Unit: 3

HTML5- SVG

Introduction to SVG:

- SVG stands for Scalable Vector Graphics
- SVG is used to define vector-based graphics for the Web
- SVG defines the graphics in XML format
- Every element and every attribute in SVG files can be animated
- SVG is a W3C recommendation
- SVG integrates with other W3C standards such as the DOM and XSL

SVG is a W3C Recommendation:

- SVG 1.0 became a W3C Recommendation on 4 September 2001.
- SVG 1.1 became a W3C Recommendation on 14 January 2003.
- SVG 1.1 (Second Edition) became a W3C Recommendation on 16 August 2011.

SVG Advantages:

Advantages of using SVG over other image formats (like JPEG and GIF) are:

- SVG images can be created and edited with any text editor.
- SVG images can be searched, indexed, scripted, and compressed.
- SVG images are scalable.
- SVG images can be printed with high quality at any resolution.
- SVG images are zoomable.
- SVG graphics do NOT lose any quality if they are zoomed or resized.
- SVG is an open standard.
- SVG files are pure XML.

Viewing SVG Files:

Most of the web browsers can display SVG just like they can display PNG, GIF, and JPG. Internet Explorer users may have to install the Adobe SVG Viewer to be able to view SVG in the browser.

Embedding SVG in HTML5:

HTML5 allows embedding SVG directly using **<svg>...</svg>** tag which has following simple syntax –

```
<svg xmlns = "http://www.w3.org/2000/svg">
   ...
</svg>
```

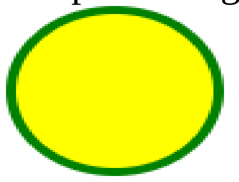
Firefox 3.7 has also introduced a configuration option ("about:config") where you can enable HTML5 using the following steps −

- Type **about:config** in your Firefox address bar.
- Click the "I'll be careful, I promise!" button on the warning message that appears (and make sure you adhere to it!).
- Type **html5.enable** into the filter bar at the top of the page.
- Currently it would be disabled, so click it to toggle the value to true.

Now your Firefox HTML5 parser should be enabled and you should be able to experiment with the following examples.

Embed SVG Directly Into HTML Pages:

You can embed SVG elements directly into your HTML pages.
Here is an example of a simple SVG graphic:



and here is the HTML code:

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>My first SVG</h1>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-
width="4" fill="yellow" />
</svg>

</body>
</html>
```

SVG Code explanation:

- An SVG image begins with an <svg> element
- The width and height attributes of the <svg> element define the width and height of the SVG image
- The <circle> element is used to draw a circle
- The cx and cy attributes define the x and y coordinates of the center of the circle. If cx and cy are not set, the circle's center is set to (0, 0)
- The r attribute defines the radius of the circle
- The stroke and stroke-width attributes control how the outline of a shape appears. We set the outline of the circle to a 4px green "border"
- The fill attribute refers to the color inside the circle. We set the fill color to yellow
- The closing </svg> tag closes the SVG image

Note: Since SVG is written in XML, all elements must be properly closed!

1) SVG Circle:

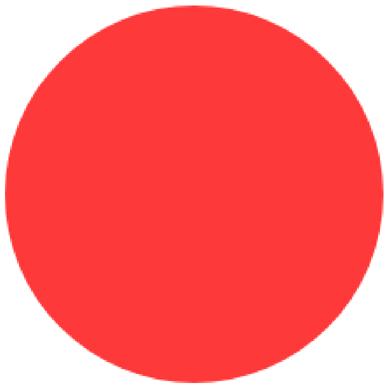
To draw a circle inside HTML SVG, you should use the *<circle>* element inside your SVG.

Inside this element, you need to set these attributes –

- *cx* – The x-coordinate of the center of the circle
- *cy* – The y-coordinate of the center of the circle
- *r* – The radius of the circle

In the below example, the radius of the circle is 30% relative to the SVG so if you change the width and height values of SVG, the area of the circle will also change.

To fix the radius irrespective of the SVG width and height values, you can use a fixed value of radius like **r:”120″.**



Syntax:

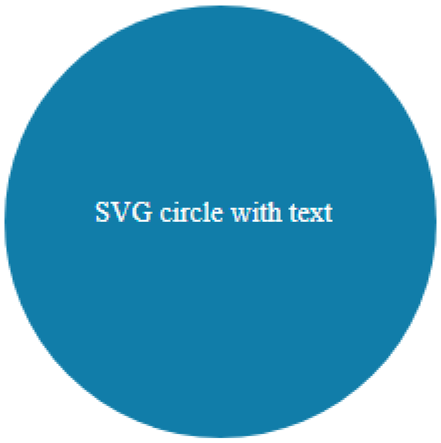
```
<svg>
  <circle cx=”x-center-val” cy=”y-center-val” r=”radius-val”/>
</svg>
```

Example

