Package: svgparser (via r-universe)

May 8, 2026

Type Package

Title Read SVG as Grobs and Data Frames

Version 0.1.2

Author mikefc

Maintainer mikefc <mikefc@coolbutuseless.com>

Description Read SVG to grid graphics objects (i.e. grobs), and data.frames.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

URL https://github.com/coolbutuseless/svgparser, https://coolbutuseless.github.io/package/svgparser

Imports xml2, stringi, cssparser (>= 0.1.1)

Suggests rmarkdown, knitr, testthat (>= 3.0.0), vdiffr, ggplot2, gggrid, minisvg, magick, openssl

Remotes coolbutuseless/cssparser, coolbutuseless/minisvg, pmur002/gggrid

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev libxml2-dev

Repository https://trevorld.r-universe.dev

Date/Publication 2021-12-25 00:04:53 UTC

RemoteUrl https://github.com/coolbutuseless/svgparser

RemoteRef HEAD

RemoteSha eed57af52189344d69aded270b4378c2254a4b92

Contents

apply_transform . . . 2
arc_to_df . . . 3
bezier3_to_df . . . 3
create_rect_df . . . 4
css_calc_all_styles . . . 4
css_value_as_numeric . . . 5
data_frame . . . 5
ellipse_to_df . . . 6
get_element_by_id . . . 6
gpar_to_df . . . 7
lex . . . 7
load_supertinyicon . . . 8
parse_command_to_matrix . . . 9
parse_svg_group . . . 9
parse_svg_path_d . . . 10
parse_svg_radialGradient . . . 10
parse_transform_string_to_matrices . . . 11
parse_transform_string_to_matrix . . . 12
path_list_expand_beziers . . . 12
path_list_to_abs_path_list . . . 13
path_list_to_df . . . 13
rbind_dfs . . . 14
read_file . . . 14
read_svg . . . 15
shotgun_match . . . 16
style_to_gpar . . . 17
style_to_viewport . . . 18
supertinyicon_names . . . 18
svg_colour_to_r_colour . . . 19
trim . . . 19
update_transform . . . 20
update_transform_with_string . . . 20

Index **21**

`apply_transform` *Apply the given transform to x,y coordds*

Description

Apply the given transform to x,y coordds

Usage

```
apply_transform(transform, coords_df)
```

Arguments

transform	matrix transform to apply
coords_df	data.frame containing x,y coords

Value

data.frame with updated coordinates

---

arc_to_df *Convert <path> arc to a data.frame of coordinates*

Description

Reference https://svgwg.org/svg2-draft/implnote.html#ArcConversionEndpointToCenter

Usage

```
arc_to_df(params, npoints)
```

Arguments

params	named list of arc parameters from the path segment i.e. xo, y0, x, y, arcflag, sweepflag, rx, ry, rot (degrees)
npoints	number of points along arc

Details

ToDo: actual rotation of arc coords by 'phi'

Value

data.frame of coodinates of arc

---

bezier3_to_df *Convert cubic and quadratic beziers to data.frame of coordinates along curve*

Description

Convert cubic and quadratic beziers to data.frame of coordinates along curve

Usage

```
bezier3_to_df(x, y, N = 5)

bezier2_to_df(x, y, N = 5)
```

Arguments

x, y	coords of control points
N	Number of output points

Value

data.frame of coords

---

create_rect_df *Create a data.frame for a rectangle with rounded corners*

Description

Create a data.frame for a rectangle with rounded corners

Usage

```
create_rect_df(x, y, width, height, rx, ry, npoints)
```

Arguments

x, y, width, height, rx, ry	rect parameters
npoints	number of points along each of the rounded corners

Value

data.frame of (x, y) coordinates

---

css_calc_all_styles *Create a named list to store computed styles indexed by the xpath of each element*

Description

Create a named list to store computed styles indexed by the xpath of each element

Usage

```
css_calc_all_styles(svg, user_css = NULL)
```

Arguments

svg	root node of svg
user_css	user supplied CSS (as a single text string)

