This report will show how to use the pattern editor that I created and explain the codes used to create the pattern editor. It compare the original plan(FDs) against the actual program(Pattern Editor) to evaluate what are things that are achieved and what are not.

## Critical analysis

Comparing to the original plan which was specified to the last page of the FDS there are some functionalities that I wasn't able to include on the actual pattern editor such as alternating colours and gradient for the colour because it doesn't seem to make sense to do both of it since I am assigning a colour for each shape. i.e. if I do a gradient from green to red, it doesn't make sense on which way it would be implemented. Whether it will be for each shape or it will be for the whole SVG pattern that's why I excluded it. I also added a little preview for each shape at the top of the pattern controls/ pattern UI which shows the shape for each shape. I also excluded save function and DOM function this is because you can just do a screenshot towards the pattern and inspect the element to view the elements used to create the svg pattern. Another reason is that there is just little to only bits of information available on how to create your own svg using just javascript making this assignment a harder task than it should be.

As for the performance and functionality of the pattern editor itself. It creates an awkward pattern only when shape 5 is chosen. i.e. rather than a grid type of pattern is produced a crooked grid type pattern is produced. Because you can't render just the half of the pattern and attach that to the next row to make a perfect square grid. As for the other shapes/patterns, they are working just as expected and produces a grid type pattern. Also, by letting the user choose which colour is to be assigned to each shape you can just camouflage the inner shape or create a new pattern using the pattern editor (screenshot of performance at screenshot section).

Overall, I think that even though some of the original planed functionality have been strip off from the pattern editor it didn't affect the actual function of the pattern editor itself. Although the pattern editor is a grid type pattern it produces many types of pattern based on simple shapes which is what I envision before creating and during planning of this task. What I didn't expect is that the assigning of colours towards each shape will camouflage other shapes thus creating other pattern possibilities on 2 shape composition up until 5.

## The code/s

My Pattern editor is separated into 3 files the myPtrn.css which consist of all the styles for the general tags on the html document; the PtnEdit.htm which is where the SVG Viewer and Pattern controls is located and the PtrnFunc.js which is where all of the DOM functions needed to create SVG is located.

To be able to create an SVG pattern I will need to be able to access the html document where I wanted to show my SVG pattern by using:

```
var body = document.getElementById("displaySVG");
```

Which is an ID to the <svg> tag within the html document. I can just then create a child node and append that as a child towards the much like:

```
body.appendChild(useIt("#_ptrn",ax*j,ay*i));
```

where useIt() function recalls a shape already defined somewhere before appending it to the parent node which is the <svg> tag. Such as

```
var defs = document.createElementNS(NS,"defs");

        var circle = doCircle(20,28,12 , "_c", s1Colour);
        var triangle = doTri(s2Colour ,"_t");
        var rectangle = doRect(40,40, "_r", s3Colour);
        var triangle2 = doTri2(s4Colour,"_tr");

        defs.appendChild(circle);
        defs.appendChild(triangle);
        defs.appendChild(rectangle);
        defs.appendChild(triangle2);
```

the document.createElementNS takes in two attribute the namespace used for the element and the element itself much like xlink:href on which xlink is the namespace and href is a reference to an xml document. I did this so I can access the SVG elements that I needed to create the svg pattern. additionally I used defs much like the same reason on doing it on an actual svg coding which rather than creating a shape over and over again you could just define the shape an use that shape over and over again. Another advantage is that by using def, I could just assign the attributes that I wanted towards the defs node so when I use it. The svg shape will be identical to everyone. The doCircle ,doTri,doRect and doTri2 are functions that I created to create the shapes I wanted

