

# TUTORIAL GUIADO 3 -2020-2

## SVG.

SVG (Scalable Vector Graphics) es una especificación XML para gráficos y animaciones. Varios navegadores de la web han incorporado la capacidad de interpretar archivos SVG y desplegar los gráficos y animaciones especificadas.

En este tutorial se espera que usted trabaje con un editor HTML y un navegador replicando cada uno de estos ejercicios. Se le pedirá algunas reflexiones al respecto.

Cada foto debe incluir claramente el fondo de pantalla de su escritorio y, en esta, la hora de su PC y parte del fondo de escritorio. Las fotos resultados deben subirse en el chat personal IDENTIFICANDO CLARAMENTE la respuesta de acuerdo a lo pedido.

### Parte 1. (20%). Un elemento básico SVG

En esta actividad usted confeccionará gráficos sencillos SVG como parte de una página HTML. Complete todos los campos pedidos y luego suba su archivo de actividades.

**Paso 1-1.** Asegúrese de tener un editor de texto, idealmente con visualización de sintaxis XML, HTML, Javascript. Las fotos de referencia de esta actividad usan Sublime, pero cualquier otro editor de texto cumplirá la misma misión. Asegúrese también de tener un navegador que acepta incrustaciones SVG. Para el caso de este documento las fotos son sobre el navegador Firefox Quantum 61.0.1



**Paso 1-2.** Los atributos `cx` y `cy` corresponden a las coordenadas en el plano. El atributo `fill` corresponde al color de relleno de la figura. Copie la estructura `circle` DOS veces, y modifique la primera copia con posición en (0,0) y con color rojo (“red”). La segunda copia debe estar en la posición (100,100) y el color de relleno debe ser verde (“green”).

De acuerdo a los resultados, ADEMÁS genere una explicación de dónde se encuentra la posición 0,0 en un gráfico SVG. Escriba esto en el chat de Slack. Note qué es lo que pasa con la parte del gráfico fuera del marco.

**Respuesta 1 debe ser:**

Tut-3-Parte-1-A. Foto del código del paso 1-2.

Tut-3-Parte-1-B. Explicación de ubicación de la coordenada 0,0

## Parte 2. (20%) Diferentes elementos SVG


Paso 2-1. Copie el siguiente código y verifique que la salida es como se muestra en la figura.

```
<svg width="500" height="500">
  <rect
    width="300"
    height="100"
    rx = "20"
    ry = "20"
    style="fill:rgb(200,100,50);stroke-width:3;stroke:rgb(0,0,0)" />

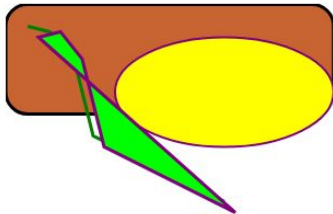
  <ellipse
    cx="200"
    cy="80"
    rx="100"
    ry="50"
    style="fill:yellow;stroke:purple;stroke-width:2" />

  <polyline
    points="20,20 40,25 60,40 80,120 120,140 200,180"
    style="fill:none;stroke:green;stroke-width:3" />

  <polygon
    points="30,30 50,25 70,50, 90,130 130,150 210,190"
    style="fill:lime;stroke:purple;stroke-width:3" />
</svg>
```



### Figuras SVG



De acuerdo a los resultados genere una explicación respecto si el orden de las figuras (cuál primero y cuál después) en el fragmento de código tiene alguna relación con el resultado gráfico. Escriba su conclusión en el chat.

**Respuesta 2 debe ser:**

- |                  |   |
|------------------|---|
| Tut-3-Parte-2-A. | Foto del código del paso 2-1.                     |
| Tut-3-Parte-2-B. | Foto de la Figura                                 |
| Tut-3-Parte-2-C. | Explicación de ubicación de relevancia del orden. |