Value

list of style lookup by xpath

`css_value_as_numeric` *Convert a string value from CSS into a numeric value*

Description

Convert a string value from CSS into a numeric value

Usage

```
css_value_as_numeric(x, percentage_as_fraction = TRUE, ...)
```

Arguments

`x`	Character string of a CSS value e.g. "12", "12px", "3em", "47%"
`percentage_as_fraction`	Should percentages like "12 as a fraction (e.g. 0.12)? Default: TRUE. If FALSE then returns the percentage as a numeric value (e.g. 12)
`...`	other arguments passed to `css_length_as_pixels()`

Value

a numeric value as best we can with limited knowledge

`data_frame` *data.frame creation*

Description

20x Faster than 'data.frame()', but with *ZERO* sanity checking and no support for row.names

Usage

```
data_frame(...)
```

Arguments

`...`	named arguments

Value

data.frame

ellipse_to_df — *Convert Ellipse parameters into a data.frame of coords*

Description

`cx`, `cy`, `rx`, `ry` are attribute parameters of the `<ellipse>` tag. `npoints` is defined by the ser.

Usage

```
ellipse_to_df(cx, cy, rx, ry, npoints)
```

Arguments

`cx`, `cy`, `rx`, `ry`	ellipse parameters
`npoints`	number of points around ellipse

Value

data.frame of x,y coordinates around the ellipse

get_element_by_id — *Get an element from an XML document by ID. Return NA if ID does not exist.*

Description

This function is called when a `<use>` tag references another element or a a colour tag references a gradient element.

Usage

```
get_element_by_id(svg, id)
```

Arguments

`svg`	xml document containing svg
`id`	id, #id, url(#id), url('#id')

Value

SVG element with the given id, otherwise NA

gpar_to_df *Convert a gpar object to a data.frame*

Description

This converts a gpar object to a 1-row data.frame representation. This is used when accumulating a data.frame representation to return to the user.

Usage

```
gpar_to_df(gp)
```

Arguments

gp	gpar object with a single specification. no vectors allowed

Details

Linear and Radial gradient objects are replaced with their first colour. Although this could be something fancier e.g. list object, or some text representation of the gradient parameters. Maybe just `deparse1()`?

Value

data.frame with single row.

lex *Break a string into labelled tokens based upon a set of patterns*

Description

Break a string into labelled tokens based upon a set of patterns

Usage

```
lex(text, regexes, verbose = FALSE)
```

Arguments

text	a single character string
regexes	a named vector of regex strings. Each string represents a regex to match a token, and the name of the string is the label for the token. Each regex can contain an explicit captured group using the standard `()` brackets. If a regex doesn't not define a captured group then the entire regex will be captured. The regexes will be processed in order such that an early match takes precedence over any later match.
verbose	print more information about the matching process. default: FALSE

Value

a named character vector with the names representing the token type with the value being the element extracted by the corresponding regular expression.

Examples

```
## Not run:
lex("hello there 123.45", regexes=c(number=re$number, word="(\\w+)", whitespace="(\\s+)"))

## End(Not run)
```

`load_supertinyicon` *Load an SVG icon from the* `supertinyicons` *icon pack*

Description

Load an SVG icon from the `supertinyicons` icon pack

Usage

```
load_supertinyicon(
  name,
  obj_type = c("grob", "data.frame", "list", "debug", "svg"),
  ...
)
```

Arguments

`name`	name of icon. e.g. `"twitter"`. If the exact name is not found, then the icon with the closest matching name is returned. See `supertinyicon_names` for a full list of supported icons in this set.
`obj_type`	What kind of R object to return - choices `c('grob', 'data.frame', 'list', 'debug', 'svg')`. Default: 'grob'. See documentation for `read_svg()` for more details.
`...`	other arguments passed to `read_svg()`

parse_command_to_matrix *Parse tokens representing a single transform instruction into a matrix*

Description

Parse tokens representing a single transform instruction into a matrix

Usage

```
parse_command_to_matrix(tokens)
```

Arguments

tokens	character vector of c(command, number, number, ...)

