Package 'ggiraph'

November 4, 2016

Type Package

Title Make 'ggplot2' Graphics Interactive

Description Create interactive 'ggplot2' graphics using 'htmlwidgets'.

Version 0.3.2

License GPL-3

Copyright See file COPYRIGHTS.

Depends ggplot2 (>= 2.1.0)

Imports grid, htmlwidgets (>= 0.6), htmltools, stats, plyr, rvg (>= 0.1.1), xml2 (>= 1.0.0)

Suggests knitr, rmarkdown, maps, shiny

VignetteBuilder knitr

URL https://davidgohel.github.io/ggiraph

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

RoxygenNote 5.0.1.9000

NeedsCompilation no

Author David Gohel [aut, cre],
Mike Bostock [cph] (d3.js)

Maintainer David Gohel <david.gohel@ardata.fr>

Repository CRAN

Date/Publication 2016-11-04 22:27:22

R topics documented:

`drawDetails.interactive_points_grob` *interactive_points_grob drawing*

Description

draw an interactive_points_grob

Usage

```
## S3 method for class 'interactive_points_grob'
drawDetails(x, recording)
```

Arguments

`x`	A grid grob.
`recording`	A logical value indicating whether a grob is being added to the display list or redrawn from the display list.

drawDetails.interactive_polygon_grob

interactive_polygon_grob drawing

Description

draw an interactive_polygon_grob

Usage

```
## S3 method for class 'interactive_polygon_grob'
drawDetails(x, recording)
```

Arguments

x	A grid grob.
recording	A logical value indicating whether a grob is being added to the display list or redrawn from the display list.

drawDetails.interactive_polyline_grob

interactive_polyline_grob drawing

Description

draw an interactive_polyline_grob

Usage

```
## S3 method for class 'interactive_polyline_grob'
drawDetails(x, recording)
```

Arguments

x	A grid grob.
recording	A logical value indicating whether a grob is being added to the display list or redrawn from the display list.

`drawDetails.interactive_rect_grob` *interactive_rect_grob drawing*

Description

draw an interactive_rect_grob

Usage

```
## S3 method for class 'interactive_rect_grob'
drawDetails(x, recording)
```

Arguments

`x`	A grid grob.
`recording`	A logical value indicating whether a grob is being added to the display list or redrawn from the display list.

`drawDetails.interactive_segments_grob` *interactive_segments_grob drawing*

Description

draw an interactive_segments_grob

Usage

```
## S3 method for class 'interactive_segments_grob'
drawDetails(x, recording)
```

Arguments

`x`	A grid grob.
`recording`	A logical value indicating whether a grob is being added to the display list or redrawn from the display list.

drawDetails.interactive_text_grob *interactive_text_grob drawing*

Description

draw an interactive_text_grob

Usage

```
## S3 method for class 'interactive_text_grob'
drawDetails(x, recording)
```

Arguments

x	A grid grob.
recording	A logical value indicating whether a grob is being added to the display list or redrawn from the display list.

geom_bar_interactive *interactive bars*

Description

The geometry is based on geom_bar. See the documentation for those functions for more details.

Usage

```
geom_bar_interactive(mapping = NULL, data = NULL, stat = "count",
  position = "stack", ..., width = NULL, na.rm = FALSE,
  show.legend = NA, inherit.aes = TRUE)
```

Arguments

mapping	The aesthetic mapping, see geom_point.
data	A data frame, see geom_point.
stat	The statistical transformation to use on the data for this layer, as a string, see geom_point.
position	Postion adjustment, see geom_point.
...	other arguments passed on to layer. See geom_point.
width	Bar width.
na.rm	See geom_point.
show.legend	See geom_point.
inherit.aes	See geom_point.

