SVG en HTML5

Introducción

Qué es SVG

SVG (Scalable Vector Grafics) significa Vectores Gráficos Escalables. Dicho así tal vez no lo entendamos, pero lo entenderemos mejor si decimos que es una aplicación para hacer dibujos, banners, gráficos, etc. tanto estáticos como animados.

Con dibujos SVG se crean mediante gráficos vectoriales. Esto significa que el dibujo se adapta al tamaño, de forma que al ampliarlo o reducirlo en la pantalla se muestra siempre uniforme y no muestra los contornos de los píxeles para ampliaciones muy grandes. Esto en pantalla tal vez no tenga mucha importancia, pero si queremos imprimirlo en un tamaño grande los dibujos hechos con SVG muestran un mejor acabado.

Características de SVG

SVG es fácil de usar, ya que se basa en el lenguaje XML, es decir el mismo lenguaje de etiquetas que usa HTML. Esto significa que no tenemos que usar (en principio) javascript ni otro lenguaje diferente del HTML para incorporar figuras SVG en la página, ya que el XML es el mismo lenguaje de base que utilizan tanto HTML como SVG. Podemos por tanto crear las figuras y dibujos con SVG mediante etiquetas que incorporamos al lenguaje HTML.

Incluir elementos SVG en la página se puede hacer desde el mismo código HTML. SVG dispone de una serie de etiquetas y atributos para introducir los elementos. vemos aquí a la derecha un círculo hecho mediante SVG. el código que hemos puesto para poder verlo es el siguiente:

```
<svg width=200 height=200>
<circle cx=100 cy=100 r=90 stroke="blue" stroke-width=3 fill="aqua"/>
</svg>
```

No vamos a explicar de momento este código, ya que en páginas posteriores veremos cómo trabajar con SVG. Lo que queremos ver aquí es que con un simple código de etiquetas y atributos podemos crear dibujos en la página.

Al estar el código SVG formado por los mismos elementos que el HTML podemos también aplicarle los lenguajes CSS y javascript.

Con CSS podemos indicar el estilo de los distintos elementos. Para ello disponemos además de las propiedades habituales de CSS de algunas propiedades especiales para estos elementos que veremos más adelante.

Con javascript podemos dar movimiento a los elementos creados con SVG y realizar otra serie de efectos (aparición, desaparición, agrandar, etc.). También podemos realizar estos efectos mediante CSS3.

Introducir código SVG en la página

Cómo se ha visto anteriormente para introducir un elemento SVG en la página podemos utilizar la etiqueta `<svg> ... </svg>`. Este es el elemento contenedor dentro del cual dibujaremos los demás elementos.

Debemos indicar mediante los atributos `witdh` y `height` la anchura y la altura del elemento contenedor, o podemos indicarlo también mediante codigo CSS. En todo caso hay que dar siempre una altura y anchura este elemento.

También podemos poner el elemento SVG en un archivo aparte, para después incorporarlo a la página en el sitio que queramos. Para ello creamos un nuevo archivo en el que incorporamos el código SVG.

El archivo tendrá la siguiente sintaxis:

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
... código del contenido de SVG
</svg>
```

El código va encerrado dentro de la etiqueta "svg". Al éste un lenguaje XML es conveniente poner en la etiqueta los atributos de este lenguaje es decir

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">

Después pondremos el código en SVG y al final cerramos la etiqueta "svg".

Todo ello lo guardaremos en un archivo aparte con la extension `.svg`, por ejemplo *circulo.svg*.

Para incluir después este archivo en la página podemos hacerlo de varias maneras:

- Como imagen: incluyéndolo dentro del atributo src:
  `<img src="circulo.sgv" alt="circulo" width="200" height="200">`
- Como iframe: incluyéndolo dentro del atributo src:
  `<iframe src="circulo.svg" width="200" height="200"></iframe>`
- Con la etiqueta embed; atributos src, width y height:
  `<embed src="circulo.svg" height="200" width="200" />`
- Con la etiqueta object; atributos data width y height:
  `<object data="circulo.svg" width="200" height="200"></object>`

En cualquier caso debemos incluir la ruta del archivo, y la altura y anchura de la ventana donde veremos el SVG (atributos "height" y "width").

Compatibilidad con navegadores.

SVG es compatible con todos los navegadores en sus versiones modernas, si bien en Internet explorer sólo es compatible con las versiones 9 en adelante. Para versiones anteriores es neceario usar un plugin. Podemos usar "Adobe SVG Viewer, el cual nos lo podemos descargar desde la página de Adobe:

http://www.adobe.com/devnet/svg/adobe-svg-viewer-download-area.html

Si el navegador del usuario no es compatible podemos advertirselo para que cambie a otro dentro de la etiqueta "svg", como texto de la misma, por ejemplo:

```
<svg width=200 height=200>
<circle cx=100 cy=100 r=90 stroke="blue" stroke-width=3 fill="aqua"/>
Tu navegador no es compatible con SVG, por favor cambia a Firefox, Chrome, Opera o Safari.
</svg>
```

Hemos incluido un texto dentro de la etiqueta SVG. Este texto sólo se verá si el navegador no es compatible, de esta manera advertimos al usuario de que la página no se verá de forma completa.

SVG (II)

Figuras básicas

Dibujar figuras básicas

Cualquier dibujo o figura que dibujemos con SVG debe tener su código dentro de la etiqueta `<svg> ... </svg>`, hecho que daremos por supuesto en algunos de los ejemplos que aquí pongamos. No hay que olvidar que debemos indicar también la altura y la anchura del contenedor que genera esta etiqueta.

Los dibujos que hagamos en esta página estarán todos incluidos dentro de una etiqueta "svg" como ésta:

<svg width=200 height=200> ... código del dibujo ... <svg>

En la etiqueta indicamos la altura y anchura del contenedor del dibujo. Además para que encaje en nuestra página hemos puesto el siguiente código CSS en la hoja de estilos:

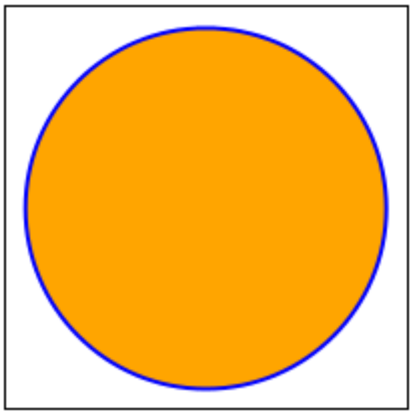
svg { margin: 1em 3em 1em 1em; float: right; border: 1px solid black; }

Dibujar un círculo

Para dibujar un círculo básico pondremos el siguiente código:

```
<svg width=200 height=200>
<circle cx=100 cy=100 r=90 stroke="blue" stroke-
width=2 fill="orange">

<svg>
```



Explicamos a continuación el código. En él hay unos elementos que son imprescindibles:

- `circle`: Esta etiqueta con este nombre indica que se dibujará un círculo.
- `cx=n cy=n`: Los atributos `cx` y `cy` indican las coordenadas del centro del círculo. con `cx` se indica la distancia horizontal en pixeles desde el borde izquierdo del contenedor al centro del círculo. `cy` es la distancia en píxeles desde el borde superior al centro del círculo.
- `r=n`: Con este atributo indicamos la longitud del radio de la circunferencia medido en píxeles.

El resto de atributos definen el estilo y se pueden utilizar con otras figuras:

- `stroke="color"`: indica el color de la línea exterior que delimita a la figura. Como valor se pone un color. Este puede ir escrito en cualquier nomenclatura aceptada por HTML5
- `stroke-width=n`: indica el grosor de la línea exterior que delimita a la figura. El valor erá un número que indica el número de pixeles.
- `fill="color"`: indica el color de relleno del interior de la figura. Como valor se pone un color que puede ir escrito en cualquier nomenclatura aceptada por HTML5.

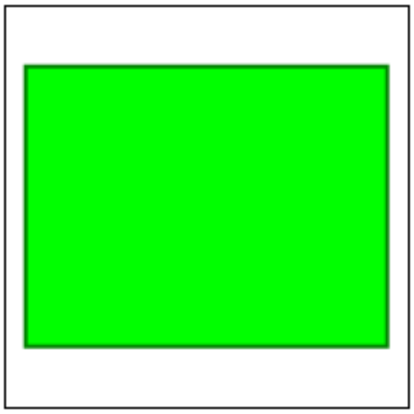
Cabe destacar que en todo contenedor SVG tenemos un origen de coordenadas que en principio estará en la esquina superior derecha con el valor 0,0. desde ahí se miden los pixeles para trazar los elementos del dibujo, tal como hemos hecho aquí para buscar el centro del círculo.

Todos los valores numéricos que indican distancia, de no indicar otra cosa se miden en píxeles. También podemos ponerlos en porcentaje siempre y cuando lo indiquemos. Esto es válido para también para el resto de figuras que hagamos.

Cuadrados y rectángulos:

Para dibujar un rectángulo lo haremos con el siguiente código:

```
<svg width=200 height=200>
<rect x=10 y=30 width=180 height=140 stroke="green"
stroke-width=2 fill="lime"/>
</svg>
```



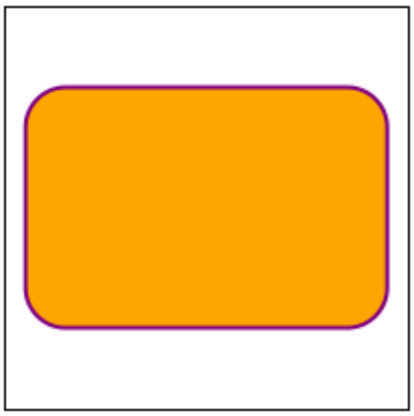
Como en el dibujo anterior hemos utilizado los atributos `stroke`, `stroke-width` y `fill` para indicar el color del borde, el grosor del borde y el color de relleno respectivamente. Los demás elementos de la etiqueta son los siguientes:

- `rect`: con esta etiqueta indicamos que vamos a dibujar un rectángulo.
- `x=n y=n`: indicamos en estos atributos las coordenadas de la esquina superior izquierda del rectángulo. `x` es la coordenada horizontal e `y` es la coordenada vertical.
- `width=n height=n`: Indicamos aquí la anchura y altura del rectángulo. con `width` indicamos la anchura y con `height` la altura.

Podemos también dibujar el rectángulo con las esquinas redondeadas, para ello sólo tenemos que añadir los atributos `rx` y `ry cuyo valor será el radio horizontal y vertical de la esquina.`

El ejemplo que vemos a la derecha tiene el siguiente código:

```
<svg width=200 height=200>
<rect x=10 y=40 rx=20 ry=20 width=180 height=120
fill="orange" stroke="purple" stroke-width=2 />
<svg>
```

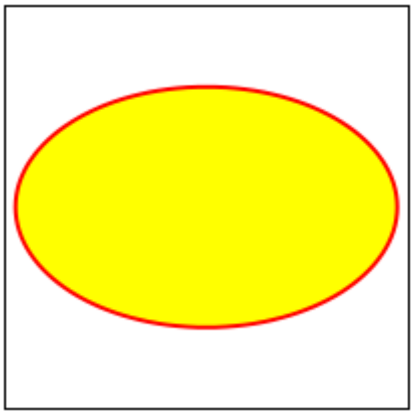


Para dibujar un cuadrado no tenemos más que dibujar un rectángulo en el que la anchura y la altura sean iguales.

Dibujar una elipse

Para dibujar una elipse utilizaremos el siguiente código:

```
<svg height="200" width="200">
  <ellipse cx=100 cy=100 rx=95 ry=60
  stroke="red" stroke-width=2 fill="yellow" />
</svg>
```



Como en las figuras anteriores utilizamos los atributos stroke, stroke-width y fill para indicar el color del borde, el grosor del borde, y el color de relleno.

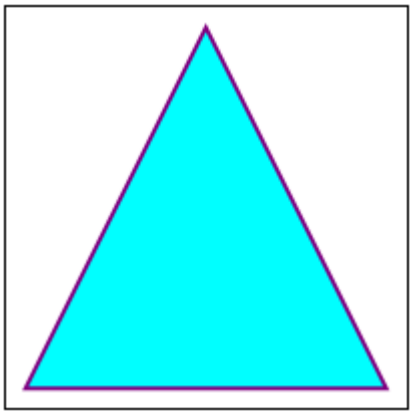
El resto de elementos de esta etiqueta son:

- `ellipse`: el nombre de la etiqueta indica que lo que dibujamos es una elipse.
- `cx=n cy=n`: coordenadas del centro de la elipse. cx es la coordenada horizontal y cy es la coordenada vertical.
- `rx=n`: radio de la elipse en el plano horizontal.
- `ry=n`: radio de la elipse en el plano vertical.

Polígonos

Para dibujar un polígono cualquiera utilizaremos un código como el siguiente:

```
<svg height="200" width="200">
  <polygon points="100,10 190,190 10,190"
       stroke="purple" stroke-width=2
fill="aqua"/>
</svg>
```



Como en las figuras anteriores indicamos el borde y el relleno con los atributos `stroke, stroke-width y fill.`

El nombre de la etiqueta `polygon` indica que lo que estamos dibujando es un polígono.

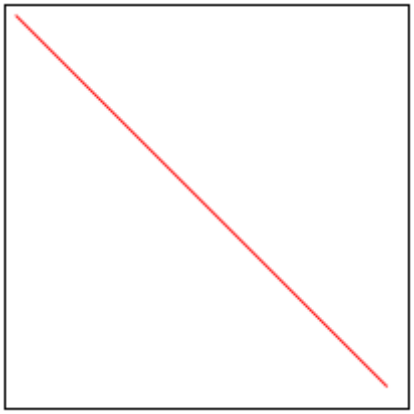
El atributo points indica los puntos de las esquinas del polígono. Escribimos aquí una serie de números por parejas. Los dos números de la pareja van separados por una coma, y cada pareja se separa de las demás por espacios en blanco. La primera pareja de números indica las coordenadas x e y respectivamente del primer punto. La segunda pareja indica las coordenadas x e y del segundo punto, y así sucesivamente, cada pareja de números más que pongamos indicamos las coordenadas del siguiente punto. Al llegar al último punto el polígono busca el primer punto para cerrarse automáticamente.

Al ir poniendo los puntos se va trazando la línea que delimita el polígono, por lo que es importante el orden en el que éstos se ponen para trazar la línea correctamente. Por otra parte debemos poner un mínimo de tres puntos para que el polígono tenga una superficie.

Líneas rectas

Para dibujar una línea recta lo haremos con un código como el siguiente:

```
<svg height=200 width=200>
  <line x1=5 y1=5 x2=190 y2=190 stroke="red" stroke-
width=2/>
</svg>
```



Los atributos `stroke y stroke-width` indican el color y grosor de la línea respectivamente.

El nombre de la etiqueta line indica que lo que dibujamos es una línea.

los atributos `x1 y y1` indican respectivamente las coordenadas horizontal y vertical del principio de la línea.

los atributos `x2 y y2` indican respectivamente las coordenadas horizontal y vertical del final de la línea.

La línea se traza desde el punto indicado como principio al punto indicado como final.

Estilo de líneas

Podemos poner un remate al final de la línea con el atributo stroke-linecap, el cual puede tener los siguientes valores:

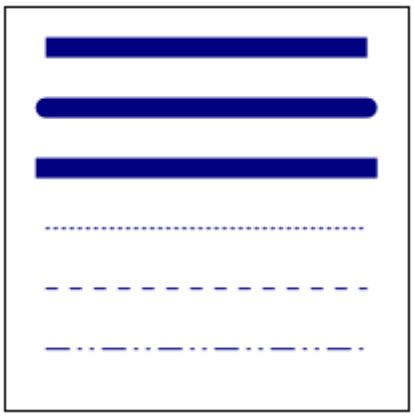
- `stroke-linecap="butt"`: valor por defecto, la línea se queda como está.
- `stroke-linecap="round"`: añade un semicírculo al final de la línea, poniendo un remate redondeado.
- `stroke-linecap="square"`: añade medio cuadrado al final de la línea, da un remate cuadrado.

Para poder apreciar las diferencias las líneas deben ser bastante gruesas como para que se vea el remate.

El atributo `stroke-dasharray` permite hacer líneas discontinuas. Como valor se ponen una o varias parejas de números. en cada par de números el primero es la longitud de línea que se ve y el segundo la longitud que está oculta. El patron se repite a lo largo de toda la línea.

Por ejemplo la ultima línea del ejemplo de la derecha tiene el siguiente atributo stroke-dasharray="12,4 2,4 2,4".

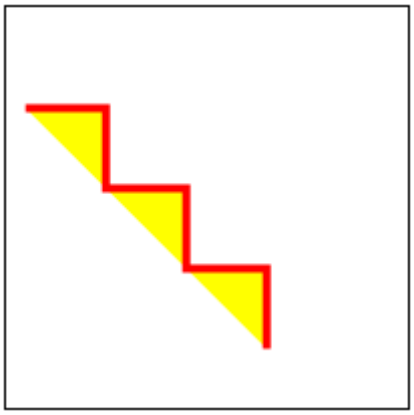
El contenedor SVG de la derecha muestra una serie de líneas con estos atributos. su código es el siguiente:



```
<svg width=200 height=200>
<line x1=20 y1=20 x2=180 y2=20 stroke="navy" stroke-
width=10 stroke-linecap="butt"/>
<line x1=20 y1=50 x2=180 y2=50 stroke="navy" stroke-
width=10 stroke-linecap="round"/>
<line x1=20 y1=80 x2=180 y2=80 stroke="navy" stroke-
width=10 stroke-linecap="square"/>
<line x1=20 y1=110 x2=180 y2=110 stroke= navy stroke-
with=2 stroke-dasharray="2,2"/>
<line x1=20 y1=140 x2=180 y2=140 stroke= navy stroke-
with=2 stroke-dasharray="6,6"/>
<line x1=20 y1=170 x2=180 y2=170 stroke= navy stroke-
with=2 stroke-dasharray="12,4 2,4 2,4"/>
</svg>
```

Polilíneas

Una polilínea es una sucesión de líneas rectas en la que cada una empieza donde acaba la anterior. También se le llama línea quebrada. Para dibujar una polilínea pondremos un código como el siguiente:



```
<svg height=200 width=200>
  <polyline points="10,50 50,50 50,90 90,90
90,130 130,130 130,170"
        stroke="red" stroke-width=4 fill="yellow"
/>

</svg>
```

El nombre de la etiqueta `polyline` indica que se traza una polilínea.

Los atributos `stroke y stroke-width` nos dan el color y el grosor de la línea.

El atributo `fill` crea un relleno en la figura. Para ello traza una línea recta del primer punto al último y rellena con el color indicado las zonas que quedan dentro.

El atributo `points` funciona como en el polígono, es decir, cada pareja de números indica las coordenadas de un punto. Al contrario que en el polígono, aquí el último punto no se cierra con el primero.

Dibujar rutas

definir una ruta

Una ruta o camino (path en inglés) es un trazado de una o varias líneas que se hacen una detrás de otra. es como cuando dibujamos con un lápiz. Aquí vamos indicando el movimiento del lapiz de un punto a otro para trazar el dibujo.

Definir una ruta en SVG se hace mediante la etiqueta `path` y el atributo `d`.

```
<path d="..comandos de la ruta .." />
```

Además como en las figuras vistas anteriormente debemos poner también los atributos de estilo para poder verlo en pantalla, es decir `stroke` para el color de las líneas, `stroke width` para el grosor de las líneas y `fill` para el color de relleno.

Respecto al color de relleno (atributo `fill`) hay que indicar que la mayoría de los navegadores ponen por defecto el color negro, por lo que si queremos otro color hay que indicarlo.

Al trazar la ruta el lapiz o puntero tiene unas coordenadas determinadas, las cuales vienen indicadas siempre por el último punto que hemos marcado.

Los comandos de la ruta, es decir lo que ponemos como valor del atributo `d` consiste en un serie de letras. Cada letra indica un tipo de trazado (línea recta, curva, mano alzada, etc) y va seguida de una serie de números que indican la posición y características del trazado.

Los comandos que podemos poner en la ruta son las siguientes letras:

- **`M`** : El lápiz o puntero se sitúa en la posición indicada.
- **`L`** : Traza una línea recta desde la posición actual a la indicada.

- **H** : Traza una línea recta horizontal desde la posición actual a la indicada.
- **V** : Traza una línea recta vertical desde la posición actual a la indicada.
- **C** : Traza una línea curva de Bézier desde la posición actual a la indicada.
- **S** : Traza una línea curva de Bézier desde la posición actual a la indicada. La curvatura de la línea es aquí la misma que en la anterior línea de Bézier.
- **Q** : Traza una línea curva cuadrática de Bézier desde la posición actual a la indicada.
- **T** : Traza una línea curva cuadrática de Bézier desde la posición actual a la indicada. La curvatura de la línea es aquí la misma que en la anterior línea cuatrática de Bézier.
- **A** : Traza un arco elíptico desde la posición actual a la indicada.
- **Z** : Cierra la ruta actual.

todos estos comandos pueden escribirse en mayúsculas o en minúsculas. Si escribimos la letra en mayúsculas, se toma una posición absoluta y las coordenadas se miden desde su origen. Si escribimos la letra en minúsculas se toma una posición relativa y las coordenadas se miden desde el último punto que se ha puesto en la ruta.

Cada comando va seguido de una serie de números, llamados parámetros, que idican o bien unas coordenadas o las características del elemento trazado.

Vemos a continuación todo esto de una forma mucho más detallada, explicando estos comandos uno a uno.

Principio y fin de la ruta

Al iniciar una ruta lo normal es que ésta no empiece en el origen de coordenadas, por lo que debemos mover el lápiz a mano alzada hasta el punto de inicio. Para ello utilizamos el comando `M`.

Los dos números o parámetros que siguen a la letra M son las nuevas coordenadas x e y del lápiz o puntero. El próximo trazo que indiquemos mediante un comando empezará en este punto.

El comando Z cierra la ruta y no lleva parámetros. Es por tanto indiferente escribirlo en mayúscula o en minúscula.

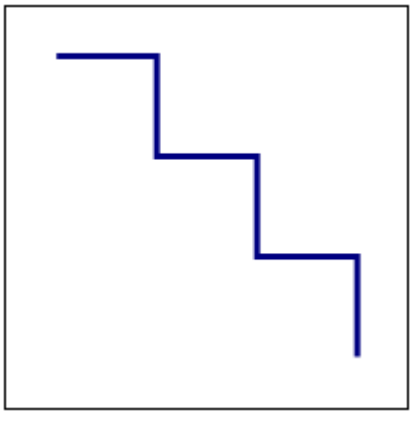
Líneas horizontales y verticales.

La línea horizontal se define mediante el comando H. éste lleva un sólo parámetro que es la nueva posición de coordenada "x". Al ser línea horizontal la coordenada "y" sique siendo la misma y no hace falta indicarla.

La línea vertical se define mediante el comando V. éste lleva un sólo parámetro que es la nueva posición de coordenada "y". Al ser línea vertical la coordenada "x" sique siendo la misma y no hace falta indicarla.

Veamos u ejemplo en el que dibujamos varias líneas horizontales y verticales