Value

matrix representing this transform

parse_svg_group *Parse xml representing some SVG*

Description

Parse xml representing some SVG

Usage

```
parse_svg_group(elem, state)
```

Arguments

elem	as SVG grouping element e.g. <svg> <g> <clipPath>
state	environment containing state information for this SVG

parse_svg_path_d *Parse path into a list of parameters and meta-information*

Description

Parse path into a list of parameters and meta-information

Usage

```
parse_svg_path_d(svg_path)
```

Arguments

svg_path	SVG path string (single character string)

Value

list of lists of information - 1 list per element in the path. each element is itself a list of parameters and meta-information about the path element e.g. index of the element in the path

parse_svg_radialGradient *Parse an SVG tag of given type*

Description

Parse an SVG tag of given type

Usage

```
parse_svg_radialGradient(elem, state)

parse_svg_linearGradient(elem, state)

parse_svg_switch(elem, state)

parse_svg_image(elem, state)

parse_svg_use(elem, state)

parse_svg_path(elem, state)

parse_svg_polyline(elem, state)

parse_svg_polygon(elem, state)
```

```
parse_svg_line(elem, state)

parse_svg_rect(elem, state)

parse_svg_circle(elem, state)

parse_svg_ellipse(elem, state)

parse_svg_text(elem, state)
```

Arguments

elem	SVG element which is of known type
state	environment containing state information for this SVG

Value

grob

`parse_transform_string_to_matrices` *Parse a transform string to a list of matrices - 1 per command. Mainly for debugging*

Description

Note there can be multiple transform instructions in a transform string

Usage

```
parse_transform_string_to_matrices(transform_string)
```

Arguments

transform_string	e.g. "rotate(10) translate(3, 3)"

Value

list of matrices

`parse_transform_string_to_matrix`
Parse a transform string to asingle transform matrix

Description

Note there can be multiple transform instructions in a transform string

Usage

```
parse_transform_string_to_matrix(transform_string)
```

Arguments

`transform_string`
e.g. "rotate(10) translate(3, 3)"

Value

combined transform marix

`path_list_expand_beziers`
Prepare complete list of control point coordinates for all beziers

---

Description

Prepare complete list of control point coordinates for all beziers

Usage

```
path_list_expand_beziers(path_list, state)
```

Arguments

`path_list`	path as returned by parse_svg_path_d
`state`	named list of current state. Default list(x=0, y=0)

Value

`path_list` with all bezier elements now having 'x' and 'y' vectors with coords of the control points.

path_list_to_abs_path_list *Convert path relative coordinates to absolute coordinates*

Description

Convert path relative coordinates to absolute coordinates

Usage

```
path_list_to_abs_path_list(path_list, state = list(x = 0, y = 0))
```

Arguments

path_list	path as returned by parse_svg_path_d()
state	named list of current state. Default list(x=0, y=0)

Value

path_list with only absolute elements

path_list_to_df *Convert a path list to a data.frame*

Description

Convert a path list to a data.frame

Usage

```
path_list_to_df(path_list, state = list(x = 0, y = 0))
```

Arguments

path_list	path as returned by parse_svg_path_d
state	named list of current state. Default list(x=0, y=0)

Value

data.frame of x,y coordinates

rbind_dfs *Internal function for perfroming do.call(rbind...)*

Description

This function expands all data.frames to have the same columns, whereas base R just complains a throws an error

Usage

```
rbind_dfs(dfs)
```

Arguments

dfs	list of data.frames

Value

single data.frame

read_file *Brute force file reader that is not concerned with file endings*

Description

`readLines()` in combination with `unz()` fails o read the last line of a file if it does not end in a CR/LF. I haven't found a workaround besides a manual read loop like below.

Usage

```
read_file(conn)
```

Arguments

conn	connection. For svgparser this is always an `unz()` connection to the zipped version of the icon set

Details

This function reads all character in a file regardless of line endings or file endings.

