Package ‘easySVG’

January 9, 2018

Type Package

Title An Easy SVG Basic Elements Generator

Version 0.1.0

Description This SVG elements generator can easily generate
SVG elements such as rect, line, circle, ellipse, polygon,
polyline, text and group. Also, it can combine and
output SVG elements into a SVG file.

Depends R (>= 3.3.0)

URL https://github.com/ytdai/easySVG

BugReports https://github.com/ytdai/easySVG/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Yuting Dai [aut, cre]

Maintainer Yuting Dai <forlyrna@sjtu.edu.cn>

Repository CRAN

Date/Publication 2018-01-09 11:02:08 UTC

R topics documented:

circle.svg .......................................................... 2
defs.svg .......................................................... 3
easySVG ............................................................ 3
ellipse.svg .......................................................... 4
gtext.svg ............................................................ 5
circle.svg

Generate circle SVG element

Description

This function can generate a circle form SVG element

Usage

circle.svg(cx = NULL, cy = NULL, r = NULL, fill, fill.opacity, stroke, stroke.width, stroke.opacity, stroke_dasharray, style_sheet = NULL)

Arguments

- **cx**: a number, x coordinate information
- **cy**: a number, y coordinate information
- **r**: a number, radius of the circle
- **fill**: a character, color of the circle, eg. "#000000"(default), "red"
- **fill.opacity**: a number, stroke opacity of the circle, default: 1. If the fill opacity is 0, the circle's internal color is invisible
- **stroke**: a character, color of the circle line, eg. "#000000"(default), "red"
- **stroke.width**: a number, stroke width of the circle line, default: 1
- **stroke.opacity**: a number, stroke opacity of the circle line, default: 1. If the stroke opacity is 0, the line is invisible
- **stroke_dasharray**: a vector, plot the dotted circle line, eg. c(9, 5)
- **style_sheet**: a vector or a character, other style of the circle, eg. "stroke-linecap: round"

Details

The `<circle>` SVG element is an SVG basic shape, used to create circles based on a center point and a radius.

Value

the character type of SVG element
Examples

```r
circle.svg(cx = 10, cy = 20, r = 10, fill = "blue")
circle.svg(cx = 10, cy = 20, r = 10, fill = "blue", stroke.width = 2)
```

defs.svg

**Description**

make SVG `defs` element

**Usage**

defs.svg(defs.content = NULL)

**Arguments**

defs.content  a character or a list, group content

**Value**

the character type of SVG element

**Examples**

defs.svg(defs.content = "<text x="10" y="20"> an SVG element </text>")
defs.content <- list(svg1 = "<text x="10" y="30"> an SVG element </text>",
                     svg2 = "<text x="10" y="40"> an SVG element </text>"
)
defs.svg(defs.content = defs.content)

easySVG

**Description**

easySVG package can generate SVG elements easily

**Author(s)**

Yuting Dai forlynna@sjtu.edu.cn
See Also

Useful links:

https://github.com/ytdai/easySVG

Report bugs at https://github.com/ytdai/easySVG/issues

Scalable Vector Graphics (SVG) is an XML-based vector image format for two-dimensional graphics with support for interactivity and animation. The SVG specification is an open standard developed by the World Wide Web Consortium (W3C) since 1999.

Examples

```r
line <- line.svg(x1 = 50, y1 = 20, x2 = 150, y2 = 20)
rect <- rect.svg(x = 50, y = 60, width = 100, height = 10, fill = "blue")
circle <- circle.svg(cx = 80, cy = 100, r = 10, fill = "blue")
ellipse <- ellipse.svg(cx = 100, cy = 120, rx = 20, ry = 5, fill = "blue")

points <- matrix(c(50, 100, 120, 140, 135, 145), nrow = 3, ncol = 2)
polygon <- polygon.svg(points = points, fill = "green", stroke = "none")
polyline <- polyline.svg(points = points)
text <- get.text.svg(x = 10, y = 20, text.content = "This is a text element", font.size = 6)

group.content <- list(line, rect,
circle, ellipse,
polygon, polyline,
text)
group <- group.svg(id = "group_1", group.content = group.content)

## Not run:
svg.name <- paste0(tempfile(), ".svg")
pack.svg(pack.content = group, output.svg.name = svg.name)

## End(Not run)
```

**ellipse.svg**

Generate ellipse SVG element

Description

This function can generate a ellipse form of SVG element. The ellipse element is an SVG basic shape, used to create ellipses based on a center coordinate, and both their x and y radius.