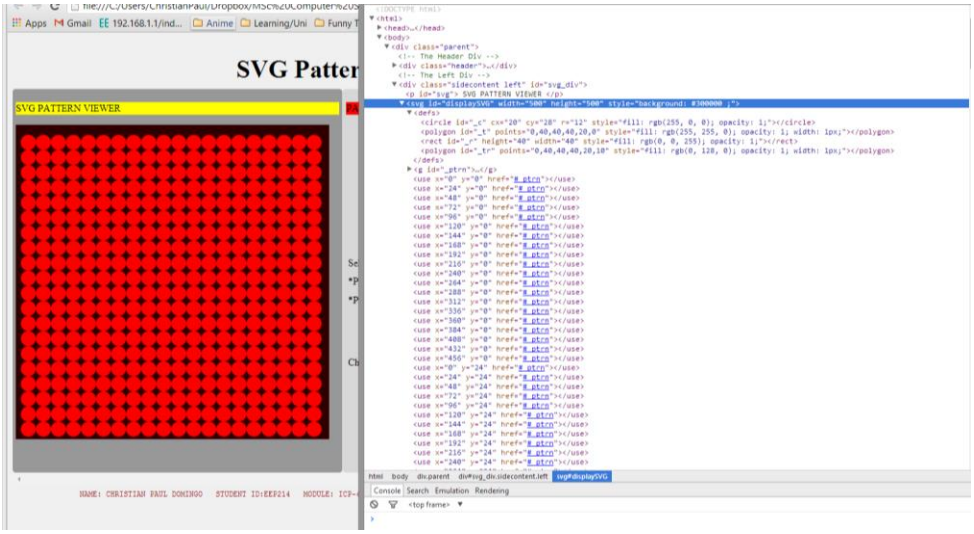
```
function doCircle(cx, cy, r, myID, colour){
        var c = document.createElementNS(NS,"circle");
        c.setAttributeNS(null,"id",myID)
        c.setAttributeNS(null,"cx",cx);
        c.setAttributeNS(null,"cy",cy);
        c.setAttributeNS(null,"r",r)
        c.style.fill=colour;
        c.style.opacity=1;
        return c;
}

function doRect(h,w, myID,colour){
        var r = document.createElementNS(NS,"rect");
        r.setAttributeNS(null,"id",myID);
        r.setAttributeNS(null,"height",h);
        r.setAttributeNS(null,"width",w);
        r.style.fill=colour
        r.style.opacity=1;

        return r;
}

function doTri(colour,myID){
        var t = document.createElementNS(NS,"polygon");
        t.setAttributeNS(null,"id",myID);
        var mypoints = [0,40, 40,40, 20,0];
        t.setAttributeNS(null,"points", mypoints );
        t.style.fill=colour;
        t.style.opacity=1;
        t.style.width=1;
```

```
	return t;
}

function doTri2(colour,myID){
	var t2 = document.createElementNS(NS,"polygon");
	t2.setAttributeNS(null,"id",myID);
	var mypoints = [0,40, 40,40, 20,10];
	t2.setAttributeNS(null,"points", mypoints );
	t2.style.fill=colour;
	t2.style.opacity=1;
	t2.style.width=1;

	return t2;
}
```

As I mentioned before xlink:href also uses namespace of xlink. On my useIt function I created a function that will do just that, the xlink namespace is declared globally as