See Also

ggiraph

Examples

```
g <- ggplot(mpg, aes( x = class, tooltip = class,
        data_id = class ) ) +
  geom_bar_interactive()
ggiraph(code = print(g))

dat <- data.frame( name = c( "David", "Constance", "Leonie" ),
    gender = c( "Male", "Female", "Female" ),
    height = c(172, 159, 71 ) )
g <- ggplot(dat, aes( x = name, y = height, tooltip = gender,
        data_id = name ) ) +
  geom_bar_interactive(stat = "identity")
ggiraph(code = print(g))
```

---

geom_boxplot_interactive

interactive boxplot

Description

The geometry is based on geom_boxplot. See the documentation for those functions for more details.

Usage

```
geom_boxplot_interactive(mapping = NULL, data = NULL, stat = "boxplot",
  position = "dodge", ..., outlier.colour = NULL, outlier.color = NULL,
  outlier.shape = 19, outlier.size = 1.5, outlier.stroke = 0.5,
  notch = FALSE, notchwidth = 0.5, varwidth = FALSE, na.rm = FALSE,
  show.legend = NA, inherit.aes = TRUE)
```

Arguments

mapping	The aesthetic mapping, see geom_point.
data	A data frame, see geom_point.
stat	see geom_boxplot.
position	Postion adjustment, see geom_point.
...	other arguments passed on to layer. See geom_point.
outlier.colour	see geom_boxplot.
outlier.color	see geom_boxplot.
outlier.shape	see geom_boxplot.

outlier.size	see geom_boxplot.
outlier.stroke	see geom_boxplot.
notch	see geom_boxplot.
notchwidth	see geom_boxplot.
varwidth	see geom_boxplot.
na.rm	See geom_point.
show.legend	See geom_point.
inherit.aes	See geom_point.

See Also

ggiraph

Examples

```
# add interactive boxplot -------
p <- ggplot(mpg,
  aes(x = class, y = hwy, tooltip = class)) +
  geom_boxplot_interactive()

ggiraph(code = print(p), width = .5)


p <- ggplot(mpg, aes(x = drv, y = hwy, tooltip = class, fill = class)) +
  geom_boxplot_interactive(outlier.colour = "red") +
  guides(fill = "none") + theme_minimal()

ggiraph(code = print(p), width = .5)
```

---

geom_map_interactive *interactive polygons from a reference map.*

Description

The geometry is based on geom_map. See the documentation for those functions for more details.

Usage

```
geom_map_interactive(mapping = NULL, data = NULL, map, stat = "identity",
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE, ...)
```

Arguments

mapping	The aesthetic mapping, see geom_point.
data	A data frame, see geom_point.
map	Data frame that contains the map coordinates. See geom_map.
stat	The statistical transformation to use on the data for this layer, as a string, see geom_point.
na.rm	See geom_point.
show.legend	See geom_point.
inherit.aes	See geom_point.
...	other arguments passed on to layer. See geom_point.

See Also

ggiraph

Examples

```
# add interactive maps to a ggplot -------
crimes <- data.frame(state = tolower(rownames(USArrests)), USArrests)

# create tooltips and onclick events
states_ <- sprintf("<p>%s</p>",
                   as.character(crimes$state) )
table_ <- paste0(
  "<table><tr><td>UrbanPop</td>",
  sprintf("<td>%.0f</td>", crimes$UrbanPop),
  "</tr><tr>",
  "<td>Assault</td>",
  sprintf("<td>%.0f</td>", crimes$Assault),
  "</tr></table>"
)

onclick <- sprintf(
  "window.open(\"%s%s\")",
  "http://en.wikipedia.org/wiki/",
  as.character(crimes$state)
)


crimes$labs <- paste0(states_, table_)
crimes$onclick = onclick

if (require("maps") ) {
  states_map <- map_data("state")
  gg_map <- ggplot(crimes, aes(map_id = state))
  gg_map <- gg_map + geom_map_interactive(aes(
                  fill = Murder,
                  tooltip = labs,
                  data_id = state,
```

```
                  onclick = onclick
                ),
                map = states_map) +
    expand_limits(x = states_map$long, y = states_map$lat)
  ggiraph(code = print(gg_map), width = .8)
}
```

---

geom_path_interactive *interactive observations connections*

Description

These geometries are based on geom_path and geom_line. See the documentation for those functions for more details.

