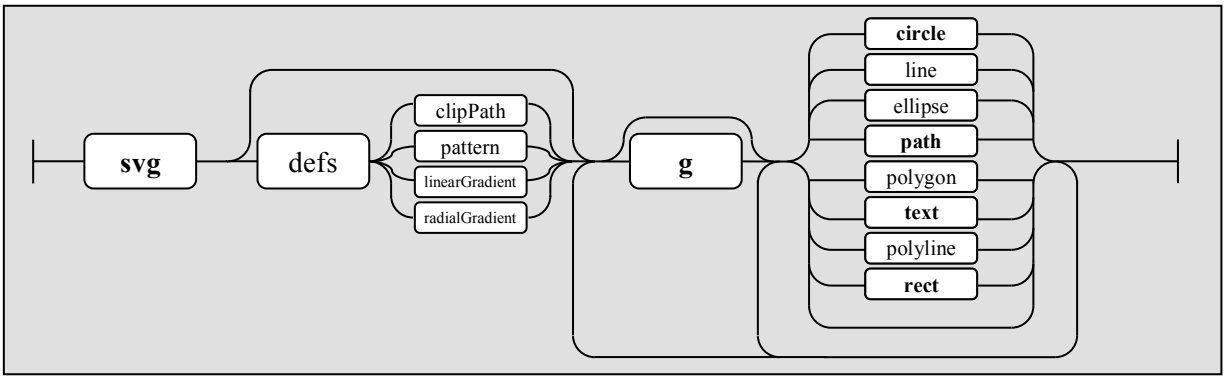
SVG



Positioning :

		svg	g	rect	text	line	path	polygon	polyline	circle	ellipse	style	attr
transform	"translate(x,y)"		●	●	●	●	●	●	●	●	●		●
x,y	number			●	●								●
x1,x2,y1,y2	number					●							●
d	(special)						●						●
points	"x,y x,y x,y"							●	●				●
cx,cy	number									●	●		●

Sizing:

		svg	g	rect	text	line	path	polygon	polyline	circle	ellipse	style	attr
transform	"scale(k)"		●	●	●	●	●	●	●	●	●		●
width, height	number	●		●									●
r	number									●			●
rx,ry	number										●		●

Colors :

		svg	g	rect	text	line	path	polygon	polyline	circle	ellipse	style	attr
fill	*name*,#RRGGBB, rgb(R,G,B); url(#title)			●	●		●	●		●	●	●	
stroke	*name*,#RRGGBB, rgb(R,G,B)			●	●	●	●	●	●	●	●	●	
opacity	number 0-1			●	●	●	●	●	●	●	●	●	
fill-opacity	number 0-1			●	●		●	●		●	●	●	
stroke-opacity	number 0-1			●	●	●	●	●	●	●	●	●	

For text, fill refers to the color of the letters, stroke to that of the outline of the letters. If a pattern or a gradient is defined, it can be used with fill with its id.

Lines :

		svg	g	rect	text	line	path	polygon	polyline	circle	ellipse	style	attr
stroke-width	number			●	●	●	●	●	●	●	●	●	
stroke-dasharray	numbers (separated by commas, blanks)*			●	●	●	●	●	●	●	●	●	
stroke-linecap	butt|round|square					●	●	●	●			●	
stroke-linejoin	miter|round|bevel						●	●				●	
stroke-miterlimit	number					●	●	●	●			●	

the numbers represent dash lengths and gaps.

Text :

			style	attr
text-anchor	start|middle|end	Determine if the text is aligned to the left, centered or aligned to the right.		●
writing-mode	tb	If specified, text is written vertically	●	
glyph-orientation-vertical	0|1	If 0, and writing-mode is set to tb, letters are horizontal (stacked), else if set to 1 when text mode is set to tb, the text will be rotated (written from top to bottom).	●	
textLength	Number	If specified, the text will be adjusted to fit the specified length.		●
lengthAdjust	spacing | spacingAndGlyphs	Which strategy is used to adjust the length?		●
textPath	xlink:href(#id)	Writes the text along a path (which should be in the defs and given an id attribute)		

Special :

		svg	g	rect	text	line	path	polygon	polyline	circle	ellipse	style	attr
transform	"rotate(a)"		●	●	●	●	●	●	●	●	●		●
clip-path	url(#title)		●	●	●	●	●	●	●	●	●	●	
mask	url(#title)		●	●	●	●	●	●	●	●	●	●	

d3 cheat sheet

Select > data > enter > append :

`select`	`CSS selector (string)`	Selects the *first* element that matches the selector.
`selectAll`	`CSS selector (string)`	Selects *all* the elements that match the selector.
		Selections can be chained : each new statement will look *inside* the current selection.
`data`	`array of items`	Provides the data that will be matched against the selection
`enter`	`nothing - ()`	Selects elements to be added: the items in the data which are not yet matched by elements.
`exit`	`Nothing - ()`	Selects elements to be removed: the elements which are no longer matched by items in the data
`append`	`name of element`	creates specified element inside the selection, either once (if data/enter is not used) or as many times as there are elements in enter().
`remove`	`Nothing - ()`	removes selected elements.

