Package ‘apexcharter’

February 27, 2022

Version 0.3.1

Title Create Interactive Chart with the JavaScript 'ApexCharts'

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 htmltools, htmlwidgets (>= 1.5.3), magrittr, rlang, ggplot2,
   jsonlite, shiny (>= 1.1.0)

Suggests testthat, knitr, scales, rmarkdown, covr

RoxygenNote 7.1.2

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 2022-02-27 17:40:02 UTC
R topics documented:

- apexcharter-package ............................ 3
- add-line ...................................... 3
- add-shade .................................... 5
- add-vh-lines .................................. 7
- add_event ..................................... 9
- add_event_marker .............................. 10
- add_point .................................... 11
- apex .......................................... 13
- apex-facets .................................. 15
- apexchart ..................................... 19
- apexcharter-exports ......................... 20
- apexcharter-shiny ............................. 20
- apexcharter-shiny-facets ..................... 22
- apexcharter-shiny-grid ....................... 23
- apexchartProxy ............................... 24
- apex_grid .................................... 25
- ax-series .................................... 26
- ax_annotations ............................... 27
- ax_chart ..................................... 30
- ax_colors .................................... 33
- ax_colors_manual ............................. 34
- ax_dataLabels ................................. 36
- ax_fill ....................................... 37
- ax_grid ....................................... 39
- ax_labels .................................... 40
- ax_labs ....................................... 41
- ax_legend ..................................... 42
- ax_markers .................................... 44
- ax_nodata ..................................... 45
- ax_plotOptions ............................... 46
- ax_proxy_options ............................. 48
- ax_proxy_series ............................... 49
- ax_responsive ................................ 51
- ax_states ..................................... 52
- ax_stroke ..................................... 53
- ax_subtitle .................................... 54
- ax_theme ....................................... 56
- ax_title ....................................... 57
- ax_tooltip ..................................... 58
- ax_xaxis ....................................... 60
- ax_yaxis ....................................... 62
- ax_yaxis2 ...................................... 64
- bar_opts ....................................... 65
- bubble_opts ................................... 67
- candles ....................................... 68
- climate_paris .................................. 68
apexcharter-package

Description

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

Author(s)

Victor Perrier (@dreamRs_fr)

add-line

Description

Add a line to an existing chart (bar, scatter and line types supported). On scatter charts you can also add a smooth line.
Usage

```
add_line(
  ax,
  mapping,
  data = NULL,
  type = c("line", "spline"),
  serie_name = NULL
)

add_smooth_line(
  ax,
  formula = y ~ x,
  model = c("lm", "loess"),
  n = 100,
  ...,
  type = c("line", "spline"),
  serie_name = NULL
)
```

Arguments

- **ax**
  An `apexchart()` htmlwidget object.
- **mapping**
  Default list of aesthetic mappings to use for chart.
- **data**
  A `data.frame` to use to add a line, if `NULL` (default), the `data.frame` provided in `apex()` will be used.
- **type**
  Type of line.
- **serie_name**
  Name for the serie displayed in tooltip and legend.
- **formula**
  Formula passed to the method, default to `y ~ x` from main aesthetics.
- **model**
  Model to use between `lm` or `loess`.
- **n**
  Number of points used for predictions.
- **...**
  Arguments passed to model.

Value

An `apexchart()` htmlwidget object.

Examples

```
library(apexcharter)

# Bar ----

data("climate_paris")

# Add a line on a column's chart
apex(climate_paris, aes(month, precipitation), type = "column")
```
add_line(aes(month, temperature))

# Add secondary axis
apex(climate_paris, aes(month, precipitation), type = "column") %>%
  add_line(aes(month, temperature)) %>%
  ax_yaxis(
    title = list(text = "Precipitation (in mm)"
  ) %>%
  ax_yaxis2(
    opposite = TRUE,
    decimalsInFloat = 0,
    title = list(text = "Temperature (in degree celsius)"
  ) %>%
  ax_dataLabels(
    enabled = TRUE, enabledOnSeries = list(1)
  )

# Scatter ----
# add smooth line on scatter plot
apex(cars, aes(speed, dist), type = "scatter") %>%
  add_line(aes(x, y), data = lowess(cars), serie_name = "lowess")

# or directly
apex(cars, aes(speed, dist), type = "scatter") %>%
  add_smooth_line()

apex(cars, aes(speed, dist), type = "scatter") %>%
  add_smooth_line(model = "loess", span = 1)

apex(cars, aes(speed, dist), type = "scatter") %>%
  add_smooth_line(model = "loess", degree = 1)

apex(cars, aes(speed, dist), type = "scatter") %>%
  add_smooth_line(formula = y ~ poly(x, 2))

apex(cars, aes(speed, dist), type = "scatter") %>%
  add_smooth_line(model = "lm", serie_name = "lm") %>%
  add_smooth_line(model = "loess", serie_name = "loess")
Description

`add_shade()` allow to add a shaded area on specified range, `add_shade_weekend()` add a shadow on every week-end.

Usage

```r
add_shade(ax, from, to, color = "#848484", opacity = 0.2, label = NULL, ...)
add_shade_weekend(ax, color = "#848484", opacity = 0.2, label = NULL, ...)
```

Arguments

- **ax**: An `apexchart()` htmlwidget object.
- **from**: Vector of position to start shadow.
- **to**: Vector of position to end shadow.
- **color**: Color of the shadow.
- **opacity**: Opacity of the shadow.
- **label**: Add a label to the shade, use a character or see `label` for more controls.
- **...**: Additional arguments, see `https://apexcharts.com/docs/options/annotations/` for possible options.

Value

An `apexchart()` htmlwidget object.

Note

`add_shade_weekend` only works if variable used for x-axis is of class Date or POSIXt.