Usage

```
geom_path_interactive(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", lineend = "butt", linejoin = "round",
  linemitre = 1, na.rm = FALSE, arrow = NULL, show.legend = NA,
  inherit.aes = TRUE, ...)

geom_line_interactive(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, ...)
```

Arguments

mapping	The aesthetic mapping, see geom_point.
data	A data frame, see geom_point.
stat	The statistical transformation to use on the data for this layer, as a string, see geom_point.
position	Postion adjustment, see geom_point.
lineend	Line end style (round, butt, square)
linejoin	Line join style (round, mitre, bevel)
linemitre	Line mitre limit (number greater than 1)
na.rm	See geom_point.
arrow	Arrow specification, as created by arrow
show.legend	See geom_point.
inherit.aes	See geom_point.
...	other arguments passed on to layer. See geom_point.

See Also

ggiraph

Examples

```
# add interactive paths to a ggplot -------
# geom_line_interactive example -----

gg <- ggplot(economics_long,
    aes(date, value01, colour = variable, tooltip = variable, data_id = variable)) +
  geom_line_interactive(size = .75)
ggiraph(code = {print(gg)}, hover_css = "stroke:red;")


# create datasets -----
id = paste0("id", 1:10)
data = expand.grid(list(
variable = c("2000", "2005", "2010", "2015"),
id = id
)
)
groups = sample(LETTERS[1:3], size = length(id), replace = TRUE)
data$group = groups[match(data$id, id)]
data$value = runif(n = nrow(data))
data$tooltip = paste0('line ', data$id )
data$onclick = paste0("alert(\"", data$id, "\")" )

cols = c("orange", "orange1", "orange2", "navajowhite4", "navy")
dataset2 <- data.frame(x = rep(1:20, 5),
y = rnorm(100, 5, .2) + rep(1:5, each=20),
z = rep(1:20, 5),
grp = factor(rep(1:5, each=20)),
color = factor(rep(1:5, each=20)),
label = rep(paste0( "id ", 1:5 ), each=20),
onclick = paste0(
  "alert(\"",
  sample(letters, 100, replace = TRUE),
  "\")" )
)


# plots ---
gg_path_1 = ggplot(data, aes(variable, value, group = id,
colour = group, tooltip = tooltip, onclick = onclick, data_id = id)) +
geom_path_interactive(alpha = 0.5)

gg_path_2 = ggplot(data, aes(variable, value, group = id, data_id = id,
tooltip = tooltip)) +
geom_path_interactive(alpha = 0.5) +
facet_wrap( ~ group )

gg_path_3 = ggplot(dataset2) +
```

```
geom_path_interactive(aes(x, y, group=grp, data_id = label,
color = color, tooltip = label, onclick = onclick), size = 1 )

# ggiraph widgets ---
ggiraph(code = {print(gg_path_1)}, hover_css = "stroke-width:3px;")
ggiraph(code = {print(gg_path_2)}, hover_css = "stroke:orange;stroke-width:3px;")
ggiraph(code = {print(gg_path_3)}, hover_css = "stroke-width:10px;")
```

`geom_point_interactive` *interactive points*

Description

The geometry is based on `geom_point`. See the documentation for those functions for more details.

Usage

```
geom_point_interactive(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, ...)
```

Arguments

`mapping`	The aesthetic mapping, see `geom_point`.
`data`	A data frame, see `geom_point`.
`stat`	The statistical transformation to use on the data for this layer, as a string, see `geom_point`.
`position`	Postion adjustment, see `geom_point`.
`na.rm`	See `geom_point`.
`show.legend`	See `geom_point`.
`inherit.aes`	See `geom_point`.
`...`	other arguments passed on to layer. See `geom_point`.

Note

The following shapes id 3, 4 and 7 to 14 are composite symbols and should not be used.

See Also

`ggiraph`

Examples

```
# add interactive points to a ggplot -------
# create dataset
dataset = iris
dataset$tooltip = dataset$Species
dataset$clickjs = paste0("alert(\"",dataset$Species, "\")" )

# plots
gg_point = ggplot(dataset, aes(x = Sepal.Length, y = Petal.Width,
color = Species, tooltip = tooltip, onclick = clickjs) ) +
geom_point_interactive()

ggiraph(code = {print(gg_point)})
```

---

geom_polygon_interactive

interactive polygons

Description

The geometry is based on geom_polygon. See the documentation for those functions for more details.