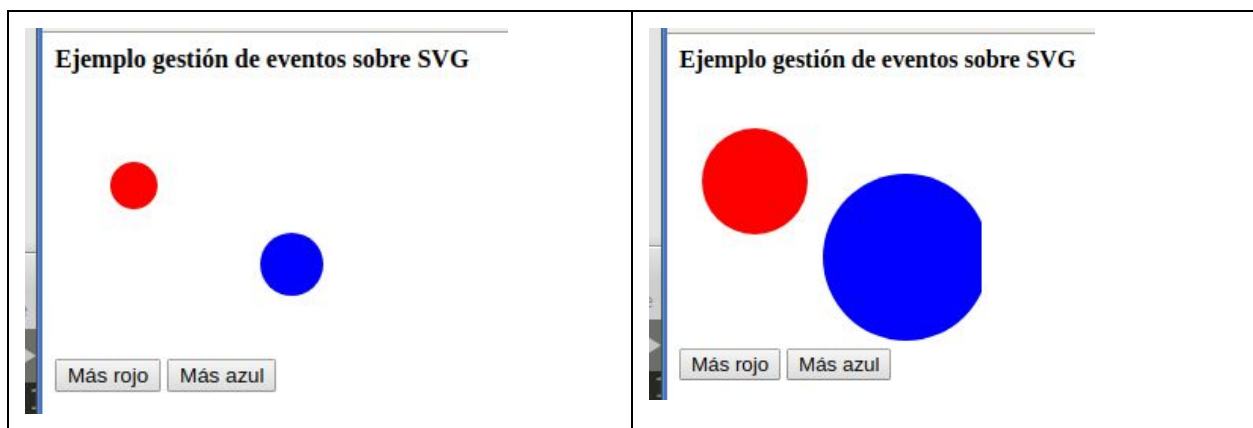
## Parte 3. (20%) Control de objetos SVG desde JS

### Paso 3-1.

El siguiente código es un ejemplo de control de los objetos SVG usando las instrucciones de la gestión de objetos (DOM). Copie el código y pruebe

```
1 <html>
2 <meta http-equiv="Content-Type" content="text/html" charset="UTF-8">
3 <head>
4   <title>Ejemplo SVG</title>
5   <script>
6     function agranda_circulo(id,delta) {
7       var cir = document.getElementById(id);
8       var dim = parseInt(cir.getAttribute("r"));
9       cir.setAttribute("r",delta+dim);
10    }
11  </script>
12 </head>
13 <body>
14 <h4>Ejemplo gestión de eventos sobre SVG</h4>
15 <svg width="200" height="160">
16   <circle id="c01" cx="50" cy="50" r="15" fill="red"/>
17   <circle id="c02" cx="150" cy="100" r="20" fill="blue"/>
18 </svg>
19 <br>
20 <button onclick="agrandar_circulo('c01',4)">Más rojo</button>
21 <button onclick="agrandar_circulo('c02',7)">Más azul</button>
22 </body>
23 </html>
```

Básicamente entre las líneas 15 y 18 está el código SVG de dos círculos. El cambio importante es que hemos agregado el atributo "id" para identificar cada círculo. Los 2 botones incluidos en las líneas 20 y 21 llaman la función `agrandar_circulo`, usando las identificaciones anteriores. La función utilizada se crea en el HEAD en las líneas 6 a 10. La función es sencilla, se lee el atributo "r" y se le suma el parámetro delta. Se usa `parseInt` porque el tipo de dato de retorno es un string, entonces usando `parseInt` se transforma a entero. La salida es la que sigue. La primera foto es al cargar la página y la siguiente foto es 5 clics después en cada botón.



**Respuesta 3 debe ser:**

Tut-3-Parte-3-A. Foto del código del paso 3-1.