Examples

```r
library(apexcharter)
data("consumption")

# specify from and to date
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_shade(from = "2020-01-06", to = "2020-01-20")

# you can add several shadows
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_shade(from = "2020-01-06", to = "2020-01-20") %>%
  add_shade(from = "2020-02-04", to = "2020-02-10")

# or use a vector
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_shade(
    from = c("2020-01-06", "2020-02-04"),
    to = c("2020-01-20", "2020-02-10")
  )
```
# Add a label
apex(consumption, aes(date, value, group = type), "spline") %>%
add_shade(
  from = "2020-01-06", to = "2020-01-20",
  label = "interesting period"
)

# add label with more options
apex(consumption, aes(date, value, group = type), "spline") %>%
add_shade(
  from = "2020-01-06", to = "2020-01-20",
  color = "firebrick",
  label = label(
    text = "something happened",
    background = "firebrick",
    color = "white",
    fontWeight = "bold",
    padding = c(3, 5, 3, 5)
  )
)

# automatically add shadow on week-ends
apex(consumption, aes(date, value, group = type), "spline") %>%
add_shade_weekend()

---

## add-vh-lines

### Add horizontal or vertical line

**Description**

Add horizontal or vertical line

**Usage**

```r
add_hline(ax, value, color = "#000", dash = 0, label = NULL, ...)
```

```r
add_vline(ax, value, color = "#000", dash = 0, label = NULL, ...)
```

**Arguments**

- `ax` An apexchart() htmlwidget object.
- `value` Vector of position for the line(s).
**add-vh-lines**

- **color**: Color(s) of the line(s).
- **dash**: Creates dashes in borders of SVG path. A higher number creates more space between dashes in the border. Use 0 for plain line.
- **label**: Add a label to the shade, use a character or see `label` for more controls.

... Additional arguments, see [https://apexcharts.com/docs/options/annotations/](https://apexcharts.com/docs/options/annotations/) for possible options.

**Value**

An `apexchart()` htmlwidget object.

**Examples**

```r
library(apexcharter)

# On a column chart
unhcr_ts %>%
  subset(year == 2017 & population_type == "Asylum-seekers") %>%
  apex(aes(continent_origin, n), "column") %>%
  add_hline(value = 5e5)

# On a scatter chart
apex(data = cars, aes(speed, dist), "scatter") %>%
  add_hline(value = mean(cars$dist)) %>%
  add_vline(value = mean(cars$speed))

# With labels
apex(data = cars, aes(speed, dist), "scatter") %>%
  add_hline(value = mean(cars$dist), label = "Mean of dist") %>%
  add_vline(value = mean(cars$speed),
            label = label(text = "Mean of speed",
                          borderColor = "red"))
```


**add_event**

*Add an event to a chart*

**Description**

Add a vertical line to mark a special event on a chart.

**Usage**

```r
add_event(ax, when, color = "#E41A1C", dash = 4, label = NULL, ...)
```

**Arguments**

- `ax` An `apexchart()` htmlwidget object.
- `when` Vector of position to place the event.
- `color` Color of the line.
- `dash` Creates dashes in borders of SVG path. A higher number creates more space between dashes in the border. Use 0 for plain line.
- `label` Add a label to the shade, use a character or see `label` for more controls.
- `...` Additional arguments, see [https://apexcharts.com/docs/options/annotations/](https://apexcharts.com/docs/options/annotations/) for possible options.

**Value**

An `apexchart()` htmlwidget object.

**See Also**

- `add_event_marker` to add a point.

**Examples**

```r
library(apexcharter)
data("consumption")

# specify from and to date
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_event(when = "2020-01-11")

# several events
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_event(when = c("2020-01-11", "2020-01-29"))

# Add labels on events
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_event(
    when = c("2020-01-11", "2020-01-29"),
```
add_event_marker

label = label(text = c("Am", "Ar"))

# can be combined with shade
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_shade(from = "2020-01-06", to = "2020-01-20") %>%
  add_event(when = c("2020-01-11", "2020-01-29"))

add_event_marker  Add an event marker to a chart

Description
Add a point with a label based on a datetime.

Usage
add_event_marker(
  ax,
  when,
  y,
  size = 5,
  color = "#000",
  fill = "#FFF",
  width = 2,
  shape = "circle",
  radius = 2,
  label = NULL,
  ...
)

Arguments

ax  An apexchart() htmlwidget object.
when  Vector of position to place the event.
y  Coordinate(s) on the y-axis.
size  Size of the marker.
color  Stroke Color of the marker point.
fill  Fill Color of the marker point.
width  Stroke Size of the marker point.
shape  Shape of the marker: "circle" or "square".
radius  Radius of the marker (applies to square shape).
label  Add a label to the shade, use a character or see label for more controls.
...  Additional arguments, see https://apexcharts.com/docs/options/annotations/ for possible options.
add_point

Add an annotation point

Description

Add an annotation point

Usage

```r
add_point(ax, x, y, size = 5, color = "#000", fill = "#FFF")
```

Examples

```r
library(apexchart)
data("consumption")

# add a marker
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_event_marker(when = "2020-01-22", y = 1805)

# with a label
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_event_marker(when = "2020-01-22", y = 1805, label = "Consumption peak")

# add several markers
apex(consumption, aes(date, value, group = type), "spline") %>%
  add_event_marker(
    y = c(1545, 1659, 1614, 1805, 1637, 1636, 1597, 1547, 1631),
    size = 10,
    color = "firebrick"
  )
```
arg

Arguments

- **ax**
  - An `apexchart()` htmlwidget object.
- **x**
  - Coordinate(s) on the x-axis.
- **y**
  - Coordinate(s) on the y-axis.
- **size**
  - Size of the marker.
- **color**
  - Stroke Color of the marker point.
- **fill**
  - Fill Color of the marker point.
- **width**
  - Stroke Size of the marker point.
- **shape**
  - Shape of the marker: "circle" or "square".
- **radius**
  - Radius of the marker (applies to square shape).
- **label**
  - Add a label to the shade, use a character or see `label` for more controls.
- **...**
  - Additional arguments, see https://apexcharts.com/docs/options/annotations/ for possible options.

Value

An `apexchart()` htmlwidget object.

See Also

- `add_event_marker` to add a point when x-axis is a datetime.

