

Package ‘RSVGTipsDevice’

January 2, 2012

Version 1.0-4

Date 10/15/2011

Title An R SVG graphics device with dynamic tips and hyperlinks

Author Tony Plate <tplate@acm.org>, based on RSvgDevice by T Jake Luciani <jakeluciani@yahoo.com>

Maintainer Tony Plate <tplate@acm.org>

Depends R (>= 2.6)

Description A graphics device for R that uses the w3.org xml standard for Scalable Vector Graphics. This version supports tooltips with 1 to 3 lines, hyperlinks, and line styles.

License GPL (>= 2)

Repository CRAN

Date/Publication 2011-11-27 09:14:08

R topics documented:

devSVGTips	2
encodeSVGSpecialChars	4
getSVGToolTipMode	5
RSVGTipsDevice	6
RSVGTipsDevice.design	11
setSVGShapeContents	13
setSVGShapeToolTip	14
setSVGShapeURL	15
SVG.viewing	16

Index	17
--------------	-----------

Description

This function starts up an SVG graphics device that will write to a file. The SVG shapes and text have optional tooltips and/or hyperlinks.

Usage

```
devSVGTips(file = "Rplots.svg", width = 7, height = 7,
           bg = "white", fg = "black", onefile=TRUE,
           xmlHeader=FALSE, useStyleAttributes=FALSE,
           toolTipMode = 1, toolTipFontSize = 10,
           toolTipOpacity = 1.0, title = "R SVG Plot", sub.special=TRUE)
```

Arguments

<code>file</code>	the file where output will appear
<code>width</code>	The width of the plot in inches
<code>height</code>	the height of the plot in inches
<code>bg</code>	the background color for the plot
<code>fg</code>	the foreground color for the plot
<code>onefile</code>	merge plot calls into onefile or separate them to separate pages
<code>xmlHeader</code>	Print XML header or not (it is recommended to not print a header)
<code>useStyleAttributes</code>	Specify shape attributes in a <code>style</code> attribute or as plain attributes (plain attributes is recommended)
<code>toolTipMode</code>	Mode of tool tips: 0 (no tool tips), 1, or 2 (number of lines of tool tip)
<code>toolTipFontSize</code>	Size of font in tool tips (in points/pixels)
<code>toolTipOpacity</code>	Opacity of tool tips: between 0 and 1. A value of 0.9 allows the image behind the tooltip to be seen, but can render slowly on some viewers
<code>title</code>	Title of plot
<code>sub.special</code>	Unless <code>sub.special=FALSE</code> , special SVG characters (ampersand, less-than, etc.) in <code>title</code> will be substituted by the appropriate XML encoding, except for an ampersand followed by lower-case letters and then a semi-colon (if these substitutions are not made, it is easy to inadvertently create unusable SVG files)

Details

This graphics do not appear in any window while they are being drawn – SVG commands are written to a file and can be viewed in an SVG viewer after the plot is completed.

After the plot is completed, the function `dev.off` must be called to close the graphics device and flush all unwritten SVG commands to the file.

Text drawn on the graphic by commands like `text()` is subject to XML special characters being replaced by the corresponding XML encoding, except for an ampersand followed by lower-case letters and then a semi-colon. This behavior is NOT controlled by `sub.special` and there is currently no way to turn it off.

This device is not implemented as one that can do clipping. Thus, R commands like `text` will not draw the text if part of the text might be outside the plotting region. This can be the cause of legend (or other functions that call `text`) omitting text. The solution is to make sure that the text is drawn inside the plotting region, e.g., using commands like `legend(..., inset=0.01)`. If text is not appearing, use `par(xpd=NA)`, which will do the least clipping of text. The example plot `svgplot13.svg` in [RSVGTipsDevice](#) shows clipping that happens for various settings of `par("xpd")`.

The clipping behavior can result in labels unexpectedly not appearing in lattice graphics. This can sometime be solved by making the region in which labels are drawn larger, e.g., by doing something like `xyplot(strip.par.text=list(lines=2), ...)`.