Tut-3-Parte-3-B. DOS Fotos de la secuencia en el navegador

## Parte 4. (20%) Eventos en los objetos SVG

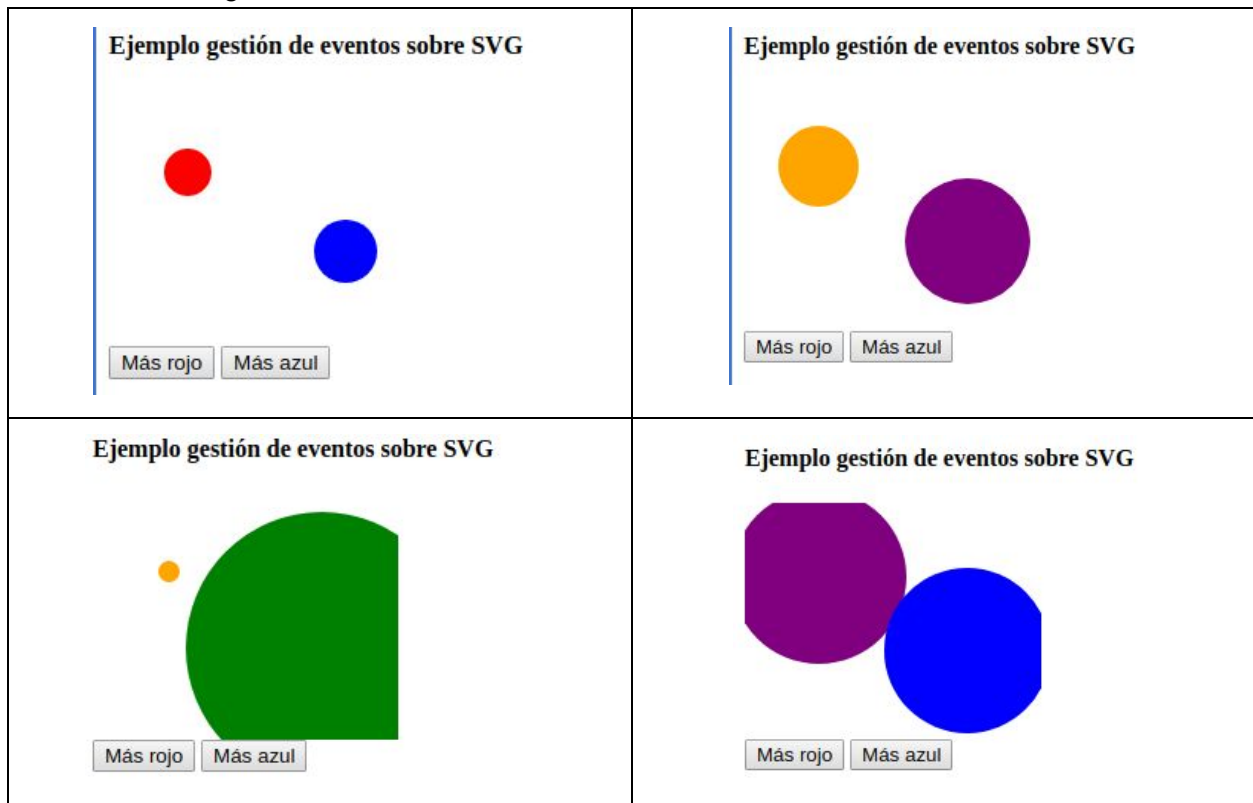
### Paso 4-1.

El siguiente código agrega eventos en los objetos SVG. Copie y pruebe el código. Pero note que los cambios son pocos.

```
1 <html>
2 <meta http-equiv="Content-Type" content="text/html" charset="UTF-8">
3 <head>
4   <title>Ejemplo SVG</title>
5   <script>
6     function agranda_circulo(id,delta) {
7       var cir = document.getElementById(id);
8       var dim = parseInt(cir.getAttribute("r"));
9       cir.setAttribute("r",delta+dim);
10    }
11    function cambia_color(id) {
12      var c = ['orange','green','blue','purple'];
13      var cir = document.getElementById(id);
14      var col = cir.getAttribute("fill");
15      var idx = c.indexOf(col);
16      if(++idx>3) idx=0;
17      cir.setAttribute("fill",c[idx]);
18    }
19  </script>
20 </head>
21 <body>
22 <h4>Ejemplo gestión de eventos sobre SVG</h4>
23 <svg width="200" height="160">
24   <circle id="c01" cx="50" cy="50" r="15" fill="red"
25     onclick="agranda_circulo('c01',-4)"
26     onmouseover="cambia_color('c01')"/>
27   <circle id="c02" cx="150" cy="100" r="20" fill="blue"
28     onclick="agranda_circulo('c02',-4)"
29     onmouseover="cambia_color('c02')"/>
30 </svg>
31 <br>
32 <button onclick="agranda_circulo('c01',4)">Más rojo</button>
33 <button onclick="agranda_circulo('c02',7)">Más azul</button>
34 </body>
35 </html>
```

Se ha agregado la función `cambia_color`. El cambio es una rotación sobre 4 colores. Se obtiene el atributo `fill`, se busca qué color es, y se elige el siguiente color en el listado. Si el color original no se encuentra, parte desde el primer color. Lo otro que se ha agregado es el manejo de los eventos `onclick` y `onmouseover` sobre los objetos SVG. Note que se ha vuelto usar la función `agranda_circulo`, pero ahora con un parámetro `delta` negativo, es decir, el círculo se achicará. En el caso de `mouseover` se invoca la función nueva de cambio de color.