Examples

```r
library(apexcharter)

# On scatter chart
apex(
  data = iris,
  aes(Sepal.Length, Sepal.Width),
  "scatter"
) %>%
  add_point(
    x = mean(iris$Sepal.Length),
    y = mean(iris$Sepal.Width)
  )

# Some options
apex(
  width = 2,
  shape = "circle",
  radius = 2,
  label = NULL,
  ...  
)  
```
data = iris,
aes(Sepal.Length, Sepal.Width),
"scatter"
) %>%
  add_point(
    x = mean(iris$Sepal.Length),
    y = mean(iris$Sepal.Width),
    fill = "firebrick",
    color = "firebrick",
    size = 8,
    label = label(text = "Mean", offsetY = 0)
  )

# Several points
clusters <- kmeans(iris[, 1:2], 3)
apex(
  data = iris,
  aes(Sepal.Length, Sepal.Width),
  "scatter"
) %>%
  add_point(
    x = clusters$centers[, 1],
    y = clusters$centers[, 2]
  )

---

**Quick ApexCharts**

**Description**

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

**Usage**

```r
apex(
  data,
  mapping,
  type = "column",
  ...,  
  auto_update = TRUE,
  synchronize = NULL,
  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.

**synchronize**
Give a common id to charts to synchronize them (tooltip and zoom).

**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.

**elementId**
Use an explicit element ID for the widget.

Value

An `apexchart()` htmlwidget object.

Examples

```r
library(ggplot2)
library(apexchart)

# make a barchart with a frequency table
data("mpg", package = "ggplot2")
apex(mpg, aes(manufacturer), 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"
)

### Description

Create matrix of charts by row and column faceting variable (ax_facet_grid), or by specified number of row and column for faceting variable(s) (ax_facet_wrap).

### Usage

#### ax_facet_wrap(

`ax`,
facets,
nrow = NULL,
col = NULL,
scales = c("fixed", "free", "free_y", "free_x"),
labeller = label_value,
chart_height = "300px"
)

#### ax_facet_grid(

`ax`,
rows = NULL,
cols = NULL,
scales = c("fixed", "free", "free_y", "free_x"),
labeller = label_value,
chart_height = "300px"
)

### Arguments

- **ax**: An apexchart() htmlwidget object.
- **facets**: Variable(s) to use for facetting, wrapped in vars(…).
- **nrow, ncol**: Number of row and column in output matrix.
- **scales**: Should scales be fixed ("fixed", the default), free ("free"), or free in one dimension ("free_x", "free_y")?
- **labeller**: A function with one argument containing for each facet the value of the faceting variable.
- **chart_height**: Individual chart height.
- **rows, cols**: A set of variables or expressions quoted by vars() and defining faceting groups on the rows or columns dimension.
Value

An `apexchart()` htmlwidget object with an additional class "apex_facet".

Warning

To properly render in Shiny applications, use `apexfacetOutput()` (in UI) and `renderApexfacet()` (in Server).

Examples

```r
### Wrap --------
if (interactive()) {
  library(apexchart)

  # Scatter ----
  data("mpg", package = "ggplot2")

  # Create facets
  apex(mpg, aes(displ, cty), type = "scatter") %>%
  ax_facet_wrap(vars(drv))

  # Change number of columns
  apex(mpg, aes(displ, cty), type = "scatter") %>%
  ax_facet_wrap(vars(drv), ncol = 2)

  # Free axis
  apex(mpg, aes(displ, cty), type = "scatter") %>%
  ax_facet_wrap(vars(drv), ncol = 2, scales = "free")

  # labels
  apex(mpg, aes(displ, cty), type = "scatter") %>%
  ax_facet_wrap(
    vars(drv), ncol = 2,
    labeller = function(x) {
      switch(
        x,
        "f" = "front-wheel drive",
        "r" = "rear wheel drive",
        "4" = "4wd"
      )
    }
  )

  # Title and subtitle are treated as global
  apex(mpg, aes(displ, cty), type = "scatter") %>%
  ax_labs(
    title = "Facet wrap example",
    subtitle = "mpg data from ggplot2"
  ) %>%
  ax_facet_wrap(vars(drv), ncol = 2)
```
# Multiple variables
```r
apex(mpg, aes(displ, cty), type = "scatter") %>%
  ax_facet_wrap(vars(year, drv))
```
```r
apex(mpg, aes(displ, cty), type = "scatter") %>%
  ax_facet_wrap(vars(year, drv), ncol = 2, nrow = 3)
```
```r
apex(mpg, aes(displ, cty), type = "scatter") %>%
  ax_chart(toolbar = list(show = FALSE)) %>%
  ax_facet_wrap(
    vars(year, drv),
    labeller = function(x) {
      paste(x, collapse = " / ")
    }
  )
```

# Lines ----
```r
data("unhcr_ts")
refugees <- unhcr_ts %>%
  subset(population_type == "Refugees (incl. refugee-like situations)") %>%
  transform(date = as.Date(paste0(year, "-01-01")))
```
```r
apex(refugees, aes(date, n), type = "line") %>%
  ax_yaxis(tickAmount = 5) %>%
  ax_facet_wrap(vars(continent_origin))
```
```r
# Free y-axis and synchronize
apex(refugees, aes(date, n), type = "line", synchronize = "my-id") %>%
  ax_yaxis(tickAmount = 5) %>%
  ax_xaxis(tooltip = list(enabled = FALSE)) %>%
  ax_tooltip(x = list(format = "yyyy")) %>%
  ax_facet_wrap(vars(continent_origin), scales = "free_y")
```

# Bars ----
```r
data("unhcr_ts")
refugees <- unhcr_ts %>%
  subset(year == 2017)
```
```r
apex(refugees, aes(continent_origin, n), type = "column") %>%
  ax_yaxis(
    labels = list(
      formatter = format_num("-s")
    )
  )
```
### Grid --------

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

  # Scatter ----
  data("mpg", package = "ggplot2")

  # Only rows
  apex(mpg, aes(displ, cty), type = "scatter") %>%
    ax_facet_grid(rows = vars(drv), chart_height = "200px")

  # Only cols
  apex(mpg, aes(displ, cty), type = "scatter") %>%
    ax_facet_grid(cols = vars(year))

  # Rows and Cols
  apex(mpg, aes(displ, cty), type = "scatter") %>%
    ax_facet_grid(rows = vars(drv), cols = vars(year))

  apex(mpg, aes(displ, cty), type = "scatter") %>%
    ax_chart(toolbar = list(show = FALSE)) %>%
    ax_facet_grid(vars(drv), vars(cyl))

  # Labels
  apex(mpg, aes(displ, cty), type = "scatter") %>%
    ax_facet_grid(
      vars(drv),
      labeller = function(x) {
        switch(
          x,
          "f" = "front-wheel drive",
          "r" = "rear wheel drive",
          "4" = "4wd"
        )
      }
    )