See Also

[SVG viewing](#) Ways to view SVG files. [Design and future](#) of the RSVGTips device. [Overview](#) of the RSVGTips device (many more examples). [setSVGShapeToolTip](#), [setSVGShapeURL](#), [getSVGToolTipMode](#), [pictex](#), [postscript](#), [Devices](#).

Examples

```
## Not run:
library("RSVGTipsDevice")
devSVGTips("svgplot1.svg", toolTipMode=1, title="SVG example plot 1: shapes and points, tooltips are title + 1 line")
plot(c(0,10),c(0,10), type="n", xlab="x", ylab="y", main="Example SVG plot with title+ 1 line tips (mode=1)")
setSVGShapeToolTip(title="A rectangle", desc="that is yellow")
rect(1,1,4,6, col='yellow')
setSVGShapeToolTip(title="1st circle with title only")
points(5.5,7.5,cex=20,pch=19,col='red')
setSVGShapeToolTip(title="A triangle", desc="big and green")
polygon(c(3,6,8), c(3,6,3), col='green')
# no tooltips on these points
points(2:8, 8:2, cex=3, pch=19, col='black')
# tooltips on each these points
invisible(sapply(1:7, function(x) {setSVGShapeToolTip(title=paste("point", x)); points(x+1, 8-x, cex=3, pch=1, col='black')
setSVGShapeToolTip(title="Text", desc="can have a tool tip too!")
text(x=4, y=9, lab="Poke me!", col="blue")
dev.off()

# Not run in tests because uses the SemiPar package for the fuel.frame data
# A plot of fuel mileage vs weight
library("RSVGTipsDevice")
library("SemiPar")
```

```

data(fuel.frame)
fuel.frame <- cbind(fuel.frame,
  US=is.element(substring(fuel.frame$car.name, 1, 5),
    c("Buick", "Chevr", "Chrys", "Dodge", "Eagle",
      "Ford ", "Mercu", "Oldsm", "Plymo", "Ponti")))
devSVGTips("mlgvswt1.svg", height=5, width=7, toolTipMode=1, title="Mileage vs Weight for autos, tooltips are tit
plot(fuel.frame$Weight, fuel.frame$Mileage, type="n", xlab="Weight",
ylab="Miles per gallon", main="US cars in blue, imports in yellow")
for (i in seq(len=nrow(fuel.frame))) {
  setSVGShapeToolTip(title=fuel.frame[i,"car.name"],
    desc=paste(fuel.frame[i, "Type"], ", disp=", fuel.frame[i,"Disp."]))
  points(fuel.frame[i,"Weight"], fuel.frame[i,"Mileage"], pch=19,
    cex=2, col=if (fuel.frame[i,"US"]) "blue" else "yellow")
}
dev.off()

## End(Not run)

```

encodeSVGSpecialChars *Encode SVG special chars (ampersand, etc)*

Description

Encode SVG special chars (ampersand, etc) as the appropriate XML encoding.

Usage

```
encodeSVGSpecialChars(x, sub.special = TRUE, xent = FALSE)
```

Arguments

x	A vector of character data to be encode.
sub.special	If FALSE, do not encode.
xent	If FALSE, encode as XML encoding, if TRUE, encode as XML entity

Details

The following substitutions are made:

Character	XML encoding	XML entity
&	\&	\&\#38
\'	\'	\&\#30
\"	\"	\&\#34
\<	\<	\&\#60
\>	\>	\&\#62

Value

The vector *x* with encodings as appropriate.

Author(s)

Tony Plate <tplate@acm.org>

`getSVGToolTipMode` *Return the tool tip mode of the RSVGTips device*

Description

Return the tool tip mode of the RSVGTips device.

Usage

```
getSVGToolTipMode()
```

Details

Returns the mode that was set in the call to [devSVGTips](#) (this mode cannot be changed while a plot is being created.)

