pySVG Tutorial

Working with pySVG is relatively straight forward. You (still) need some knowledge on what values the appropriate svg elements need. Apart of that it's a piece of cake. Note that parameters that are numbers normally can be given either as ints or strings. Should both work fine.

Text

Lets start with the typical "hello world" which I have already packed into a method.

```
def HelloWorld1():
  svg=SVG("Hello World Example") # title is ignored still
  t=text("Hello World",0,200)
  svg.addElement(t)
  svg.saveSVG('HelloWorld1.svg')
```

Easy enough isn't it? The steps as a summary:

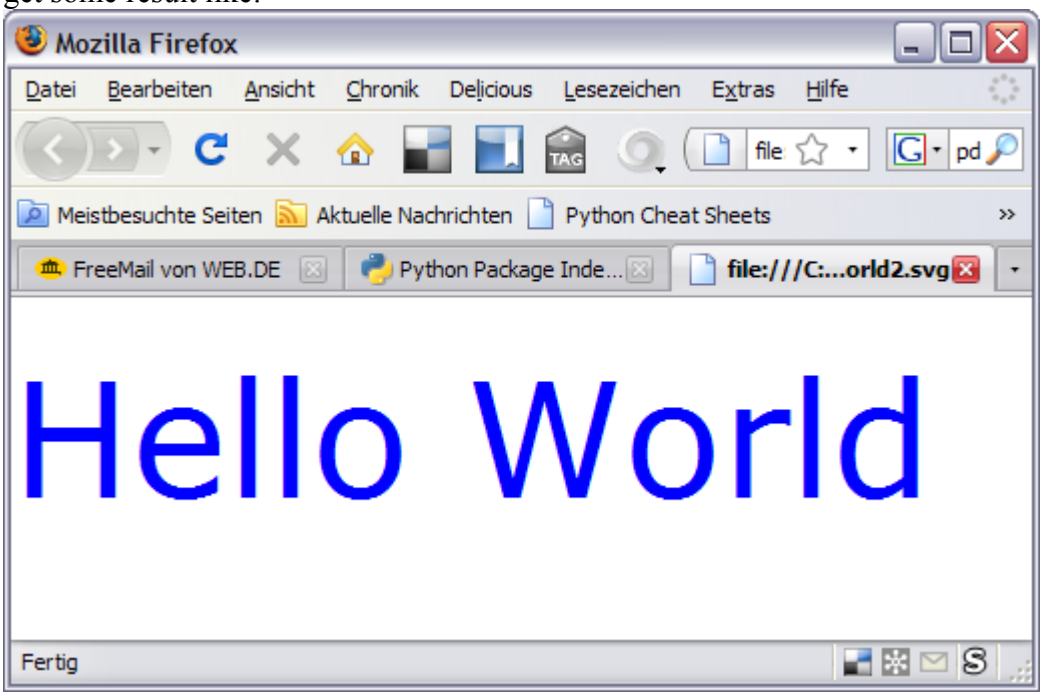
1. Create a svg document.
2. Create a text element with the text "Hello World" at position x=0, y=200.
3. Add the text to the svg document.
4. Save the document as “HelloWorld1.svg”

Run the script and open the saved document it with your svg capable browser or imageviewer.