  # Title and subtitle are treated as global
  apex(mpg, aes(displ, cty), type = "scatter") %>%
    ax_labs(
      title = "Facet grid example",
      subtitle = "mpg data from ggplot2"
    )
}
```
Create an ApexCharts 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, height**: A numeric input in pixels.
- **elementId**: Use an explicit element ID for the widget.

Value

An `apexchart()` htmlwidget object.

See Also

For quickly create a chart, see `apex()`.

Examples

library(apexcharter)

# 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

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

---

**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)
sparkBoxOutput(outputId, width = "100\%", height = "160px")
renderSparkBox(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 calendar
- **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

Output element that can be included in UI. Render function to create output in server.

Note

To render a chart with facets (using `ax_facet_wrap()` or `ax_facet_grid()`) in Shiny, see `apexfacetOutput()` (in UI) and `renderApexfacet()` (in Server).

Examples

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

  ui <- fluidPage(
    fluidRow(
      column(
        width = 8, offset = 2,
        tags$h2("Apexchart in Shiny"),
        actionButton("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]
      )
    })
  }
}
```
Shiny bindings for faceting with apexcharter

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

Usage
apexfacetOutput(outputId)
renderApexfacet(expr, env = parent.frame(), quoted = FALSE)

Arguments
- outputId: output variable to read from
- expr: An expression that generates a apexcharter facet with \texttt{ax_facet_wrap()} or \texttt{ax_facet_grid()}.
- env: The environment in which to evaluate expr.
- quoted: Is expr a quoted expression (with \texttt{quote()})? This is useful if you want to save an expression in a variable.

Value
An Apexcharts output that can be included in the application UI.

See Also
\texttt{ax_facet_wrap()}, \texttt{ax_facet_grid()}

Examples

```r
library(shiny)
library(apexcharter)

data("unhcr_ts")
refugees <- unhcr_ts %>%
  subset(  
```
population_type == "Refugees (incl. refugee-like situations)"
) %>%
transform(date = as.Date(paste0(year, "-01-01")))

ui <- fluidPage(
  tags$h2("Apexcharts Facets Example"),
  apexfacetOutput("myfacet")
)

server <- function(input, output, session) {
  output$myfacet <- renderApexfacet({
    apex(refugees, aes(date, n), type = "column") %>%
    ax_yaxis(tickAmount = 5) %>%
    ax_facet_wrap(vars(continent_origin),
      scales = "free"
  )
})

if (interactive())
  shinyApp(ui, server)
Is expr a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

**Value**

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

**Examples**

```r
library(shiny)
library(apexchart)

ui <- fluidPage(
  tags$h2("Apexcharts Grid Example"),
  apexgridOutput("myfacet")
)

server <- function(input, output, session) {

  output$myfacet <- renderApexgrid({
    a1 <- apex(mpg, aes(manufacturer), type = "bar")
    a2 <- apex(mpg, aes(trans), type = "column")
    a3 <- apex(mpg, aes(drv), type = "pie")

    apex_grid(
      a1, a2, a3,
      grid_area = c("1 / 1 / 3 / 2", "1 / 2 / 2 / 4", "2 / 2 / 3 / 4"),
      ncol = 3,
      nrow = 2,
      height = "600px"
    )
  })
}

if (interactive())
  shinyApp(ui, server)
```

**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.

---

**apex_grid**

*Create a grid of ApexCharts*

---

Description

Create a grid of ApexCharts

Usage

```r
apex_grid(
  ..., 
  nrow = NULL, 
  ncol = NULL, 
  row_gap = "10px", 
  col_gap = "0px", 
  grid_area = NULL, 
  height = NULL, 
  width = NULL, 
  .list = NULL
)
```

Arguments

- ...: Several `apexcharts htmlwidget` objects.
- **nrow, ncol**: Number of rows and columns.
- **row_gap, col_gap**: Gap between rows and columns.
- **grid_area**: Custom grid area to make elements take more than a single cell in grid, see [https://cssgrid-generator.netlify.app/](https://cssgrid-generator.netlify.app/) for examples.
- **height, width**: Height and width of the main grid.
- **.list**: A list of `apexcharts htmlwidget` objects.

Value

Custom `apex_grid` object.
Note

You have to provide either height for the grid or individual chart height to make it work.

Examples

```r
if (interactive()) {
  library(apexchart)
  data("mpg", package = "ggplot2")

  # Two chart side-by-side
  a1 <- apex(mpg, aes(manufacturer), type = "bar")
  a2 <- apex(mpg, aes(trans), type = "column")
  apex_grid(a1, a2, height = "400px")

  # More complex layout:
  a3 <- apex(mpg, aes(drv), type = "pie")
  apex_grid(
    a1, a2, a3,
    grid_area = c("1 / 1 / 3 / 2", "1 / 2 / 2 / 4", "2 / 2 / 3 / 4"),
    ncol = 3, nrow = 2,
    height = "600px"
  )
}
```

ax-series

Add data to a chart

Description

Add data to a chart

Usage

```r
ax_series(ax, ...)
ax_series2(ax, l)
```

Arguments

- `ax` An `apexchart()` htmlwidget object.
- `...` Lists containing data to plot, typically list with two items: name and data.
- `l` A list.
Value

An apexchart() htmlwidget object.

Examples

```r
# 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)
    )
  )
```

---

<table>
<thead>
<tr>
<th>ax_annotations</th>
<th>Annotations properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

Annotations properties

Usage

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

Arguments

- **ax**: An apexchart() htmlwidget object.
- **position**: Whether to put the annotations behind the charts or in front of it. Available Options: "front" or "back".
ax_annotations

- **yaxis**: List of lists.
- **xaxis**: List of lists.
- **points**: List of lists.
- ... Additional parameters.

**Value**

An `apexchart()` `htmlwidget` object.