Usage

```
geom_polygon_interactive(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, ...)
```

Arguments

mapping	The aesthetic mapping, see geom_point.
data	A data frame, see geom_point.
stat	The statistical transformation to use on the data for this layer, as a string, see geom_point.
position	Postion adjustment, see geom_point.
na.rm	See geom_point.
show.legend	See geom_point.
inherit.aes	See geom_point.
...	other arguments passed on to layer. See geom_point.

See Also

ggiraph

Examples

```
# add interactive polygons to a ggplot -------
# create data
ids <- factor(c("1.1", "2.1", "1.2", "2.2", "1.3", "2.3"))

values <- data.frame(
id = ids,
value = c(3, 3.1, 3.1, 3.2, 3.15, 3.5) )
positions <- data.frame(
id = rep(ids, each = 4),
x = c(2, 1, 1.1, 2.2, 1, 0, 0.3, 1.1, 2.2, 1.1, 1.2, 2.5, 1.1, 0.3,
0.5, 1.2, 2.5, 1.2, 1.3, 2.7, 1.2, 0.5, 0.6, 1.3),
y = c(-0.5, 0, 1, 0.5, 0, 0.5, 1.5, 1, 0.5, 1, 2.1, 1.7, 1, 1.5,
2.2, 2.1, 1.7, 2.1, 3.2, 2.8, 2.1, 2.2, 3.3, 3.2) )

datapoly <- merge(values, positions, by=c("id"))

datapoly$oc = "alert(this.getAttribute(\"data-id\"))"

# create a ggplot -----
gg_poly_1 <- ggplot(datapoly, aes( x = x, y = y ) ) +
geom_polygon_interactive(aes(fill = value, group = id,
tooltip = value, data_id = value, onclick = oc))

# display ------
ggiraph(code = {print(gg_poly_1)})
```

---

geom_rect_interactive *interactive rectangles*

Description

These geometries are based on geom_rect and geom_tile. See the documentation for those functions for more details.

Usage

```
geom_rect_interactive(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, ...)

geom_tile_interactive(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)
```

Arguments

mapping	The aesthetic mapping, see geom_point.
data	A data frame, see geom_point.
stat	The statistical transformation to use on the data for this layer, as a string, see geom_point.
position	Postion adjustment, see geom_point.
na.rm	See geom_point.
show.legend	See geom_point.
inherit.aes	See geom_point.
...	other arguments passed on to layer. See geom_point.

Note

Converting a raster to svg elements could inflate dramatically the size of the svg and make it unreadable in a browser. Function geom_tile_interactive should be used with caution, total number of rectangles should be small.

