Programación Avanzada
Taller de Programación 3 SVG -2021-2

Ph.D. Carlos Cares

En cada problema formulado usted 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.

Problema 1. (15%). Estructura básica

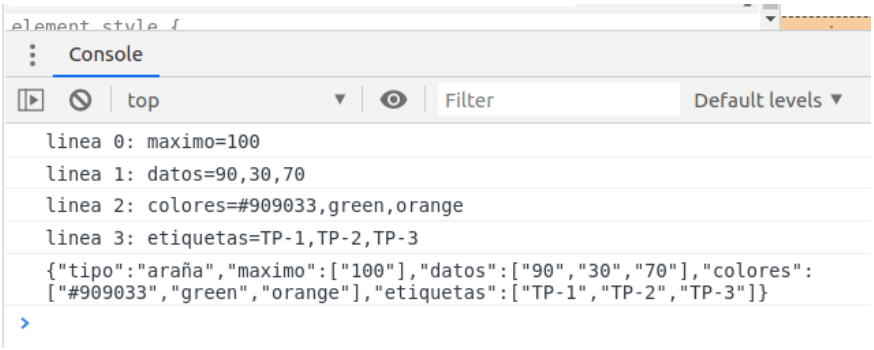
En esta actividad usted confeccionará gráficos sencillos SVG en base a información de objetos JavaScript. El programa base se ofrece a continuación.

```html
<html>
<meta http-equiv="Content-Type" content="text/html" charset="UTF-8">
<head>
    <title>Ejemplo Inyección SVG usando DOM</title>
    <script>
        class AranaSVG {
            constructor(dato) {

            }
        }

        function datosParaArana(tx) {
            var obj = document.getElementById(tx);
            str = obj.innerHTML.replace(/(\r\n|\n|\r|\s)/gm, "");
            var texto = str.split(";");
            texto.splice(-1,1);  //borra último elemento vacío
            var obj = {tipo:"araña"};
            var linea = 0;
            for(lin of texto) {
                console.log("linea "+linea+": "+lin);
                var sep = lin.split("=");
                if(sep.length==2) {
                    var ele = sep[1].split(",");
                    obj[sep[0]] = ele;
                }
                else {
                    console.log("error en línea "+linea);
                }
                linea++;
            }
            console.log(JSON.stringify(obj));
        }
    </script>
</head>
```

El cuerpo del HTML es el siguiente:

```
<body>
<h4>Ejemplo Graficación Araña</h4>
<table>
    <tr>
        <td>
            <textarea id="tx_" rows="6" cols="40">
maximo=100;
datos=90,30,70;
colores=#909033,green,orange;
etiquetas=TP-1,TP-2,TP-3;
            </textarea>
            <br>
            <button onclick="datosParaArana('tx_')">Crea Gráfico</button>
        </td>
        <td>
            <div id="grafico"></div>
        </td>
    </tr>
</table>
</body>
</html>
```

El script ofrecido, lo que hace es leer el área de texto, limpiar saltos de línea y convertir los datos que hay en dentro del área de texto en un objeto. Básicamente los “;” separan las líneas, los símbolos “=” separan nombres de atributos de valores, y los valores separados por “,” (o no) se entienden que son parte de un arreglo. El código provisto envía algunos mensajes a la consola de depuración. Al presionar el botón la salida de consola debería ser la siguiente.



Primero note que la siguiente línea es la que produce la salida del objeto. Es decir, el objeto Javascript obj, se transforma en JSON y se escribe como string. Note que, incluso, el valor “maximo” es en realidad un arreglo de 1. Valor. De modo que para acceder a “maximo” lo puede hacer de dos maneras:

- obj.maximo[0]
- obj[“maximo”][0]

```
            }
            console.log(JSON.stringify(obj));
        }
```

Para demostrar que ha implementado esta estructura básica debe hacer lo siguiente:

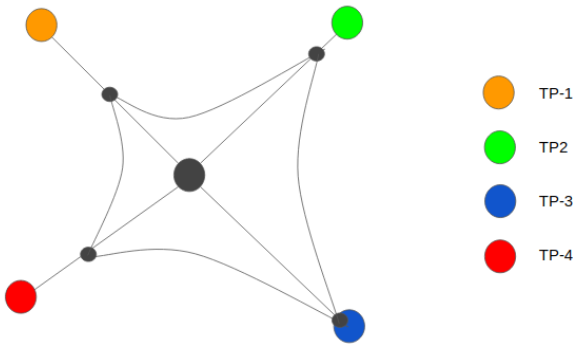
1.- Corra el programa en el navegador
2.-Agregue un cuarto dato en el textarea, un cuarto color y una cuarta etiqueta (no en el código fuente en la edición luego presione el botón.

