04. D3JS SVG Transformations.

SVG stands for Scalable Vector Graphics. SVG is an XML-based vector graphics format. It provides options to draw different shapes such as Lines, Rectangles, Circles, Ellipses, etc. Hence, designing visualizations with SVG gives you more power and flexibility.

4.1. SVG Features

Some of the salient features of SVG are as follows:

- SVG is a vector based image format and it is text-based.
- SVG is similar in structure to HTML.
- SVG can be represented as a Document object model.
- SVG properties can be specified as attributes.
- SVG should have absolute positions relative to the origin (0, 0).
- SVG can be included as is in the HTML document.

4.2. SVG Minimal Example

Let us create a minimal SVG image and include it in the HTML document.

Example 1: Create a SVG image and set width as 300 pixel and height as 300 pixel.

```
<svg width = "300" height = "300">
</svg>
```

Here, the svg tag starts an SVG image and it has width and height as attributes. The default unit of the SVG format is pixel.

Example 2: Create a line starting at (100, 100) and ending at (200, 100) and set red color for the line.

```
<line x1 = "100" y1 = "100" x2 = "200" y2 = "200"
   style = "stroke:rgb(255,0,0);stroke-width:2"/>
```

Here, the line tag draws a line and its attributes x1, y1 refers to the starting point and x2, y2 refers to the ending point. The style attribute sets color and thickness of the line using the stroke and the stroke-width styles.

- x1 − This is the x-coordinate of the first point.
- y1 − This is the y-coordinate of the first point.
- x2 − This is the x-coordinate of the second point.
- y2 − This is the y-coordinate of the second point.
- stroke − Color of the line.
- stroke-width − Thickness of the line.

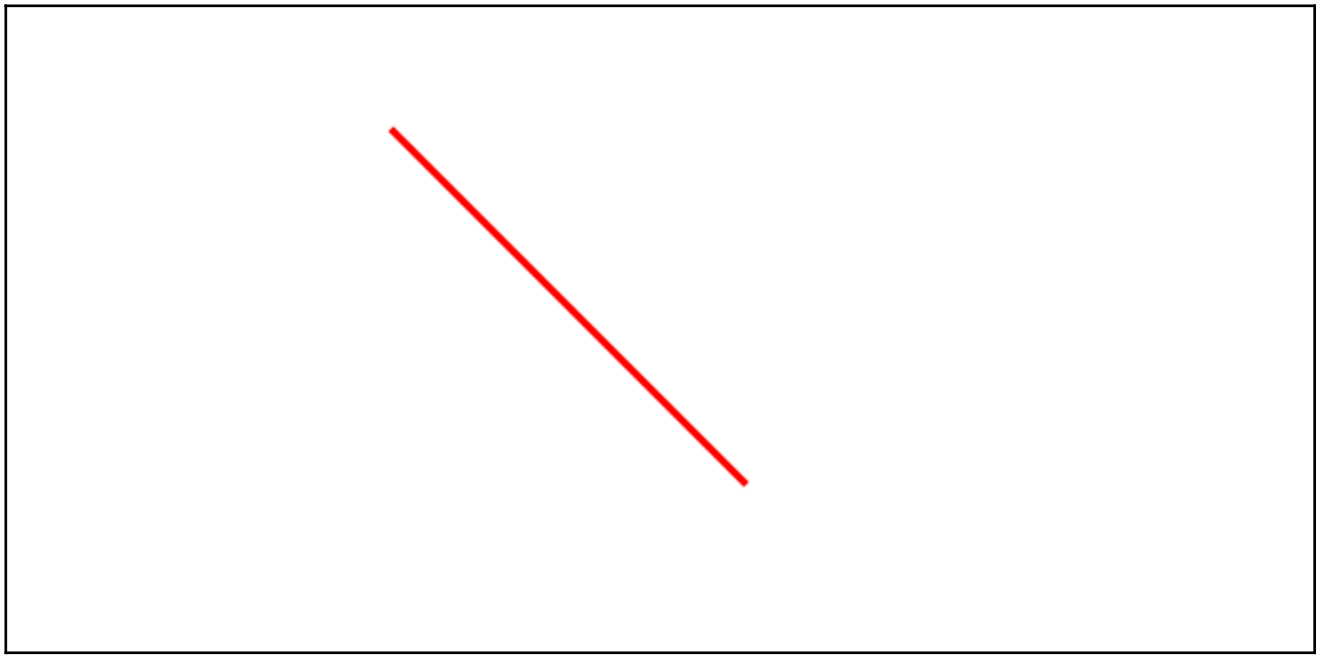
Example 3: Create a HTML document, "svg_line.html" and integrate the above SVG as shown below.

```
<!DOCTYPE html>
<html>
   <head>
      <script type = "text/javascript" src =
"https://d3js.org/d3.v4.min.js"></script>
      <style>
         body { font-family: Arial; }
      </style>
   </head>

   <body>
```

```
            <div id = "svgcontainer">
               <svg width = "300" height = "300">
                  <line x1 = "100" y1 = "100"
                     x2 = "200" y2 = "200" style = "stroke:rgb(255,0,0);
                     stroke-width:2"/>
               </svg>
            </div>
            <p></p>
            <p></p>
         </body>
      </html>
```

The above program will yield the following result.