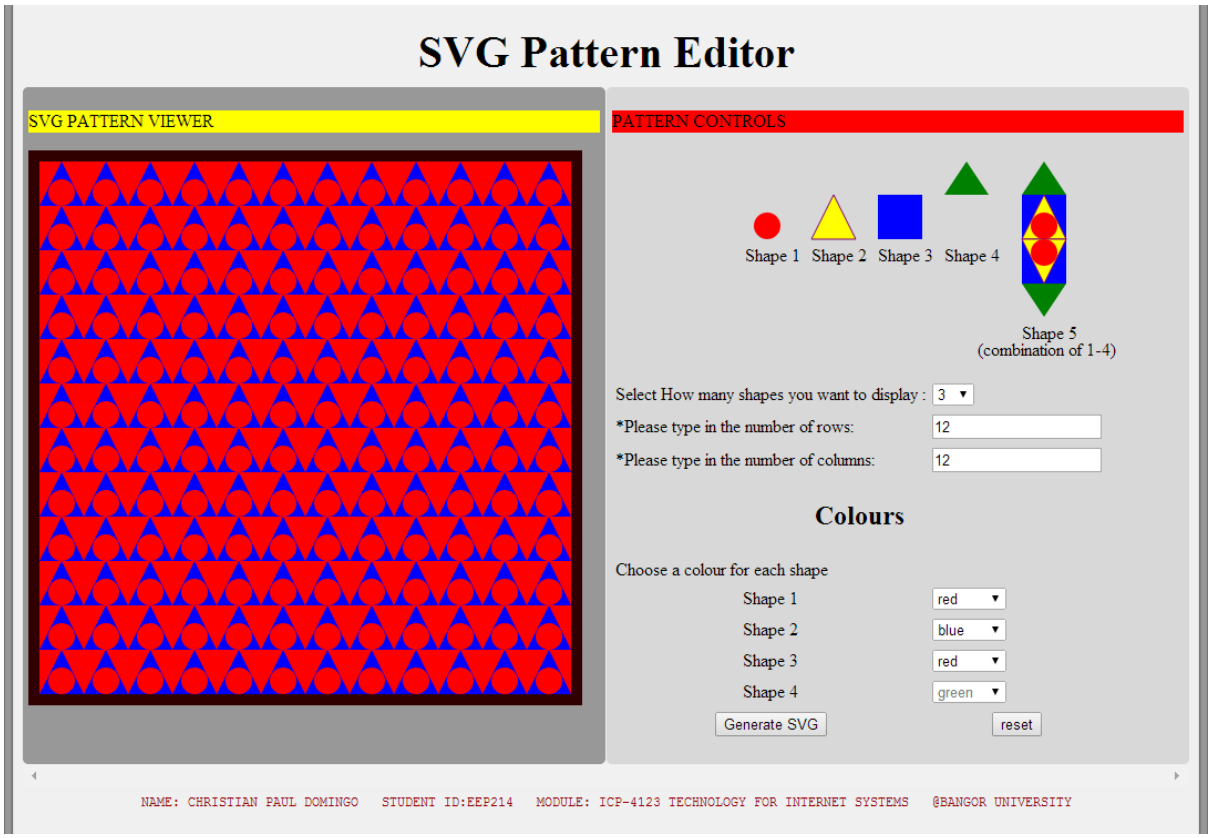
```
var xrf = "http://www.w3.org/1999/xlink";
function useIt(shape, x, y){
	var use = document.createElementNS(NS,"use");

	use.setAttributeNS(null,"x",x);
	use.setAttributeNS(null,"y",y);
	use.setAttributeNS(xrf,"href",shape);
	return use;
}
```

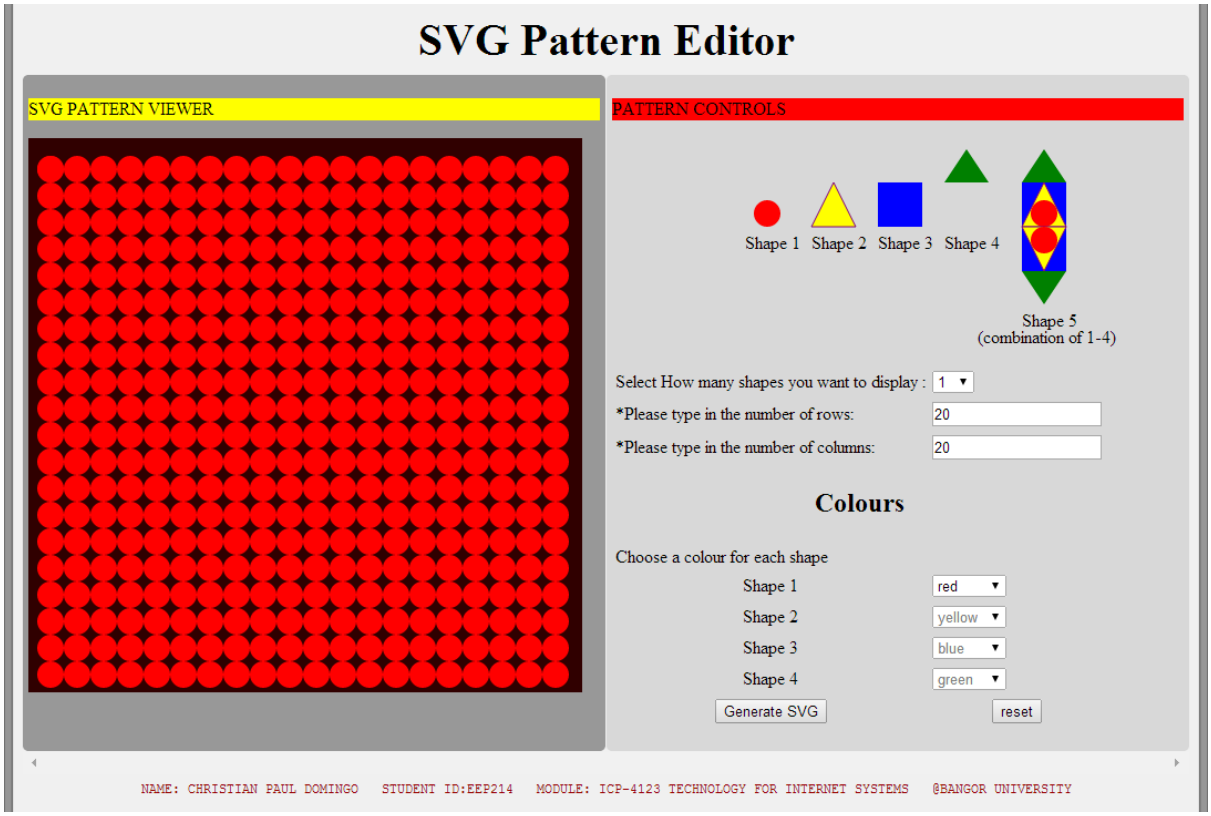
So when you run the my pattern editor the element it generates is the same as what you would see when you created your own svg document

