray = 4) %>%
  ax_series(70) %>%
  ax_labels("Indicator")
```
run_input_demo  
**Run Shiny input events examples**

**Description**

Run Shiny input events examples

**Usage**

```r
run_input_demo(example = c("click", "zoom", "selection"))
```

**Arguments**

- `example` Name of the example.

**Examples**

```r
if (interactive()) {
  run_input_demo("click")
  run_input_demo("zoom")
  run_input_demo("selection")
}
```

set_input_click  
**Retrieve click information in Shiny**

**Description**

According to type of chart, different values are retrieved:

- **bar and column**: retrieve category (x-axis).
- **pie and donut**: retrieve label.
- **time-series**: retrieve x-axis value, you have to display markers with size > 0 and set tooltip’s options intersect = TRUE and shared = FALSE.
- **scatter**: retrieve XY coordinates.

**Usage**

```r
set_input_click(
  ax,
  inputId,
  multiple = FALSE,
  effect_type = c("darken", "lighten", "none"),
  effect_value = 0.35,
  session = shiny::getDefaultReactiveDomain()
)
```
set_input_selection

Arguments

- **ax**: An apexcharts htmlwidget object.
- **inputId**: The id that will be used server-side for retrieving click.
- **multiple**: Allow multiple selection: TRUE or FALSE (default).
- **effect_type**: Type of effect for selected element, default is to use lightly darken color.
- **effect_value**: A larger value intensifies the select effect, accept value between 0 and 1.
- **session**: The Shiny session.

Value

An apexcharts htmlwidget object.

Note

If x-axis is of type datetime, value retrieved is of class POSIXct.

Examples

```r
library(apexcharter)

# Not in Shiny but you can still click on bars
data.frame(
  month = month.abb,
  value = sample(1:100, 12)
)%>%
apex(aes(month, value)) %>%
set_input_click("month_click", multiple = TRUE)

# Interactive examples:
if (interactive()) {
  run_input_demo("click")
}
```

Description

Retrieve selection information in Shiny
set_input_selection

Usage

set_input_selection(
  ax,
  inputId,
  type = c("x", "xy", "y"),
  fill_color = "#24292e",
  fill_opacity = 0.1,
  stroke_width = 1,
  stroke_dasharray = 3,
  stroke_color = "#24292e",
  stroke_opacity = 0.4,
  xmin = NULL,
  xmax = NULL,
  ymin = NULL,
  ymax = NULL,
  session = shiny::getDefaultReactiveDomain()
)

Arguments

- **ax**: An `apexcharts htmlwidget` object.
- **inputId**: The id that will be used server-side for retrieving selection.
- **type**: Allow selection either on x-axis, y-axis or on both axis.
- **fill_color**: Background color of the selection rect which is drawn when user drags on the chart.
- **fill_opacity**: Opacity of background color of the selection rectangle.
- **stroke_width**: Border thickness of the selection rectangle.
- **stroke_dasharray**: Creates dashes in borders of selection rectangle. Higher number creates more space between dashes in the border.
- **stroke_color**: Colors of selection border.
- **stroke_opacity**: Opacity of selection border.
- **xmin, xmax**: Start value of x-axis. Both min and max must be provided.
- **ymin, ymax**: Start value of y-axis. Both min and max must be provided.
- **session**: The Shiny session.

Value

An `apexcharts htmlwidget` object.

Examples

```r
library(apexcharter)
data("economics", package = "ggplot2")
```
set_input_zoom

# Not in Shiny so no events
# but you can still select an area on chart
apex(economics, aes(date, psavert), type = "line") %>%
set_input_selection("selection")

# Default selection at start
apex(economics, aes(date, psavert), type = "line") %>%
set_input_selection(
  inputId = "selection",
  xmin = format_date("1980-01-01"),
  xmax = format_date("1985-01-01")
)

---

set_input_zoom Retrieve zoom information in Shiny

Description

Retrieve zoom information in Shiny

Usage

set_input_zoom(ax, inputId, session = shiny::getDefaultReactiveDomain())

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ax</td>
<td>An apexcharts htmlwidget object.</td>
</tr>
<tr>
<td>inputId</td>
<td>The id that will be used server-side for retrieving zoom.</td>
</tr>
<tr>
<td>session</td>
<td>The Shiny session.</td>
</tr>
</tbody>
</table>

Value

An apexcharts htmlwidget object.

Note

If x-axis is of type datetime, value retrieved is of class POSIXct.

Examples

if (interactive()) {
  run_input_demo("zoom")
}

unhcr_popstats_2017  UNHCR data for 2017

Description

The dataset contains data about UNHCR’s populations of concern for the year 2017.

Usage

unhcr_popstats_2017

Format

A data frame with 11237 observations on the following 6 variables.

country_origin  Country of origin of population

country_residence  Country / territory of asylum/residence of population

population_type  Populations of concern: Refugees, Asylum-seekers, Internally displaced persons (IDPs), Returned refugees, Returned IDPs, Stateless persons, Others of concern.

type  Number of people concerned

continent_residence  Continent of origin of population

continent_origin  Continent of residence of population

Source

UNHCR (The UN Refugee Agency) (https://www.unhcr.org/)

unhcr_ts  UNHCR data by continent of origin

Description

The dataset contains data about UNHCR’s populations of concern summarised by continent of origin.

Usage

unhcr_ts
unhcr_ts

Format

A data frame with 913 observations on the following 4 variables.

year Year concerned.
population_type Populations of concern: Refugees, Asylum-seekers, Internally displaced persons (IDPs), Returned refugees, Returned IDPs, Stateless persons, Others of concern.
continent_origin Continent of residence of population.
n Number of people concerned.

Source

UNHCR (The UN Refugee Agency) (https://www.unhcr.org/)
Index

*Topic datasets
  unhcr_popstats_2017, 60
  unhcr_ts, 60
  `%%(apexcharter-exports), 6`

aes (apexcharter-exports), 6
apex, 3
apexchart, 5
apexcharter-exports, 6
apexcharter-package, 3
apexcharter-shiny, 6
apexchartOutput (apexcharter-shiny), 6
apexchartProxy, 8
ax-series, 8
ax_annotations, 9
ax_chart, 12, 48
ax_colors, 15
ax_dataLabels, 16
ax_fill, 17
ax_grid, 19
ax_labels, 21
ax_labels2 (ax_labels), 21
ax_labs, 22
ax_legend, 22
ax_markers, 24
ax_plotOptions, 26, 45, 51, 53–55
ax_proxy_options, 27
ax_proxy_series, 29
ax_responsive, 30
ax_series, 52
ax_series (ax-series), 8
ax_series2 (ax-series), 8
ax_states, 31
ax_stroke, 32
ax_subtitle, 34
ax_theme, 35
ax_title, 36
ax_tooltip, 38
ax_xaxis, 40
ax_yaxis, 42, 44
ax_yaxis2, 44
bar_opts, 26, 45
config_update, 3, 5, 46
events_opts, 13, 47
format_date, 49
format_num, 49
heatmap_opts, 26, 51
JS (apexcharter-exports), 6
parse_df, 52
pie_opts, 26, 53
radialBar_opts, 26, 54
renderApexchart (apexcharter-shiny), 6
run_input_demo, 56
set_input_click, 56
set_input_selection, 57
set_input_zoom, 59
unhcr_popstats_2017, 60
unhcr_ts, 60