**Note**

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

**Examples**

```r
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
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
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 = "#B3F7CA"
        )
      )
    )
  )
)
style = list(
    color = "#fff",
    background = "#FF4560"
)
)
)
)

ax_chart

Description

Chart parameters

Usage

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  An apexchart() 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%".

defaultLocale

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
An apexchart() htmlwidget object.

Examples

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

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

# stack
apex(
data = diamonds,
# Stack filled

```r
apex(
  data = diamonds,
  mapping = aes(x = cut, fill = color)
) %>%
  ax_chart(stacked = TRUE, stackType = "100%")
```

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

# Hide the toolbar

```r
apex(
  data = diamonds,
  mapping = aes(x = cut, fill = color)
) %>%
  ax_chart(toolbar = list(show = FALSE))
```

# Hide download buttons

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

```r
apex(
  data = economics,
  mapping = aes(x = date, y = pce),
  type = "line"
) %>%
  ax_chart(
    toolbar = list(tools = list(download = FALSE))
)
```

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

# Disable

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

# Auto-scale Y axis

```r
apex(
  data = economics,
  mapping = aes(x = date, y = pce),
  type = "line"
) %>%
  ax_chart(stacked = TRUE)
```
```r
# 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: ax_colors

<table>
<thead>
<tr>
<th>Colors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>ax_colors(ax, ...)</td>
</tr>
</tbody>
</table>
Arguments

\textbf{ax} \hspace{1cm} \textbf{Values} \hspace{1cm} \textbf{Note}

\textbf{ax} \hspace{1cm} An \texttt{apexchart()} htmlwidget object.

Colors for the chart’s series. When all colors are used, it starts from the beginning.

Value

An \texttt{apexchart()} htmlwidget object.

Note

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

Examples

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

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

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

\begin{tabular}{ll}
\textbf{ax\_colors\_manual} & \textit{Set specific color’s series} \\
\end{tabular}

**Description**

Set specific color’s series

**Usage**

\texttt{ax\_colors\_manual(ax, values)}

**Arguments**

\begin{tabular}{ll}
\texttt{ax} & An \texttt{apexchart()} htmlwidget object. \\
\texttt{values} & Named list, names represent data series, values colors to use. \\
\end{tabular}
Value

An `apexchart()` htmlwidget object.

Examples

```r
## scatter

apex(
  data = mtcars,
  type = "scatter",
  mapping = aes(x = wt, y = mpg, fill = cyl)
) %>%
  ax_colors_manual(list(
    "4" = "steelblue",
    "6" = "firebrick",
    "8" = "forestgreen"
  ))

# If missing level, colors are recycled

apex(
  data = mtcars,
  type = "scatter",
  mapping = aes(x = wt, y = mpg, fill = cyl)
) %>%
  ax_colors_manual(list(
    "4" = "steelblue",
    "8" = "forestgreen"
  ))

# Ignore levels not present in data

apex(
  data = mtcars,
  type = "scatter",
  mapping = aes(x = wt, y = mpg, fill = cyl)
) %>%
  ax_colors_manual(list(
    "4" = "steelblue",
    "6" = "firebrick",
    "8" = "forestgreen",
    "99" = "yellow"
  ))

## Bar

tab <- table(sample(letters[1:5], 100, TRUE), sample(LETTERS[1:5], 100, TRUE))
dat <- as.data.frame(tab)

apex(
  data = dat,
  type = "column",
)
ax_dataLabels

Labels on data

Description

Labels on data

Usage

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

Arguments

- **ax**: An apexchart() 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

An apexchart() htmlwidget object.

Note

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

Examples

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

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

---

ax_fill

### Fill property

Description

Fill property

Usage

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

Arguments

- **ax** An apexchart() 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

An apexchart() htmlwidget object.

Note

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

Examples

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

# Use a pattern to fill bars
apex(
  data = diamonds,
  mapping = aes(x = color, fill = cut)
) %>%
  ax_fill(
    type = "pattern",
    opacity = 1,
    pattern = list(
      style = c("circles", "slantedLines", "verticalLines", "horizontalLines", "squares")
    )
  )


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

# Customise gradient
apex(
  data = economics,
  mapping = aes(x = date, y = psavert),
  type = "area"
) %>%
  ax_fill(gradient = list(
    enabled = TRUE,
    shadeIntensity = 1,
    inverseColors = FALSE,
    opacityFrom = 0,
    opacityTo = 1,
    stops = c(0, 2000)
  ))
**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** An `apexchart()` 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

An `apexchart()` htmlwidget object.
Note

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

Examples

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

# Hide Y-axis and gridlines
apex(
  data = mpg,
  mapping = aes(x = manufacturer)
) %>%
  ax_grid(show = FALSE)

# just grid lines
apex(
  data = mpg,
  mapping = aes(x = manufacturer)
) %>%
  ax_grid(yaxis = list(lines = list(show = FALSE)))

# 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**

ax_labels(ax, ...)

ax_labels2(ax, labels)
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 An apexchart() 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.
Value

An `apexchart()` htmlwidget object.

Examples

```r
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)"
)
```

Legend properties

Description

Legend properties

Usage

```r
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, 
  onClick = NULL, 
  onItemHover = NULL, 
  floating = NULL,
)```
Arguments

ax
An `apexchart()` 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.

Value
An `apexchart()` htmlwidget object.

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

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

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

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

---

**ax_markers**

**Markers properties**

**Description**

Markers properties

**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` An `apexchart()` 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
An apexchart() htmlwidget object.

Note
See 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_nodata Configuration for charts with no data

Description
Configuration for charts with no data

Usage

ax_nodata(
  ax,
  text = "No data",
  align = "center",
  verticalAlign = "middle",
  color = NULL,
)
Arguments

- `ax` An `apexchart()` htmlwidget object.
- `text` The text to display when no-data is available.
- `align` Horizontal alignment: "left", "center" or "right".
- `verticalAlign` Vertical alignment: "top", "middle" or "bottom".
- `color` ForeColor of the text.
- `fontSize` FontSize of the text.
- `fontFamily` FontFamily of the text.
- `offsetX`, `offsetY` Text offset.

Value

An `apexchart()` htmlwidget object.

Examples

```r
ext <- data.frame(
  var1 = character(0),
  var2 = numeric(0)
)
apex(ex, aes(var1, var2), "column") %>%
  ax_nodata(
    text = "Sorry no data to visualize",
    fontSize = "30px"
  )
```

Description

Specific options for chart
Usage

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

Arguments

- **ax**: An `apexchart()` htmlwidget object.
- **bar**: See `bar_opts`.
- **heatmap**: See `heatmap_opts`.
- **radialBar**: See `radialBar_opts`.
- **pie**: See `pie_opts`.
- **bubble**: See `bubble_opts`.
- **...**: Additional parameters.

Value

An `apexchart()` htmlwidget object.