See Also

ggiraph

Examples

```
# add interactive polygons to a ggplot -------
dataset = data.frame( x1 = c(1, 3, 1, 5, 4),
x2 = c(2, 4, 3, 6, 6),
y1 = c( 1, 1, 4, 1, 3),
y2 = c( 2, 2, 5, 3, 5),
t = c( 'a', 'a', 'a', 'b', 'b'),
r = c( 1, 2, 3, 4, 5),
tooltip = c("ID 1", "ID 2", "ID 3", "ID 4", "ID 5"),
uid = c("ID 1", "ID 2", "ID 3", "ID 4", "ID 5"),
oc = rep("alert(this.getAttribute(\"data-id\"))", 5)
)

gg_rect = ggplot() +
scale_x_continuous(name="x") +
scale_y_continuous(name="y") +
geom_rect_interactive(data=dataset,
mapping = aes(xmin = x1, xmax = x2,
ymin = y1, ymax = y2, fill = t,
tooltip = tooltip, onclick = oc, data_id = uid ),
color="black", alpha=0.5) +
geom_text(data=dataset,
aes(x = x1 + ( x2 - x1 ) / 2, y = y1 + ( y2 - y1 ) / 2,
label = r ),
size = 4 )
```

```
ggiraph(code = {print(gg_rect)})
df <- data.frame(
  id = rep(c("a", "b", "c", "d", "e"), 2),
  x = rep(c(2, 5, 7, 9, 12), 2),
  y = rep(c(1, 2), each = 5),
  z = factor(rep(1:5, each = 2)),
  w = rep(diff(c(0, 4, 6, 8, 10, 14)), 2)
)
ggiraph( code = {
  print(
    ggplot(df, aes(x, y, tooltip = id)) + geom_tile_interactive(aes(fill = z))
  )
})


# correlation dataset ----
cor_mat <- cor(mtcars)
diag( cor_mat ) <- NA
var1 <- rep( row.names(cor_mat), ncol(cor_mat) )
var2 <- rep( colnames(cor_mat), each = nrow(cor_mat) )
cor <- as.numeric(cor_mat)
cor_mat <- data.frame( var1 = var1, var2 = var2,
  cor = cor, stringsAsFactors = FALSE )
cor_mat[["tooltip"]] <-
  sprintf("<i>`%s`</i> vs <i>`%s`</i>:</br><code>%.03f</code>",
  var1, var2, cor)

# ggplot creation and ggiraph printing ----
p <- ggplot(data = cor_mat, aes(x = var1, y = var2) ) +
  geom_tile_interactive(aes(fill = cor, tooltip = tooltip), colour = "white") +
  scale_fill_gradient2(low = "#BC120A", mid = "white", high = "#BC120A", limits = c(-1, 1)) +
  coord_equal()
ggiraph( code = print(p))
```

`geom_segment_interactive` *Line interactive segments*

Description

The geometry is based on `geom_segment`. See the documentation for those functions for more details.

Usage

```
geom_segment_interactive(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", arrow = NULL, lineend = "butt", na.rm = FALSE,
  show.legend = NA, inherit.aes = TRUE, ...)
```

Arguments

mapping	The aesthetic mapping, see geom_point.
data	A data frame, see geom_point.
stat	The statistical transformation to use on the data for this layer, as a string, see geom_point.
position	Postion adjustment, see geom_point.
arrow	Arrow specification, as created by ?grid::arrow
lineend	Line end style (round, butt, square)
na.rm	See geom_point.
show.legend	See geom_point.
inherit.aes	See geom_point.
...	other arguments passed on to layer. See geom_point.

See Also

ggiraph

Examples

```
# add interactive segments to a ggplot -------
counts <- as.data.frame(table(x = rpois(100,5)))
counts$x <- as.numeric( as.character(counts$x) )
counts$xlab <- paste0("bar",as.character(counts$x) )

gg_segment_1 <- ggplot(data = counts, aes(x = x, y = Freq,
yend = 0, xend = x, tooltip = xlab ) ) +
geom_segment_interactive( size = I(10))

dataset = data.frame(x=c(1,2,5,6,8),
y=c(3,6,2,8,7),
vx=c(1,1.5,0.8,0.5,1.3),
vy=c(0.2,1.3,1.7,0.8,1.4),
labs = paste0("Lab", 1:5))
dataset$clickjs = paste0("alert(\"",dataset$labs, "\")" )

gg_segment_2 = ggplot() +
geom_segment_interactive(data=dataset, mapping=aes(x=x, y=y,
xend=x+vx, yend=y+vy, tooltip = labs, onclick=clickjs ),
arrow=grid::arrow(length = grid::unit(0.03, "npc")),
size=2, color="blue") +
geom_point(data=dataset, mapping=aes(x=x, y=y),
size=4, shape=21, fill="white")

ggiraph(code = {print(gg_segment_1)})
ggiraph(code = {print(gg_segment_2)})
```

geom_text_interactive *interactive textual annotations.*

Description

The geometry is based on geom_text. See the documentation for those functions for more details.

