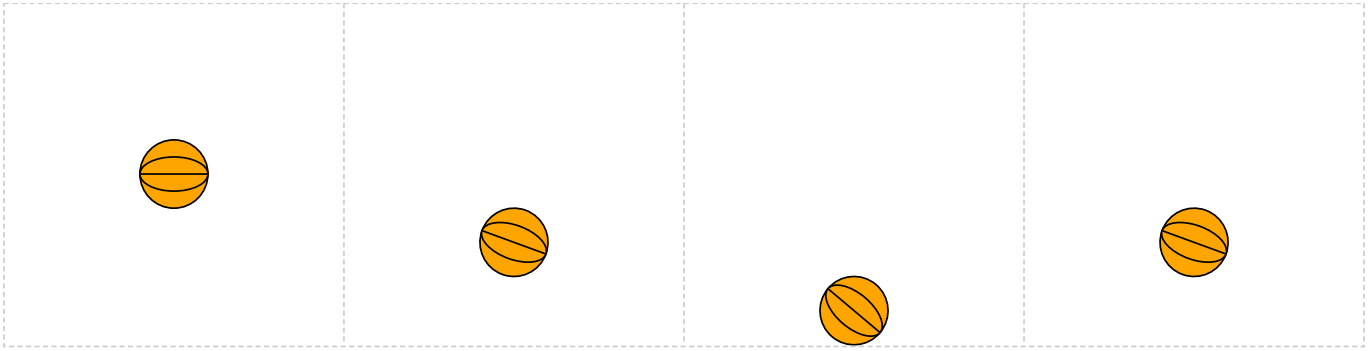
Learning to Code with SVG

Lesson Plan:	Coding a Bouncing Ball in SVG on a 800 by 200 grid
Objective:	Hands-on learning of SVG by drawing a ball symbol with basic shapes; circles, ellipses, and lines. Then creating a sprite sheet using the ball symbol in different positions. Introducing the <script> element and javascript to display frames from the sprite sheet in rapid succession.
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:	4-12th grades, or any age student unfamiliar with SVG and/or Javascript
Requirements:	Familiar with a simple text editor like Notepad on Windows, or Text Wrangler on a Mac. Familiar with copy and paste shortcuts (ctrl-c and ctrl-v on windows and command-c and command-v on a Mac). Ability to save file with a .svg extension. Should be familiar with adding <circle>, <ellipse>, <path>, <symbol>, and <use> elements in SVG
Resources:	http://steamcoded.org/lessons/grid200x200.svg http://www.w3schools.com/svg/default.asp http://www.w3schools.com/js/default.asp http://www.w3schools.com/jsref/dom_obj_document.asp https://www.w3.org/TR/SVG11/ Free eBook for the iPad *Learn SVG Interactively*, by Jay Nick
Lecture:	Review the <symbol> and the <use> elements, the translate and rotate transformations, and using a clip path from previous lessons. Introduce how javascript is added to the SVG image using the <script> element, some basic javascript syntax i.e. use of { } symbols, expressions (usually lines) end with semicolon (forgiving if missing), variables, and how to call a function. The lesson plan will walk through the details. **Important:** Line up lines of code correctly - learn to type clean, readable code. Much easier to find errors.
Procedure:	Give the students the instructions on pages 4-6. Have students create an SVG element with a viewBox="0 0 800 200" in the text editor. Save the file as ball.svg then open in a browser. Keep the text editor and browser windows open. Save and refresh the browser regularly to see if errors were introduced and what effect the code had.
Take Away:	Students should feel comfortable creating SVG images from scratch. Students should better understand the <symbol> and <use> elements and transformations. Students should get a feel for javascript syntax, variables, and functions. Students should understand that SVG elements can be modified with javascript code to create animation.

Additional Activity	Students can change the ball symbol, or change the angles of rotation. Advanced students can create additional frames for the animation, then modify the javascript to display them.

Bouncing Ball Sprite Sheet



Coding a Bouncing Ball in SVG

In a text editor, create an <svg> element with a viewBox from (0,0) to (800,200) and save the file as ball.svg and open the file in a browser. In the editor, add the SVG elements (per instructions below).

1: Create a <defs> element.

2: Add a <symbol> element to the <defs> element. Give the symbol element an id="ball" and a style="fill:orange;stroke:black;"

3: Add 3 elements to the "ball" symbol:
Circle centered at (100,100), radius: 20
Ellipse centered at (100,100), x-radius: 20, y-radius: 10
Path drawing a line from (80,100) to (120,100)

4: Add a <clipPath> element to the <defs> element. Give the clipPath element an id="cp1" and a viewBox="0 0 200 200"

5: Add a <rect> element to the <clipPath> element. Give the rect element an id="cp" starting at x=0, y=0 and a width and height of 200

6: Create a grouping <g> element with id="sprites".