Examples

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

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

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

```r
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

```r
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**

`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$df)
            # Append new line of data
            df <- rbind(
                df, data.frame(
                    date = df$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
            )
        })
    }
}
```
Description
Responsive options

Usage
ax_responsive(ax, ...)

Arguments
ax An apexchart() htmlwidget object.
... Additional parameters.

Value
An apexchart() htmlwidget object.

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

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

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

ax_legend(position = "right")
ax_responsive(
  list(
    breakpoint = 1000,
    options = list(
      plotOptions = list(""
ax_states

Charts' states

Description
Charts' states

Usage

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

Arguments

- `ax` An `apexchart()` htmlwidget object.
- `normal` A list of parameters.
- `hover` A list of parameters.
- `active` A list of parameters.
- `...` Additional parameters.

Value
An `apexchart()` htmlwidget object.

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

Examples

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

# Inverse effect on hover
apex(
  data = mpg,
  mapping = aes(x = manufacturer),
  type = "bar"
) %>%
```
**ax_stroke**

```r
ax_states(
  hover = list(
    filter = list(
      type = "darken"
    )
  )
)
```

---

**ax_stroke**  
*Stroke properties*

**Description**

Stroke properties

**Usage**

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

**Arguments**

- **ax**  
  An `apexchart()` 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

An apexchart() 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,
    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)
  )

ax_subtitle

Chart's subtitle

Description

Chart's subtitle

Usage

ax_subtitle(
  ax,
  text = NULL,
  align = NULL,
  margin = NULL,
  offsetX = NULL,
  offsetY = NULL,
  floating = NULL,
  style = NULL,
Arguments

ax An apexchart() 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.
style List with two items: fontSize (Font Size of the subtitle text) and color (Font color of the subtitle text).

Value

An apexchart() htmlwidget object.

Note

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

Examples

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"
)
ax_theme

Theme for charts

Description
Theme for charts

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

Arguments
- ax: An apexchart() htmlwidget object.
- mode: use light or dark theme.
- palette: Character. Available palettes: "palette1" to "palette10".
- monochrome: A list of parameters.
- ...: Additional parameters.

Value
An apexchart() htmlwidget object.

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

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

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

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

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

## ax_title

### Chart's title

#### 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` An `apexchart()` 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

An `apexchart()` htmlwidget object.
Note

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

Examples

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

ax(
  data = economics,
  mapping = aes(x = date, y = uempmed),
  type = "line"
) %>%
  ax_title(
    text = "Median duration of unemployment, in weeks"
  )

ax_tooltip

# Tool tip options

Description

Tooltip options

Usage

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**: An `apexchart()` 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**

An `apexchart()` htmlwidget object.

**Note**

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

**Examples**

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

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

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

data = mpg,
mapping = aes(x = manufacturer, 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**  
**X-axis options**

### 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

- **ax**  
  An `apexchart()` htmlwidget object.
- **type**  
  Character. Available Options: "categories" and "datetime".
categories  Categories are labels which are displayed on the x-axis.
labels  A list of parameters.
axisBorder  A list of parameters.
axisTicks  A list of parameters.
tickAmount  Number of Tick Intervals to show.

min  Lowest number to be set for the x-axis. The graph drawing beyond this number will be clipped off.
max  Highest number to be set for the x-axis. The graph drawing beyond this number will be clipped off.

range  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.

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  
An \texttt{apexchart()} htmlwidget object.

Note  
See \url{https://apexcharts.com/docs/options/xaxis/}

Examples  
\begin{verbatim}
data("mpg", package = "ggplot2")

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

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

# force to not rotate
apex(
  data = mpg,
  mapping = aes(x = manufacturer)
)
) %>%
  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"
      )
    )
  )

---

ax_yaxis

Y-axis options

Description

Y-axis options
ax_yaxis

Usage

ax_yaxis(
  ax,
  opposite = NULL,
  tickAmount = NULL,
  max = NULL,
  min = NULL,
  floating = NULL,
  labels = NULL,
  axisBorder = NULL,
  axisTicks = NULL,
  title = NULL,
  tooltip = NULL,
  crosshairs = NULL,
  ...
)

Arguments

- **ax**: An `apexchart()` 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

An `apexchart()` htmlwidget object.

Note

See [https://apexcharts.com/docs/options/yaxis/](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") %>%
apex_yaxis(
    labels = list(
        formatter = htmlwidgets::JS("function(value) {return value + '\u00b0C';}")
    )
)

---

ax_yaxis2  Secondary Y-axis options

Description

Secondary Y-axis options

Usage

ax_yaxis2(ax, ...)

Arguments

ax  An apexchart() htmlwidget object.
...
See arguments from ax_yaxis.

Value

An apexchart() htmlwidget object.
Examples

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

eco <- economics_long %>%
  subset(variable %in% c("pce", "pop")) %>%
  transform(value = round(value))

# add second y-axis
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"))

# Customize axis a bit more
apex(eco, aes(x = date, y = value, color = variable), type = "line") %>%
  ax_yaxis(
    title = list(text = "Pce"),
    axisBorder = list(
      show = TRUE,
      color = "#008FFB"
    ),
    labels = list(
      style = list(
        colors = "#008FFB"
      )
    ),
    tooltip = list(
      enabled = TRUE
    )
  ) %>%
  ax_yaxis2(
    opposite = TRUE,
    min = 160000,
    forceNiceScale = TRUE,
    title = list(text = "Pop"),
    axisBorder = list(
      show = TRUE,
      color = "#00E396"
    ),
    labels = list(
      style = list(
        colors = "#00E396"
      )
    ),
    tooltip = list(
      enabled = TRUE
    )
  )
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/.