La salida es la siguiente en diversos momentos



**Respuesta 4 debe ser:**

- |                  |   |
|------------------|---|
| Tut-3-Parte-4-A. | Foto del código del paso 4-1.   |
| Tut-3-Parte-4-B. | CUATRO Fotos similares a las mostradas  |
| Tut-3-Parte-4-C. | Listado de las acciones en el navegador para lograr cada figura<br>Igualmente dispuestas en 4 cuadrantes. |



## Parte 5. (20%) Creación de objetos SVG usando DOM.

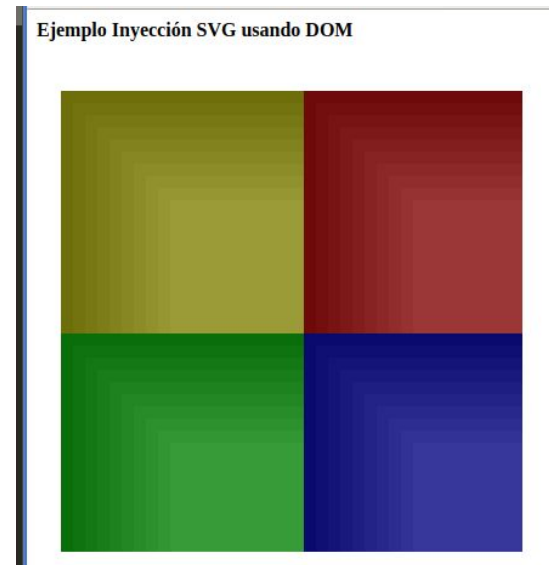
Paso 5-1. Se presenta un código nuevo que usa el método `createElement` para crear los elementos SVG. En este caso cuatro cuadrantes de cuadrados que se van achicando. El código es el siguiente:

```
4      <title>Ejemplo Inyección SVG usando DOM</title>
5      <script>
6          function inyectaPaleta(idDiv) {
7              var base = document.createElement("div");
8              var svg = document.createElement("SVG");
9              var w=400;
10             svg.setAttribute("width",w);
11             svg.setAttribute("height",w);
12             var x=20,size=200;
13             var cc=10;
14             var col,c;
15             for(var i=0; i<10; i++, x+=10, size-=10, cc+=5) {
16                 c = creaCuadrado(x,x,size,size,cc+100,cc+100,cc);
17                 svg.appendChild(c);
18                 c = creaCuadrado(x+w/2,x,size,size,cc+100,cc,cc);
19                 svg.appendChild(c);
20                 c = creaCuadrado(x,x+w/2,size,size,cc,cc+100,cc);
21                 svg.appendChild(c);
22                 c = creaCuadrado(x+w/2,x+w/2,size,size,cc,cc,cc+100);
23                 svg.appendChild(c);
24             }
25             base.appendChild(svg);
26             document.getElementById(idDiv).innerHTML = base.innerHTML;
27         }
28         function creaCuadrado(x,y,ancho,alto,colr,colg,colb) {
29             var c = document.createElement("rect");
30             c.setAttribute("x",x);
31             c.setAttribute("y",y);
32             c.setAttribute("width",ancho);
33             c.setAttribute("height",alto);
34             c.setAttribute("fill","rgb("+colr+","+colg+","+colb+)");
35             return c;
36         }
37     </script>
```

Ya hemos revisado las instrucciones `setAttribute`, y `createElement`. Los elementos diferentes a considerar es en la línea 34, el uso de la función `rgb()`. Lo que hay que considerar aquí es que **rgb no es una función javascript**, es una función css, por lo tanto es necesario generar el código de la invocación, por eso se hace como una secuencia de strings. Lo segundo a considerar es que el código **no se agrega con `appendChild`**, sino como copias del atributo `innerHTML`. Esto se hizo de esta manera porque en Chrome (al menos) no funciona el re-despliegue de un namespace diferente (en este caso SVG) aunque la generación sea correcta. El contenido HTML es el siguiente.

```
39 <body>
40 <h4>Ejemplo Inyección SVG usando DOM</h4>
41 <div id="destino"></div>
42 <script>
43   inyectaPaleta("destino");
44 </script>
45 </body>
46 </html>
47
```

Con todo esto la salida es como se indica:



**Respuesta 5 debe ser:**

- Tut-3-Parte-5-A. Foto del código del paso 5-1.
- Tut-3-Parte-5-B. Foto de la salida en el navegador