Value

An integer representing the tooltip mode of the current device:

- -1: the current device is not an RSVGTips device
- 0: the tooltips are not being used
- 1: tool tips are title + one line of description
- 2: tool tips are title + two lines of description

Author(s)

Tony Plate <tplate@acm.org>

See Also

[devSVGTips](#)

Description

This package contains a SVG (Scalable Vector Graphics) device that will write to a file. The SVG shapes have optional tooltips and/or hyperlinks. The resulting SVG files should conform the W3C Scalable Vector Graphics (SVG) standard (with the possible exception of `toolTipMode=2`) and do display correctly in the SVG viewer in Mozilla Firefox under Windows (XP) and Linux (Ubuntu).

Details

This graphics device supplies the underlying functions that are called by `plot`, `lines`, etc, and the following functions that are called by the user to access the special capabilities of the SVG device:

- `devSVGTips` Open an SVG device
- `setSVGShapeToolTip` Set a tool tip to be used for the next graphics shape or text drawn
- `setSVGShapeURL` Set a URL to be used in the hyperlink for the next graphics shape or text drawn
- `getSVGToolTipMode` Get the tool tip mode used by the current graphics device

Author(s)

This driver is a modification by Tony Plate <tplate@acm.org> of the `RSvgDevice` driver written by T Jake Luciani <jakeluciani@yahoo.com>

See Also

[Design and future possible features](#) of the RSVGTips device.
[pictex](#), [postscript](#), [Devices](#).

Examples

```
library("RSVGTipsDevice")
sessionInfo()
devSVGTips("svgplot1.svg", toolTipMode=1,
  title="SVG example plot 1: shapes and points, tooltips are title + 1 line")
plot(c(0,10),c(0,10), type="n", xlab="x", ylab="y",
  main="Example SVG plot with title + 1 line tips (mode=1)")
setSVGShapeToolTip(title="A rectangle", desc="that is yellow")
rect(1,1,4,6, col='yellow')
setSVGShapeToolTip(title="1st circle with title only")
points(5.5,7.5,cex=20,pch=19,col='red')
setSVGShapeToolTip(title="A triangle", desc="big and green")
polygon(c(3,6,8), c(3,6,3), col='green')
# no tooltips on these points
points(2:8, 8:2, cex=3, pch=19, col='black')
# tooltips on each these points
```

```

invisible(sapply(1:7, function(x)
{setSVGShapeToolTip(title=paste("point", x))
  points(x+1, 8-x, cex=3, pch=1, col='black')}}))
setSVGShapeToolTip(title="Text", desc="can have a tool tip too!")
text(x=4, y=9, lab="Poke me!", col="blue")
dev.off()

devSVGTips("svgplot2.svg", toolTipMode=2,
  title="SVG example plot 2: shapes and points, tooltips are title + 2 lines")
getSVGToolTipMode()
setSVGShapeToolTip(title="First circle title only")
plot(1:3, cex=10, main="Example SVG plot with 2 line tips (mode=2)")
setSVGShapeToolTip(title="A rectangle", desc="with a 1 line tip")
rect(1,1,2,2)
setSVGShapeToolTip(title="second circle", desc1="first line of description", desc2="second line of description")
points(1.5,2.5,cex=20,pch=19,col='red')
dev.off()

devSVGTips("svgplot3.svg", toolTipMode=2,
  title="SVG example plot 3: shapes, tooltips are title + 2 lines")
plot(c(0,10),c(0,10), type="n", xlab="x", ylab="y",
  main="Example SVG plot, tooltips are title + 2 lines (mode=2)")
setSVGShapeToolTip(title="A rectangle", desc="that is yellow")
rect(1,1,4,6, col='yellow')
setSVGShapeToolTip(title="1st circle", desc1="1st line of description",
  desc2="2nd line of description")
points(5.5,7.5,cex=20,pch=19,col='red')
setSVGShapeToolTip(title="A triangle", desc1="green", desc2="big")
polygon(c(3,6,8), c(3,6,3), col='green')
# no tooltips on these points
points(2:8, 8:2, cex=3, pch=19, col='black')
# tooltips on each of these points
invisible(sapply(1:7, function(x)
{setSVGShapeToolTip(title=paste("point", x))
  points(x+1, 8-x, cex=3, pch=1, col='black')}}))
dev.off()

devSVGTips("svgplot4.svg", title="SVG example plot 4: points and no tooltips")
plot(1:11,(-5:5)^2, type='b', main="Simple Example Plot with no tooltips")
dev.off()

devSVGTips("svgplot5.svg",
  title="SVG example plot 5: points and a rectangle, no tooltips")
plot(1:3)
rect(1,1,2,2)
dev.off()

devSVGTips("svgplot6.svg",
  title="SVG example plot 6: supply XML code for tips (title + 1 line)")
setSVGShapeContents("<title>first circle</title>")
plot(1:3, cex=10)
setSVGShapeContents("<title>hah!</title>")
rect(1,1,2,2)