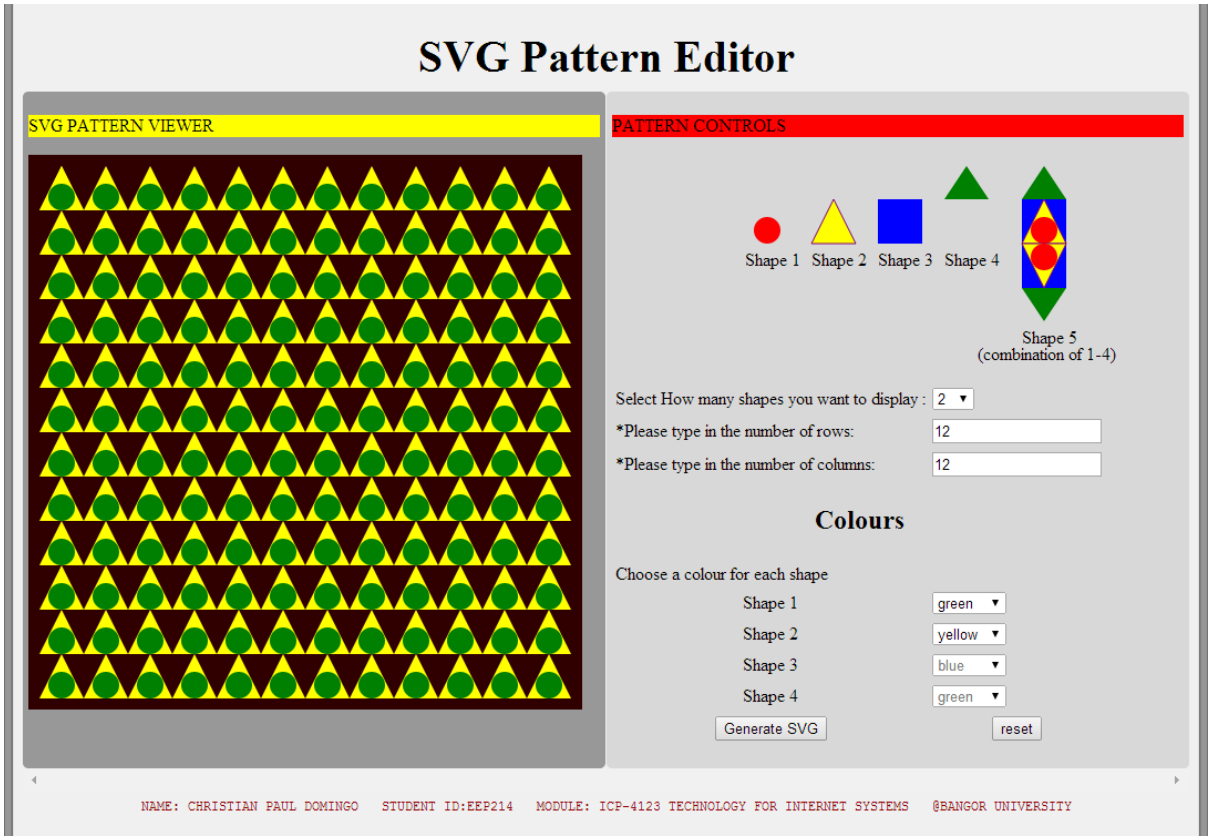


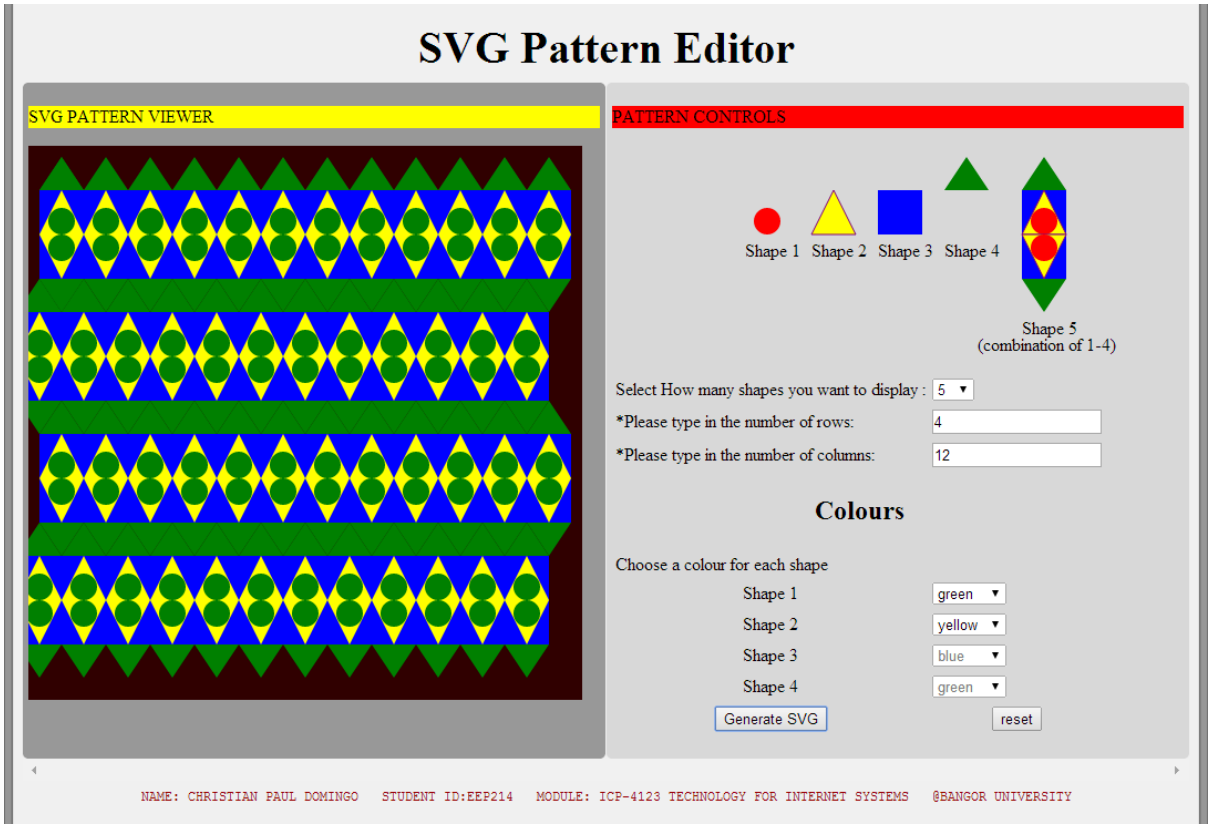
Screen shot

Sample of pattern that can be produced




SVG Pattern Editor
SVG PATTERN VIEWER
PATTERN CONTROLS
Shape 1
Shape 2
Shape 3
Shape 4
Shape 5
(combination of 1-4)
Select How many shapes you want to display : 1
*Please type in the number of rows: 20
*Please type in the number of columns: 20
Colours
Choose a colour for each shape
Shape 1 red
Shape 2 yellow
Shape 3 blue
Shape 4 green
Generate SVG
reset
NAME: CHRISTIAN PAUL DOMINGO STUDENT ID:EEP214 MODULE: ICP-4123 TECHNOLOGY FOR INTERNET SYSTEMS @BANGOR UNIVERSITY


SVG Pattern Editor
SVG PATTERN VIEWER
PATTERN CONTROLS
Shape 1
Shape 2
Shape 3
Shape 4
Shape 5
(combination of 1-4)
Select How many shapes you want to display : 2
*Please type in the number of rows: 12
*Please type in the number of columns: 12
Colours
Choose a colour for each shape
Shape 1 green
Shape 2 yellow
Shape 3 blue
Shape 4 green
Generate SVG
reset
NAME: CHRISTIAN PAUL DOMINGO STUDENT ID:EEP214 MODULE: ICP-4123 TECHNOLOGY FOR INTERNET SYSTEMS @BANGOR UNIVERSITY




## How to use

to be able to use my pattern editor you can just try it out and generate svg already since I've supplied default values in it as you load it.