```
<h2>Example for SVG Circle </h2>
<svg height="400" width="400">
 <circle cx="180" cy="180" r="30%" fill="#fe3939" />
</svg>
```

SVG Circle with Text:

Let us combine the *<circle>* and *<text>* element to create text inside the circle of SVG.



Example

```
<h2>Example of SVG Circle with Text </h2>
<svg height="400" width="400">
  <circle cx="120" cy="120" r="30%" fill="#117da9" />
  <text x="50" y="120" fill="white">SVG circle with text</text>
</svg>
```

SVG Half Circle

Just set the width and height of the SVG and keep on changing the values of *cx*, *cy* and *r* values in such a way that the value of *cx*, *cy* and *r* should not be enough to create a full circle.

Example

```
<svg height="400" width="100">
 <circle cx="120" cy="120" r="30%" fill="navy" />
</svg>
```

SVG Stroke

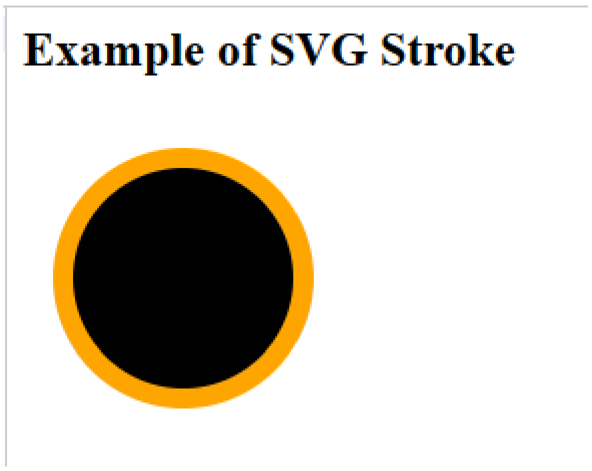
The *stroke* property adds a stoke color(outline color). Optionally, you can use the *stroke-width* property to set the width of the stroke.

Syntax:

```
<svg stroke="stroke-color" stroke-width="s-width">
   <SVG elements…>
</svg>
```

Example

```
<h2>Example of SVG Stroke</h2>
<svg width="200" height="200" stroke-width="10" stroke="orange">
 <circle cx="80" cy="80" r="60"/>
</svg>
```



2) SVG Line

The *<line>* element creates line inside your SVG.
Set these attributes –

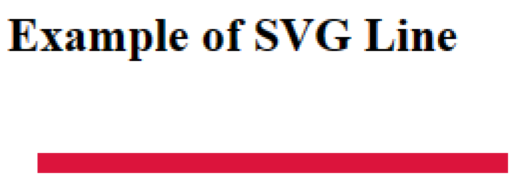
- *x1* – The x-coordinate of the starting point of a line
- *y1* – The y-coordinate of the starting point of a line
- *x2* – The x-coordinate of the ending point of a line
- *y2* – The y-coordinate of the ending point of a line
- *stroke* – To create a stroke

Syntax:

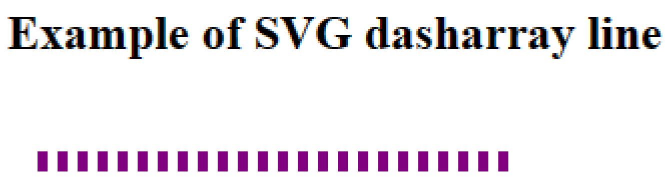
```
<svg>
<line x1=”x-start-pos” y1=”y-start-pos”
    x2=”x-end-pos” y2=”y-end-pos”
    stroke=”crimson”/>
</svg>
```

Example