Examples

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

apex(mpg, aes(manufacturer)) %>%
  ax_plotOptions(
    bar = bar_opts(
      endingShape = "rounded",
    ),
  )
```
bubble_opts

```
columnWidth = 100,
distributed = TRUE
```

---

**bubble_opts**  
*Bubble options*

---

**Description**

Use these options in `ax_plotOptions`.

**Usage**

```
bubble_opts(minBubbleRadius, maxBubbleRadius, ...)
```

**Arguments**

- `minBubbleRadius`
  - Minimum radius size of a bubble. If a bubble value is too small to be displayed, this size will be used.
- `maxBubbleRadius`
  - Maximum radius size of a bubble. If a bubble value is too large to cover the chart, this size will be used.
- `...`
  - Additional parameters.

**Value**

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

**Note**

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

**Examples**

```
apex(
  data = mtcars,
  type = "scatter",
  mapping = aes(x = wt, y = mpg, z = qsec)
) %>%
ax_plotOptions(
  bubble = bubble_opts(
    minBubbleRadius = 1,
    maxBubbleRadius = 20
  )
)
```
**candles**  
*Candlestick demo data*

**Description**

Candlestick demo data

**Usage**

candles

**Format**

A data frame with 60 observations and the following 5 variables:

- **datetime** Timestamp.
- **open** Open value.
- **high** Highest value.
- **low** Lowest value.
- **close** Close value.

**Source**


---

**climate_paris**  
*Paris Climate*

**Description**


**Usage**

climate_paris

**Format**

A data frame with 12 observations and the following 3 variables:

- **month** Month
- **temperature** Temperature (in degree celsius).
- **precipitation** Precipitation (in mm).

**Source**

**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 = TRUE,
    update_synced_charts = 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.
- `update_synced_charts` All the charts in a group should also update when one chart in a group is updated.

**consumption**

Electricity consumption and forecasting

**Description**

Electricity consumption per day in France for January and February of year 2020.

**Usage**

```r
consumption
```
**Format**

A data frame with 120 observations and the following 3 variables:

- **date** date.
- **type** Type of data: realized or forecast.
- **value** Value in giga-watt per hour.

**Source**

Rte (Electricity Transmission Network in France) ([https://data.rte-france.com/](https://data.rte-france.com/))

---

<table>
<thead>
<tr>
<th>events_opts</th>
<th>Events options</th>
</tr>
</thead>
</table>

**Description**

Events options

**Usage**

```r
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.
events_opts

- **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('click', config.selectedDataPoints)
            }
          )
        )
    })
  }
}
```
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

\[ \text{format_num}(\text{format}, \text{prefix} = \text{""}, \text{suffix} = \text{""}, \text{locale} = \text{"en-US"}) \]

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
\[
\text{dat} \leftarrow \text{data.frame}(
\quad \text{labels} = \text{c("apex", "charts")},
\quad \text{values} = \text{c(1e4, 2e4)}
\)
\]

\[
\text{apex}((\text{dat}, \text{aes(\text{labels}, \text{values}), "column"}) \%\%
\quad \text{ax_yaxis(\text{labels} = \text{list(}
\quad \quad \text{formatter} = \text{format_num("-s")}
\quad \))}
\]

\[
\text{apex}((\text{dat}, \text{aes(\text{labels}, \text{values} \times 100), "column"}) \%\%
\quad \text{ax_yaxis(\text{labels} = \text{list(}
\quad \quad \text{formatter} = \text{format_num("-s")}
\quad ))}
\]

# Percentage
\[
\text{dat} \leftarrow \text{data.frame}(
\quad \text{labels} = \text{c("apex", "charts")},
\quad \text{values} = \text{c(0.45, 0.55)}
\)
\]
```r
# 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("%.0%")
  ))

# 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**

<table>
<thead>
<tr>
<th>Description</th>
<th><strong>Heatmap options</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use these options in <em>ax_plotOptions</em>.</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

```r
heatmap_opts(
  radius = NULL,
```
heatmap_opts

```r
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**

```r
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"
) %>%
apx_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")
      )
    )
  )
)
```
label  

**Description**

Label for annotations

**Usage**

```r
label(
  text = NULL,
  borderColor = NULL,
  borderWidth = NULL,
  textAnchor = NULL,
  position = NULL,
  offsetX = NULL,
  offsetY = NULL,
  background = NULL,
  color = NULL,
  fontSize = NULL,
  fontWeight = NULL,
  fontFamily = NULL,
  cssClass = NULL,
  padding = c(2, 5, 2, 5)
)
```

**Arguments**

- **text**: Text for the annotation label.
- **borderColor**: Border color for the label.
- **borderWidth**: Border width for the label.
- **textAnchor**: The alignment of text relative to label’s drawing position.
- **position**: Available options: left or right.
- **offsetX**: Sets the left offset for annotation label.
- **offsetY**: Sets the top offset for annotation label.
- **background**: Background Color for the annotation label.
- **color**: ForeColor for the annotation label.
- **fontSize**: FontSize for the annotation label.
- **fontWeight**: Font-weight for the annotation label.
- **fontFamily**: Font-family for the annotation label.
- **cssClass**: A custom Css Class to give to the annotation label elements.
- **padding**: Padding for the label: top, right, bottom, left.
parse_df

Value
A list that can be used in `add_shade, add_point, add_event, add_event_marker`.

---

**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**

```r
# 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.

### Value

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

### Note

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

```r
data("mpg", package = "ggplot2")
apex(mpg, aes(cyl), 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

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

Examples

```r
apexchart() %>%
  ax_chart(type = "radialBar") %>%
  ax_plotOptions(
    radialBar = radialBar_opts(
      startAngle = -135,
      endAngle = 135,
      dataLabels = list(
        name = list(
          fontSize = "16px",
         [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")
```

Description

Run Shiny input events examples

Usage

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

Arguments

- `example` Name of the example.
run_demo_sparkbox

Examples
if (interactive()) {
    run_demo_input("click")
    run_demo_input("zoom")
    run_demo_input("selection")
}

run_demo_sparkbox  Run Shiny spark boxes example

Description
Run Shiny spark boxes example

Usage
run_demo_sparkbox()

Examples
if (interactive()) {
    run_demo_sparkbox()
}

run_demo_sync  Run Shiny synchronization example

Description
Run Shiny synchronization example

Usage
run_demo_sync()

Examples
if (interactive()) {
    run_demo_sync()
}
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()
)
```

Arguments

- `ax`: An `apexchart()` 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 `apexchart()` 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_demo_input("click")
}
```

---

**Description**

Retrieve chart’s base64 dataURI.

**Usage**

```r
set_input_export(ax, inputId, session = shiny::getDefaultReactiveDomain())
```

**Arguments**

- `ax` An `apexchart()` htmlwidget object.
- `inputId` The id that will be used server-side for retrieving data.
- `session` The Shiny session.

**Value**

An `apexchart()` htmlwidget object.

**Examples**