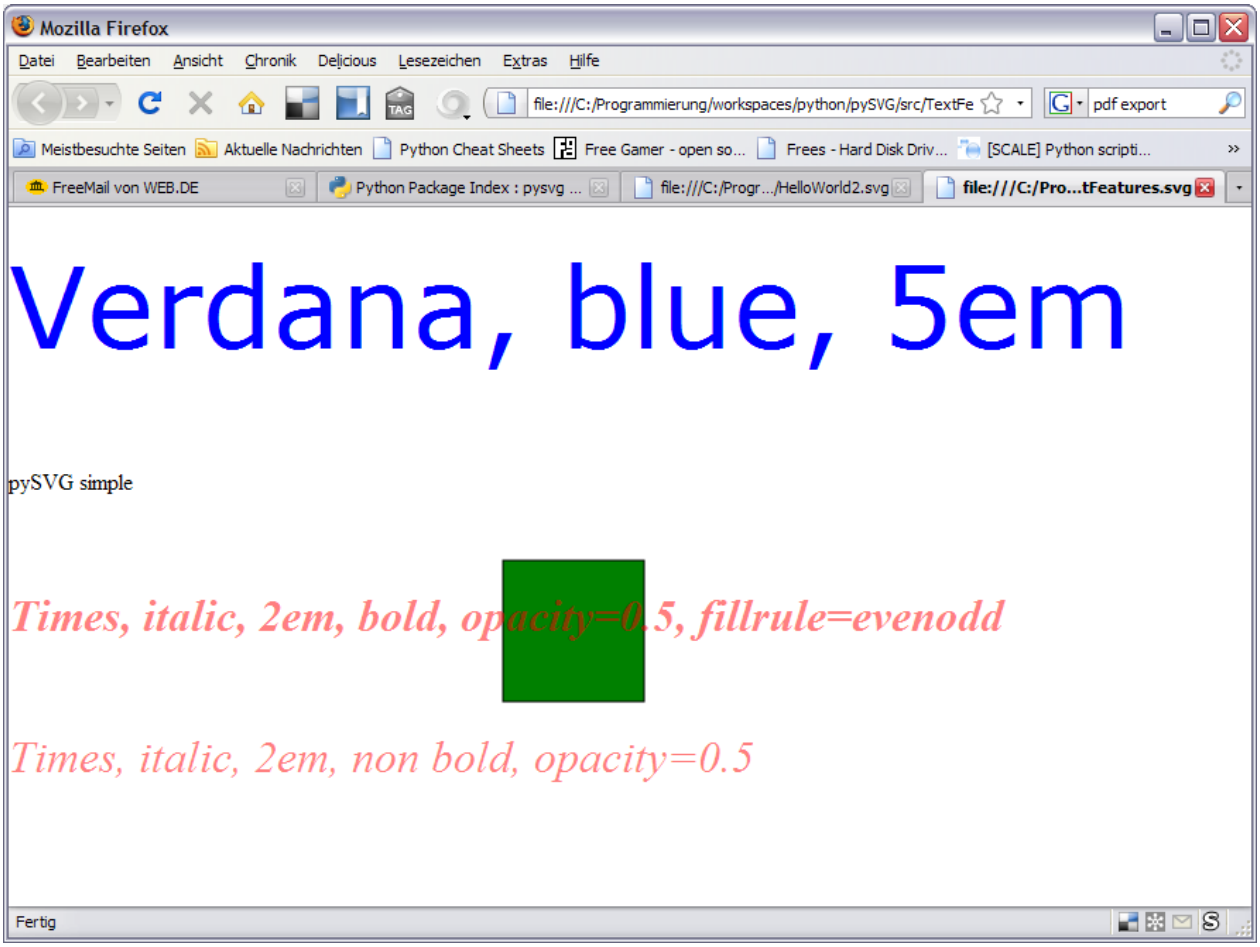
Now let's do some more stuff with the styling. The class stylehelper will provide you with many methods for fonts, filling, stroke and so on. (Some might not work yet ;-))

```
def HelloWorld2():
  svg=SVG("Hello World Example") # title is ignored still
  style=StyleHelper()
  style.setFontFamily(fontfamily="Verdana")
  style.setFontSize('5em') #no need for the keywords all the time
  style.setFilling("blue")
  t1=text("Hello World",0,100,style_dict=style.getStyleDict())
  svg.addElement(t1)
  svg.saveSVG('HelloWorld2.svg')
```

Just create the style, set the font and filling. Rest stays the same. Run the program and you should get some result like:



Look at the function TextFeatures() so get a result like the one below:



Basic Shapes

Including shapes is as easy as it is to produce text.
Since svg is a bit nasty when it comes to defaults, you can have a look at the objecthelper class. It provides you with some defaults and some extra parameters that I think are needed often. I give you an example below.
In order to draw a line you would normally do something like:

```
svg=SVG("test")
style=StyleHelper()
style.setStrokeWidth(2)
style.setStroke('black')
l=line(0,0,300,300, style.getStyleDict())
svg.addElement(l)
svg.saveSVG('Line.svg')
```

With the objecthelper class however it is far shorter:

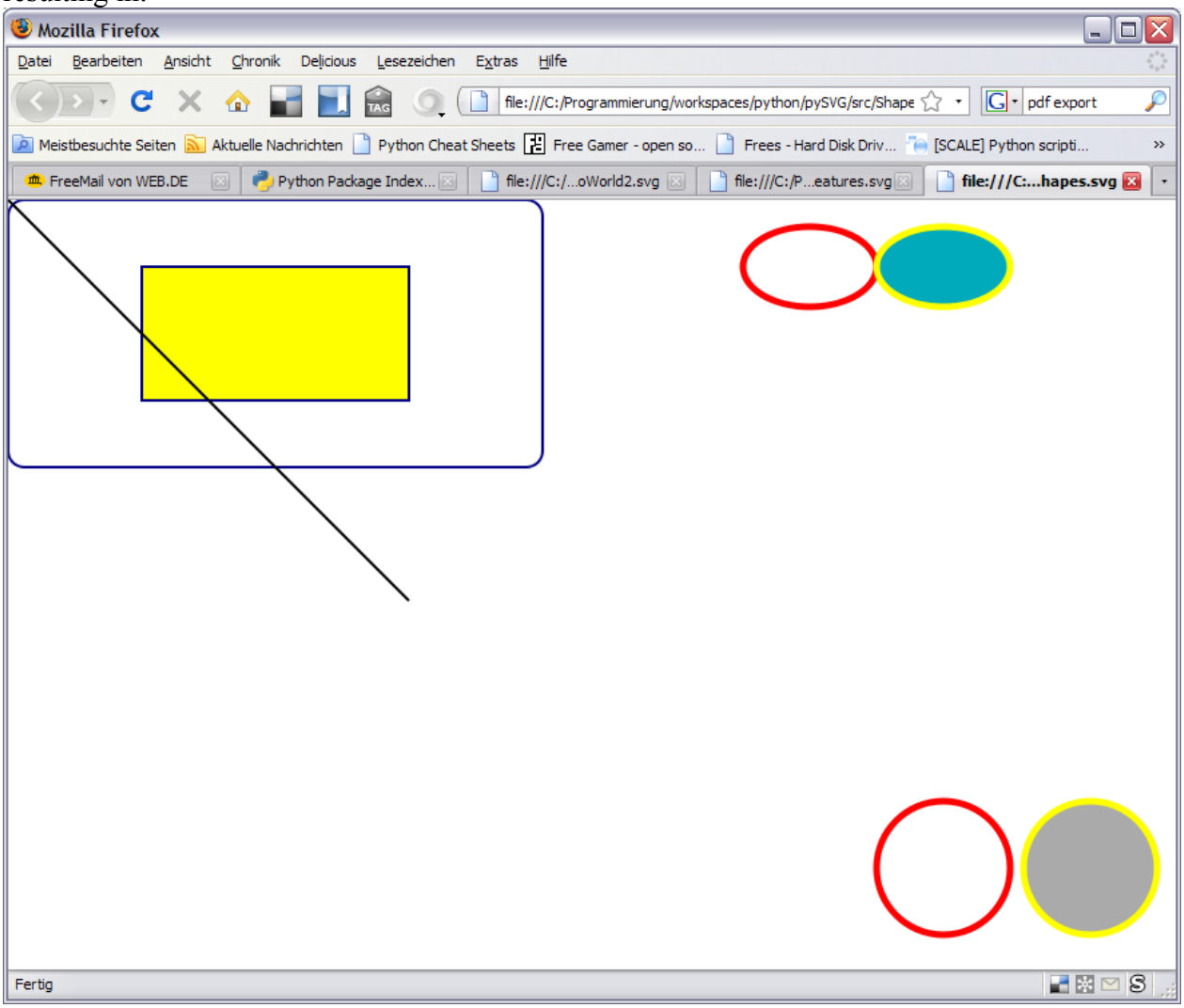
```
svg=SVG("test")
oh=ObjectHelper()
svg.addElement(oh.createLine(10,0,300,300,strokewidth=2,stroke="blue"))
svg.saveSVG('Line.svg')
```