Output:



4.3. SVG Using D3.js

To create SVG using D3.js, let us follow the steps given below.

Step 1: Create a container to hold the SVG image as given below.

```
<div id = "svgcontainer"></div>
```

Step 2: Select the SVG container using the select() method and inject the SVG element using the append() method. Add the attributes and styles using the attr() and the style() methods.

```
var width = 300;
var height = 300;
var svg = d3.select("#svgcontainer")
   .append("svg").attr("width", width).attr("height", height);
```

Step 3: Similarly, add the line element inside the svg element as shown below.

```
svg.append("line")
   .attr("x1", 100)
   .attr("y1", 100)
   .attr("x2", 200)
   .attr("y2", 200)
   .style("stroke", "rgb(255,0,0)")
   .style("stroke-width", 2);
```

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script type = "text/javascript" src =
"https://d3js.org/d3.v4.min.js"></script>
      <style>
         body { font-family: Arial; }
```

```
            </style>
        </head>

        <body>
            <div id = "svgcontainer">
            </div>
            <script language = "javascript">
                var width = 300;
                var height = 300;
                var svg = d3.select("#svgcontainer")
                    .append("svg")
                    .attr("width", width)
                    .attr("height", height);
                svg.append("line")
                    .attr("x1", 100)
                    .attr("y1", 100)
                    .attr("x2", 200)
                    .attr("y2", 200)
                    .style("stroke", "rgb(255,0,0)")
                    .style("stroke-width", 2);
            </script>
        </body>
    </html>
```

4.4. Rectangle Element

A rectangle is represented by the <rect> tag as shown below.

Example:

```
<rect x = "20" y = "20" width = "300" height = "300"></rect>
```

The attributes of a rectangle are as follows −

- x − This is the x-coordinate of the top-left corner of the rectangle.
- y − This is the y-coordinate of the top-left corner of the rectangle.
- width − This denotes the width of the rectangle.
- height − This denotes the height of the rectangle.

A simple rectangle in SVG is defined as explained below.

Example:

```
<svg width = "300" height = "300">
   <rect x = "20" y = "20" width = "300" height = "300" fill =
"green"></rect>
</svg>
```

The same rectangle can be created dynamically as described below.

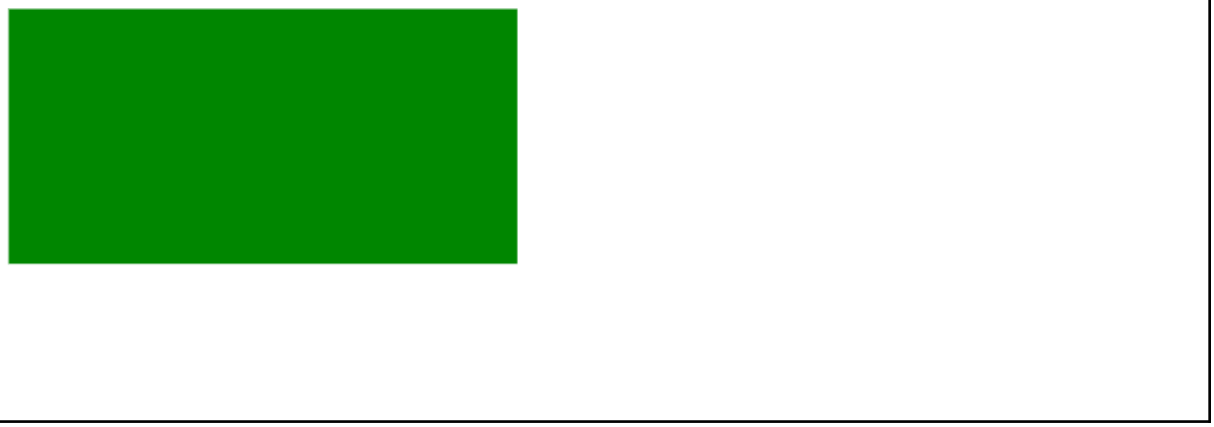
Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script type = "text/javascript" src =
"https://d3js.org/d3.v4.min.js"></script>
   </head>

   <body>
      <div id = "svgcontainer"></div>
      <script>
         var width = 300;
         var height = 300;
         //Create SVG element
```

```
            var svg = d3.select("#svgcontainer")
                .append("svg")
                .attr("width", width)
                .attr("height", height);
            //Create and append rectangle element
            svg.append("rect")
                .attr("x", 20)
                .attr("y", 20)
                .attr("width", 200)
                .attr("height", 100)
                .attr("fill", "green");
        </script>
    </body>
</html>
```

The above code will yield the following result.

