Learning to Code with SVG

Lesson Plan:	Introducing Scalable Vector Graphics (SVG)
Objective:	Hands-on learning of Scalable Vector Graphics (SVG) by coding a mouse in SVG on a 600 by 600 grid with basic shapes; circles and ellipses. Understanding SVG file requirements.
Lab Time:	Approximately 1 hour, not including Lecture time. Students should copy and paste lines, and then change the attribute values to greatly reduce typing time and typos.
Age range:	4-8th grades, or any age student unfamiliar with SVG
Requirements:	A Computer with a browser and a simple text editor like Notepad on Windows, or Text Wrangler (free in the App Store) on a Mac.
Resources:	http://steamcoded.org/lessons/numberedgrid600x600.svg http://www.w3schools.com/svg/default.asp https://www.w3.org/TR/SVG11/ Free eBook for the iPad *Learn SVG Interactively*, by Jay Nick

Lecture: Scalable Vector Graphics (SVG) is a web standard markup language for graphics. Its syntax is very similar to Hyper-Text Markup Language (HTML), a web standard used in all web pages. These standards are agreed upon and voted on by the World Wide Web Consortium (W3C) whose membership includes all the companies making browsers, i.e. Apple. Google, Microsoft, Mozilla, etc. as well as many other companies and individuals.

An SVG file is just text and is identified by its .svg file extension. SVG elements consist of tagnames, attributes, and their values in the form <tagname attribute="value"> Each image file starts with the tagname svg, i.e. <svg> and ends with its closing tag, i.e. </svg>

The <svg> element can have many attributes, but for the purpose of this lesson plan, it includes attributes; width, height, and viewBox. The width and height attributes set the size of the image that will be inserted in the browser's container for the image. For an SVG file, the container is the browser's window. This lesson will set the width="100%" and height="100%" to use the whole browser window. SVG defaults to preserving the aspect ratio of the image, so the browser will reduce either the height or width as needed to keep the image from being distorted.

The viewBox attribute defines the rectangular shape of the image and has the

form viewBox="min-x min-y width height" where the point (min-x,min-y) is the upper left corner of the image and has a width and height. In this lesson, the viewBox="0 0 600 600" which sets the upper left corner of the image to (0,0) and the lower right corner to (600,600).

Additionally, the <svg> element contains some namespace attributes that refer to the W3C specifications for the markup used in the file. Suffice it to say, they are needed. This lesson includes 2 common ones. xmlns="http://www.w3.org /2000/svg" and xmlns:xlink="http://www.w3.org/1999/xlink" The first one is for SVG and the second one is needed anytime there is a reference link used in the SVG elements. While this lesson has none, it is used so often it is better to include it now than leave it out and copy and paste this in future work when it will be needed.

So our minimum requirements for an SVG file are:

```
<svg width="100%" height="100%" viewBox="0 0 600 600"
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink">

</svg>
```

In this lesson <circle> and <ellipse> elements will be added as child elements by adding them between the <svg> and </svg> tags. The <circle> element has attributes; cx, cy, and r where the center of the circle is defined to be at point (cx,cy) and has a radius defined by the r attribute. A circle with a center at point (300,300) with a radius of 100 would look like

```
<circle cx="300" cy="300" r="100" />
```

Similarly, The <ellipse> element has attributes; cx, cy, rx, and ry where the center of the ellipse is defined to be at point (cx,cy), has a x-radius defined by the rx attribute, and has a y-radius defined by the ry attribute. An ellipse with a center at point (300,300) with an x-radius of 100 and a y-radius or 50 would look like

```
<ellipse cx="300" cy="300" rx="100" ry="50" />
```

Note that since the <circle> and <ellipse> elements have no child elements, their closing tag, i.e. </circle> and </ellipse> can be omitted by including a / at the end of the element. This is an alternative way to close the element that is in standard use.

Lastly, the lesson will style the circles and ellipses using another web standard called CSS, short for Cascading Style Sheets. CSS styles consist of name:value

pairs separated by a semi-colon. Each name is separated from its value with a colon. For example, style="name:value;name:value;"

This lesson will add 3 CSS styles, fill, stroke, and stroke-width, where fill paints the inside of the element and stroke draws the outline of the element. The stroke-width sets the thickness of the outline. The fill and stroke styles take colors as their values or the word none. Since the default fill is usually black, this lesson will use the name:value pair fill:none regularly. If omitted, expect to see black where it is not desired. The stroke-width value is a number followed by its units. This lesson will define the stroke-width in pixels, abbreviated px.