```
<svg height="50" width="300">
 <line x1="250" y1="30" x2="15" y2="30" stroke-width="10"
stroke="crimson"/>
</svg>
<h2>Example of SVG dasharray line </h2>
<svg height="150" width="300">
 <line x1="250" y1="30" x2="15" y2="30" stroke-width="10"
stroke="purple"
  stroke-dasharray="5,5"/>
</svg>
```



Example of SVG Line



Example of SVG dasharray line

3) HTML5 SVG Rectangle

To draw a Rectangle inside SVG, you should use the *<rect>* element inside your SVG.
Set these attributes –

- *x* – The x-coordinate of the top left corner
- *y* – The y-coordinate of the top left corner
- *width* – The width of the rectangle
- *height* – The height of the rectangle

In the below example, the *<fill>* is optional. This sets the color of the rectangle as darkred.

Syntax:

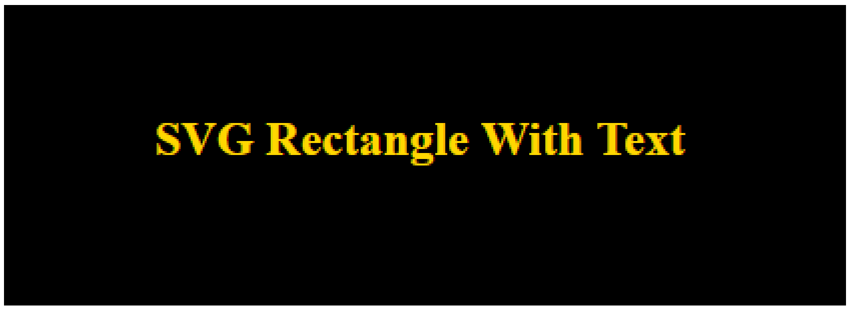
```
<svg>
   <rect x=”x-top-left” y=”y-top-left” width=”w-val” height=”h-val” />
</svg>
```

Example

```
<svg height="300" width="300">
 <rect x="20" y="20" width="200" height="100" fill="darkred" />
</svg>
```

SVG Rectangle with Text

Now, we are going to combine the *<rect>* element with the *<text>* element to create text inside the rectangle of SVG.



Example

```
<svg height="300" width="300">
 <rect x="20" y="20" width="300" height="100" />
 <text x="70" y="70" fill="gold" font-weight="bold">SVG Rectangle With
Text</text>
</svg>
```

SVG Rounded Rectangle

Just code the *<rx>*, *<ry>* or both the elements to create rounded corners for X point and y point respectively.

Example

```
.flexbox-container17aa {
 <svg height="300" width="300">
  <rect x="20" y="20" width="200" height="100" rx="20" fill="purple" />
</svg>
```

SVG Text

To draw HTML5 SVG containing Texts, you should use the *<text>* element inside your SVG.
Set these attributes –

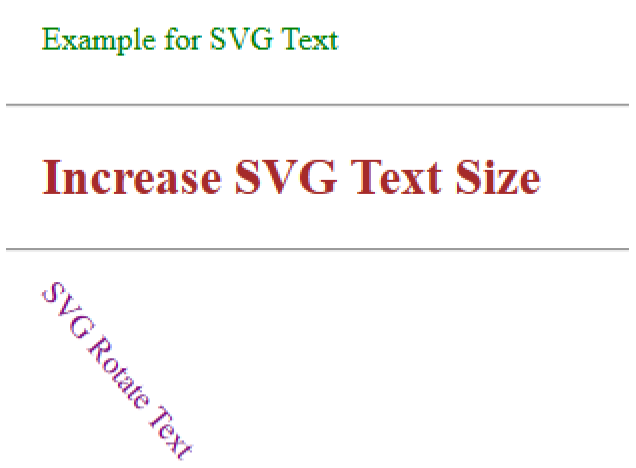
- *x* – The x-coordinate of the starting point of the text
- *y* – The y-coordinate of the starting point of the text
- Setting the width and height of the SVG is optional but often a good idea

Syntax:

```
<svg>
  <text x="x-pos" y="y-pos"> Text… </text>
</svg>
```

Example

```
<svg height="50" width="300" >
  <text x="20" y="35" fill="green">Example for SVG Text</text>
</svg>
```



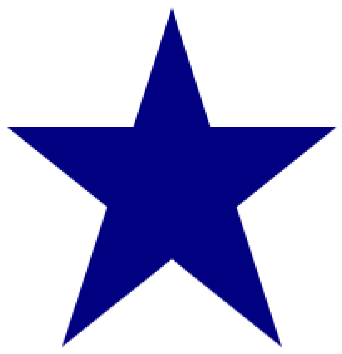
Example for SVG Text

Increase SVG Text Size

SVG Rotate Text

4) SVG Star

Let us create a star using the SVG *<polygon>* element. In this case, you have to set the x & y coordinate of the 1st, 2nd, 3rd, 4th and 5th angle.



Syntax:

```
<svg>
<polygon points=”x1,y1  x2,y2  x3,y3  x4,y4  x5,y5 “/>
</svg>
```

Example

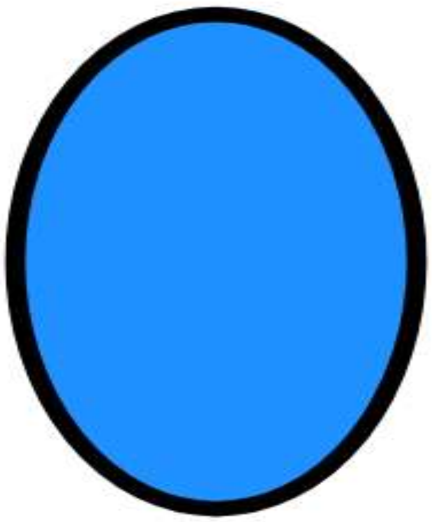
```
<svg height="300" width="300">
 <polygon points="100,15 40,200 190,80, 10,80 160,200" fill="navy" />
</svg>
```

5) SVG Ellipse

The SVG *<ellipse>* element create ellipse.
To create the SVG, you must set the center of the ellipse and the radius along x and y axis.

Set these attributes –

- *cx* – X-coordinate of the center of the ellipse
- *cy* – Y-coordinate of the center of the ellipse
- *rx* – Radius of the ellipse along X axis
- *ry* – Radius of the ellipse along Y axis



Syntax:

```
<svg>
   <ellipse cx=”x-center” cy=”y-center”
rx=”x-radius” ry=”y-radius”/>
</svg>
```

Example

```
<h2>Example of SVG ELLIPSE </h2>
<svg height="200" width="300">
 <ellipse cx="150" cy="100" rx="50" ry="80" fill="dodgerblue"
stroke="black" stroke-width="5"/>
</svg>
```

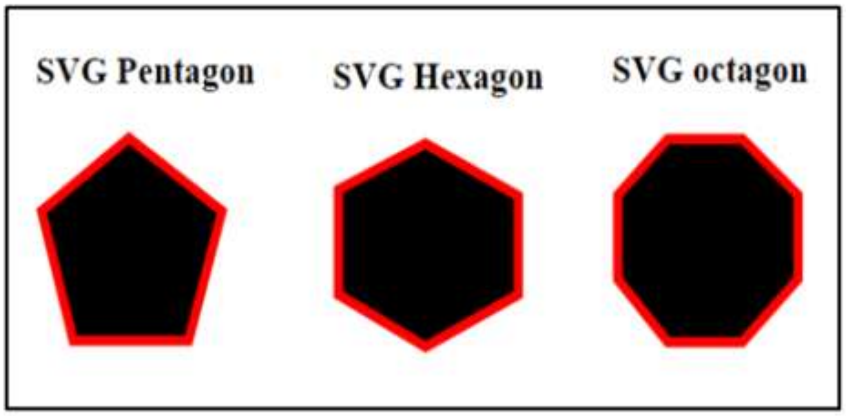
6) SVG Polygon

Now, we are going to create multiple ***SVG Polygon*** objects like a pentagon, Hexagon, and Octagon.

- To create **Pentagon**, set the x,y coordinates of **5** angles.
- For **Hexagon**, set the x,y coordinates of **6** angles.
- And for **Octagon**, we need set x,y coordinates of **8** angles.

Example