Output:



4.5. Circle Element

A circle is represented by the <circle> tag as explained below.

Example:

```
<circle cx = "200" cy = "50" r = "20"/>
```

The attributes of circle are as follows :

- cx − This is the x-coordinate of the center of the circle.
- cy − This is the y-coordinate of the center of the circle.
- r − This denotes the radius of the circle.

Example: A simple circle in SVG is described below.

```
<svg width = "300" height = "300">
   <circle cx = "200" cy = "50" r = "20" fill = "green"/>
</svg>
```

The same circle can be created dynamically as described below.

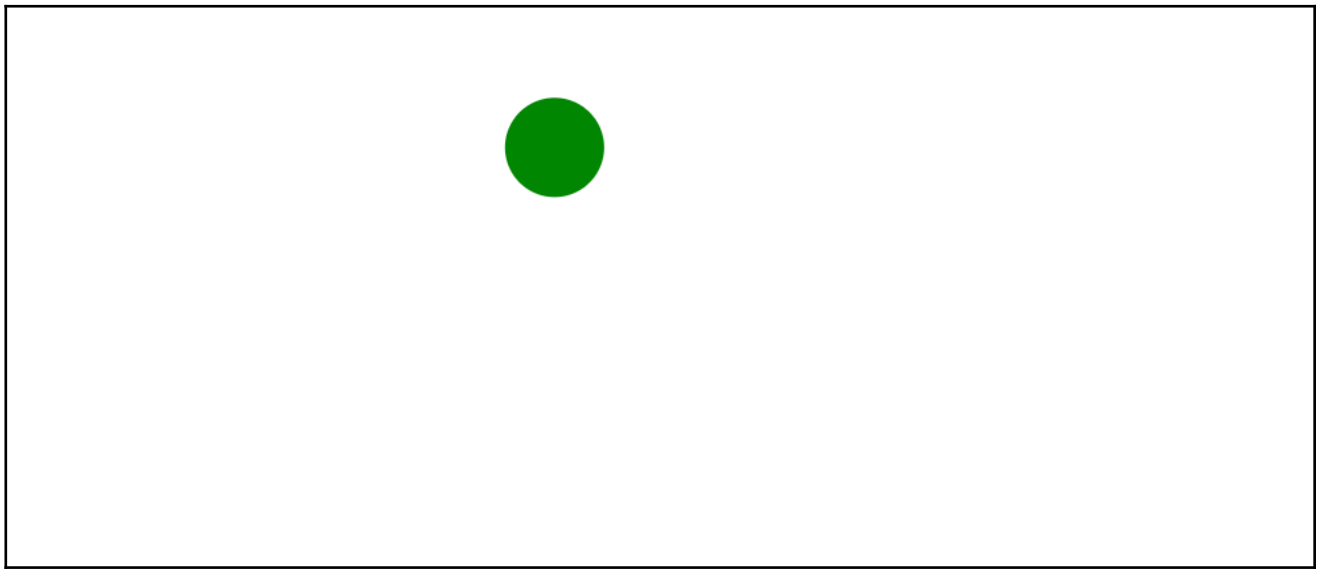
Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script type = "text/javascript" src =
"https://d3js.org/d3.v4.min.js"></script>
   </head>

   <body>
      <div id = "svgcontainer"></div>
      <script>
         var width = 300;
```

```
            var height = 300;
            //Create SVG element
            var svg = d3.select("#svgcontainer")
                .append("svg")
                .attr("width", width)
                .attr("height", height);
            //Append circle
            svg.append("circle")
                .attr("cx", 200)
                .attr("cy", 50)
                .attr("r", 20)
                .attr("fill", "green");
        </script>
    </body>
</html>
```

The above code will yield the following result.

Output:



4.6. Ellipse Element

The SVG Ellipse element is represented by the <ellipse> tag as explained below.

Example:

```
<ellipse cx = "200" cy = "50" rx = "100" ry = "50"/>
```

The attributes of an ellipse are as follows −

- cx − This is the x-coordinate of the center of the ellipse.
- cy − This is the y-coordinate of the center of the ellipse.
- rx − This is the x radius of the circle.
- ry − This is the y radius of the circle.

A simple ellipse in the SVG is described below.

Example:

```
<svg width = "300" height = "300">
   <ellipse cx = "200" cy = "50" rx = "100" ry = "50" fill = "green" />
</svg>
```

The same ellipse can be created dynamically as below,

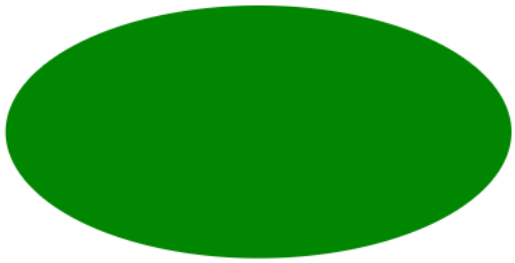
Example:

```
<html>
   <head>
      <script type = "text/javascript" src =
"https://d3js.org/d3.v4.min.js"></script>
   </head>

   <body>
      <div id = "svgcontainer"></div>
```

```
        <script>
            var width = 300;
            var height = 300;
            var svg = d3.select("#svgcontainer")
                .append("svg")
                .attr("width", width)
                .attr("height", height);
            svg.append("ellipse")
                .attr("cx", 200)
                .attr("cy", 50)
                .attr("rx", 100)
                .attr("ry", 50)
                .attr("fill", "green")
        </script>
    </body>
</html>
```

The above code will yield the following result.

Output:



4.7. SVG Transformations

SVG provides options to transform a single SVG shape element or group of SVG elements. SVG transform supports **Translate, Scale, Rotate and Skew**. Let us learn transformation in this chapter.

SVG introduces a new attribute, **transform** to support transformation. The possible values are one or more of the following:

- **Translate** − It takes two options, tx refers translation along the x-axis and ty refers to the translation along the y-axis.

  For Example− *translate(30 30).*

- **Rotate** − It takes three options, angle refers rotation angle, cx and cy refers to the center of the rotation in the x and y axis. If cx and cy are not specified, then it defaults to the current origin of the coordinate system.

  For Example − *rotate(60).*

- **Scale** − It takes two options, sx refers to the scaling factor along the x-axis and sy refers to the scaling factor along the y-axis. Here, sy is optional and it takes the value of sx, if it is not specified.

  For Example − *scale(10).*

- **Skew** (**SkewX** and *SkewY*) − It takes a single option; the skew-angle refers to the angle along the x-axis for SkewX and the angle along the y-axis for SkewY.

  For Example − *skewx(20).*

Example: An example of the SVG rectangle with translate, which is described as follows

```
<html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
   </head>
   <body>
      <svg width = "300" height = "300">
         <rect x = "20"
            y = "20"
            width = "60"
            height = "60"
            fill = "green"
            transform = "translate(30 30)">
         </rect>
      </svg>
   </body>
</html>
```

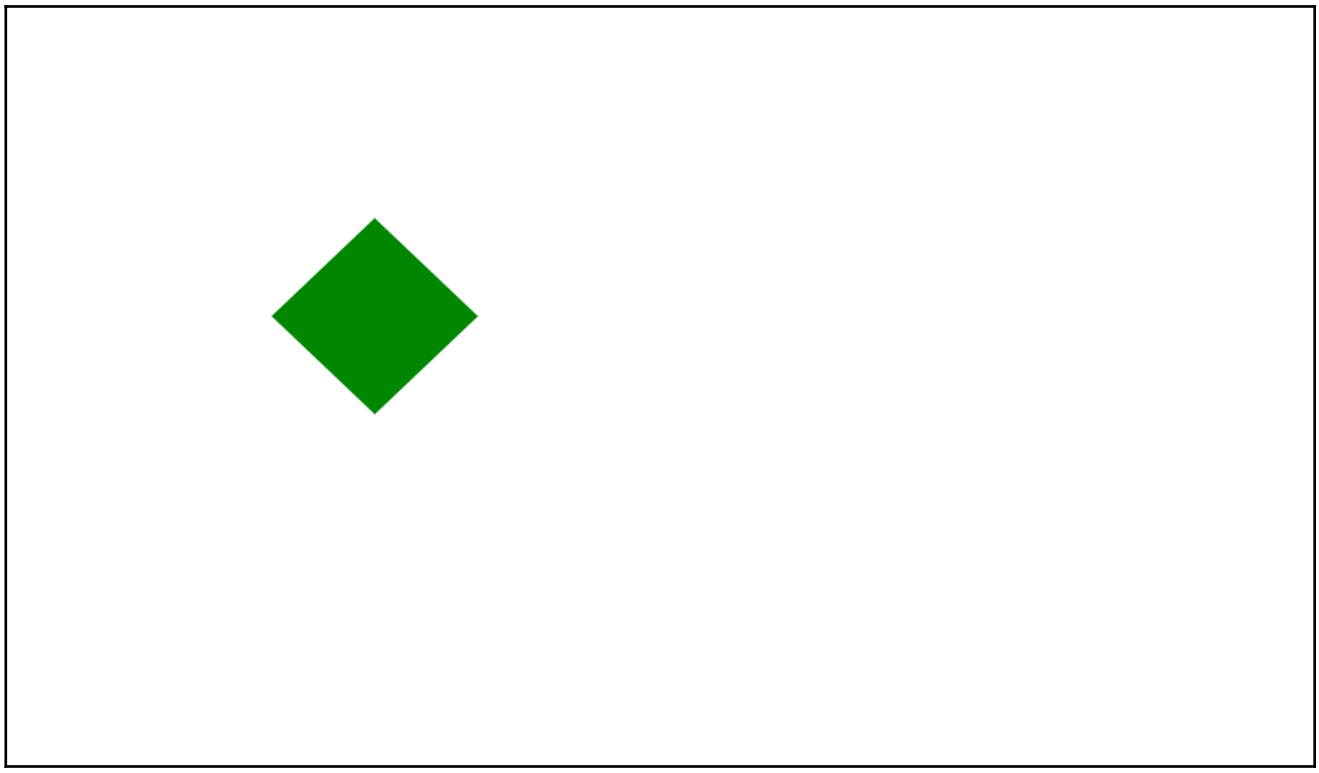
Output: The above code will yield the following result.

More than one transformation can be specified for a single SVG element using space as separation. If more than one value is specified, the transformation will be applied one by one sequentially in the order specified.

Example: An example of the SVG rectangle with translate and rotate, which is described as follows