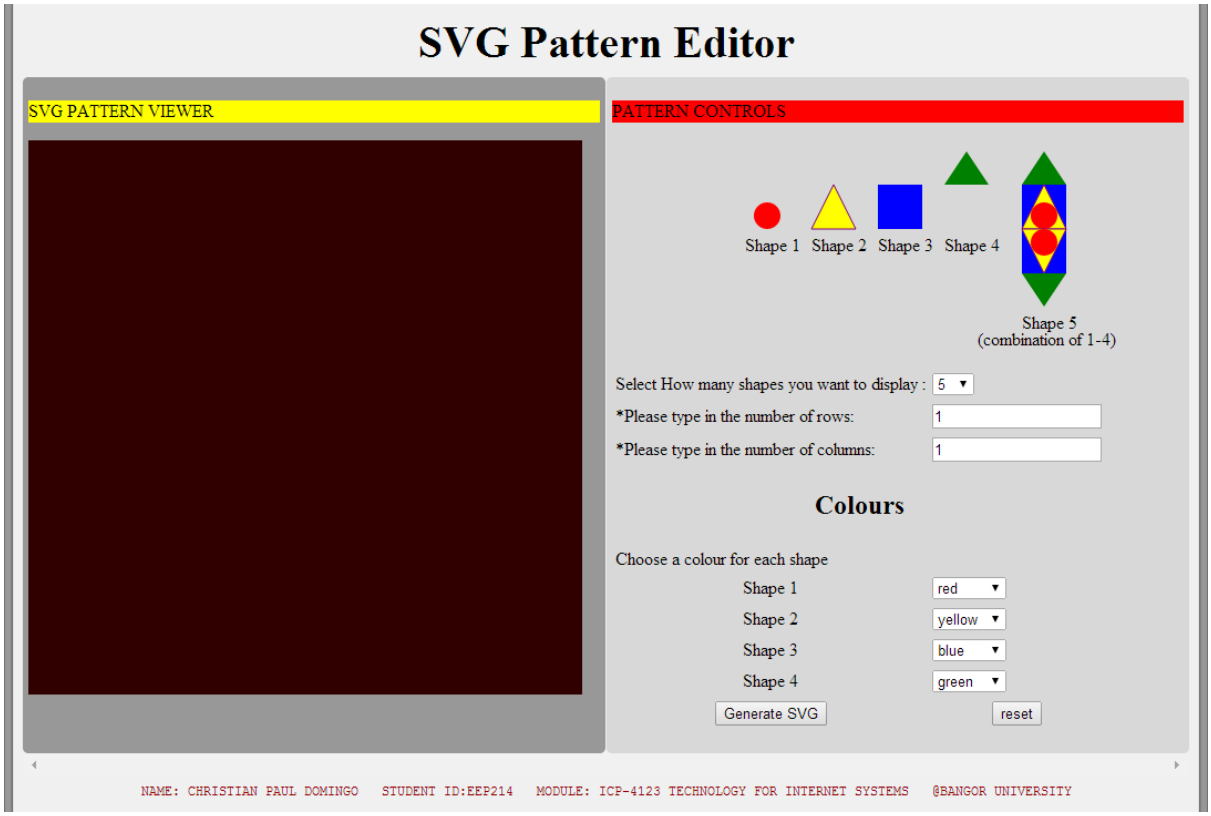
However if you want better control over the shapes being produced.

Just select how many shapes you want to display from the dropdown which is from 1-5.

Then specify how many times you wanted to repeat this shape in rows and in column.

> Warning* bigger shapes such as shape 5 can only be viewed in a limited space because the svg dimension is 500 by 500.

Finally choose a colour for each shape using the dropdown again and press generate SVG button and it should generate an SVG pattern for you




Resources/websites

http://www.i-programmer.info/programming/graphics-and-imaging/3254-svg-javascript-and-the-dom.html

http://www.w3schools.com/svg/svg_inhtml.asp

http://msdn.microsoft.com/en-us/library/gg589525(v=vs.85).aspx