```
<h2>Example of SVG POLYGONS </h2>
<h3>SVG Pentagon</h3>
<svg height="120" width="300">
 <polygon points="50 3, 100 38, 82 100, 20 100, 3 38" stroke="red" stroke-
width="5" />
</svg>
```



Example:

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Example of SVG POLYGONS</h2>
    <h3>SVG Pentagon</h3>
    <svg height="120" width="300">
      <polygon points="50 3, 100 38, 82 100, 20 100, 3 38" stroke="red"
stroke-width="5" />
    </svg>
    <hr>
    <h3>SVG Hexagon</h3>
    <svg height="120" width="300">
      <polygon points="50 3, 100 28, 100 75, 50 100, 3 75, 3 25" stroke="red"
stroke-width="5" />
    </svg>
    <hr>
    <h3>SVG octagon</h3>
    <svg height="300" width="300">
      <polygon points="30 3, 70 3, 100 30, 100 70, 70 100, 30 100, 3 70, 3 30"
stroke="red" stroke-width="5" />
    </svg>
  </body>
</html>
```

7) SVG Polyline

The meaning of **Poly** is **multiple** so Polyline means multiple connected lines.

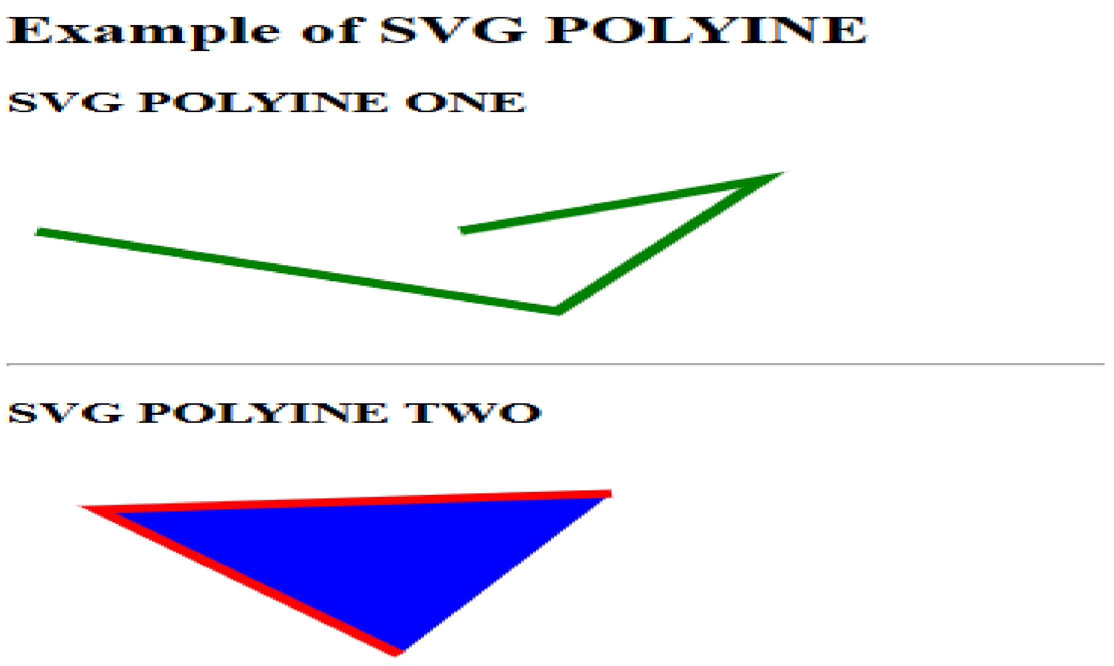
It is easy to create SVG Polyline using the *<polyline>* element. You must set the x & y coordinate of each connected multiple points of the line.

Syntax:

```
<svg>
   <polyline points=”x1,y1  x2,y2  …  xn,yn“/>
</svg>
```

Example

```
<svg height="120" width="300">
 <polyline points="150 50, 250 18, 182 100, 10 50" fill="none"
stroke="green" stroke-width="5" />
</svg>
```



8) SVG Gradients

The SVG element can help to create a smooth texture of colors in a linear way which is called a linear SVG gradient. This must contain an **id** attribute to refer it later.

We can create a vertical gradient as well as a horizontal gradient. Another way is to create a SVG gradient in the same direction as to how SVG is defined.