```
<html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
   </head>
   <body>
      <svg width = "300" height = "300">
         <rect x = "20"
            y = "20"
            width = "60"
            height = "60"
            fill = "green"
            transform = "translate(60 60) rotate(45)">
         </rect>
      </svg>
   </body>
</html>
```

Output: The above code will yield the following result.



Transformation can be applied to the SVG group element as well. This enables to transform complex graphics defined in the SVG as described below.

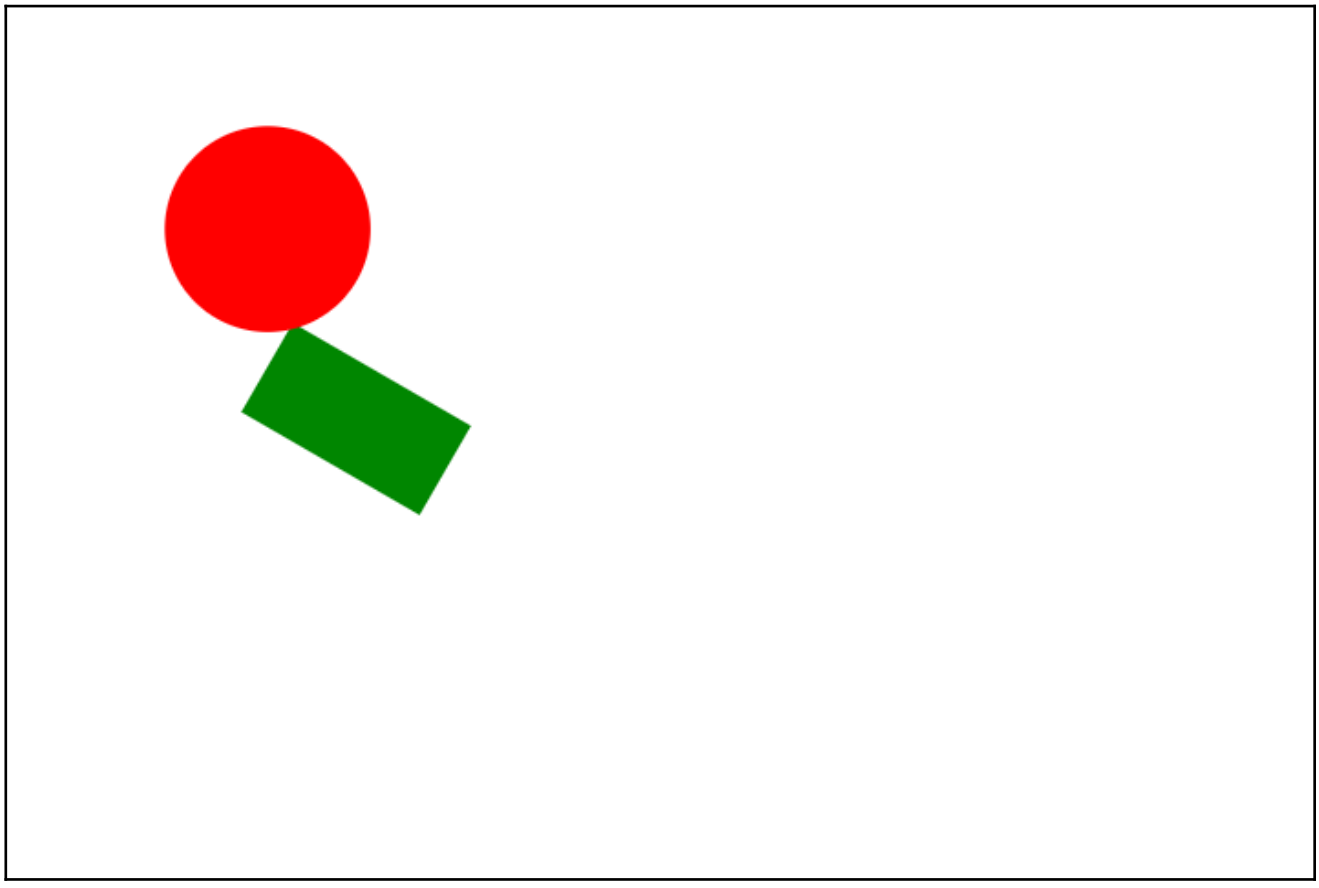
Example: An example of the SVG group with translate and rotate, which is described as follows

