

Scalar Vector Graphics (SVG)

Introduction:

SVG is a W3C (World Wide Web Consortium) recommended XML based file format that is used to describe 2D vector graphics. SVG describes and defines the vector –based graphics in XML formats for the web. This is a feature rich two-dimensional graphics language which allows for the combination of vector graphic shapes, raster images (jpeg, bmp, png, etc) and text.

Like other markup languages, SVG is integrated with these features to provide styling template. It also supports DOM (Document Object Model), ECMAScript (aka JavaScript) and [SMIL](#) (Synchronized Multimedia Integration Language).

Browser Support:

To view any web page the browser must be SVG enabled since it is written in XML otherwise we need to have plug in. Mozilla is one of the browsers that support SVG. Internet Explorer requires plug in and that are available free.

Features of SVG includes

- Presentation of fully scalable images in case of zooming and panning
- Perfect Positioning of Pixel.
- high resolution gradients, drop shadows, and other filter effects
- color control and its accuracy got enhanced
- printing resolution possibility turned highest
- editable and search able text
- ability to search text elements within graphics
- dynamic content, animation, and interactivity through scripting
- download sizes turns compact
- Cascading Style Sheet support enabling global Web site changes
- multiple levels of transparency
- other devices such as palmtops, GPS, cellphones supported by the frame

Origin of SVG

With the requirement of vector graphic format by 1998, the landscape had settled somewhat and there were five competing submissions to the W3C in the Web vector graphics area that year:

- PGML, from Adobe, IBM, Netscape, and Sun
- VML Autodesk, Hewlett-Packard, Macromedia, and Microsoft
- Hyper Graphics Markup Language, by Orange, PCSL, and PRP
- DrawML, from ExcOSOFT
- WebCGM from Boeing, CCLRC, Inso, JISC, and Xerox

As we know, object-oriented graphics uses software and hardware to represent images using geometrical formulas. Bitmap is the other method of representing graphical image in which the images are composed of a pattern of dots. But Bitmap is limited when resizing and stretching comes into play, which is flexibly handled by vector oriented images and better resolution factor is also found in case of vector orientation of images.

So all the sophisticated graphics system including CAD systems and other animated software uses vector graphics.

Microsoft's browser supports XML – based a Vector markup language a reduced functionality SVG. So before the introduction of SilverLight the appropriate vector graphic language was SVG, even now also SVG users gives preference to SVG over XAML which is a vector markup used in silverLight.

XAML and SVG

In case of XAML, vector graphic markup is used this is very similar to SVG. But when we talk about XAML it's a complete application framework where as SVG is not. When we talk about WPF Window Presentation Foundation it deploys XAML to define UI. In the verge to combine both the designer and the developer in a single premise Microsoft came up with a new user interface platform Windows Presentation Foundation (WPF).

W3C vs. Adobe vs. Microsoft (SVG vs. Flash vs. SilverLight)

Let's compare the situation we know that SVG is a cross-browser, cross-platform presentation of vector graphics and animation. Adobe product Flash also provides a cross-browser, cross-platform way of doing the same thing as does SilverLight. All three are mutually working with the same goal. Flash and SilverLight are "proprietary", and they are controlled by the two largest names in computer industry – Adobe and Microsoft respectively. SVG on the other hand, is governed by W3C from the beginning and is hence "open".

Basic Shapes in SVG:

Shapes that are supported by SVG are as follows:

- a. Rectangles
- b. Circles
- c. Ellipses
- d. Line
- e. Polygon

f. Polyline

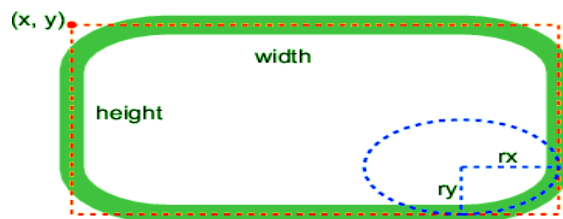
Rectangles:

Rectangle is one of the major shapes supported by SVG. Basic syntax of representing a rectangle in SVG format is as below