Value

single string representing file contents

read_svg *Read an svg file (or text) into a* grid::grobTree *object or* data.frame

Description

Read an svg file (or text) into a grid::grobTree object or data.frame

Usage

```
read_svg(
  svg_file,
  xoffset = 0,
  yoffset = 0,
  npoints = 30,
  scale = 1,
  default.units = "snpc",
  stroke_scale = 1,
  font_scale = 1,
  style_default = list(),
  user_css = NULL,
  obj_type = c("grob", "data.frame", "list", "debug"),
  false_colour = NULL
)
```

Arguments

svg_file	either a filename, or a single character string containing all the text of an SVG. Filenames may either be '.svg' or '.svgz' (gzip compressed SVG)
xoffset, yoffset	Extra offsets to element coordinates applied in the grob coordinate system (not the SVG coordinate system). Default: (0, 0)
npoints	number of segmentation points per section of bezier, arc, circle or ellipse. Default 30 Increase this number if the curves look too jaggy for your use case.
scale	Scale factor to apply to all coordinates. Default: 1.
default.units	the grid units to use throughout. The default ('snpc') is a pretty safe bet that will give you an auto-resizing vector object. Uou could also set it to an *absolute* unit (like 'mm') and then play with the scale argument to get a fixed size grid object.
stroke_scale	Default: 1. Multiplication factor for width of strokes. The value to use here is heavily dependent upon what size output you are rendering to.
font_scale	extra scaling applied to font parameters. Default: 1. The value to use here is heavily dependent upon the output size you are rendering to.

`style_default` a named list of CSS properties which should override the defaults. default: `list()`. E.g. set `style_default = list(fill = 'red')` to set the default fill style for all elements to 'red'. This style will still be overridden by inline styles, css styles, or presentation attributes. It is a useful way of setting the 'color' property which is often used in SVG icon sets (which make heavy use of the 'currentColor' property)

`user_css` single string containing CSS text e.g. "circle { fill: red !important; }". Note: Normal cascading style rules apply i.e. more specific rules override those with lower specificity, and inline style specifications have the highest specificty. You may need to use `!important` to override styles consistently.

`obj_type` What kind of R object to return - choices `c('grob', 'data.frame', 'list', 'debug')`. Default: 'grob'.

The 'list' and 'data.frame' options are for advanced/debugging use, but some users may find them useful if trying to extract coordinate information etc.

The `data.frame` option could be used to recreate much of the SVG but it is missing key information such as clipping paths, and gradients (as these are pretty difficult to capture nicely in a data.frame).

The `debug` option returns all the possible information. Currently this returned object is undocumented. Use at your peril!

The `list` option returns a list containing the following elements for each parsed SVG element

- svg - the SVG string for this element
- tag - the SVG tag e.g. "path", "circle"
- svg_df - data.frame of coordinates in SVG coordinate system
- transform - the transform matrix for this element
- grid_df - the transformed coordinates in R/grid coordinate space
- style - caclculated style for this element. Named list of sttyle attributes
- gp - the `gpar()` equivalent of the style
- grob - the final generated grob for this element

`false_colour` Use false colouring on all elements, by selecting random colours from palettes in `grDevices` Default: NULL means to use actual colours. Possible values: 'rainbow', 'hcl', 'heat', 'terrain', 'topo', 'cm'

Value

Return type determined by `obj_type` argument

`shotgun_match` *Pick the closest matching word from a list of words.*

Description

As long as 'word' & 'words' are not empty, this function should always return a valid member of 'words'.

Usage

```
shotgun_match(word, words)
```

Arguments

word	user word
words	list of all words

---

style_to_gpar *Convert a list of style information for an element to a* gpar *object*

Description

Convert a named list of CSS style information into a gpar object. E.g. `list(stroke = 'black')` is converted to `gpar(col = 'black')`.

Usage

```
style_to_gpar(style, state)
```

Arguments

style	named list of style properties for an element. E.g. `list(stroke = 'black')`
state	state information. See inline documentation for `read_svg` for more information.