Usage

```
geom_text_interactive(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", parse = FALSE, ..., nudge_x = 0, nudge_y = 0,
  check_overlap = FALSE, na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)
```

Arguments

mapping	The aesthetic mapping, see geom_point.
data	A data frame, see geom_point.
stat	The statistical transformation to use on the data for this layer, as a string, see geom_point.
position	Postion adjustment, see geom_point.
parse	See geom_point.
...	other arguments passed on to layer. See geom_point.
nudge_x, nudge_y	See geom_point.
check_overlap	See geom_point.
na.rm	See geom_point.
show.legend	See geom_point.
inherit.aes	See geom_point.

See Also

ggiraph

Examples

```
# add interactive polygons to a ggplot -------
## the data
dataset = mtcars
dataset$label = row.names(mtcars)

dataset$tooltip = paste0( "cyl: ", dataset$cyl, "<br/>",
       "gear: ", dataset$gear, "<br/>",
       "carb: ", dataset$carb)
```

```
## the plot
gg_text = ggplot(dataset,
                 aes(x = mpg, y = wt, label = label,
                     color = qsec,
                     tooltip = tooltip, data_id = label ) ) +
  geom_text_interactive() +
  coord_cartesian(xlim = c(0,50))

## display the plot
ggiraph(code = {print(gg_text)}, hover_css = "fill:#FF4C3B;font-style:italic;")
```

---

ggiraph *ggiraph*

Description

Create an interactive graphic to be used in a web browser.

Use `geom_zzz_interactive` to create interactive graphical elements.

Difference from original functions is that the following aesthetics are understood: `tooltip`, `onclick` and `data_id`.

Tooltips can be displayed when mouse is over graphical elements. On click actions can be set with javascript instructions. If id are associated with points, they get animated when mouse is over and can be selected when used in shiny apps.

Usage

```
ggiraph(code, ggobj = NULL, pointsize = 12, width = 0.7, width_svg = 6,
  height_svg = 6, tooltip_extra_css, hover_css, tooltip_opacity = 0.9,
  tooltip_offx = 10, tooltip_offy = 0, zoom_max = 1,
  selection_type = "multiple", selected_css, ...)
```

Arguments

`code`	Plotting code to execute
`ggobj`	ggplot objet to print. argument `code` will be ignored if this argument is supplied.
`pointsize`	the default pointsize of plotted text in pixels, default to 12.
`width`	widget width ratio (0 < width <= 1)
`width_svg, height_svg`	svg viewbox width and height in inches
`tooltip_extra_css`	extra css (added to `position: absolute;pointer-events: none;`) used to customize tooltip area.
`hover_css`	css to apply when mouse is hover and element with a data-id attribute.
`tooltip_opacity`	tooltip opacity

tooltip_offx	tooltip x offset
tooltip_offy	tooltip y offset
zoom_max	maximum zoom factor
selection_type	row selection mode ("single", "multiple", "none") when widget is in a Shiny application.
selected_css	css to apply when element is selected (shiny only).
...	arguments passed on to dsvg

Examples

```
# ggiraph simple example -------
# create dataset
dataset = iris
dataset$tooltip = dataset$Species
dataset$clickjs = paste0("alert(\"",dataset$Species, "\")" )

# plots
gg_point = ggplot(dataset, aes(x = Sepal.Length, y = Petal.Width,
color = Species, tooltip = tooltip, onclick = clickjs) ) +
geom_point_interactive()

ggiraph(code = {print(gg_point)})
```

---

ggiraphOutput *Create a ggiraph output element*

Description

Render a ggiraph within an application page.

Usage

```
ggiraphOutput(outputId, width = "100%", height = "500px")
```

Arguments

outputId	output variable to read the ggiraph from.
width	widget width
height	widget height

Examples

```
## Not run:
if( require(shiny) && interactive() ){
  app_dir <- file.path( system.file(package = "ggiraph"), "shiny/cars" )
  shinyAppDir(appDir = app_dir )
 }
if( require(shiny) && interactive() ){
  app_dir <- file.path( system.file(package = "ggiraph"), "shiny/crimes" )
  shinyAppDir(appDir = app_dir )
}

## End(Not run)
```

`interactive_points_grob` *Generate interactive grob points*

Description

This function can be used to generate interactive grob points.