Procedure: Start the editor application, i.e. Notepad or Text Wrangler. Then have students get an SVG template with 600x600 grid from: http://steamcoded.org/lessons /numberedgrid600x600.svg.txt
Copy the code and paste it into a text editor.

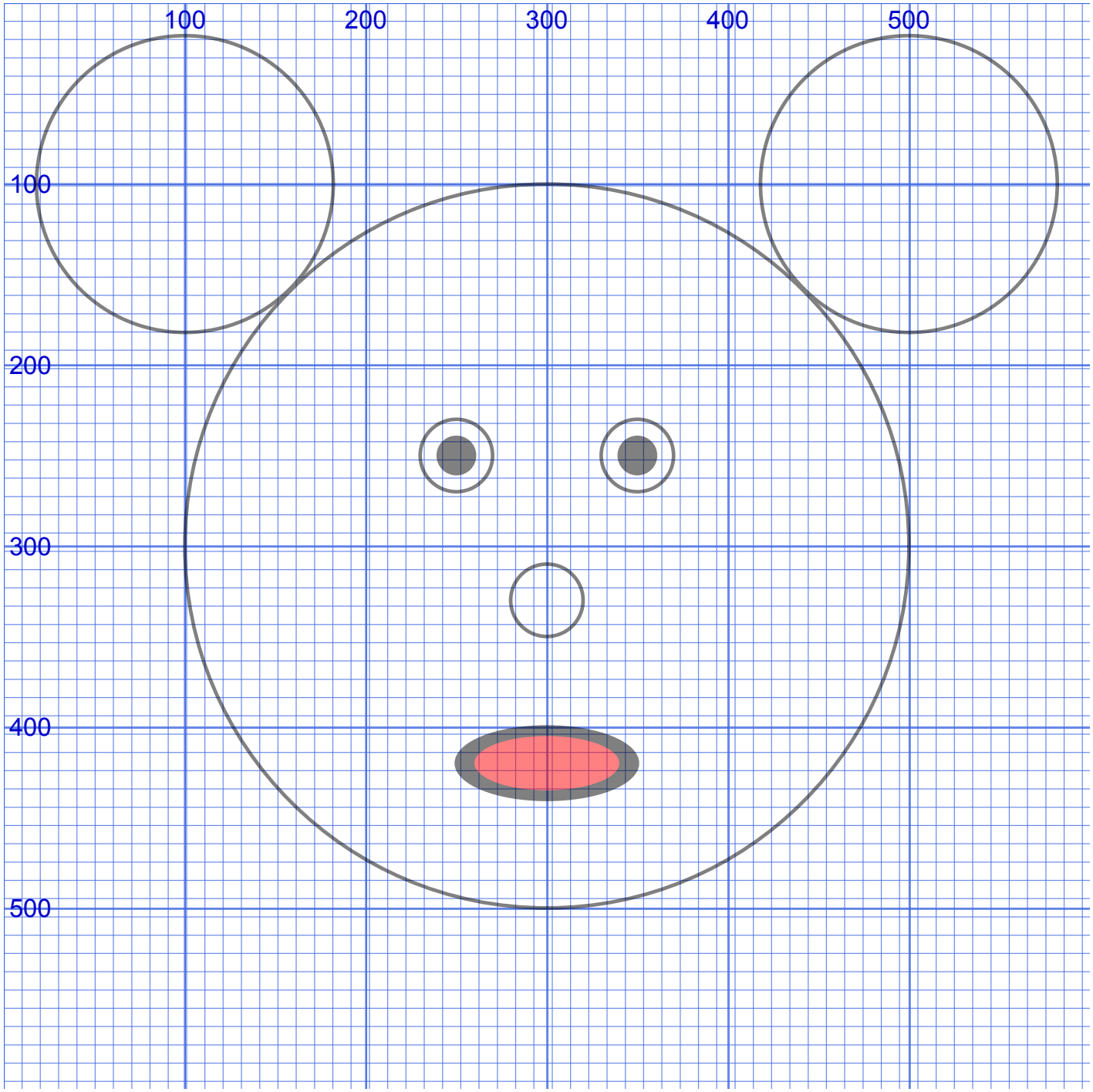
Alternatively, put a copy of the file somewhere that the students have access to and let them copy it from there. Once the code has been copied to the editor, save the file giving the file the name Mouse.svg then open the file in a browser. The browser should display the 600 by 600 grid. Keep the text editor and browser windows open.