Usted debe subir lo siguiente:

Respuesta 1-A. Pantalla de su código HTML
Respuesta 1-B. Pantalla de la consola con el nuevo objeto de salida.
Respuesta 1-C. En la pantalla de inspección, hay una pestaña que se llama `Elementos`, (o `Elements` si ha mantenido el inglés), ubique la etiqueta textarea, ábrala, y tome una foto de su contenido. Esta foto es lo esperado como respuesta 1-C.

Problema 2. (25%). Base para la reconversión de coordenadas.

El gráfico que se pretende comenzar a construir será algo similar al siguiente, pero para un sólo conjunto de valores. Deberá recordar conceptos trigonométricos básicos y de transformaciones de reglas de 3 simple.



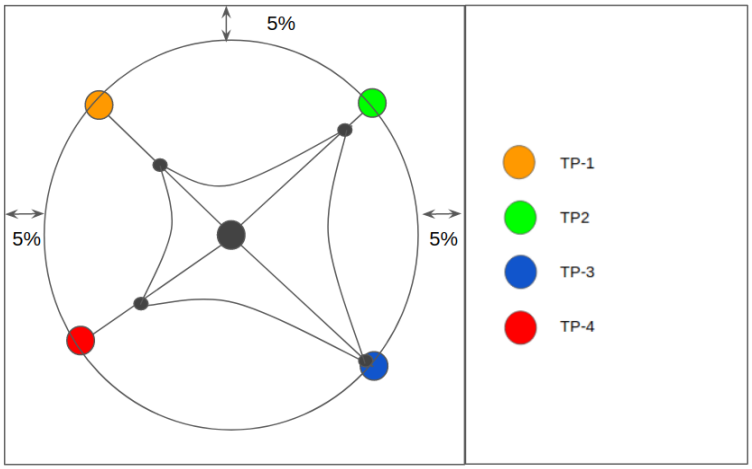
¿Cómo enfrentamos este problema?

El problema se enfrentará mediante la clase `AranaSVG`. En esta clase tendremos dos objetos internos `this.svg` y `this.cartesian`. La idea básica es que el objeto construido en la función `datosParaArana` sea el parámetro del constructor y sea, inicialmente este objeto, la propiedad `this.cartesian`;

Vamos a considerar que existe otra propiedad dentro de la clase denominada ancho (es decir `this.ancho`, para este problema lo consideraremos iniciado en 400, configuración que debe ser hecha en el constructor.

En este problema se debe crear un ambiente SVG de 400 de alto (`this.ancho`) por 600 de ancho (`this.ancho*1.5`), esto estará dividido en dos áreas, un área de 400x400 y, a la derecha de esta, un área de 200x400. En el área de la izquierda, la de 400x400 estará el gráfico y en el área de 200x400 estarán las etiquetas del significado de los colores (como la figura anterior de la derecha).

También en el constructor se debe crear el centro como un atributo dentro del objeto this.polar, simplemente haciendo `this.cartesian.centro = {x:0,y:0}`. Bien, pero el centro 0,0 no está en el centro del gráfico. Entonces lo que se pide es que cree un método que se llame `a_svg()`, este método tomará una coordenada real (objeto con propiedad x e y), y la transformará al espacio svg, El máximo sin embargo no será el máximo SVG (en este caso this.alto/2), sino que todas las cifras en `this.cartesian` estarán referidas a el máximo reportado (en esta caso 100). Adicionalmente y por estética, consideraremos un márgen de 5% de pixels. La siguiente figura ilustra la división pedida. El círculo debe tener un radio de 100 (el máximo) pero para tener un 5% de margen (arriba y abajo) significa un 10% de la distancia, que en SVG es 400. El 10% de 400 es 40, luego el espacio entre el máximo posible es 20 pixels, de donde nos queda que el alto máximo de graficar son 200-20 = 180, o lo que es lo mismo `(this.ancho/2)*0.9`, es decir, 180 unidades SVG serían 100 del gráfico. De este modo 70 a graficar será una distancia del centro de (regla de 3) 180->100, x->70 => x = 70*(180/100) = 126, es decir, si queremos graficar 70 debe estar distanciado 126 desde el centro.