7: Add a <use> element to the "sprites" grouping referencing the ball symbol, i.e. xlink:href="#ball".

8: Add 3 additional <use> elements to the "sprites" grouping each referencing the ball symbol, but transform each one as follows:
translate(200,40) rotate(20,100,100)
translate(400,80) rotate(40,100,100)
translate(600,40) rotate(20,100,100)

9: Now that the 4 frame sprite sheet is complete, add a <script> element after the <defs> element as shown below:

```
<script type="application/x-javascript"><![CDATA[

]]>
</script>
```

The CDATA section tells the browser to ignore symbols contained in it when parsing the SVG elements.

10: Inside the CDATA section, create a javascript variable called SpriteSheetAnimation that will contain the code as follows:

```
var SpriteSheetAnimation = new function()
{

}();
```

Be sure to use proper indentation to help troubleshooting errors like missing or unmatched { } symbols. These symbols group code similar to the way a <g> element groups SVG elements.

11: Inside the SpriteSheetAnimation function, add a line of code:

```
alert("Hello World");
```

Refresh the browser, to see that the code is executing and displaying an alert box. The alert() function is useful when troubleshooting code errors.

12: Add an init() function around the alert box and add an event listener to call the init() function after the SVG image loads.

```
window.addEventListener("load",init);

function init()
{
  alert("Hello World");
}
```

It is important to know that the SVG image has finished loading before executing any javascript code that tries to access any SVG elements - otherwise the SVG element may not be found.

13: Define a variable called frame (first line inside the SpriteSheetAnimation function) that will keep track of which frame is to be displayed.

```
var frame=0;
```

frame will be a 0 relative variable counter, counting the 4 frames as 0 ,1 ,2, and 3.

14: Add an DisplayNextFrame() function after the init() function with the following lines of code:

```
function DisplayNextFrame()
{
  document.documentElement.setAttribute("viewBox",frame*200+" 0 200 200");
  frame++;
  if(frame>3) frame=0;
}
```

The document object, in this case, is the SVG image. Objects have properties and methods, where properties are like variables, and methods are like functions. Access to these properties and methods uses a dot syntax, so the documentElement property of the document object can be accessed with the code "document.documentElement". The documentElement property contains the root element of the document. It is the first element in the document, which in this case is the <svg> element.

The line "frame++;" is a simplified way to increment a variable. It is equivalent to "frame = frame + 1;"

The last line checks if the frame variable is greater than 3 and, if so, sets the frame variable back to 0.

15: In the init() function, comment out the alert() function, and a add line of code:

```
function init()
{
  //alert("Hello World");
  setInterval(DisplayNextFrame,150);
}
```

The javascript function setInterval() takes 2 passed parameters, the name of the function to call when the timing interval expires and the number of milliseconds for the interval. It sets a timer that will call the DisplayNextFrame function every 150 milliseconds.

At this point, the animation works, but overflow is showing the other frames. To solve this, the next step will add a clip-path to display only the desired frame.

16: Add a line of code to the beginning of the init() function:

```
document.getElementById("sprites").setAttribute("style","clip-path:url(#cp1);");
```

The document object has a getElementById() method (function) to find the element with the passed id name, in this case "sprites". Then calls the setAttribute() method (function) which adds a "style" attribute to the "sprites" element. The style attribute contains the name:value pair "clip-path:url(#cp1);", which add the clip path with id="cp1". Since the rectangle element in the clip path is set to the dimentions of the first frame, that

is all that is displayed. To solve this, add the line of code in the next step.

17: Add a line of code as the second line of code in the DisplayNextFrame() function:

```
document.getElementById("cp").setAttribute("x",frame*200);
```

Use the document object getElementById() method (function) to access the element with id="cp". Then call the setAttribute() method (function) to modify the "x" attribute to 0 when frame=0, 200 when frame=1, 400 when frame=2, and 600 when frame=3.

18: To eliminate the flicker, add a line of code in front of the setInterval() function call in the init() function:

```
DisplayNextFrame();
```

This eliminates the 150 millisecond delay before calling the DisplayNextFrame() function. Otherwise the frame is displayed with a viewBox="0 0 800 200" in the first 150 milliseconds.

19: To display the sprite sheet without animation, just comment out the line of code as shown:

```
//window.addEventListener("load",init);
```

With this line of code commented out, the init() and therefore the DisplayNextFrame() don't get called.