```
html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
   </head>

   <body>
      <svg width = "300" height = "300">
         <g transform = "translate(60,60) rotate(30)">
            <rect x = "20"
               y = "20"
               width = "60"
               height = "30"
               fill = "green">
```

```
                </rect>
                <circle cx = "0"
                   cy = "0"
                   r = "30"
                   fill = "red"/>
            </g>
        </svg>
    </body>
</html>
```

The above code will yield the following result.

Output:



4.8. A Minimal Example

To create an SVG image, try to scale, and rotate it using transformation, let us follow the steps given below.

Step 1: Create an SVG image and set width as 300 pixels and height as 300 pixels.

```
<svg width = "300" height = "300">
</svg>
```

Step 2: Create an SVG group.

```
<svg width = "300" height = "300">
   <g>
   </g>
</svg>
```

Step 3: Create a rectangle of length 60 and height 30 and fill it with green color.

```
<svg width = "300" height = "300">
   <g>
      <rect x = "20"
         y = "20"
         width = "60"
         height = "30"
         fill = "green">
      </rect>
   </g>
</svg>
```

Step 4: Create a circle of radius 30 and fill it with red color.

```
<svg width = "300" height = "300">
   <g>
      <rect x = "20"
         y = "20"
         width = "60"
         height = "30"
         fill = "green">
      </rect>
      <circle cx = "0"
         cy = "0"
         r = "30"
         fill = "red"/>
   </g>
</svg>
```

Step 5: Add a transform attribute and add translate and rotate as shown below.

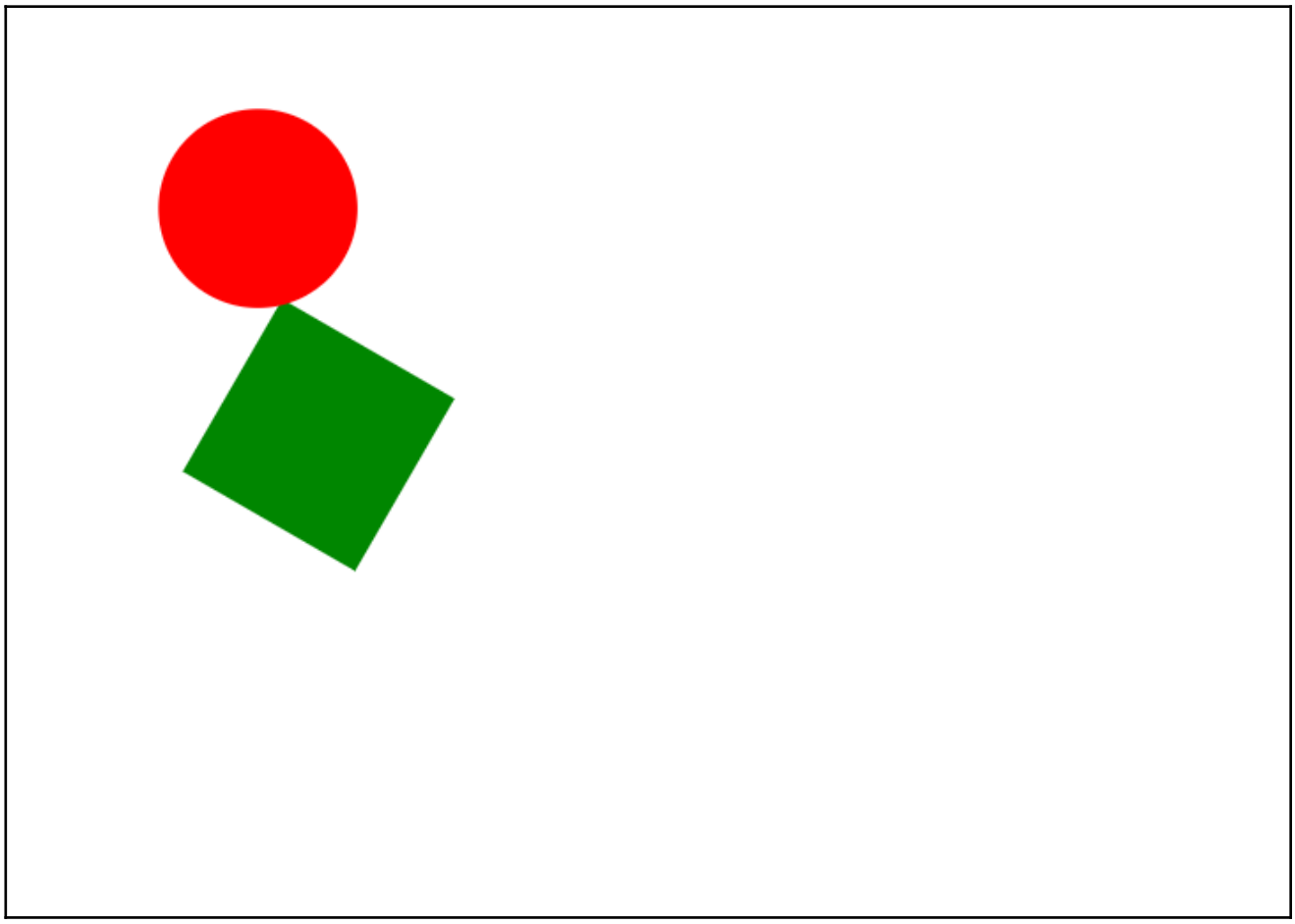
```
<svg width = "300" height = "300">
   <g transform = "translate(60,60) rotate(30)">
      <rect x = "20"
         y = "20"
         width = "60"
         height = "60"