```

```

setSVGShapeContents("<title>second circle</title><desc>description</desc>")
points(1.5,2.5,cex=20,pch=19,col='red')
dev.off()

devSVGTips("svgplot7.svg", toolTipMode=2,
  title="SVG example plot 7: supply XML code for tips (title + 2 lines)")
setSVGShapeContents("<title>first circle</title>")
plot(1:3, cex=10)
setSVGShapeContents("<title>hah!</title>")
rect(1,1,2,2)
setSVGShapeContents("<title>second circle</title><desc1>first line of description</desc1><desc2>second line of de
points(1.5,2.5,cex=20,pch=19,col='red')
dev.off()

devSVGTips("svgplot8.svg", toolTipMode=1,
  title="SVG example plot 8: tooltips + hyperlink")
plot(c(0,10),c(0,10), type="n", xlab="x", ylab="y",
  main="Example SVG plot with title + 1 line tips (mode=1) + hyperlink")
setSVGShapeToolTip(title="A rectangle", desc="that is yellow")
rect(1,1,4,6, col='yellow')
setSVGShapeToolTip(title="1st circle with title only")
points(5.5,7.5,cex=20,pch=19,col='red')
setSVGShapeToolTip(title="A triangle", desc="big and green")
polygon(c(3,6,8), c(3,6,3), col='green')
# no tooltips on these points
points(2:8, 8:2, cex=3, pch=19, col='black')
# tooltips on each these points
invisible(sapply(1:7, function(x)
{setSVGShapeToolTip(title=paste("point", x))
  points(x+1, 8-x, cex=3, pch=1, col='black')}}))
# Hyperlink to www.r-project.org
setSVGShapeToolTip(title="www.r-project.org", desc="click to visit!")
setSVGShapeURL("http://www.r-project.org")
rect(8,6,10,7, col='blue')
# Hyperlink to www.r-project.org that opens in a new browser window or tab
setSVGShapeToolTip(title="www.r-project.org", desc="click to visit in a new page!")
setSVGShapeURL("http://www.r-project.org", "_blank")
rect(8,8,10,9, col='green')
dev.off()

devSVGTips("svgplot9.svg", toolTipMode=1,
  title="SVG example plot 9: line and point types")
plot(c(0,20),c(0,5), type="n", xlab="x", ylab="y",
  main="Example SVG plot with different line and point types")
for (i in 0:16) {
  lines(i+(0:4), (1:5), col=max(i,1), pch=i, lty=i, type="b")
  text(i, 0.5, lab=as.character(i), cex=2^(abs((i-8)/4)-1))
}
dev.off()

# This example checks encoding of characters that are special in XML: # <&'\">
# To avoid tripping up people who have already used the XML entity
# (e.g., "&amp"), an ampersand followed by lower-case letters and a