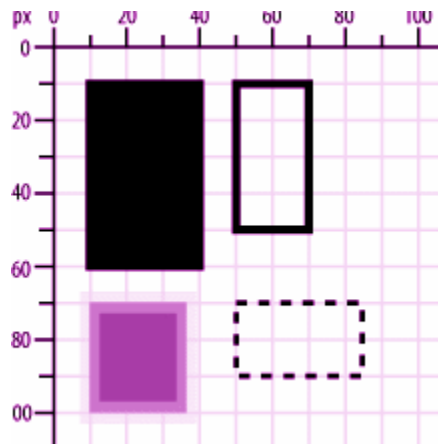
```
<rect x="40" y="30" rx="0" ry="0" width="40" height="60" fill="blue" />
```

Basic attributes of rectangle can be listed in tabular form as below

Attributes	Representation
X	X coordinates of upper left corner of the rectangle.
Y	Y coordinates of upper left corner of the rectangle.
rx	For rounded rectangles. x-axis radius used to round the element
ry	For rounded rectangles. y-axis radius used to round the element
width	Represents width of rectangle
Heights	Represents Height of the rectangle



Let's discuss some more examples analyzing the graph



In the above shown figure we can find four rectangles so starting from **first rectangle** if we look the upper left corner of the rectangle we can find that the x and y coordinates value of the first rectangle, in this instance the x coordinate is 10. While measuring the width we can find that width in x direction extends from 10 – 40 so width is 30, measuring the length the extension of rectangle in Y direction is from 10 - 60 so the length is 50. So the syntax of my rectangle in SVG will be

```
<rect x="10" y="10" width="30" height="50"/>
```

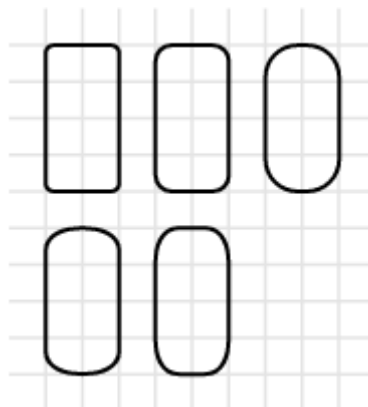
Second Rectangle attributes from the graph can be determined as x =50, y=10 measuring the extension of the rectangle in x direction 20(50 – 70) and that in Y direction 40(10 – 50). But this rectangle is different to first rectangle in terms of **fill** and **stroke** displayed. Looking at the rectangle we can determine the fill attribute of the rectangle require no specification and there is border stroke. Packing them all the syntax will be

```
<rect x="50" y="10" width="20" height="40" style="fill: none; stroke: black ;"/>
```

Coming to the third rectangle x and y coordinates will be 10 and 70 respectively. And looking at the extension of the rectangle width and length can be determined as 25 and 30 respectively. But we are coming across lots of different attributes. Here the fill is different; stroke width is a factor and looking more precisely we are getting opacity in stroke also. Again the syntax will be

```
<rect x="10" y="70" width="25" height="30" style="fill: #0000ff; stroke: gray; stroke-width: 7; stroke-opacity: 0.5;"/>
```

Let's look into rounded rectangle



So in case of first rectangle we can find both x axis radius and y axis radius for rounded rectangle are same so we can represent the above first rectangle with SVG Format as

```
<rect x="10" y="10" width="50" height="70" rx="4" ry="4" style="stroke: black; fill: none;"/>
```

Similarly we can find that in second rectangle rx is having some value and ry is 0 whereas in third rectangle ry is having some value and rx is 0. We can represent it as follows

```
<rect x="40" y="10" width="50" height="70" rx="5" style="stroke: black; fill: none;"/>
```

```
<rect x="70" y="10" width="50" height="70" ry="10" style="stroke: black; fill: none;"/>
```

In fourth rectangle if we pay attention we can find that rx value is more than ry while fifth rectangle is doing just the opposite of fourth rectangle. Representation of the same will be

```
<rect x="10" y="60" width="50" height="70" rx="10" ry="5" style="stroke: black; fill: none;"/>
```

```
<rect x="40" y="60" width="50" height="70" rx="5" ry="10" style="stroke: black; fill: none;"/>
```

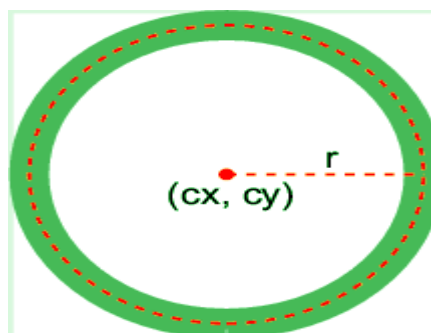
Circles:

Another important shape is Circle. Representation of the same in SVG is as follows

```
<circle cx="10" cy="10" r="5s" fill="blue" />
```

Basic attributes of circle can be listed in tabular form as below

Attributes	Representation
cx	X coordinates center of the circle.
cy	Y coordinates center of the circle
r	Radius of the circle



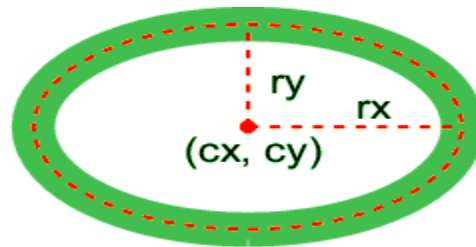
Ellipses:

Syntax for representing ellipse in SVG format is as below

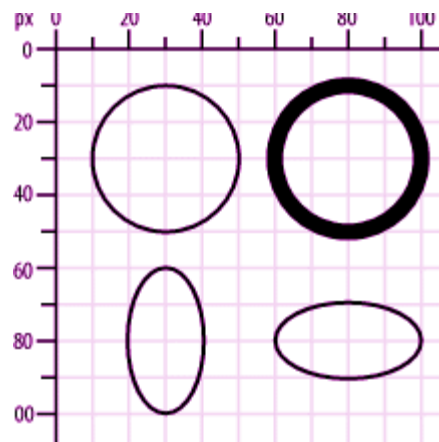
```
<ellipse cx="200" cy="200" rx="190" ry="180" fill="blue" />
```

Basic attributes of circle can be listed in tabular form as below

Attributes	Representation
cx	The x-axis center of the ellipse
cy	The Y-axis center of the ellipse
rx	Ellipse's radius length along the x-axis
ry	Ellipse's radius length along the y-axis



Let's analyze circle and ellipse through examples



The above graph comprises two rectangles and two ellipses lets checkout their syntax

For the first circle we can see the center x and y coordinates to be 30, 30 and if we look into the graph we will find the diameter to be 40 which means radius is 20. The SVG syntax for the same will be

```
<circle cx="30" cy="30" r="20" style="stroke: black; fill: none ;"/>
```

Second Circle is the circle with same radius but X and Y Coordinates are 80 and 30. One important attributes that comes into play is the stroke and its width which is making it different from the previous one . Packing them all will let to

```
<circle cx="80" cy="30" r="20" style="stroke-width: 5; stroke: black; fill: none;"/>
```

Let's move to the section of ellipse the

First ellipse in the above graph is having cx and cy co-ordinates to be 30 and 80 respectively and if we look, rx can be measured as 10 and ry can be measured as 20. Second ellipse coordinate calculations will come up to cx and cy with 80 and 80 respectively and rx and ry to be 20 and 10 respectively. Both can be represented as

```
<ellipse cx="30" cy="80" rx="10" ry="20" style="stroke: black; fill: none;"/>
```

```
<ellipse cx="80" cy="80" rx="20" ry="10" style="stroke: black; fill: none;"/>
```

Lines:

The Syntax for representing a line goes like

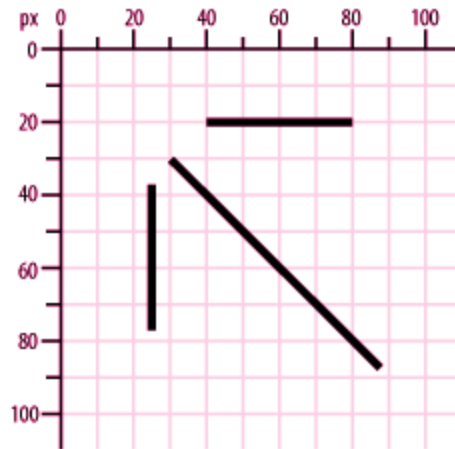
```
<line x1="10" y1="10" x2="190" y2="190" stroke="blue" stroke-width="4" />
```

Basic attributes for the lines can be listed below:

Attributes	Representation
X1,y1	The x1 and y1 are the initial coordinates of the Line
X2,y2	The x2 and y2 are the final coordinates of the Line