The file contains two grouping <g> elements. The first displays a 600 by 600 grid so the students can get a better understanding of the coordinate system of the image and understand why the circles and ellipses are placed where they appear on the screen. The SVG elements the students add should be placed in the second grouping and will be child elements of that <g> element. Note that this grouping has a CSS style of opacity:0.5. This will allow the background grid to show through the elements contained in that group. When the lesson is completed, the style attribute of these 2 groupings will be changed to hide the grid and not allow the background to show through.

Add SVG elements where indicated using the instructions on page 6 (see below). Students should copy and paste lines, and then change the attribute values to greatly reduce typing time and typos. The copy and paste shortcuts are (ctrl-c and ctrl-v on windows and command-c and command-v on a Mac).
Important: Students should save the file and refresh the browser after adding each SVG element to their file.

When complete, change the style attribute of the first <g> element from "display:initial" to "display:none" which hides the grid Then change the style attribute of the second <g> element from "opacity:0.5" to "opacity:1"

Take Away:	Students should understand the minimum requirements of an SVG file, including its coordinate system. Students should feel comfortable adding circles and ellipses to an SVG image and understand CSS styling; fill, stroke, and stroke-wdith.
Additional Activity	Students can change the fill and stroke styling colors to make a more interesting mouse.


100
200
300
400
500
100
200
300
400
500

Coding a Mouse in SVG on a 600 by 600 grid

To get started, copy the code at this link into your editor: http://steamcoded.org/lessons/numberedgrid600x600.svg.txt
Save the file as **Mouse.svg** and open the file in a browser. Keep both applications open.

In the editor, add the SVG elements (per instructions below) where indicated in the SVG code, i.e. inside the second grouping (<g> element).
Important: Save the file and refresh the browser after each step.

Syntax Examples:
An circle centered at (300,300) with a radius of 200 would have the following code:
<circle cx="300" cy="300" r="200" />

An ellipse centered at (300,420) with an x-radius of 50 and a y-radius of 20 would have the following code:
<ellipse cx="300" cy="420" rx="50" ry="20" />

1: Draw a circle at (300,300) with a radius of 200
<circle cx="300" cy="300" r="200" />
2: Draw a circle at (100,100) with a radius of 82
3: Draw a circle at (500,100) with a radius of 82
4: Draw a circle at (250,250) with a radius of 20
5: Draw a circle at (250,250) with a radius of 10 and attribute: style="fill:black;"
<circle cx="250" cy="250" r="10" style="fill:black;" />
6: Draw a circle at (350,250) with a radius of 20
7: Draw a circle at (350,250) with a radius of 10 and attribute: style="fill:black;"
8: Draw a circle at (300,330) with a radius of 20
9: Draw an ellipse at (300,420) with an x-radius of 50 and a y-radius of 20 and attribute: style="fill:black;"
<ellipse cx="300" cy="420" rx="50" ry="20" style="fill:black;" />
10: Draw an ellipse at (300,420) with an x-radius of 40 and a y-radius of 15 and attribute: style="fill:red;stroke:none;"

When complete, change the style attribute of the first element from "display:initial" to "display:none" which hides the grid. Then change the style style attribute of the second element from "opacity:0.5" to "opacity:1"

For added fun, change the fill and stroke colors. There are 16 million colors to choose from, but only about 140 have names. All colors can be entered using a hex number like #FF0000 for red.
For example: style="fill:#00FFFF;stroke:#000000;" is the same as style="fill:aqua;stroke:black;"

Reference: http://www.w3schools.com/colors/colors_names.asp
Note the Hex column. Click on the Shades link for more colors.

Coding a Mouse in SVG on a 600 by 600 grid

Answer Sheet

Common mistakes are missing double quote marks around attribute values, missing space between attributes, missing the start < and ending /> symbols.

```
1:  <circle cx="300" cy="300" r="200" />
2:  <circle cx="100" cy="100" r="82" />
3:  <circle cx="500" cy="100" r="82" />
4:  <circle cx="250" cy="250" r="20" />
5:  <circle cx="250" cy="250" r="10" style="fill:black;" />
6:  <circle cx="350" cy="250" r="20" />
7:  <circle cx="350" cy="250" r="10" style="fill:black;" />
8:  <circle cx="300" cy="330" r="20" />
9:  <ellipse cx="300" cy="420" rx="50" ry="20" style="fill:black;" />
10: <ellipse cx="300" cy="420" rx="40" ry="15" style="fill:red;stroke:none;" />
```