Usage

```
interactive_points_grob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
  tooltip = NULL, onclick = NULL, data_id = NULL, pch = 1,
  size = unit(1, "char"), default.units = "native", name = NULL,
  gp = gpar(), vp = NULL)
```

Arguments

x	numeric vector or unit object specifying x-values.
y	numeric vector or unit object specifying y-values.
tooltip	tooltip associated with points
onclick	javascript action to execute when point is clicked
data_id	identifiers to associate with points
pch	numeric or character vector indicating what sort of plotting symbol to use. See points for the interpretation of these values, and note fill below.
size	unit object specifying the size of the plotting symbols.
default.units	string indicating the default units to use if x or y are only given as numeric vectors.
name	character identifier.
gp	an R object of class gpar, typically the output from a call to the function gpar. This is basically a list of graphical parameter settings; note that fill (and not bg as in package **graphics** points) is used to "fill", i.e., color the background of symbols with pch = 21:25.
vp	A Grid viewport object (or NULL).

interactive_polygon_grob

Generate interactive grob polygons

Description

This function can be used to generate interactive grob polygons.

Usage

```
interactive_polygon_grob(x = unit(c(0, 1), "npc"), y = unit(c(0, 1), "npc"),
  id = NULL, id.lengths = NULL, tooltip = NULL, onclick = NULL,
  data_id = NULL, default.units = "npc", name = NULL, gp = gpar(),
  vp = NULL)
```

Arguments

x	A numeric vector or unit object specifying x-locations.
y	A numeric vector or unit object specifying y-locations.
id	A numeric vector used to separate locations in x and y into multiple polygons. All locations with the same id belong to the same polygon.
id.lengths	A numeric vector used to separate locations in x and y into multiple polygons. Specifies consecutive blocks of locations which make up separate polygons.
tooltip	tooltip associated with polygons
onclick	javascript action to execute when polygon is clicked
data_id	identifiers to associate with polygons
default.units	A string indicating the default units to use if x, y, width, or height are only given as numeric vectors.
name	A character identifier.
gp	An object of class gpar, typically the output from a call to the function gpar. This is basically a list of graphical parameter settings.
vp	A Grid viewport object (or NULL).

interactive_polyline_grob

Generate an Interactive Grob Path

Description

This function can be used to generate an interactive grob path.

Usage

```
interactive_polyline_grob(x = unit(c(0, 1), "npc"), y = unit(c(0, 1),
  "npc"), id = NULL, id.lengths = NULL, tooltip = NULL, onclick = NULL,
  data_id = NULL, default.units = "npc", arrow = NULL, name = NULL,
  gp = gpar(), vp = NULL)
```

Arguments

x	A numeric vector or unit object specifying x-values.
y	A numeric vector or unit object specifying y-values.
id	A numeric vector used to separate locations in x and y into multiple lines. All locations with the same id belong to the same line.
id.lengths	A numeric vector used to separate locations in x and y into multiple lines. Specifies consecutive blocks of locations which make up separate lines.
tooltip	tooltip associated with polylines
onclick	javascript action to execute when polyline is clicked
data_id	identifiers to associate with polylines
default.units	A string indicating the default units to use if x or y are only given as numeric vectors.
arrow	A list describing arrow heads to place at either end of the line, as produced by the arrow function.
name	A character identifier.
gp	An object of class gpar, typically the output from a call to the function gpar. This is basically a list of graphical parameter settings.
vp	A Grid viewport object (or NULL).

---

interactive_rect_grob *Generate interactive grob rectangles*

Description

This function can be used to generate interactive grob rectangles.