```

```

# semicolon is NOT encoded.
devSVGTips("svgplot10.svg", toolTipMode=1,
  title="SVG example plot 10: shapes and points, tooltips are title + 1 line, special chars")
plot(c(0,10),c(0,10), type="n", xlab="x", ylab="y",
  main="Example SVG plot with title + 1 line tips <mode=1>")
setSVGShapeToolTip(title="A rectangle", desc="that is <yellow> &amp; yellow")
rect(1,1,4,6, col='yellow')
setSVGShapeToolTip(title="1st circle with title only")
points(5.5,7.5,cex=20,pch=19,col='red')
setSVGShapeToolTip(title="A triangle", desc="big & green")
polygon(c(3,6,8), c(3,6,3), col='green')
text(lab="Special chars: <&'\">", 6, 9, adj=0, cex=1.3)
text(lab="Already encoded: &lt;&amp;&gt;", 6, 9.5, adj=0, cex=1.3)
# no tooltips on these points
points(2:8, 8:2, cex=3, pch=19, col='black')
# tooltips on each these points
invisible(sapply(1:7, function(x)
{setSVGShapeToolTip(title=paste("<point ", x, ">", sep="")}
  points(x+1, 8-x, cex=3, pch=1, col='black'))}))
dev.off()

# This example checks that the fontwidth information is accurate
# by drawing a box closely around letters.
# Compare appearance to other devices if you want to make things
# more accurate. Note that some descenders and ascenders will go
# outside of the bounds. This info is in src/devSVG.c:charwidth.
devSVGTips("svgplot11.svg", toolTipMode=0, title="SVG example plot 11: text width with cex=1")
charbysize0 <- " 'ijlIJ,./\\|:;f!\"()[]rt-*?FLTYcksxyzv'_0123456789EKPRXabdeghnopqu{ }\\177$ wABCSDVGHNOQUZ#&+<=>M
allchars <- " !\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNQRSTUUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{ }
charbysize <- paste(c(charbysize0, setdiff(strsplit(allchars, NULL)[[1]], strsplit(charbysize0, NULL)[[1]])), col
textinbox <- function(x, y, i) {
  cc <- substring(charbysize, i-31, i-31)
  # cc <- rawToChar(as.raw(i))
  lab <- paste(rep(cc,10), collapse="")
  w <- strwidth(lab)
  if (cc==" ")
    lab <- paste(lab, "<", sep="")
  text(x, y, lab=lab, adj=c(0))
  rect(x, y-strheight(lab)/2, x+w, y+strheight(lab)/2)
}
par(mar=c(0,0,0,0))
plot(y=c(0,25),x=c(0,4), type="n", axes=FALSE, xlab="", ylab="")
text(2,25,"Character metric info: box is drawn around strings using strwidth & strheight")
for (i in 32:127) textinbox(x=0.1+(i-32)%/24, y=24-(i-32)%/24, i)
# print(data.frame(letters, strwidth(letters)))
dev.off()

# This example checks that the character alignment information is
# accurate.
devSVGTips("svgplot12.svg", toolTipMode=0, title="SVG example plot 12: character alignment", height=6, width=6)
par(mar=c(0,0,0,0))
plot(y=c(-5,5),x=c(-10,10), type="n", axes=FALSE, xlab="", ylab="")
text(0,4.5,"Character alignment using adj= or pos= in text()")

```

```

text(0,4,"Text is drawn at red '+' using args shown")
text(0,3.5,lab="XXX")
points(0,3.5,pch=3,col='red')
for (xadj in (0:2)/2) {
  text(-3+6*(1-xadj),3,paste("X adj=", xadj, " X", sep=""),adj=xadj)
  points(-3+6*(1-xadj),3,pch=3,col='red')
}
for (pos in 1:4) {
  text(c(0,-1,0,1)[pos], 1.5+0.5*c(-1,0,1,0)[pos], paste("X pos=", pos, " X", sep=""), pos=pos)
  points(c(0,-1,0,1)[pos], 1.5+0.5*c(-1,0,1,0)[pos], pch=3, col='red')
}
for (xadj in (0:2)/2)
  for (yadj in (0:2)/2) {
    text(6*(0.5-xadj),-1.5+(1-yadj),paste("X adj=", xadj, ", ",yadj, " X", sep=""),adj=c(xadj,yadj))
    points(6*(0.5-xadj),-1.5+(1-yadj),pch=3,col='red')
  }
dev.off()