Some parameters are often needed and so I tried to include them (with defaults) in the objecthelper. Feel free to mail me if you think something usefull is missing.
Now let us simply paste the code for common shapes here. I think you will get the idea fast.

```
def Shapes():
  oh=ObjectHelper()
  svg=SVG("test")

  svg.addElement(oh.createRect(0,0,400,200,12,12,strokewidth=2,
stroke='navy'))
  svg.addElement(oh.createRect(100,50,200,100,strokewidth=2, stroke='navy',
fill='yellow'))
  svg.addElement(oh.createCircle(700,500,50,strokewidth=5, stroke='red'))
  svg.addElement(oh.createCircle(810,500,50,strokewidth=5, stroke='yellow',
fill='#AAAAAA'))
  svg.addElement(oh.createEllipse(600,50,50,30,strokewidth=5, stroke='red'))
  svg.addElement(oh.createEllipse(700,50,50,30,strokewidth=5, stroke='yellow',
fill='#00AABB'))
  svg.addElement(oh.createLine(0,0,300,300,strokewidth=2,stroke="black"))
  svg.saveSVG('Shapes.svg')
```

resulting in:



Complex Shapes

Complex shapes are done using polylines, polygons and paths. Now these (for now) are still the ugliest folks in the lib. For example if you have a polyline you must specify ALL points as a string. Something like:

```
pl=oh.createPolyline(points="50,375 150,375 150,325 250,325 250,375 350,375
350,250 450,250 450,375 550,375 550,175 650,175 650,375 750,375 750,100
850,100 850,375 950,375 950,25 1050,25 1050,375 1150,375 ",strokewidth=10,
stroke='blue')
```

This is rather ugly. The objecthelper has a function that allows you to convert a list of points (tuples) into that format. You can use that for polylines or for polygons for example like here:

```
pointsAsTuples=[(350,75),(379,161),(469,161),(397,215),(423,301),(350,250),
(277,301),(303,215),(231,161),(321,161)]
pg=oh.createPolygon(points=oh.convertTupleArrayToPoints(pointsAsTuples),stroke
width=10, stroke='blue', fill='red')
```

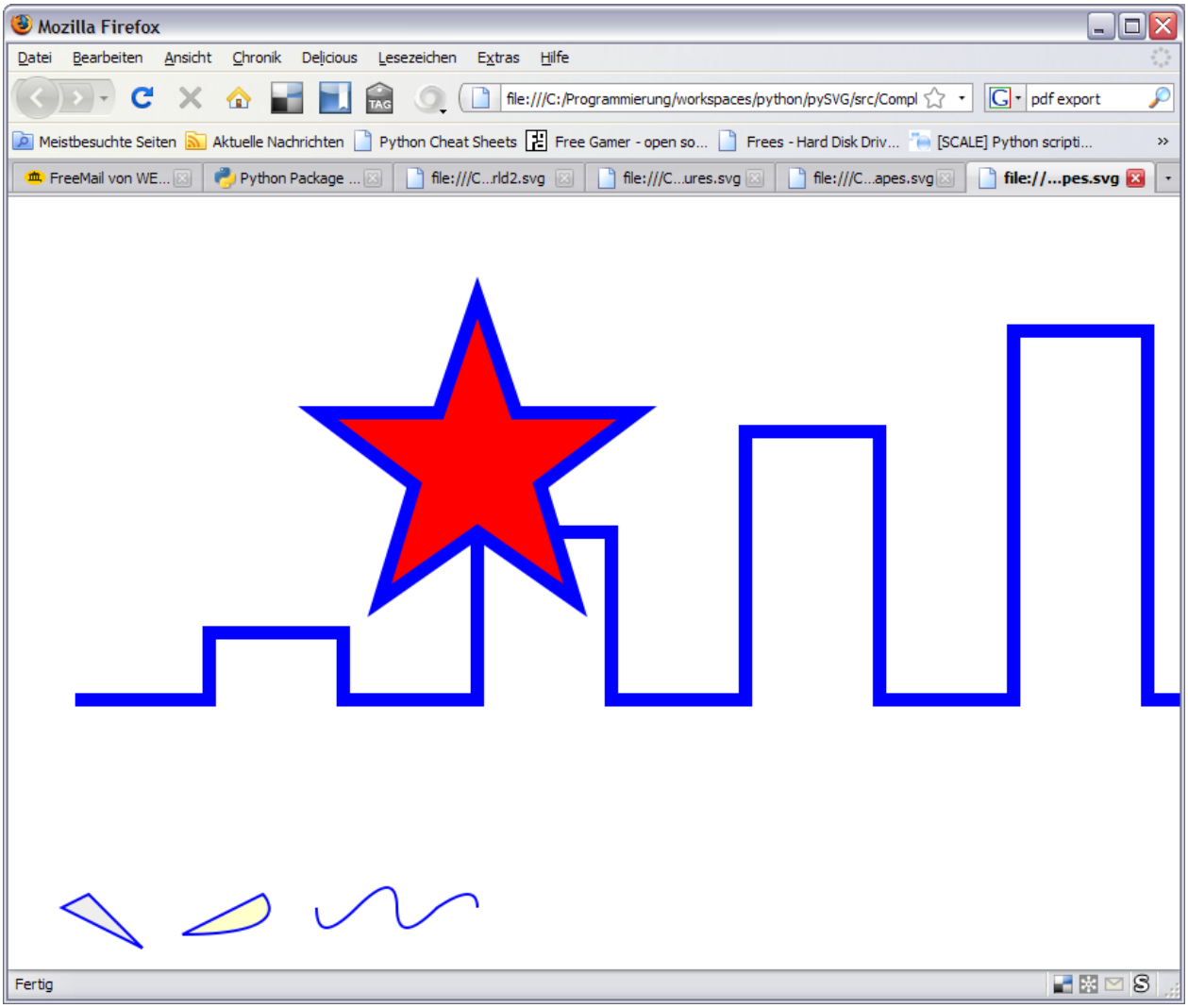
pySVG also supports paths now but it doesn't yet OPTIMIZE them. If for example you have several moveto commands you can normally append the coordinates and skip the command repetitions. In pySVG you must append each move separately and it will create slightly bigger svg files.