Usage

```
interactive_rect_grob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
  width = unit(1, "npc"), height = unit(1, "npc"), tooltip = NULL,
  onclick = NULL, data_id = NULL, just = "centre", hjust = NULL,
  vjust = NULL, default.units = "npc", name = NULL, gp = gpar(),
  vp = NULL)
```

Arguments

x	A numeric vector or unit object specifying x-location.
y	A numeric vector or unit object specifying y-location.
width	A numeric vector or unit object specifying width.
height	A numeric vector or unit object specifying height.
tooltip	tooltip associated with rectangles
onclick	javascript action to execute when rectangle is clicked
data_id	identifiers to associate with rectangles
just	The justification of the rectangle relative to its (x, y) location. If there are two values, the first value specifies horizontal justification and the second value specifies vertical justification. Possible string values are: "left", "right", "centre", "center", "bottom", and "top". For numeric values, 0 means left alignment and 1 means right alignment.
hjust	A numeric vector specifying horizontal justification. If specified, overrides the just setting.
vjust	A numeric vector specifying vertical justification. If specified, overrides the just setting.
default.units	A string indicating the default units to use if x, y, width, or height are only given as numeric vectors.
name	A character identifier.
gp	An object of class gpar, typically the output from a call to the function gpar. This is basically a list of graphical parameter settings.
vp	A Grid viewport object (or NULL).

---

interactive_segments_grob

Generate interactive grob segments

Description

This function can be used to generate interactive grob segments.

Usage

```
interactive_segments_grob(x0 = unit(0, "npc"), y0 = unit(0, "npc"),
  x1 = unit(1, "npc"), y1 = unit(1, "npc"), tooltip = NULL,
  onclick = NULL, data_id = NULL, default.units = "npc", arrow = NULL,
  name = NULL, gp = gpar(), vp = NULL)
```

Arguments

x0	Numeric indicating the starting x-values of the line segments.
y0	Numeric indicating the starting y-values of the line segments.
x1	Numeric indicating the stopping x-values of the line segments.
y1	Numeric indicating the stopping y-values of the line segments.
tooltip	tooltip associated with segments
onclick	javascript action to execute when segment is clicked
data_id	identifiers to associate with segments
default.units	A string.
arrow	A list describing arrow heads to place at either end of the line segments, as produced by the arrow function.
name	A character identifier.
gp	An object of class gpar.
vp	A Grid viewport object (or NULL).

---

interactive_text_grob *Generate interactive grob text*

Description

This function can be used to generate interactive grob text.

Usage

```
interactive_text_grob(label, x = unit(0.5, "npc"), y = unit(0.5, "npc"),
  tooltip = NULL, onclick = NULL, data_id = NULL, just = "centre",
  hjust = NULL, vjust = NULL, rot = 0, check.overlap = FALSE,
  default.units = "npc", name = NULL, gp = gpar(), vp = NULL)
```

Arguments

label	A character or expression vector. Other objects are coerced by as.graphicsAnnot.
x	A numeric vector or unit object specifying x-values.
y	A numeric vector or unit object specifying y-values.
tooltip	tooltip associated with rectangles
onclick	javascript action to execute when rectangle is clicked
data_id	identifiers to associate with rectangles
just	The justification of the text relative to its (x, y) location. If there are two values, the first value specifies horizontal justification and the second value specifies vertical justification. Possible string values are: "left", "right", "centre", "center", "bottom", and "top". For numeric values, 0 means left alignment and 1 means right alignment.

hjust	A numeric vector specifying horizontal justification. If specified, overrides the just setting.
vjust	A numeric vector specifying vertical justification. If specified, overrides the just setting.
rot	The angle to rotate the text.
check.overlap	A logical value to indicate whether to check for and omit overlapping text.
default.units	A string indicating the default units to use if x or y are only given as numeric vectors.
name	A character identifier.
gp	An object of class gpar, typically the output from a call to the function gpar. This is basically a list of graphical parameter settings.
vp	A Grid viewport object (or NULL).

---

renderggiraph *Reactive version of ggiraph object*

Description

Makes a reactive version of a ggiraph object for use in Shiny.

Usage

```
renderggiraph(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

expr	An expression that returns a ggiraph object.
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression

Examples

```
## Not run:
if( require(shiny) && interactive() ){
  app_dir <- file.path( system.file(package = "ggiraph"), "shiny" )
  shinyAppDir(appDir = app_dir )
}

## End(Not run)
```

Index