The must always be nested inside the element. The element helps to embed the properties and definition of an SVG object that can be reused inside an SVG object.

There are 2 types which decide the stopping offset position to create linear SVG gradients. Each contains a to apply a particular color.

Optionally, you can set for each to set the opacity between the stop points.

It is better to set these attributes for the element –

- *x1* – X-coordinate of the starting point of the SVG to define the SVG linear gradient
- *y1* – Y-coordinate of the starting point of the SVG to define the linear gradient
- *x2* – X-coordinate of the end point of the SVG to define the linear gradient
- *y2* – Y-coordinate of the end point of the SVG to define the linear SVG gradient

Similarly, the radial gradient creates textures of colors in a circular way. The SVG is used for this purpose. This element should also contain an **id** attribute to refer it later. In this case –

- *x1* – X-coordinate of the center of the SVG radial gradient
- *y1* – Y-coordinate of the center of the radial gradient
- *x2* – X-coordinate of the focal point of the radial gradient
- *y2* -Y-coordinate of the focal point of the radial gradient

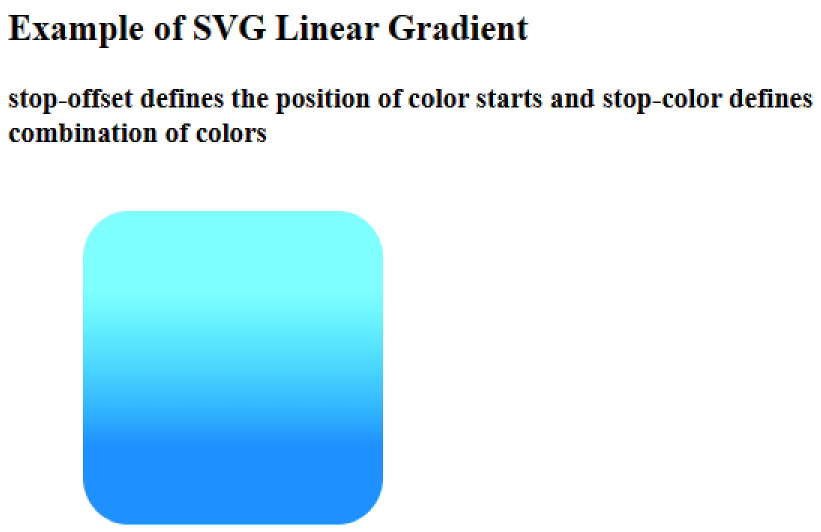
Example 1:

```
<svg height="250" width="300" >
 <defs>
  <linearGradient id="gradval" x1="0%" y="0" x2="0%" y2="100%">
   <stop offset="25%"  stop-color="cyan" stop-opacity="0.5" />
   <stop offset="75%" stop-color="dodgerblue" />
  </linearGradient>
 </defs>
 <rect x="50" y="20" width="200" height="200" rx="30"
fill="url(#gradval)"/>
</svg>
```

Example 2:

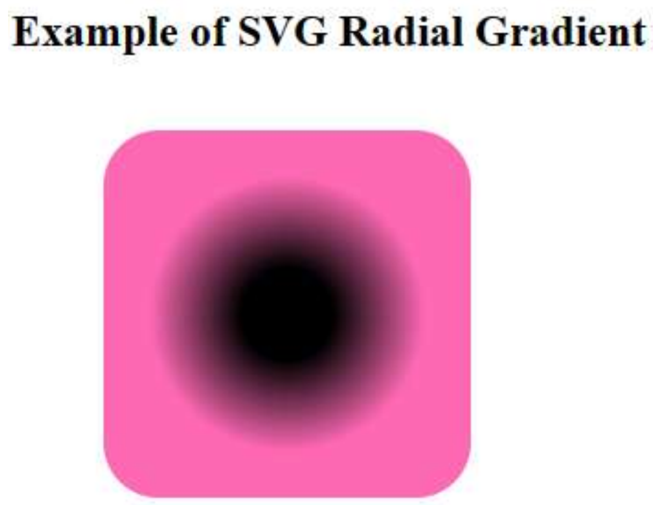
```
<!DOCTYPE html>
<html>
  <body>
    <h2>Example of SVG Linear Gradient </h2>
    <h3>stop-offset defines the position of color starts and stop-color defines combination of
colors</h3>
    <svg height="250" width="300" >
      <defs>
        <linearGradient id="gradval" x1="0%" y1="0" x2="0%" y2="100%">
          <stop offset="25%"  stop-color="cyan" stop-opacity="0.5"/>
          <stop offset="75%" stop-color="dodgerblue" />
        </linearGradient>
      </defs>
      <rect x="50" y="20" width="200" height="200" rx="30" fill="url(#gradval)"/>
    </svg>

    <hr>

    <h2>Example of SVG Radial Gradient </h2>
    <svg height="300" width="300" >
      <defs>
        <radialGradient id="gradval1" x1="0%" y1="0" x2="0%" y2="100%">
          <stop offset="25%"  stop-color="black" />
          <stop offset="75%" stop-color="hotpink" />
        </radialGradient>
      </defs>
      <rect x="50" y="20" width="200" height="200" rx="30" fill="url(#gradval1)"/>
    </svg>
  </body>
</html>
```