The following constructs are valid.

`d3.select("body").append("svg");`	Creates an svg element in the HTML document (assuming there is a body element)
`var myRects=d3.select("body").select("svg").selectAll("rect");`	Looks for all rect elements within the first svg element found within the body element.
`myRects=myRects.data([1,2,3]).enter();`	If there are fewer than 3 rect elements in myRects, this prepares the missing elements to be added.
`myRects.append("rect");`	Create these missing elements. Extra methods can be added to initiate them (attr, style…) will only apply to the elements just created
`d3.selectAll("rect");`	This selects all rect elements. Methods added there will apply to all of the rect elements, not only to those just created.
`d3.selectAll("rect").data([4,5]).exit().remove();`	This passes new data to the rect elements. Then, if there are more than 2 of them, the rest – exit() – are deleted – remove().
`d3.selectAll("rect").remove();`	This deletes all rect elements.

The following are not:

`d3.append("svg");`	All elements must be added within a container element. *First* select, *then* append.
`d3.selectAll("rect").enter().append("rect");` `d3.data([1,2,3]).enter().append("rect");`	enter() requires *first* a selection, *then* data.
`d3.select("rect").data([1,2,3]).enter().append("rect");`	To use data and enter, you need to use selectAll first. (else elements are created, but not where you'd expect.)

Path – what goes in the d attribute:

`M`	`x,y`	Begins the string. M moves the cursor to the designated position to draw a shape. If more than one pair of coordinates are provided, it's as if the other pairs are preceded by an L.
`m`	`x,y`	Same as above, the only difference is that if more than one pair of coordinates are provided, extra pairs are processed as if preceded by an l.
`L`	`x,y`	Draws a line from the last position to the specified position.
`l`	`x,y`	Draws a line from the last position to a relative position: x pixels to the right and y to the bottom.
`H`	`x`	Draws a horizontal line to the specified, absolute x position (y stays the same)
`h`	`x`	Draws a horizontal line x pixels to the right (if x is negative, the line is drawn to the left).
`V`	`Y`	Draws a vertical line to the specified, absolute y position (x stays the same)
`v`	`y`	Draws a vertical line x pixels to the bottom (if y is negative, the line is drawn to the top).
`A`	`rx,ry,alpha,large,sweep,x,y`	Draws an elliptical arc: rx and ry are the radius of the ellipse; alpha is the x-axis rotation of the ellipse; large is 0 if the arc should be the shorter arc (less than 180°), 1 if it should be the longer arc; sweep is 0 is the arc is to be drawn clockwise, 1 if counter-clockwise; x,y are the coordinate of the end point of the arc.
`Q`	`cX cY eX eY`	Draws a Bézier quadratic curve. cX,cY are the coordinates of the control point, eX,eY that of the endpoint. More pairs can be provided (extra control points and endpoints).
`q`	`cX cY eX eY`	Draws a Bézier quadratic curve, with the coordinates of the points relative to the current point.
`T`	`eX eY`	Draws a Bézier quadratic curve, using the last provided control point (or failing that the current point).
`t`	`eX eY`	Same as above, with the coordinates of the end point relative to the current point.
`C`	`cX1 cY1 cX2 cY2 eX eY`	Draws a Bézier cubic curve, with cX1,cY1, cX2, cY2 being the coordinates of the control points.
`c`	`cX1 cY1 cX2 cY2 eX eY`	Same as above, with the coordinates of the points relative to the current point.
`S`	`cX2 cY2 eX eY`	Draws a Bézier cubic curve, using the previously provided control point (cX1,xY1) or failing that the current point, cX2,cY2 as the next control point, and eX, eY as the end point
`s`	`cX2 cY2 eX eY`	Same as above, with the coordinates of the points relative to the current point.
`Z`		Optional at the end the string for a closed shape (ie line to the first point)