```

```

# This example shows what happens with text clipping.
# Too much clipping occurs! I think is is not calling
# the text method for the device whenever the text might
# be a little out-of-bounds (because of the setting
# dd->canClip=FALSE in src/devSVG.c). It seems to be too
# aggressive about this, e.g., when par(xpd=NA) (xpd=NA
# means clip to device region, xpd=TRUE means clip to
# figure region, and xpd=FALSE means clip to plot region).
devSVGtips("svgplot13.svg", tooltipMode=0, title="SVG example plot 13: character clipping", height=6, width=6)
par(xpd=FALSE)
par(mfrow=c(2,2))
par(mar=c(5, 4, 4, 2) + 0.1)
str <- paste(letters[c(1:6,21:26)],collapse="")
plot(y=c(-10,10),x=c(-10,10), type="n", xlab="X axis", ylab="Y axis",
      main=paste("par(xpd=", par("xpd"), ")", sep=""))
text(0,2,paste("clip to", switch(as.character(par("xpd")), "NA"="device", "TRUE"="figure", "FALSE"="plot"), "region"),
     -9,9,str,srt=45)
text(9,0,str,srt=0)
text(9,9,str,srt=-45)
text(0,-9,str,srt=-90)
text(9,-9,str,srt=-135)
text(-9,0,str,srt=180)
text(-9,-9,str,srt=135)
text(0,9,str,srt=90)
par(xpd=TRUE)
plot(y=c(-10,10),x=c(-10,10), type="n", xlab="X axis", ylab="Y axis",
      main=paste("par(xpd=", par("xpd"), ")", sep=""))
text(0,2,paste("clip to", switch(as.character(par("xpd")), "NA"="device", "TRUE"="figure", "FALSE"="plot"), "region"),
     -9,9,str,srt=45)
text(9,0,str,srt=0)
text(9,9,str,srt=-45)
text(0,-9,str,srt=-90)
text(9,-9,str,srt=-135)
text(-9,0,str,srt=180)
text(-9,-9,str,srt=135)

```

```

text(0,9,str,srt=90)
par(xpd=NA)
plot(y=c(-10,10),x=c(-10,10), type="n", xlab="X axis", ylab="Y axis",
      main=paste("par(xpd=", par("xpd"), ") ", sep=""))
text(0,2,paste("clip to", switch(as.character(par("xpd")), "NA"="device", "TRUE"="figure", "FALSE"="plot"), "region"),
text(-9,9,str,srt=45)
text(9,0,str,srt=0)
text(9,9,str,srt=-45)
text(0,-9,str,srt=-90)
text(9,-9,str,srt=-135)
text(-9,0,str,srt=180)
text(-9,-9,str,srt=135)
text(0,9,str,srt=90)
par(xpd=FALSE)
dev.off()

```

RSVGTipsDevice.design *Design and future of the SVG Graphics Driver with dynamic tips*

Description

The design of this driver is that it can be used with ordinary R plotting commands. The extra information needed for tooltips and hyperlinks is supplied by a call to a separate function call before the call to the plotting function that generates the shape. Extra information for only one shape can be supplied this way, so shapes that have tooltips or hyperlinks must be drawn one at a time.

SVG coding style

It appears that preferred SVG coding styles have changed over time. The defaults in this driver are those suggested at <http://jwatt.org/svg/authoring/>.

In particular:

- The driver output by default does NOT include a DOCTYPE DTD (it is recommended at <http://jwatt.org/svg/authoring/> to NOT include a DOCTYPE declaration.)
- Use shape properties directly rather than specifying them via style property, e.g., prefer this: `<circle fill="red" stroke="blue" ... />` over this: `<circle style="fill:red; stroke:blue;" ... />`.
- Use "Namespace" aware functions, e.g., `getElementByTagNameNS` instead of `getElementByTagName` (see also "SVG: Namespace Crash Course" in the references).

Limitations

- This driver currently does not have any font metric information, so the use of `plotmath` is not supported.
- This device will not record tool tips and hyperlinks if it is called by `dev.copy` - the functions to add tool tips and hyperlinks are ignored by other devices.