Details

This function handles stroke and fill colours (and separate alphas for each), fonts, linear and radial gradients, line parameters (width, join and end types etc).

Not yet done:

- Conversion of line type information from CSS to gpar

Value

gpar object

style_to_viewport *Convert a list of style information to a viewport object*

Description

Currently this only supports setting the clipping path on the viewport.

Usage

```
style_to_viewport(style, transform, state)
```

Arguments

style	named list of style properties
transform	transform matrix of element being clipped
state	state information

Details

Masks would also be set here, but I don't have any good examples to debug with, so leaving this for now.

Value

viewport object

supertinyicon_names *Names of icons in* supertinyicons *set*

Description

Names of icons in supertinyicons set

Usage

```
supertinyicon_names
```

Format

An object of class character of length 297.

svg_colour_to_r_colour *Convert an SVG colour to an R colour*

Description

Handles: False colouring, hex colours, oct colours, radial gradients.

Usage

```
svg_colour_to_r_colour(col, state)
```

Arguments

col	svg colour
state	environment containing state information for this SVG

Details

More complex CSS colour specifications like `rgb(10, 20, 30, 0.5)` are handled by `cssparser::css_colour_to_hex()`

Value

valid R colour

trim *Trim the end off a word.*

Description

Trim the end off a word.

Usage

```
trim(x, len)
```

Arguments

x	string
len	number of characters to trim

update_transform *Update a transform matrix with an element*

Description

Update a transform matrix with an element

Usage

```
update_transform(transform, elem)
```

Arguments

transform	matrix
elem	svg element

Value

new transform matrix

update_transform_with_string *Update a transform matrix with the new SVG transform string*

Description

Update a transform matrix with the new SVG transform string

Usage

```
update_transform_with_string(transform, transform_string)
```

Arguments

transform	matrix
transform_string	the transform string which is a presentation attribute on an element

Value

new transform matrix

Index

∗ **datasets**
    supertinyicon_names, 18

apply_transform, 2
arc_to_df, 3

bezier2_to_df *(*bezier3_to_df*)*, 3
bezier3_to_df, 3

create_rect_df, 4
css_calc_all_styles, 4
css_value_as_numeric, 5

data_frame, 5

ellipse_to_df, 6

get_element_by_id, 6
gpar_to_df, 7

lex, 7
load_supertinyicon, 8

parse_command_to_matrix, 9
parse_svg_circle
        *(*parse_svg_radialGradient*)*, 10
parse_svg_ellipse
        *(*parse_svg_radialGradient*)*, 10
parse_svg_group, 9
parse_svg_image
        *(*parse_svg_radialGradient*)*, 10
parse_svg_line
        *(*parse_svg_radialGradient*)*, 10
parse_svg_linearGradient
        *(*parse_svg_radialGradient*)*, 10
parse_svg_path
        *(*parse_svg_radialGradient*)*, 10
parse_svg_path_d, 10
parse_svg_polygon
        *(*parse_svg_radialGradient*)*, 10
parse_svg_polyline
        *(*parse_svg_radialGradient*)*, 10
parse_svg_radialGradient, 10
parse_svg_rect
        *(*parse_svg_radialGradient*)*, 10
parse_svg_switch
        *(*parse_svg_radialGradient*)*, 10
parse_svg_text
        *(*parse_svg_radialGradient*)*, 10
parse_svg_use
        *(*parse_svg_radialGradient*)*, 10
parse_transform_string_to_matrices, 11
parse_transform_string_to_matrix, 12
path_list_expand_beziers, 12
path_list_to_abs_path_list, 13
path_list_to_df, 13

rbind_dfs, 14
read_file, 14
read_svg, 15

shotgun_match, 16
style_to_gpar, 17
style_to_viewport, 18
supertinyicon_names, 18
svg_colour_to_r_colour, 19

trim, 19

update_transform, 20
update_transform_with_string, 20