Coding a Bouncing Ball in SVG on a 800 by 200 grid

Answer Sheet

Common mistakes are incorrectly creating group elements, closing the opening g tag with a /, missing the closing g tag, missing double quote marks around attribute values, missing space between attributes, missing the start < and ending /> symbols, and typing rbg instead of rgb. The rgb() function stands for red,green,blue and the numbers represent the amount of each color. Values of each color range from 0 to 255.

```
1:    <svg width="100%" height="100%" viewBox="0 0 800 200"
          xmlns="http://www.w3.org/2000/svg"
          xmlns:xlink="http://www.w3.org/1999/xlink">
        <defs>
       </defs>
      <svg>

2-3:  <symbol id="ball" style="fill:orange;stroke:black;">
        <circle cx="100" cy="100" r="20" />
        <ellipse cx="100" cy="100" rx="20" ry="10" />
        <path d="M80,100L120,100" />
      </symbol>

4-5:  <clipPath id="cp1" viewBox="0 0 200 200">
        <rect id="cp" x="0" y="0" width="200" height="200" />
      </clipPath>

6-8:  <g id="sprites">
        <use xlink:href="ball" />
        <use xlink:href="ball" transform="translate(200,40) rotate(20,100,100)" />
        <use xlink:href="ball" transform="translate(400,80) rotate(40,100,100)" />
        <use xlink:href="ball" transform="translate(600,40) rotate(20,100,100)" />
      </g>

9-11: <script type="application/x-javascript"><![CDATA[
        var SpriteSheetAnimation = new function()
        {
            alert("Hello World");
        }();
      ]]>
      </script>

12:   <script type="application/x-javascript"><![CDATA[
        var SpriteSheetAnimation = new function()
        {
            window.addEventListener("load",init);

            function init()
            {
                alert("Hello World");
            }
        }();
      ]]>
      </script>

13-14: <script type="application/x-javascript"><![CDATA[
        var SpriteSheetAnimation = new function()
        {
```

```
        var frame=0;

        window.addEventListener("load",init);

        function init()
        {
            alert("Hello World");
        }

        function DisplayNextFrame()
        {
            document.documentElement.setAttribute("viewBox",frame*200+" 0 200 200");
            frame++;
            if(frame>3) frame=0;
        }
    }();
]]>
</script>
```

15:
```
<script type="application/x-javascript"><![CDATA[
    var SpriteSheetAnimation = new function()
    {
        var frame=0;

        window.addEventListener("load",init);

        function init()
        {
            //alert("Hello World");
            setInterval(DisplayNextFrame,150);
        }

        function DisplayNextFrame()
        {
            document.documentElement.setAttribute("viewBox",frame*200+" 0 200 200");
            frame++;
            if(frame>3) frame=0;
        }
    }();
]]>
</script>
```

16:
```
<script type="application/x-javascript"><![CDATA[
    var SpriteSheetAnimation = new function()
    {
        var frame=0;

        window.addEventListener("load",init);

        function init()
        {
            document.getElementById("sprites").setAttribute("style","clip-path:url(#cp1);");
            //alert("Hello World");
            setInterval(DisplayNextFrame,150);
        }

        function DisplayNextFrame()
        {
            document.documentElement.setAttribute("viewBox",frame*200+" 0 200 200");
```

```
            frame++;
            if(frame>3) frame=0;
        }
    }();
]]>
</script>
```

17:

```
<script type="application/x-javascript"><![CDATA[
    var SpriteSheetAnimation = new function()
    {
        var frame=0;

        window.addEventListener("load",init);

        function init()
        {
            document.getElementById("sprites").setAttribute("style","clip-path:url(#cp1);");
            //alert("Hello World");
            setInterval(DisplayNextFrame,150);
        }

        function DisplayNextFrame()
        {
            document.documentElement.setAttribute("viewBox",frame*200+" 0 200 200");
            document.getElementById("cp").setAttribute("x",frame*200);
            frame++;
            if(frame>3) frame=0;
        }
    }();
]]>
</script>
```

18:

```
<script type="application/x-javascript"><![CDATA[
    var SpriteSheetAnimation = new function()
    {
        var frame=0;

        window.addEventListener("load",init);

        function init()
        {
            document.getElementById("sprites").setAttribute("style","clip-path:url(#cp1);");
            DisplayNextFrame();

            //alert("Hello World");
            setInterval(DisplayNextFrame,150);
        }

        function DisplayNextFrame()
        {
            document.documentElement.setAttribute("viewBox",frame*200+" 0 200 200");
            document.getElementById("cp").setAttribute("x",frame*200);
            frame++;
            if(frame>3) frame=0;
        }
    }();
]]>
</script>
```

```
19:     <script type="application/x-javascript"><![CDATA[
            var SpriteSheetAnimation = new function()
            {
                var frame=0;

                //window.addEventListener("load",init);

                function init()
                {
                    document.getElementById("sprites").setAttribute("style","clip-path:url(#cp1);");
                    DisplayNextFrame();

                    //alert("Hello World");
                    setInterval(DisplayNextFrame,150);
                }

                function DisplayNextFrame()
                {
                    document.documentElement.setAttribute("viewBox",frame*200+" 0 200 200");
                    document.getElementById("cp").setAttribute("x",frame*200);
                    frame++;
                    if(frame>3) frame=0;
                }
            }();
        ]]>
        </script>
```