         fill = "green">
      </rect>
      <circle cx = "0"
         cy = "0"
         r = "30"
         fill = "red"/>
   </g>
</svg>
```

Step 6: Create an HTML document, "svg_transform_rotate_group.html" and integrate the above SVG as explained below.

```
<!DOCTYPE html>
<html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
      <style>
         body { font-family: Arial; }
      </style>
   </head>

   <body>
      <div id = "svgcontainer">
         <svg width = "300" height = "300">
            <g transform = "translate(60,60) rotate(30)">
               <rect x = "20"
                  y = "20"
                  width = "60"
                  height = "60"
                  fill = "green">
               </rect>
               <circle cx = "0"
                  cy = "0"
                  r = "30"
                  fill = "red"/>
            </g>
         </svg>
      </div>
   </body>
</html>
```

The above code will yield the following result.

Output:



4.9. Transformation Using D3.js

To create SVG using D3.js, let us follow the steps given below.

Step 1: Create a container to hold the SVG image as explained below.

```
<div id = "svgcontainer"></div>
```

Step 2: Create a SVG image as explained below.

```
var width = 300;
var height = 300;
```

```
var svg = d3.select("#svgcontainer")
    .append("svg")
    .attr("width", width)
    .attr("height", height);
```

Step 3: Create a SVG group element and set translate and rotate attributes.

```
var group = svg.append("g").attr("transform", "translate(60, 60)
rotate(30)");
```

Step 4: Create an SVG rectangle and append it inside the group.

```
var rect = group
    .append("rect")
    .attr("x", 20)
    .attr("y", 20)
    .attr("width", 60)
    .attr("height", 30)
    .attr("fill", "green")
```

Step 5: Create an SVG circle and append it inside the group.