Future development possibilities

- Clean up the SVG code that defines the layout of a tool tip (`SVG_footer()` in `src/devSVG.c`) - currently the vertical spacing (the 'height' and 'y' attributes) is somewhat ad-hoc (coded by trial and error, and may not work for all font sizes).
- Add font metric information to support the use of `plotmath`.
 - Note that SVG uses Unicode characters encoded like `α`. The symbols for Unicode are described here: <http://www.unicode.org/charts/symbols.html>.
 - SVG allows for definitions of fonts - maybe the right approach is to include a symbol font in the doc when `plotmath` is used.
 - <http://www.w3.org/TR/SVG11/fonts.html>
 - <http://www.w3.org/TR/xsl/#font-model>
- Clean up the two-line tooltip SVG code: The two-line tooltip mode displays OK in the default SVG renderers in Firefox (under Windows 2000/XP and Linux) but does not display in the Batik standalone SVG viewer. It probably needs declarations to extend XML elements: <http://www.w3.org/TR/SVG/extend.html> 23.5 Adding private elements and attributes to the DTD.
- Provide post-processing to use CSS (cascading style sheets) to make SVG files with lots of elements more compact.
- The design of how tooltip and URL information is passed to the device is somewhat ugly - it would be nicer if the basic graphics commands allowed additional optional arguments. However, they don't, and the scheme implemented here at least works. It has the advantage that the graphics commands that specify tooltips and URLs will be ignored when other devices are used. One disadvantage is graphics recording will not record tooltips and URLs.

References

www Consortium W3C Scalable Vector Graphics (SVG) <http://www.w3.org/Graphics/SVG/Overview.htm#8>

SVG authoring guidelines <http://jwatt.org/svg/authoring>

SVG: Namespaces Crash Course [http://developer.mozilla.org/en/docs/SVG_Namespace_Tips_\(external\)](http://developer.mozilla.org/en/docs/SVG_Namespace_Tips_(external))

"SVG Essentials", J. David Eisenberg, O'Reilly and Associates, 2002.

See Also

[Overview](#) of the RSVGTips device.

[pictex](#), [postscript](#), [Devices](#).

Cairo is another device for R that can produce SVG graphics; see <http://www.rforge.net/Cairo>. However, it doesn't appear to support tooltips.

setSVGShapeContents	<i>Set raw shape contents (XML text) for the next graphics shape drawn with the RSVGTips device.</i>
---------------------	--

Description

Set raw shape contents (XML text) for the next graphics shape drawn with the RSVGTips device. This function provides lower-level access than [setSVGShapeToolTip](#).

Usage

```
setSVGShapeContents(contents)
```

Arguments

contents A character vector containing XML text.

Details

This function sets text that will be included in the SVG commands output for the next graphics shape drawn with [RSVGTipsDevice](#).

Note that contents is used as-is, with no substitutions made for special characters. Consequently, it is possible to produce an invalid XML file by using special characters in contents (i.e., "<", ">", "&", and single- and double-quotes.) Such special characters should be replaced by their XML equivalent entity – see [setSVGShapeToolTip](#) for a list of these.

Value

Returns an invisible NULL.

Author(s)

Tony Plate <tplate@acm.org>

See Also

[RSVGTipsDevice](#)

setSVGShapeToolTip *Set the tool tip for the next graphics shape drawn with the RSVGTips device.*

Description

Set the tool tip for the next graphics shape drawn with the RSVGTips device.

Usage

```
setSVGShapeToolTip(title = NULL, desc = NULL, desc1 = desc, desc2 =
NULL, sub.special=TRUE)
```

Arguments

title	A title for the tool tip box. (optional)
desc	The text of the tool tip.
desc1	The first line of text of the tool tip.
desc2	The second line of text of the tool tip (used when the tool tip mode is 2.)
sub.special	If FALSE, do NOT replace special characters by their XML encodings (see details).

Details

This function sets tool tip text that will be used by the next graphics shape drawn with [RSVGTipsDevice](#).

The RSVGTips device "consumes" the tooltip with the first shape that it draws, e.g., using `rect`, `points`, `polygon`, etc. If multiple objects are drawn (e.g., by supplying a vector to `points`), only the first shape drawn will have a tooltip. Note that `lines` does NOT use a tooltip.