Por ahora, la coordenada del centro debe ser el punto (200,200), fíjese entonces que la coordenada -70,-70, debe ser entonces por proporción -126,-126, por traslación 200-126, 200-126, (74,74), pero además, como el eje-`y` va en el sentido contrario, y tiene máximo 400, entonces y=74 significa 400-74 = 326. Note que en el caso del centro, sucede que la coordenada resultado es 200, pero dado que el eje vertical (y) va al revés, es 400-200 = 200, osea la regla general igualmente aplica al centro.

En este problema debe implementar el método `a_svg()` e implementar también el constructor de la clase como se ha indicado, entonces las líneas que deberían funcionar son:

```
this.cartesian.centro = {x:0,y:0};
this.svg.centro = this.a_svg(this.cartesian.centro);
```
En el mismo constructor se espera que haga lo siguiente para probar el despliegue:

```
this.cartesian.puntos = [{x:-70,y:-70},{x:10,y-70},{x:70,y-70}];
this.svg.puntos=  [ this.a_svg(this.cartesian.puntos[0]) ,
                this.a_svg(this.cartesian.puntos[1]),
                this.a_svg(this.cartesian.puntos[2]) ];
```

Adicionalmente se pide implementar el método `muestra`() en la clase AranaSVG, en la versión del problema 2, se debe inyectar un svg en el div de id=grafico, de 400x600, y deben aparecer los marcos de la zona de gráfico y de la zona de etiquetas, adicionalmente debe aparecer un círculo gris de radio 3 marcando el centro. Adicionalmente la función `muestra` debe desplegar un círculo de radio 4 por cada punto del arreglo de puntos (this.svg.puntos) en los colores del arreglo de colores La invocación de la inyección SVG debe ser obviamente desde fuera de la clase y deberá recibir como parámetro el identificador de dónde se desplegará. De este modo las líneas finales de la función `datosParaArana` deben ser las siguientes:

```
var graf = new AranaSVG(obj);
console.log(JSON.stringify(graf.cartesian));
graf.muestra();
console.log(JSON.stringify(graf.svg));
```

Para este problema usted debe entregar

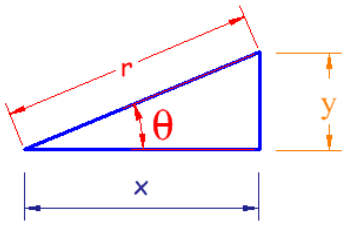
Respuesta 2-A.	El nuevo código de la clase `AranaSVG`
Respuesta 2-B.	El nuevo código de la función `datosParaArana`
Respuesta 2-C.	Foto de la salida de consola que incluya claramente los objetos `graf.cartesian`, y `graf.svg`.
Respuesta 2-D.	Foto de la salida del navegador después del clic sobre el botón con la figura de los círculos pedidos (el centro más los tres puntos asignados en `cartesian` pero reconvertidos a las coordenadas svg con los colores pre-asignados en el área de texto

Problema 3 (25%). Graficando los círculos de los extremos.

El gráfico que se hará deberá comenzar en los 45°, de tal modo que si los puntos son 90, 30, 70, entonces, lo más fácil es presentar los puntos en coordenadas polares, esto es, una distancia más un ángulo, de este modo el primer punto sería (90,45°), como son 3 puntos, obtenemos la diferencia entre ángulos es 360°/3 = 120°, de este modo la segunda coordenada es (30,45°+120°) y (70,45°+120°+120°), es decir los centros de los tres puntos deberían ser:

(90,45°) , (30,165°) , (70, 285°)

Para transformar una coordenada polar a un coordenada cartesiana es muy sencillo porque si tenemos el siguiente triángulo:



Entonces:

$$x = r \times \cos(\theta);$$
$$y = r \times \text{sen}(\theta);$$

En Javascript podemos usar Math.sin y Math.cos para seno y coseno respectivamente, el tema es que los argumentos son radianes. Puede usar la fórmula entonces,

Math.sin(45 * (Math.PI / 180)) para sen(45°), y lo mismo para coseno pero con Math.cos() o con cualquier otro ángulo reemplazando el 45 por la variable. Con estas fórmulas tendríamos en coordenadas cartesianas los centros de los círculos. Luego debemos aplicar la conversión ya hecha de coordenadas cartesianas para llevarlos al plano SVG (método `a_svg`).