```
var circle = group
    .append("circle")
    .attr("cx", 0)
    .attr("cy", 0)
    .attr("r", 30)
    .attr("fill", "red")
```

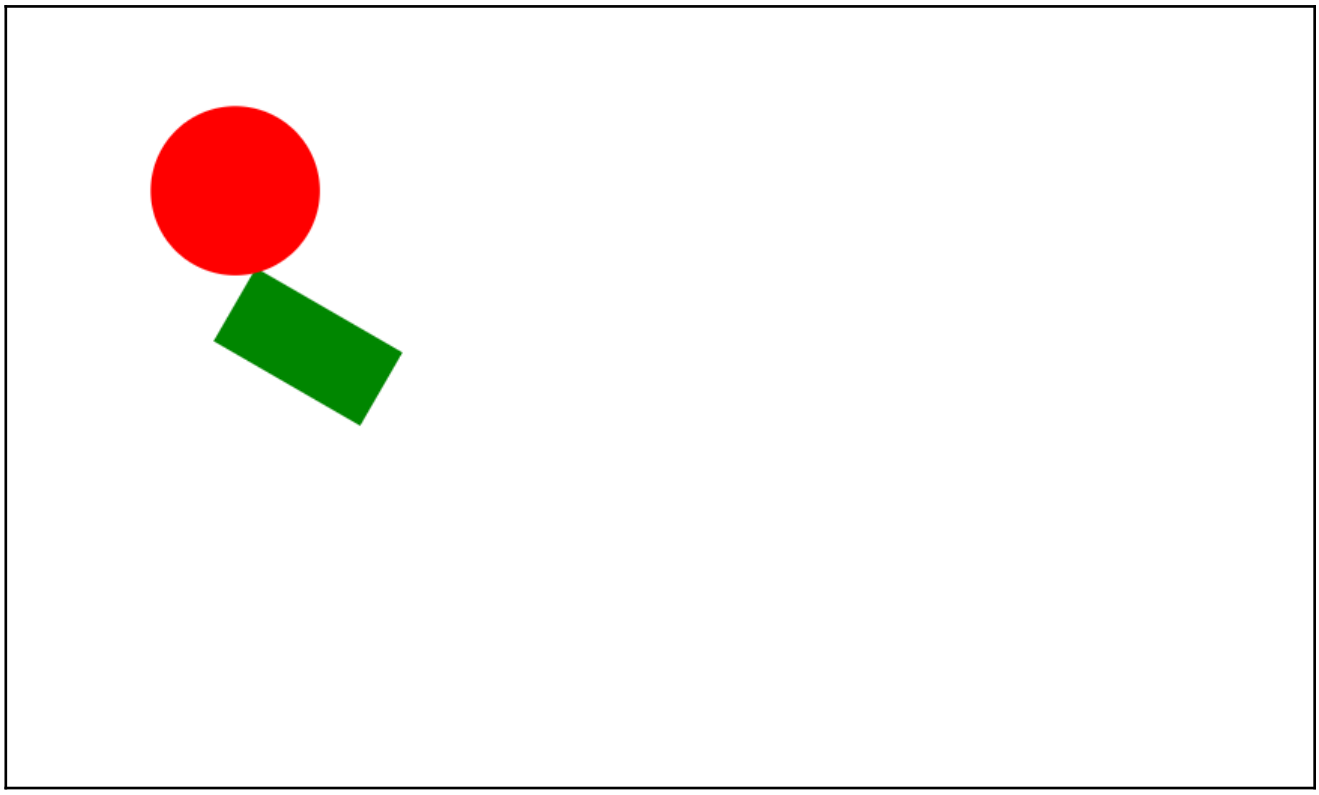
Example: The complete code is as follows:

```
<!DOCTYPE html>
<html lang = "en">
   <head>
      <title>SVG rectangle</title>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
      <style>
         body { font-family: Arial; }
      </style>
   </head>

   <body>
      <div id = "svgcontainer"></div>
         <script language = "javascript">
            var width = 300;
            var height = 300;
            var svg = d3.select("#svgcontainer")
               .append("svg")
               .attr("width", width)
               .attr("height", height);

            var group = svg.append("g")
               .attr("transform", "translate(60, 60) rotate(30)");

            var rect = group.append("rect")
               .attr("x", 20)
               .attr("y", 20)
               .attr("width", 60)
               .attr("height", 30)
               .attr("fill", "green")

            var circle = group
               .append("circle")
               .attr("cx", 0)
               .attr("cy", 0)
```

```
                    .attr("r", 30)
                    .attr("fill", "red")
                </script>
            </div>
        </body>
    </html>
```

The above code will yield the following result.

Output:



4.10. Transform Library

D3.js provides a separate library to manage transform without manually creating the transform attributes. It provides methods to handle all type of transformation. Some of the methods are transform(), translate(), scale(), rotate(), etc. You can include d3-transform in your webpage using the following script.

Example:

```
<script src = "http://d3js.org/d3.v4.min.js"></script>
<script src = "d3-transform.js"></script>
```

In the above example, the transform code can be written as shown below:

Example:

```
var my_transform = d3Transform()
   .translate([60, 60])
   .rotate(30);

var group = svg
   .append("g")
   .attr("transform", my_transform);
```