OUTPUT:



Example of SVG Linear Gradient

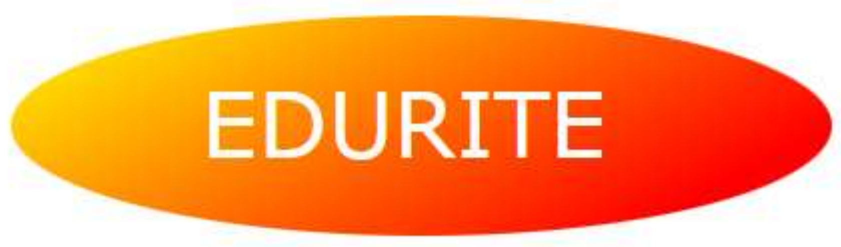
stop-offset defines the position of color starts and stop-color defines combination of colors

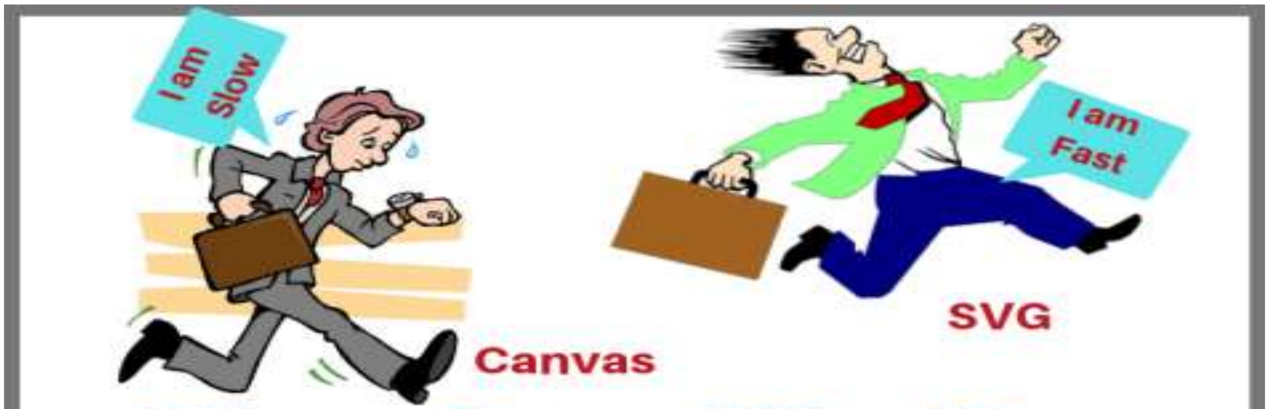


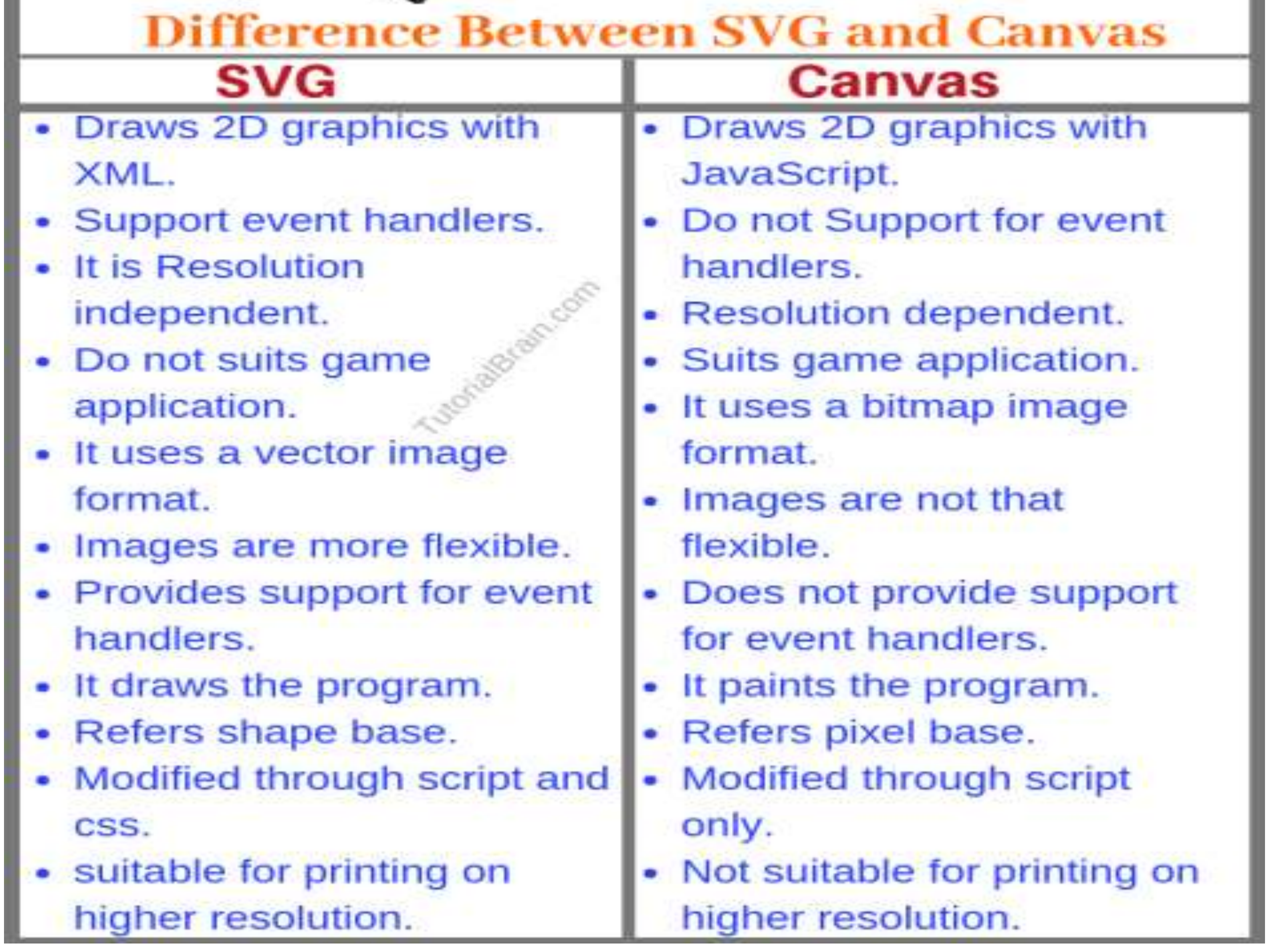
Example of SVG Radial Gradient

9) SVG Logo

```
<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-
family="Verdana" x="50" y="86">Edurite</text>

</svg>
```







Difference Between SVG and Canvas

SVG	Canvas
Draws 2D graphics with XML.	Draws 2D graphics with JavaScript.
Support event handlers.	Do not Support for event handlers.
It is Resolution independent.	Resolution dependent.
Do not suits game application.	Suits game application.
It uses a vector image format.	It uses a bitmap image format.
Images are more flexible.	Images are not that flexible.
Provides support for event handlers.	Does not provide support for event handlers.
It draws the program.	It paints the program.
Refers shape base.	Refers pixel base.
Modified through script and css.	Modified through script only.
suitable for printing on higher resolution.	Not suitable for printing on higher resolution.

******************************END OF UNIT-3*************************************