Unless `sub.special=FALSE`, the following special characters in `title`, `desc`, etc. will be substituted by the XML entity as shown:

```
&  \&amp;
\ ' \&apos;
\" \&quot;
\< \&lt;
\> \&gt;
```

The exception to this is that an ampersand followed by lower-case letters and then a semi-colon is always left alone, because it already looks like an XML entity.

Value

Returns an invisible NULL.

Author(s)

Tony Plate <tplate@acm.org>

See Also

Examples here: [RSVGTipsDevice](#)

Examples

```
## Not run: setSVGShapeToolTip(title="A rectangle", desc="that is yellow")
```

setSVGShapeURL	<i>Set raw shape URL (XML text) for the next graphics shape drawn with the RSVGTips device.</i>
----------------	---

Description

Set a URL to use as a hyperlink for the next graphics shape drawn with the RSVGTips device.

Usage

```
setSVGShapeURL(url, target=NULL, sub.special=TRUE)
```

Arguments

url	A full valid URL, e.g., "http://www.r-project.org"
target	The name of a target in the page - use "_blank" to make the link open a new browser tab or window
sub.special	If TRUE, substitute XML encodings for special characters in url and target. If FALSE, it is possible to produce an invalid XML file by supplying a URL with special characters (i.e., "<", ">", "&", and single- and double-quotes.)

Details

This function sets a hyperlink that will be included in the SVG commands output for the next graphics shape drawn with [RSVGTipsDevice](#).

Value

Returns an invisible NULL.

Author(s)

Tony Plate <tplate@acm.org>

See Also

[RSVGTipsDevice](#)

Examples

```
## Not run: setSVGShapeURL("http://www.r-project.org")
```

SVG.viewing

Ways to view SVG files

Description

The easiest way to view SVG files is in a web browser that has native SVG support. Mozilla Firefox, Opera, Safari, Google Chrome, and Microsoft Internet Explorer (version 9 onwards) all have native SVG browser support.

The section "Necessary Viewer/Browser for the examples" at the bottom of <http://www.carto.net/papers/svg/samples/> has information about SVG viewing software for various OS's.

Opera notes

The Opera viewer has automatic tooltips, which get data from the "title" element. When viewing SVG files created by RSVGTipsDevice, an extra tooltip will pop up after a short wait. To enable or disable tooltip popups in Opera, go to Tools > Preferences > Advanced > Browsing and toggle the "Show tooltips" option. (see <http://www.opera.com/support/search/view/714/>)

Index

*Topic **device**

- devSVGTips, [2](#)
- encodeSVGSpecialChars, [4](#)
- getSVGToolTipMode, [5](#)
- RSVGTipsDevice, [6](#)
- RSVGTipsDevice.design, [11](#)
- setSVGShapeContents, [13](#)
- setSVGShapeToolTip, [14](#)
- setSVGShapeURL, [15](#)
- SVG.viewing, [16](#)

*Topic **package**

- RSVGTipsDevice, [6](#)
- SVG.viewing, [16](#)

Design and future, [3](#)

Design and future possible features, [6](#)

dev.off, [3](#)

Devices, [3](#), [6](#), [12](#)

devSVGTips, [2](#), [5](#), [6](#)

encodeSVGSpecialChars, [4](#)

getSVGToolTipMode, [3](#), [5](#), [6](#)

lines, [6](#)

Overview, [3](#), [12](#)

pictex, [3](#), [6](#), [12](#)

plot, [6](#)

plotmath, [11](#), [12](#)

postscript, [3](#), [6](#), [12](#)

RSVGTipsDevice, [3](#), [6](#), [13–15](#)

RSVGTipsDevice.design, [11](#)

setSVGShapeContents, [13](#)

setSVGShapeToolTip, [3](#), [6](#), [13](#), [14](#)

setSVGShapeURL, [3](#), [6](#), [15](#)

SVG viewing, [3](#)

SVG.viewing, [16](#)