Usage

```r
ellipse.svg(cx = NULL, cy = NULL, rx = NULL, ry = NULL, fill,
fill.opacity, stroke, stroke.width, stroke.opacity, stroke.dasharray,
style.sheet = NULL)
```
get.text.svg

Arguments

- **cx**: a number, x coordinate information
- **cy**: a number, y coordinate information
- **rx**: a number, x radius of the ellipse
- **ry**: a number, y radius of the ellipse
- **fill**: a character, color of the ellipse, eg. "#000000"(default), "red"
- **fill.opacity**: a number, stroke opacity of the ellipse, default: 1. If the fill opacity is 0, the ellipse’s internal color is invisible
- **stroke**: a character, color of the ellipse line, eg. "#000000"(default), "red"
- **stroke.width**: a number, stroke width of the ellipse line, default: 1
- **stroke.opacity**: a number, stroke opacity of the ellipse line, default: 1. If the stroke opacity is 0, the line is invisible
- **stroke.dasharray**: a vector, plot the dotted ellipse line, eg. c(9, 5)
- **style.sheet**: a vector or a character, other style of the ellipse, eg. "stroke-linecap: round"

Value

the character type of SVG element

Examples

```r
ellipse.svg(cx = 10, cy = 20, rx = 10, ry = 5, fill = "blue")
ellipse.svg(cx = 10, cy = 20, rx = 10, ry = 5, fill = "blue", stroke.width = 2)
```

Description

This function can generate a text form SVG element. The SVG `<text>` element defines a graphics element consisting of text. It's possible to apply a gradient, pattern, clipping path, mask, or filter to `<text>`, just like any other SVG graphics element.

Usage

```r
get.text.svg(x = NULL, y = NULL, text.content = "", fill, stroke, stroke.width, font.family, font.size, font.weight, font.style, text.decoration, word.spacing, letter.spacing, text.anchor, rotate, text.path, style.sheet = NULL)
```
Arguments

- `x` a number, x coordinate information
- `y` a number, y coordinate information
- `text.content` a character, text content
- `fill` a character, color of the text, eg. "#000000" (default), "red"
- `stroke` a character, color of the rect text, eg. "#000000" (default), "red"
- `stroke.width` a number, stroke width of the rect text, default: 1
- `font.family` a character, font family of text, eg. "Arial"
- `font.size` a number, font size of text, default: 8
- `font.weight` a character, font weight of text, eg. "normal" (default), "bold"
- `font.style` a character, font style of text, eg. "normal" (default), "italic"
- `text.decoration` a character, text decoration, eg. "none" (default), "underline", "overline", "line-through"
- `word.spacing` a number or character, default: "normal"
- `letter.spacing` a number or character, default: "normal"
- `text.anchor` a character, eg. "start" (default), "middle", "end"
- `rotate` a number, rotation angle of text
- `text.path` a character, fit text path
- `style.sheet` a vector or a character, other style of the text, eg. "stroke-linecap: round"

Value

the character type of SVG element

Examples

```javascript
get.text.svg(x = 10, y = 20, text.content = "Hello Word", fill = "blue")
get.text.svg(x = 10, y = 20, text.content = "Hello Word", fill = "blue",
rotate = 90, font.family = "Helvetica")
```

Description

The `<g>` SVG element is a container used to group other SVG elements. Transformations applied to the `<g>` element are performed on all of its child elements, and any of its attributes are inherited by its child elements. It can also group multiple elements to be referenced later with the `<use>` element.
Usage

```r
group.svg(id = NULL, group.content = NULL, fill, fill.opacity, stroke,
stroke.width, stroke.opacity, stroke.dasharray, font.family, font.size,
font.weight, font.style, text-decoration, word.spacing, letter.spacing,
text.anchor, scale, rotate, translate, skewX, skewY, style.sheet = NULL,
transform.sheet = NULL)
```

Arguments

- **id**: a character, group id
- **group.content**: a character or a list or a vector, group content
- **fill**: a character, color of the group, eg. "#000000"(default), "red"
- **fill.opacity**: a number, stroke opacity of the group, default:1. If the fill opacity is 0, the rect’s internal color is invisible
- **stroke**: a character, color of the group line, eg. "#000000"(default), "red"
- **stroke.width**: a number, stroke width of the group line, default: 1
- **stroke.opacity**: a number, stroke opacity of the group line, default:1. If the stroke opacity is 0, the line is invisible
- **stroke.dasharray**: a vector, plot the dotted group line, eg. c(9, 5)
- **font.family**: a character, font family of text, eg. "Arial"
- **font.size**: a number, font size of text, default: 8
- **font.weight**: a character, font weight of text, eg. "normal"(default), "bold"
- **font.style**: a character, font style of text, eg. "normal"(default), "italic"
- **text.decoration**: a character, text decoration, eg. "none"(default), "underline", "overline", "line-through"
- **word.spacing**: a number or character, default: "normal"
- **letter.spacing**: a number or character, default: "normal"
- **text.anchor**: a character, eg. "start"(default), "middle", "end"
- **scale**: a number, transform scale of the object
- **rotate**: a vector, rotation of the object
- **translate**: a vector, translate of the object
- **skewX**: a number
- **skewY**: a number
- **style.sheet**: a vector or a character, other style of the group, eg. "stroke-linecap: round"
- **transform.sheet**: a vector or a character, other transform of the group

