SVG Cheat Sheet Cheat Sheet

by beccam via cheatography.com/33170/cs/10320/

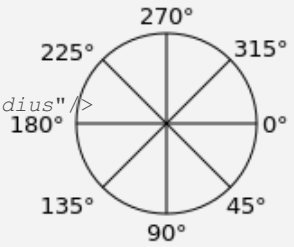
Basic Shapes

line	`<line x1="start-x" y1="start-y" x2="end-x" y2="end-y"/>`
rectangle	`<rect x="left-x" y="top-y" width="width" height="height"/>`
circle	`<circle cx="center-x" cy="center-y" r="radius"/>`
ellipse	`<ellipse cx="center-x" cy="center-y" rx="x-radius" ry="y-radius"/>`
polygon	`<polygon points="points-list"/>`
polyline	`<polyline points="points-list"/>`

Transformations

`translate(x, y)`	moves *x* horizontally, *y* vertically
`scale(xFactor, yFactor)`	multiplies by *xFactor* and *yFactor*
`scale(factor)`	equivalent to `scale(factor, factor)`
`rotate(angle, centerX, centerY)`	rotates by *angle* degrees with center of rotation `(centerX, centerY)`
`rotate(angle)`	equivalent to `rotate(angle, 0, 0)`
`skewX(angle)`	skews *x*-coordinates by *angle* degrees
`skewY(angle)`	skews *y*-coordinates by *angle* degrees
`matrix(a b c d e f)`	specifies a transformation matrix of six values

Angle Measurements



Angle measurements increase clockwise, starting from the positive *x*-axis.

Grouping and Referencing Objects

grouping	`<g id="id" style="attributes">` `</g>`
use a group	`<use xlink:href="#id" x="x1" y="y1"/>`
defining groups without displaying	`<defs>` `</defs>`
symbol	`<symbol id="id" style="attributes" preserve` `</symbol>`

Gradients

attributes	
`spreadMethod`	`pad`
	`repeat`
	`reflect`
`gradientTransform`	`skewX`
	`skewY`
	`rotate`
`<linearGradient>` attributes	
`x1 y1 x2 y2`	`="0-100%"`
`<radialGradient>` attributes	
`cx cy r fx fy`	`="0-100%"`
elements	
`<stop>`	`offset="0-100%"`
	`stop-color:`
	`stop-opacity: 0.0-1.0`

Clipping and masking

clipping

`<clipPath>`	`id`, `clipPa thUnits`
`<use xlink:href="#imageid" style="clip-path: url(#pathid);" />`	

masking

`<mask>`	`id`, `x`, `y`, `width`, `height`
`clipPathUnits` `maskUnits` `maskContentUnits`	`objectBoundingBox`, `userSpaceOnUse`
`style`	`mask: url(#maskid)`
	`fill-opacity: 0.0-1.0`
	`fill: color` ; white specified for opacity only

Filters

`<filter>`	`x`, `y`, `width`, `height`
`filterUnits` `primitiveUnits`	`objectBoundingBox`, `userSpaceOnUse`
`<feGaussianBlur>`	can create a drop shadow
`in`	`SourceAlpha`, `SourceGraphic`
`stdDeviation`	*blur* or *x-blur y-blur*

Stroke Attributes

`style="attribute:value"`	specify stroke attributes in `style`
`stroke`	stroke color; default is `none`
`stroke-width`	width of stroke; default is `one`
`stroke-opacity`	a value between `0.0` (transparent) and `1.0` (opaque, the default)
`stroke-dasharray`	a list of the lengths of dashes and gaps; default is `none`

Stroke Attributes (cont)

`stroke-linecap`	specifies shape of endpoints: `butt` (default), `round`, or `square`
`stroke-linejoin`	specifies shape of corners: `miter` (pointed, the default), `round`, or `bevel` (flat)
`stroke-miterlimit`	maximum ratio of length of the miter point to width of the lines; default is `4`

Fill Attributes

`style="attribute:value"`	specify fill attributes in `style`
`fill`	fill color; default is `black`
`fill-opacity`	a value between `0.0` (transparent) and `1.0` (opaque, the default)
`fill-rule`	determines whether a point is inside a shape; `nonzero` (default) or `evenodd`

Paths

`<path d="command arguments"/>`

uppercase commands: absolute coordinates

lowercase commands: relative coordinates

Command	Arguments	Effect
`M m`	*x y*	move to (*x*, *y*)
`L l`	*x y*	line to (*x*, *y*)
`Z`		close path
`H h`	*x*	horizontal line to *x*
`V v`	*y*	vertical line to *y*

Paths (cont)

A a *rx ry x-axis-rotation large-arc sweep x y*	elliptical arc to (*x*, *y*); points lie on ellipse with *x*-radius *rx*, *y*-radius *ry*, rotated *x-axis-rotation* degrees; if arc < 180°, *large-arc* is 0; if arc direction is positive, *sweep* is 1
Q q *x1 y1 x y*	quadratic Bézier curve to (*x*, *y*) using control point (*x1*, *y1*)
T t *x y*	quadratic Bézier curve to (*x*, *y*) using reflection of previous Q's control pt
C c *x1 y1 x2 y2 x y*	cubic Bézier curve to (*x*, *y*) using control pt 1 (*x1*, *y1*) and control pt 2 (*x2*, *y2*)
S s *x2 y2 x y*	cubic Bézier curve to (*x*, *y*) using reflection of previous C's control pt for control pt 1 and (*x2*, *y2*) for control pt 2

Text

<text x="*x*" y="*y*">	"d" baseline (*x*, *y*)
,desplayed</text>	
font-family	serif, sans-serif, monos
font-size	pt, em, ex, %
font-weight	bold, normal
font-style	italic, normal
text-decoration	none, underline, overlin
word-spacing	*+length*, normal, *-length*
letter-spacing	*+length*, normal, *-length*
text-anchor	start, middle, end
xxxLength	*value*
lengthAdjust	spacing (def), spacingAndGl
writing-mode	tb
glyph-orientation-vertical	0 (letter-spacing:-#), 90 (d
direction	rtl, ltr
unicode-bidi	bidi-override
<text>	
<textPath xlink:href="#*path-id*">text</textPath>	
</text>	
startOffset=""	*val*, *val%*
<tspan style="*attributes*">spanned text</tspan>	
dx="*x*" or dy="*y*"	offset chars by *x* or *y*
x="*x*" or y="*y*"	place chars at *x* or *y*
rotate="*angle*"	rotate chars by *angle*
baseline-shift	super, sub, em, %
xml:space=""	default, preserve