En este problema se pide que solucione y divida el círculo en todos los puntos existentes (recuerde que la propiedad `length` devuelve el largo de un string). Todo lo anterior debe hacerlo un método nuevo que se debe llamar `calculaPosicionCirculos`() de tal manera que debe borrar las líneas agregadas antes en el constructor, estas:

```
this.cartesian.puntos = [{x:-70,y:-70},{x:10,y-70},{x:70,y-70}];
this.svg.puntos= [ this.a_svg(this.cartesian.puntos[0]) ,
                   this.a_svg(this.cartesian.puntos[1]),
                   this.a_svg(this.cartesian.puntos[2]) ];
```

En su lugar debe sólo invocar el método descrito, es decir:

```
this.calculaPosicionCirculos();
```

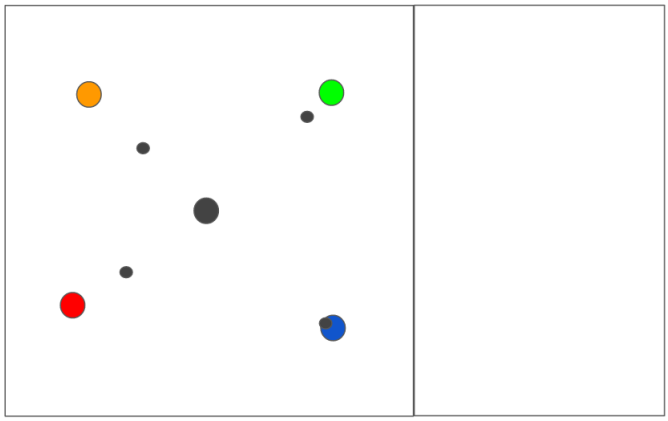
Adicionalmente, debe calcular los extremos de los ejes, es decir, las coordenadas de los círculos de color deben ser, para el caso del ejemplo (100,45°), (100,165°), (100,285°).

Adicionalmente debe actualizar el método `muestra`. Ahora en la posición de los puntos debe poner un círculo negro de 3 de radio, usando los datos de graf.svg.puntos. Así mismo, debe usar los datos de graf.svg.extremos para graficar los extremos, que deben ser de radio 6, y del color indicado en el arreglo de colores.

Para este problema usted debe entregar

Respuesta 3-A.	El nuevo código de la clase `AranaSVG`, en fotos separadas el constructor, muestra y `calculaPosicionCirculos`.
Respuesta 3-B.	El nuevo código de la función `datosParaArana`
Respuesta 3-C.	Foto de la salida del navegador después del clic sobre el botón con la figura de los círculos, esta vez considerando los tres valores existentes en el área de texto.
Respuesta 3-D.	Foto de la pestaña `Elementos` mostrando el SVG inyectado en el div “grafico”.
Respuesta 3-E.	Agregue 3 valores más al arreglo en el área de texto, y 3 colores más al arreglo de colores, presione clic. Muestre foto del nuevo gráfico y,
Respuesta 3-F.	Muestre foto de la pestaña `Elementos` mostrando el nuevo SVG inyectado.

El gráfico debería verse más o menos así, es decir, sin líneas.



Problema 4 (20%). Graficando las etiquetas

En esta fase se pide que agregue un método que se llame `inyectaEtiquetas`. No hay tips así que, se espera que entregue:

Respuesta 4-A.	Foto del nuevo método `inyectaEtiquetas`.
Respuesta 4-B.	Pantalla con el gráfico de 3 valores y 3 etiquetas
Respuesta 4-C.	Agregue 4 valores más, 4 colores más, y 4 etiquetas más. Muestre la nueva pantalla con 7 los círculos de color, y las 7 etiquetas.

Problema 5 (15%). Agregar líneas del gráfico.

En esta fase se pide que modifique el método `muestra` agregando las líneas transversales y las líneas de ejes como líneas rectas entre valores. Agregando además los valores del gráfico.

Respuesta 5-A.	Foto del método `muestra` actualizado.
Respuesta 5-B.	Pantalla con el gráfico de 3 valores y 3 etiquetas
Respuesta 5-C.	Agregue 4 valores más, 4 colores más, y 4 etiquetas más. Muestre la nueva pantalla con 7 los círculos de color, y las 7 etiquetas.
Respuesta 5-D	Suba el archivo html resultado.

El gráfico debería verse más o menos así.