Value

the character type of SVG element
Examples

```r
lim.axis.svg(id = "group_1", group.content = "this is a svg element")
group.content <- list(svg1 = "this is a svg element",
                     svg2 = "this is a svg element")
group.svg(id = "group_1", group.content = group.content)
group.svg(id = "group_1", group.content = group.content,
         style.sheet = c("stroke:red", "stroke-width:1"),
         transform.sheet = c("translate(100, 100)"))
```

---

**Generate SVG element of axis**

**Description**

This function will generate a axis form SVG element.

**Usage**

```r
lim.axis.svg(x = NULL, stroke = "#000000", stroke.width = 1,
             line.length = 100, axis.font.size = 8, digit = 2, span = 5,
             id = NULL, unit = NULL)
```

**Arguments**

- `x` a vector, the range of your number
- `stroke` a number, the line stroke of the axis
- `stroke.width` a number, the line stroke of the axis
- `line.length` a number, the line length of the axis
- `axis.font.size` a number, the axis font size of axis
- `digit` a number, the significant digits number of axis
- `span` a number, distance between number and axis line
- `id` a character, the id name of this axis
- `unit` the unit of this axis

**Value**

the character type of SVG element
Examples

```r
lim.axis.1 <- lim.axis.svg(x = c(100, 900), id = "test")
pack_info_1 <- pack.svg(pack.content = lim.axis.1)
# You can write it in a svg file
# message(pack_info_1)

lim.axis.2 <- lim.axis.svg(x = c(3.3, 4, 5), id = "test", unit = 4000, axis.font.size = 4)
pack_info_2 <- pack.svg(pack.content = lim.axis.2)
# You can write it in a SVG file
# message(pack_info_2)
```

Description

This function will generate a line form SVG element. The `<line>` element is an SVG basic shape used to create a line connecting two points.

Usage

```r
line.svg(x1 = NULL, y1 = NULL, x2 = NULL, y2 = NULL, stroke, stroke.width, stroke.opacity, stroke.dasharray, style.sheet = NULL)
```

Arguments

- `x1` a number, x1 coordinate information
- `y1` a number, y1 coordinate information
- `x2` a number, x2 coordinate information
- `y2` a number, y2 coordinate information
- `stroke` a character, color of the line, eg. "#000000"(default), "red"
- `stroke.width` a number, stroke width of the line, default: 1
- `stroke.opacity` a number, stroke opacity of the line, default: 1. If the stroke opacity is 0, the line is invisible
- `stroke.dasharray` a vector, plot the dotted line, eg. c(9, 5)
- `style.sheet` a vector or a character, other style of the line, eg. "stroke-linecap: round"

Value

the character type of SVG element
**Examples**

```r
dline <- line.svg(x1 = 1, y1 = 2, x2 = 10, y2 = 20)
dline <- line.svg(x1 = 1, y1 = 2, x2 = 10, y2 = 20, stroke = "#00FF00")
dline <- line.svg(x1 = 1, y1 = 2, x2 = 10, y2 = 20, stroke.dasharray = c(9, 5))
```

---

**Description**

`pack svg`

**Usage**

```r
pack.svg(width = 1200, height = 800, output.svg.name = NULL,
          pack.content = pack.content)
```

**Arguments**

- `width` a number, width of the plot
- `height` a number, height of the plot
- `output.svg.name` a character, the output svg file name
- `pack.content` a character or a list, group content

**Value**

the characher type of svg element

**Examples**

```r
pack.svg(pack.content = "<text x="10" y="20"> this is a svg element </text>")
pack.content <- list(svg1 = "<text x="10" y="20"> this is a svg element </text>",
                    svg2 = "<text x="10" y="40"> this is a svg element </text>")
pack_info <- pack.svg(pack.content = pack.content)
message(pack_info)
```
Generate polygon SVG element

Description

This function can generate a polygon form SVG element The <polygon> element defines a closed shape consisting of a set of connected straight line segments. The last point is connected to the first point. For open shapes see the <polyline> element.

