**REVIEW tutorial challenges**

1. **Draw a SVG object from a path string.**

Find an svg image you like on the web, and grab its path string.

Or if you’d rather, just use the path to my heart:

mybigheart="M340.8,98.4c50.7,0,91.9,41.3,91.9,92.3c0,26.2-10.9,49.8-28.3,66.6L256,407.1L105,254.6c-15.8-16.6-25.6-39.1-25.6-63.9c0-51,41.1-92.3,91.9-92.3c38.2,0,70.9,23.4,84.8,56.8C269.8,121.9,302.6,98.4,340.8,98.4z"

(Did you know? You can also create SVG paths drawing freehand in Inkscape (and other apps)!)

mybigheart is just the SVG path string at this point, not a graphics object.

1. a) Use it to actually draw a heart by using the ‘path’ function which is a property of the Raphael paper. Pass it the path string as an argument, and store the results in a variable myHeart1.

1.b) Give it some attributes using the attr function which is a property of all Raphael objects. Recall that the attr function takes an object of attribute /object pairs.

**2. The transform attribute**

<https://dmitrybaranovskiy.github.io/raphael/reference.html#Element.transform>

Raphael objects have a property called transform. You can set it like any other attribute.

object.attr({transform : val})

Its vale is a string with letters (S, T, and R, for scale, rotation, translation) and numbers. The meaning of the numbers depend on the letters they follow (like for path).

S – followed by one number specifying the scale magnification factor. Anything less than 1 shrinks the object, anything bigger than 1 magnifies it.

T – followed by 2 numbers: translation in x, translation in y

R – followed by a number specifying rotation in degrees

Thus “S2T100,50” would mean a double the size, move 100 to the right, and 50 down)

Explore! Try various translations, rotations, and scaling of the heart.

1. **Multi-stage animation**

Now, do a 2 (or more!) -stage animation.

Write one function (call it foo) that animates the transform attribute specifying new values for scale, transform, and/or rotation, and a second function (call it bar) for specifying different values.

Remember: The Raphael object animate method can take 3 arguments: the object specifying attributes to animate with their new values, the number of ms the animation will last, and the last one is the name of a function to call when the animation is complete.

Thus obj.animate( arg1, 1000, bar) would animate the obj to the attribute values specified in art1, it would last one second, and would then call bar.

1. **Rapahael.transformPath**

Unfortunately, paths are “hard coded”. They don’t have ‘x’ and ‘y’ attributes like rectangles, or ‘cx’ and ‘cy’ attributes like circles and ellipses that can be used to set their location. That means, we actually have to change each and every point in the path if we want to move it. Augh.

Luckily, Raphael provides is a method for doing that, Raphael.transformPath.

It takes 2 arguments: a path string, and a transform string.

It returns a new string.

SO – make a few more objects based on transformations of the mybigheart path.

Give them different attributes.

Animate them all at the same time! (You could “synchronize” their animations by putting the animate calls in the same functions you have already written, or you could write new animation functions for the new objects)

1. **Still have time to explore?**

Make the animations depend on some kind of interactivity (though I do not recommend using the mouse to position an svg path object).