Here are some examples for paths:

```
sh=StyleHelper()
sh.setFilling('#EEE')
sh.setStroke('#00F')
sh.setStrokeWidth('2px')
path1=path('M 40,530 L 100,560 L 60,520 Z', style_dict=sh.getStyleDict())

sh2=StyleHelper()
sh2.setFilling('#FFC')
sh2.setStroke('#00F')
sh2.setStrokeWidth('2px')
path2=path(style_dict=sh2.getStyleDict())
path2.appendMoveToPath(190, 520, False)
#as you can see we can mix strings and ints without trouble
path2.appendCubicCurveToPath('+0', '+0', 30, 30, -60, 30, True)
path2.appendCloseCurve()

sh3=StyleHelper()
sh3.setFilling('none')
sh3.setStroke('#00F')
sh3.setStrokeWidth('2px')
path3=path('M 230,530', style_dict=sh3.getStyleDict())
path3.appendQuadraticCurveToPath(0, 30, 30, 0)
path3.appendQuadraticCurveToPath(30, -30, 30, 0)
path3.appendQuadraticCurveToPath(-0, 30, 30, 0)
path3.appendQuadraticCurveToPath(30, -20, 30, 0)
```



Grouping

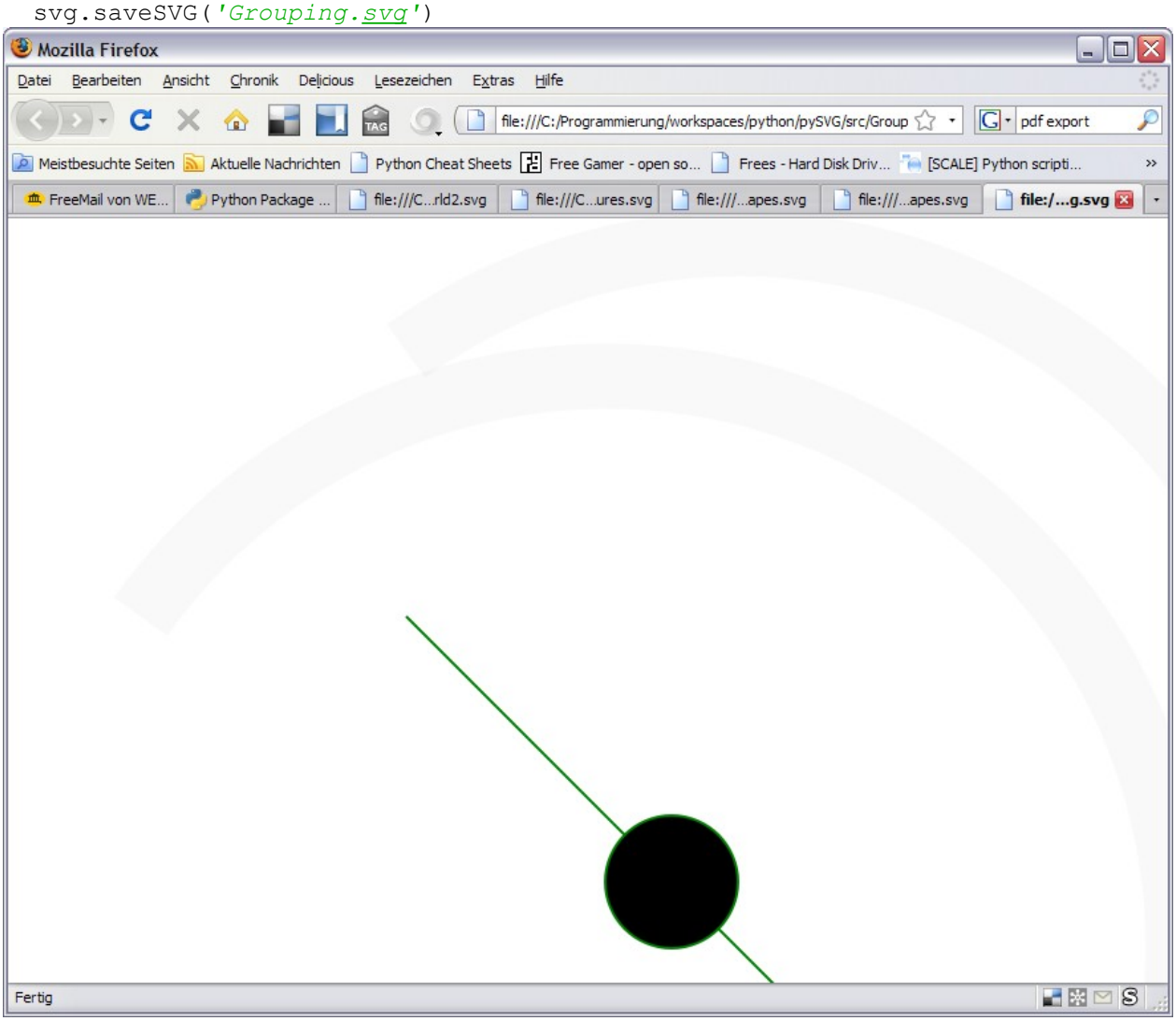
Sometimes you want to apply certain styles to many elements. And changes in that style also should have an effect on all appropriate elements. For that svg has a group element that you of course can also use in pySVG:

```
def Grouping():
  oh=ObjectHelper()
  svg=SVG("test")

  #testing container
  style=StyleHelper()
  style.setStrokeWidth(2)
  style.setStroke("green")
  style_dict=style.getStyleDict()
  g=Group(style_dict)
  g.addElement(line(300,300,600,600))
  g.addElement(circle(500,500,50))
  svg.addElement(g)

  g=Group(style_dict)
```

```
  style_dict = {"stroke":"#000000", "fill":"none" ,"stroke-
width":"49" ,"stroke-opacity":"0.027276"}
  p=path(pathData="M 300 100 A 1,1 0 0 1 802,800",style_dict=style_dict)
  p2=path(pathData="M 100 300 A 1,1 0 0 1 802,800",style_dict=style_dict)
  g.addElement(p)
  g.addElement(p2)
  svg.addElement(g)
  print svg.getXML()
  svg.saveSVG('Grouping.svg')
```



Notice that both arc have the same style. The circle and the line also have the same style using a green colored stroke with a width of 2px.

What is missing?

Frankly too much to count ☺ But I think gradients are what I miss most for now.