Usage

```r
generate_polygon <- function(points = NULL, fill, fill.opacity, stroke, stroke.width, stroke.opacity, fill.rule, style.sheet = NULL) {
  # your implementation
}
```

Arguments

- **points**: a matrix, a series of coordinates
- **fill**: a character, color of the polygon, eg. "#000000" (default), "red"
- **fill.opacity**: a number, stroke opacity of the polygon, default: 1. If the fill opacity is 0, the polygon’s internal color is invisible
- **stroke**: a character, color of the polygon line, eg. "#000000" (default), "red"
- **stroke.width**: a number, stroke width of the polygon line, default: 1
- **stroke.opacity**: a number, stroke opacity of the polygon line, default: 1. If the stroke opacity is 0, the polygon line is invisible
- **fill.rule**: a character, fill rule of polygon, eg. "nonzero", "evenodd"
- **style.sheet**: a vector or a character, other style of the polygon, eg. "stroke-linecap: round"

Value

the character type of SVG element

Examples

```r
points <- matrix(c(1,2,3, 11,12,13), nrow = 3, ncol = 2)
generate_polygon(points = points)
generate_polygon(points = points, fill = "red", stroke = "yellow", fill.rule = "evenodd")
```
Description

This function can generate a polyline form SVG element. The `<polyline>` SVG element is an SVG basic shape that creates straight lines connecting several points. Typically a polyline is used to create open shapes as the last point doesn’t have to be connected to the first point. For closed shapes see the `<polygon>` element.

Usage

```r
polyline.svg(points = NULL, fill, stroke, stroke.width, stroke.opacity, style.sheet = NULL)
```

Arguments

- **points**: a matrix, a series of coordinates
- **fill**: a character, color of the polyline, eg. "#000000" (default), "red"
- **stroke**: a character, color of the polyline line, eg. "#000000" (default), "red"
- **stroke.width**: a number, stroke width of the polyline line, default: 1
- **stroke.opacity**: a number, stroke opacity of the polyline line, default: 1. If the stroke opacity is 0, the polygon line is invisible
- **style.sheet**: a vector or a character, other style of the polyline, eg. "stroke-linecap: round"

Value

the character type of SVG element

Examples

```r
points <- matrix(c(1, 2, 3, 11, 12, 13), nrow = 3, ncol = 2)
polyline.svg(points = points)
polyline.svg(points = points, stroke = "yellow")
```
Description

This function can generate a rect form SVG element. The `<rect>` element is a basic SVG shape that creates rectangles, defined by their corner’s position, their width, and their height. The rectangles may have their corners rounded.

Usage

```r
rect.svg(x = NULL, y = NULL, width = NULL, height = NULL, rx = NULL,
         ry = NULL, fill, fill.opacity, stroke, stroke.width, stroke.opacity,
         stroke.dasharray, style.sheet = NULL)
```

Arguments

- `x` a number, x coordinate information
- `y` a number, y coordinate information
- `width` a number, width of the rect
- `height` a number, height of the rect
- `rx` a number, x coordinate of rounded rectangle
- `ry` a number, y coordinate of rounded rectangle
- `fill` a character, color of the rect, eg. "#000000" (default), "red"
- `fill.opacity` a number, stroke opacity of the rect, default: 1. If the fill opacity is 0, the rect’s internal color is invisible
- `stroke` a character, color of the rect line, eg. "#000000" (default), "red"
- `stroke.width` a number, stroke width of the rect line, default: 1
- `stroke.opacity` a number, stroke opacity of the rect line, default: 1. If the stroke opacity is 0, the line is invisible
- `stroke.dasharray` a vector, plot the dotted rect line, eg. c(9, 5)
- `style.sheet` a vector or a character, other style of the rect, eg. "stroke-linecap: round"

Value

the character type of SVG element

Examples

```r
rect.svg(x = 1, y = 2, width = 10, height = 20, fill = "blue")
rect.svg(x = 1, y = 2, width = 10, height = 20, stroke.dasharray = c(9, 5))
rect.svg(x = 1, y = 2, width = 10, height = 20, rx = 2, ry = 4, fill = "blue")
```
Description

The `<use>` element takes nodes from within the SVG document, and duplicates them somewhere else.

Usage

```use.svg(id = NULL, x = NULL, y = NULL, scale, rotate, translate, skewX, skewY, style.sheet = NULL, transform.sheet = NULL)```

Arguments

- `id` a character, target of the link
- `x` a number, x transform coordinate
- `y` a number, y transform coordinate
- `scale` a number, transform scale of the object
- `rotate` a vector, rotation of the object
- `translate` a vector, translate of the object
- `skewX` a number
- `skewY` a number
- `style.sheet` a vector or a character, other style of the link, eg. "stroke-linecap: round"
- `transform.sheet` a vector or a character, other transform of the link

Value

the character type of svg element

Examples

```use.svg(id = "target", x = 100, y = 200)
use.svg(id = "target", x = 100, y = 200, rotate = c(90, 100, 200))```
Index

circle.svg, 2
defs.svg, 3
easySVG, 3
easySVG-package (easySVG), 3
ellipse.svg, 4
get.text.svg, 5
group.svg, 6
lim.axis.svg, 8
line.svg, 9
pack.svg, 10
polygon.svg, 11
polyline.svg, 12
rect.svg, 13
use.svg, 14