Again analyzing the graphical representation and representing the syntax





Lets consider each line one by one

For the first line(vertical) the initial coordinates (x1 ,y1) and the final coordinate (x2, y2) value can easily be identified to be (40, 20) and (80 , 20) and we can look at the stroke to be black so its representation in SVG will be .

```
<line x1="40" y1="20" x2="80" y2="20" style="stroke: black;"/>
```

Similarly for vertical and diagonal line we can place the SVG format as follows

```
<line x1="25" y1="35" x2="25" y2="75" style="stroke: black;"/>
```

```
<line x1="30" y1="30" x2="90" y2="90" style="stroke: black;"/>
```

Polygon :

Its SVG representation is like

```
<polygon points="100,10 40,190 190,50 10,70 160,170 90,10" stroke="blue" fill="darkblue" stroke-width="4" />
```

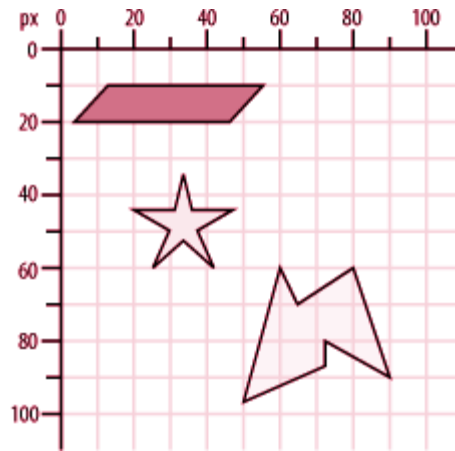
Attributes for the Polygon is represented below

Attributes	Representation
Points	Each pair of point represents coordinates

In above diagram we can list out different coordinate to be

(850,75), (958,137.5),(958,262.5),(850,325),(742,262.6),(742,137.5)

Two consecutive coordinates is representing a line . Let us discuss more with the help of graphical representation of the same



So if we list out the coordinates of the first polygon presented in the graph than it will come to (15, 10), (55, 10) , (45,20), (5,20) and more to that some attributes such as fill and stroke representing all together

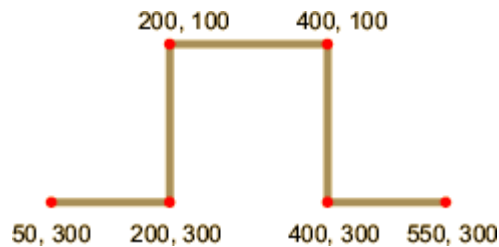
```
<polygon points="15,10 55, 10 45, 20 5, 20" style="fill: gray ; stroke: black;"/>
```

Similarly we can represent other two polygon also.

PolyLine :

This shape is very much similar to polygon mentioned above but the difference is that is PolyLine are not closed whereas polygon are closed shapes .Representation of the same in SVG is as follows

```
<polyline points="5 20, 20 20, 25 10, 35 30, 45 10, 55 30, 65 10, 75 30, 80 20, 95 20" style="stroke: black; stroke-width: 3; fill: none;"/>
```



Listing the coordinate in above figure

(50,300), (200,300), (200,100), (400,100), (400,300) , (550,300)

since the shape is not closed its a polyline and the above figure can be represented as

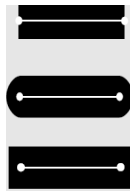
```
<polyline points="50,300 200,300 200,100 400,100 400,300 550,300" style="stroke: black; stroke-width: 2; fill: none;"/>
```

Line Caps and Joins :

This is a very essential concept in SVG. While drawing a line we need to take care the end points of the line . Depending upon the requirement we can set the endpoints to be "butt", "round ", "square". The attributes that takes the mentioned value is **stroke-linecap**

```
style="stroke-linecap:butt;"  
style="stroke-linecap:round;"  
style="stroke-linecap:square;"
```

We will get the output as



Join deals with the shape created at the corner of the line joined . Again on requirement we can set the corner shape to be "miter", " round" and bevel. The attributes that takes the mentioned value is **stroke-linecap**

```
style="stroke-linejoin: miter ;"  
style="stroke-linejoin: round ;"  
style="stroke-linejoin: bevel ;"
```

and corresponding output will be



This is Introduction to SVG and contents are covered from various site mentioned below . To know more about SVG please Refer :-

Reference :-

- http://apike.ca/prog_svg_text.html
- <http://www.princexml.com/doc/7.0/svg/>
- <http://www.w3schools.com/svg/default.asp>
- <http://www.learnsvg.com/>