```r
library(shiny)
library(apexcharter)

ui <- fluidPage(
  fluidRow(
```
set_input_selection

Retrieve selection information in Shiny

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 apexchart() 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 apexchart() htmlwidget object.

Examples

library(apexcharter)
data("economics", package = "ggplot2")
# 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**

Retrieves zoom information in Shiny

### Description

Retrieve zoom information in Shiny

### Usage

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

### Arguments

- `ax` An `apexchart()` htmlwidget object.
- `inputId` The id that will be used server-side for retrieving zoom.
- `session` The Shiny session.

### Value

An `apexchart()` htmlwidget object.

### Note

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

### Examples

```r
if (interactive()) {
  run_demo_input("zoom")
}
```
Description

Fixed tooltip

Usage

```r
set_tooltip_fixed(
  ax,
  position = c("topLeft", "topRight", "bottomLeft", "bottomRight"),
  offsetX = NULL,
  offsetY = NULL
)
```

Arguments

- `ax` An `apexchart()` htmlwidget object.
- `position` Predefined position: "topLeft", "topRight", "bottomLeft" or "bottomRight".
- `offsetX` Sets the left offset for the tooltip container in fixed position.
- `offsetY` Sets the top offset for the tooltip container in fixed position.

Value

An `apexchart()` htmlwidget object.

Examples

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

apex(
  data = tail(economics, 350),
  mapping = aes(x = date, y = uempmed),
  type = "line"
)

set_tooltip_fixed()
```
spark_box  

Create a box with a sparkline

Description

Create a box with a sparkline

Usage

spark_box(
  data,
  title = NULL,
  subtitle = NULL,
  color = "#2E93fA",
  background = "#FFF",
  type = c("area", "line", "spline", "column"),
  synchronize = NULL,
  title_style = NULL,
  subtitle_style = NULL,
  width = NULL,
  height = NULL,
  elementId = NULL
)

Arguments

data A data.frame-like object with at least two columns, first is mapped to x-axis, second to y-axis.
title Title to display in the box.
title Subtitle to display in the box.
color Color of the chart.
background Background color of the box.
type Type of chart, currently type supported are: "area" (default), "line", "spline", "column".
synchronize Give a common id to charts to synchronize them (tooltip and zoom).
title_style, subtitle_style A list of named attributes to style the title / subtitle, possible values are fontSize, fontWeight, fontFamily, color.
width, height A numeric input in pixels.
elementId Use an explicit element ID for the widget.

Value

An apexcharts htmlwidget object.
Note

In Shiny use `sparkBoxOutput` / `renderSparkBox` to render boxes, see example. Boxes have CSS class "apexchart-spark-box" if you need more styling.

Examples

```r
library(apexcharter)

spark_data <- data.frame(
  date = Sys.Date() + 1:20,
  var1 = round(rnorm(20, 50, 10)),
  var2 = round(rnorm(20, 50, 10)),
  var3 = round(rnorm(20, 50, 10))
)

spark_box(
  data = spark_data,
  title = mean(spark_data$var1),
  subtitle = "Variable 1"
)

# In Shiny
if (interactive()) {
  run_sparkbox_demo()
}
```

---

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`

Format

A data frame with 913 observations and 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

* datasets
candles, 68
climate_paris, 68
consumption, 69
 unhcr_ts, 89
%>% (apexcharter-exports), 20
add-line, 3
add-shade, 5
add-vh-lines, 7
add_event, 9, 11, 77
add_event_marker, 9, 10, 12, 77
add_hline (add-vh-lines), 7
add_line (add-line), 3
add_point, 11, 77
add_shade, 77
add_shade (add-shade), 5
add_shade_weekend (add-shade), 5
add_smooth_line (add-line), 3
add_vline (add-vh-lines), 7
aes (apexcharter-exports), 20
apex, 13
apex(), 19
apex-facets, 15
apex_grid, 25
apexchart, 19
apexcharter-exports, 20
apexcharter-package, 3
apexcharter-shiny, 20
apexcharter-shiny-facets, 22
apexcharter-shiny-grid, 23
apexchartOutput (apexcharter-shiny-grid), 20
apexchartProxy, 24
apexfacetOutput (apexcharter-shiny-facets), 22
apexfacetOutput(), 16, 21
apexgridOutput (apexcharter-shiny-grid), 23
ax-series, 26
ax_annotations, 27
ax_chart, 30, 71
ax_colors, 33
ax_colors_manual, 34
ax_dataLabels, 36
ax_facet_grid (apex-facets), 15
ax_facet_grid(), 21, 22
ax_facet_wrap (apex-facets), 15
ax_facet_wrap(), 21, 22
ax_fill, 37
ax_grid, 39
ax_labels, 40
ax_labels2 (ax_labels), 40
ax_labs, 41
ax_legend, 42
ax_markers, 44
ax_nodata, 45
ax_plotOptions, 46, 66, 67, 74, 75, 78–80
ax_proxy_options, 48
ax_proxy_series, 49
ax_responsive, 51
ax_series, 77
ax_series (ax-series), 26
ax_series2 (ax-series), 26
ax_states, 52
ax_stroke, 53
ax_subtitle, 54
ax_theme, 56
ax_title, 57
ax_tooltip, 58
ax_xaxis, 60
ax_yaxis, 62, 64
ax_yaxis2, 64
bar_opts, 47, 65
bubble_opts, 47, 67
INDEX

candles, 68
climate_paris, 68
config_update, 69
config_update(), 14, 19
consumption, 69

events_opts, 31, 70

format_date, 72
format_num, 73

heatmap_opts, 47, 74

JS (apexcharter-exports), 20

label, 6, 8–10, 12, 76
label_value (apexcharter-exports), 20

lm, 4
loess, 4

parse_df, 77
pie_opts, 47, 78

radialBar_opts, 47, 79
renderApexchart (apexcharter-shiny), 20
renderApexfacet
    (apexcharter-shiny-facets), 22
renderApexfacet(), 16, 21
renderApexgrid
    (apexcharter-shiny-grid), 23
renderSparkBox (apexcharter-shiny), 20
run_demo_input, 80
run_demo_sparkbox, 81
run_demo_sync, 81

set_input_click, 82
set_input_export, 83
set_input_selection, 84
set_input_zoom, 86
set_tooltip_fixed, 87
spark_box, 88
sparkBoxOutput (apexcharter-shiny), 20

unhcr_ts, 89

vars (apexcharter-exports), 20