```
<svg width=200 height=200>
<path d="M 25 25 h 50 v 50 h 50 v 50 h 50 v 50 m
0 0 z"
       stroke="navy" stroke-width=3
fill="white"/>
</svg>
```



Empezamos la ruta posicionando el puntero al principio (`M 25 25`).

Seguimos trazando la ruta mediante una serie de líneas horizontales y verticales (`h 50 v 50 h 50 v 50 h 50 v 50`). Al poner los comandos en minúsculas la posición es relativa, por lo que sólo tenemos que marcar la longitud de la línea.

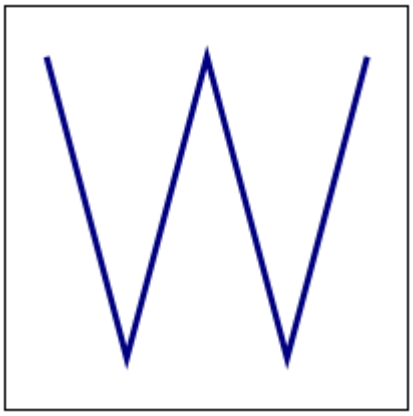
Después volvemos a posicionar el puntero en el último punto (`m 0 0`). Esto es importante antes de cerrar la ruta, ya que si no lo hacemos el puntero volverá automáticamente al principio, trazando una línea de cierre. Fíjate que también utilizamos la posición relativa, que con las coordenadas 0 0 deja el puntero en donde estaba.

Por último cerramos la ruta con el comando `z`.

líneas

Las líneas se dibujan con el comando `L`. Este lleva dos parámetros que son las coordenadas "x" e "y" del final de la línea respectivamente. El principio de línea viene marcado por la posición actual del puntero. Veamos un ejemplo.

```
<svg width=200 height=200>
<path d="M 20 25 l 40 150 l 40 -150 l 40 150 l 40
-150 m 0 0 z"
       stroke="navy" stroke-width=3
fill="white"/>
</svg>
```



Como en el ejemplo anterior empezamos posicionando el puntero al principio de la ruta (`M 20 25`).

Seguimos indicando una serie de líneas con el comando `l` (ele minúscula) (`l 40 150 l 40 -150 l 40 150 l 40 -150`). En cada una de ellas se indican las coordenadas relativas "x" e "y" respecto de la anterior.

Después posicionamos el puntero en el último punto en el que está para que éste no vuelva al inicio al cerrar la ruta (`m 0 0`).

Y por último cerramos la ruta (`z`).

Texto en SVG

Insertar texto

Para insertar texto dentro de un gráfico de SVG utilizamos la etiqueta `text`. Esta etiqueta tiene su correspondiente etiqueta de cierre, y entre la etiqueta de apertura y la de cierre incluimos el texto que queremos poner.

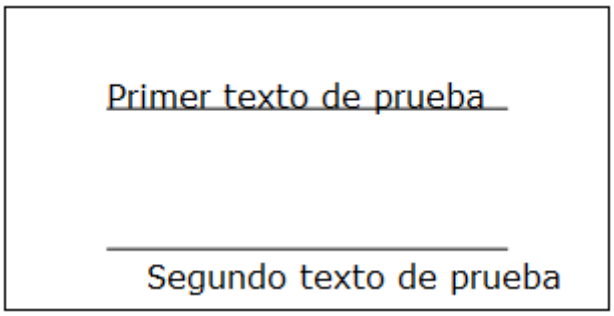
<text>mi texto </text>

Esta es la sintaxis básica de esta etiqueta, sin embargo para que funcione bien debemos poner algunos atributos que indiquen en qué lugar del contenedor queremos poner el texto. Estos son los atributos `x` e `y` que indican las coordenadas "x" e "y" respectivamente del lugar en donde queremos poner el texto.

<text x=50 y=70 >Mi texto</text>

Esto es suficiente para visualizar un texto en la mayoría de los navegadores. Estos nos dan el estilo por defecto del texto. Para insertar el texto a partir del punto dado éste traza una línea horizontal y el texto se coloca encima de la misma.

Para desplazar el texto respecto de esta línea en la que se coloca podemos añadir los atributos `dx` y `dy` que indican el desplazamiento horizontal y vertical respectivamente respecto a la línea que marca su posición original.



En el contenedor de la derecha hemos puesto dos textos y también dos líneas horizontales que empiezan en las mismas coordenadas "x" e "y" que los textos. El primer texto lo hemos dejado en sus coordenadas originales, y el segundo lo hemos desplazado mediante los atributos `dx` y `dy`. el código que hemos empleado es el siguiente:

```
<svg width=300 height=150>
<text x=50 y=50>Primer texto de prueba</text>
<text x=50 y=120 dx=20 dy=20>Segundo texto de prueba</text>
<line x1=50 y1=50 x2=250 y2=50 stroke="black" stroke-width=1 />
<line x1=50 y1=120 x2=250 y2=120 stroke="black" stroke-width=1 />
</svg>
```

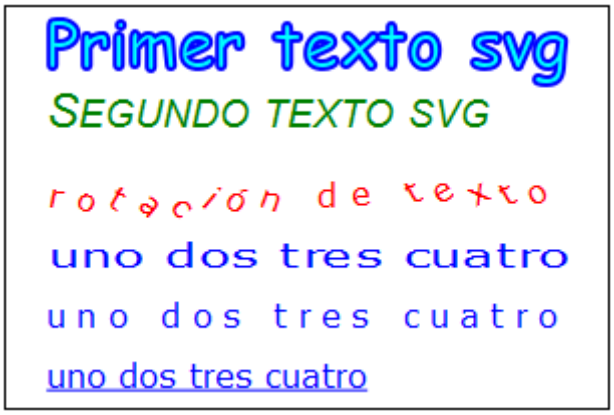
Estilo de texto

Para dar un estilo al texto emplearemos diferentes atributos:

- `fill="<color>"` : Cambia el color del texto al indicado. Se usa de igual manera que con las figuras y rutas.
- `stroke="<color>"` : Pone un borde del color indicado a las letras. Se usa de igual manera que con las figuras y rutas.
- `stroke-width="<medida>"` : Indica el grosor del borde del atributo `stroke`. Si no se indica el tipo de medida se entiende que son píxeles. Funciona igual que con las figuras y rutas.
- `font-family="<fuente>"` : Como valor podemos poner el nombre de una fuente o tipo de letra o una clase genérica o familia de fuente; tal como se hace con la propiedad `font-family` de CSS.
- `font-size="<medida>"` : Indica el tamaño de letra, igual que la propiedad `font-size` de CSS. Si ponemos sólo el número sin indicar medida la medida se tomará en píxeles.
- `font-style`="normal | italic | oblique" : indica si se quiere poner la letra en cursiva. Los valores que podemos dar son los mismos que para la propiedad `font-style` de CSS.
- `font-weight`="normal | bold | bolder | lighter" : indica el grosor de la letra (letra en negrita). Los valores que podemos poner son los mismos que en la propiedad `font-weight` de CSS.
- `font-variant`="normal | small-caps" : Pone el texto en letra versal (mayúsculas pequeñas). Se emplea igual que la propiedad `font-variant` de CSS.
- `text-decoration`=" underline | overline | line-through" : Pone una raya en el texto que puede ser de subrayado (`overline`), de tachado (`line-througt`) o una línea por encima (`overline`). Funciona igual que la propiedad `text-decoration` de CSS.
- `letter-spacing`="<medida>" : indica la medida de espaciado entre caracteres. Funciona igual que la propiedad `letter-spacing` de CSS. Si se poe un número sin indicar la medida se entiende que son píxeles. Este atributo no funciona con el navegador Firefox.
- `rotate="<número>"` : Hace rotar cada una de las letras el número de grados indicado. Podemos poner más un número, en este caso cada número hará rotar una letra: 1º num con 1ª letra, 2º num. con 2ª letra, y así sucesivamente. El último número hace rotar todas las letras que faltan hasta el final.

- `textLength="<medida>"` : indica la longitud que ocupará el texto. El espacio entre letras se adapta a la longitud indicada. Si se pone el número sin ninguna medida el espacio se medirá en píxeles.
- `lenghtAdjust="spacing | spacingAndGlyphs"` : Relacionado con el atributo anterior, una vez que hemos puesto un `textLength` ponemos este atributo para indicar si el espaciado se producirá entre letras (valor `spacing` o sólo entre palabras (valor `spacingAndGlyphs`.

Además de estos atributos podemos poner también los atributos de estilo CSS `style=" ... "`, de clase `class="nombre_clase"` y de identidad `id= "nombre_id"` que funcionan igual que con el resto de elementos de HTML.



Hemos puesto aquí un ejemplo de un contenedor SVG en el que empleamos los atributos vistos anteriormente para texto. En los dos primeros empleamos los atributos de estilo para la fuente. En el tercero ponemos el atributo para rotación del texto. En los tres últimos comparamos los atributos para ajustar la longitud del texto y el espaciado entre letras.

El código fuente de este ejemplo es el siguiente:

```
<svg width=300 height=200 >
<text x=20 y=30 font-weight="bold" fill="aqua" stroke="blue" stroke-width=2
      font-family="comic sans ms" font-size="2em" >
      Primer texto svg</text>
<text x=20 y=60 fill="green" font-family="arial" font-style="italic"
      font-variant="small-caps" font-size="1.5em">
      Segundo texto svg</text>
<text x=20 y=100 fill="red" textLength=260 rotate="10 20 30 40 50 40 30 20 10 0 -10 -
20 -30">
     rotación de texto</text>
<text x=20 y=130 fill="blue" textLength=260 lengthAdjust='spacingAndGlyphs' >
     uno dos tres cuatro</text>
<text x=20 y=160 fill="blue" textLength=260 lengthAdjust='spacing' >
     uno dos tres cuatro</text>
<text x=20 y=190 fill="blue" letter-spacing="5px" text-decoration="underline"  >
     uno dos tres cuatro</text>
</svg>
```

Estilo CSS en SVG

Gran parte de los atributos anteriores pueden ser sustituidos por propiedades CSS aplicadas al elemento de texto. Como con cualquier código CSS podemos ponerlo en el propio elemento con la etiqueta `style` o en una hoja de estilos aparte.

Los atributos relativos a la fuente (`font-family`, `font-size`, `font-weight`, `font-style` y `font-variant`) pueden sustituirse por sus correspondientes propiedades en CSS. También podemos usar la propiedad de tipo "shorthand" `font` que sustituye a todas ellas.

Los atributos `text-decoration` y `letter-spacing` también pueden sustituirse por sus correspondientes propiedades CSS.

Los atributos `fill`, `stroke` y `stroke-width` tienen también sus correspondientes propiedades CSS con el mismo nombre, por ejemplo:

text {fill: aqua; stroke: navy; stroke-width: 1px; }

Estas propiedades CSS pueden usarse también con otros elementos como figuras y rutas.

Los demás atributos no tienen correspondencia en CSS, y deben ponerse en el elemento correspondiente.

Siempre que se pueda es preferible utilizar las propiedades CSS en lugar de los atributos, ya que en caso de páginas con bastantes elementos, los atributos puede que no funcionen correctamente a causa de la herencia de los elementos padre en CSS.

El ejemplo que hemos puesto aquí simplifica bastante el código de los dos primeros textos del ejemplo anterior. Este es su código:

```
<svg width=300 height=120>
<text x=20 y=30
      style="font: bold 2em 'comic sans ms';fill: aqua; stroke: blue; stroke-width:
2px;">
      Primer texto svg</text>
<text x=20 y=100
      style="font: italic small-caps 1.5em arial; fill: green;">
      Segundo texto svg</text>
</svg>
```

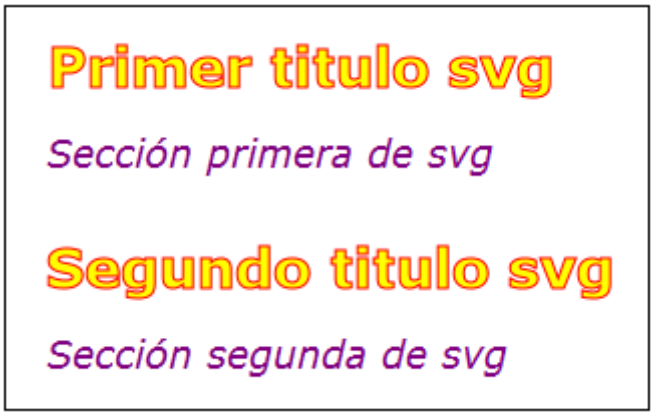
Al igual que con los demás elementos de HTML podemos usar también los atributos `class` o id para referirnos a uno o varios elementos en la hoja de estilos.

Agrupar elementos

Supongamos que tenemos un contenedor con varios elementos, y queremos dar a todos ellos, o a una parte de ellos, el mismo estilo. Hay en todos ellos varios atributos que se repiten. Para evitar esto podemos poner una serie de etiquetas que son elementos contenedores, en las cuales podemos incluir estos atributos.

Utilizamos para ello la etiqueta `<g> ... </g>` por ejemplo si queremos poner dos textos con el mismo estilo podemos incluir el estilo en la etiqueta `g` que engloba las dos etiquetas de texto

El contenedor SVG de la derecha tiene dos tipos de texto diferente. Hemos utilizado la etiqueta `g` par agrupar los elementos con igual estilo. El código es el siguiente:




```
<svg width=320 height=200>
<g style="font: bolder 1.7em arial;fill: yellow; stroke: red; stroke-width: 1px;">
<text x=20 y=40>Primer titulo svg</text>
<text x=20 y=140>Segundo titulo svg</text>
</g>
<g style="font: italic 1.2em verdana;fill: purple;">
<text x=20 y=80>Sección primera de svg</text>
<text x=20 y=180>Sección segunda de svg</text>
</g>
</svg>
```

Otras etiquetas de texto

La etiqueta tspan

Esta etiqueta se incluye dentro de la etiqueta `text` para poner un estilo diferente a una parte del texto. Es como la etiqueta `span` de HTML pero dentro de un texto en SVG.

Vemos aquí a la derecha un ejemplo de utilización de esta etiqueta. su código es el siguiente:

```
<svg width=300 height=100>
<text x=20 y=60 style="font: normal 1em arial; fill: navy;">
   texto en
         <tspan style="font: bold 1.2em 'courier new'; fill: red;">rojo</tspan>
         destacado.
</text>
</svg>
```

incluir enlaces

Podemos incluir enlaces en los elementos de un contenedor SVG.

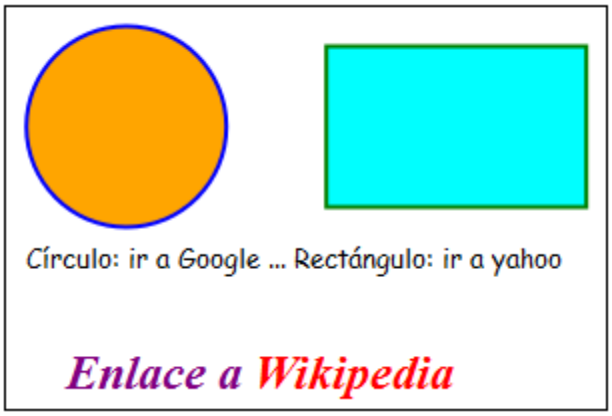
Usaremos para ello la misma etiqueta `<a>` que en HTML, pero con algunas modificaciones.

La ruta del enlace la pondremos dentro del atributo `xlink:href`. Aunque en realidad podríamos usar el atributo `href`, con `xlink:href` evitamos que el enlace tenga el estilo por defecto de los enlaces (color azul, subrayado, etc.).

Círculo: ir a Google ... Rectángulo: ir a yahoo Enlace a Wikipedia

Dentro del enlace podemos poner uno o varios elementos de SVG o podemos ponerlo dentro de una etiqueta de texto.

Los enlaces, tal como ocurre en HTML, no heredan algunas de las características de su elemento contenedor (sobre todo cuando éste es un texto), por lo que deberemos indicar éstas en el enlace.



Por lo demás admiten también los mismos atributos que el resto de enlaces, y también las pseudo-clases CSS.

Hemos hecho aquí un ejemplo en el que vemos cómo tanto figuras como texto pueden incluir enlaces. Para poder poner poner pseudo-clases hemos incluído también algo de código CSS en la hoja de estilos. Este es el código HTML-SVG:

```
<svg width=300 height=200>
<a xlink:href="http://www.google.es" target="_blank">
<circle cx=60 cy=60 r=50 class="c1"/>
</a>
<a xlink:href="http://yahoo.es" target="_blank" >
<rect x=160 y=20 width=130 height=80 class="r1"/>
</a>
```

```
<text x=10 y=130 style="font: normal 0.8em 'comic sans ms';" >
   Círculo: ir a Google ... Rectángulo: ir a yahoo</text>
<text x=30 y=190 style="fill: purple; font: italic bold 1.5em 'times new roman';">
   Enlace a <a xlink:href="http://es.wikipedia.org/" target="_blank" title="Wikipedia"
   class="e1" style="font: italic bold 1em 'times new roman';">Wikipedia</a>
<text>
</svg>
```

Para completar el código hemos puesto unas líneas en la hoja de estilos CSS que se corresponden con los atributos de clase del código anterior:

```
.c1 { fill: yellow; stroke: blue; stroke-width: 2px; }
.c1:hover { fill: orange; }
.r1 { stroke: green; stroke-width: 2px; fill: aqua;}
.r1:hover { stroke: purple; stroke-width: 5px; }
.e1 { fill: red; }
.e1:hover {fill : green; }
```

Los enlaces en SVG pueden ser una opción para sustituir los mapas de imágenes con HTML.

Degradado de color

Definición

Un degradado de color, también llamado gradiente de color, es una transición de manera suave desde un color a otro, pasando por todas las gamas intermedias entre ellos.

En SVG tenemos dos tipos de degradados, los degradados lineales y los degradados radiales.

Los degradados pueden aplicarse a cualquier elementa al que le hayamos dado color, tanto en su relleno (fill) como en el trazado del borde (stroke).

Degradado lineal

Para crear un degradado lineal utilizaremos la etiqueta `linearGradient`.

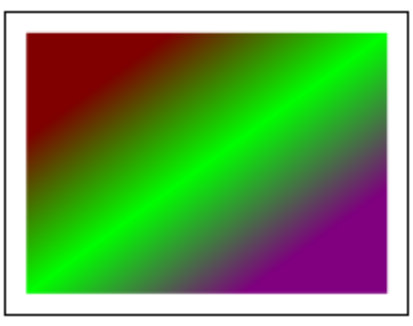
Este elemento y otros similares se suelen incluir dentro de la etiqueta `defs`:

```
<defs>
   <linearGradient ........ >
      .........
   </linearGradient>
</defs>
```

En el esquema anterior las líneas de puntos serán sustituidas par atributos y etiquetas que crean el degradado.

<="" svg="">

Vemos aquí a la derecha un ejemplo de degradado lineal en el que el fondo cambia gradualmente de unos colores a otros.



La etiqueta `defs` indica que lo que pondremos dentro serán definiciones y no parte del dibujo. En este caso definimos un degradado lineal con la

etiqueta `linearGradient`. Una vez definido tendremos que hacer una referencia al degradado desde el dibujo para aplicarlo.

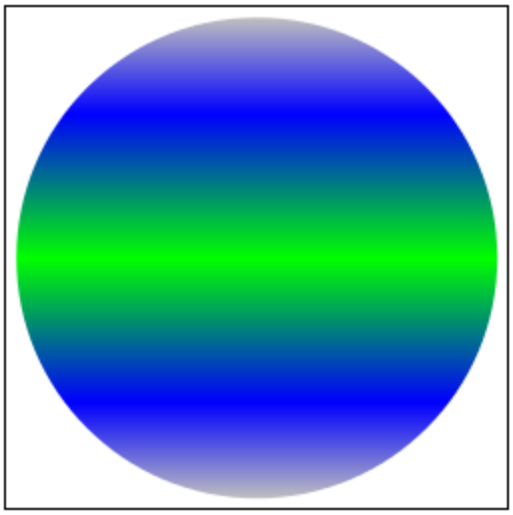
La etiqueta `linearGradient` puede llevar los siguientes atributos:

- `id="<nombre_id>"` : Como valor llevará un nombre identificador para poder referirse a este degradado en otro elemento. Es por tanto obligatorio.
- `x1="" y1=""` : Indican las coordenadas "x" e "y" del punto en donde se inicia el degradado de color. Si se pone sólo un número se entiende que las medidas son en píxeles. Se pueden poner también porcentajes.
- `x2="" y2=""` : Indican las coordenadas "x" e "y" del punto en donde termina el degradado de color. Al igual que en los anteriores podemos poner sólo el número (pixels) o porcentajes.
- `gradientUnits="userSpaceOnUse | objectBoundingBox"` : Indica desde dónde se miden las coordenadas. con el valor "`userSpaceOnUse`"Las coordenadas son las mismas que para el elemento contenedor del SVG, mientras que `objectBoudingBox` indica que toma como origen de coordenadas el propio elemento al que se le aplica, en su esquina superior derecha de la caja (imaginaria) que lo contiene. Este último es el valor por defecto, y el que se aplica si el atributo no se pone.
- `spreadMethod ="pad | reflect | repeat"` : Indica cómo debe tratarse el resultado del degradado. El valor `pad` es el valor por defecto y hace que antes y después de las coordenadas indicadas el los colores se extiendan de manera uniforme (sin degradado). El valor `reflect` repite el degradado pero en cada repetición hace un reflejo del mismo como en un espejo. El valor `repeat` repite el degradado tal cual, sin reflejarlo en cada repetición.

Nos falta definir los colores del degradado. Para ello, dentro de la etiqueta `linearGradient` idicamos los colores mediante la etiqueta `stop`. Esta etiqueta se repite para cada cambio de color en el degradado y tiene la siguiente sintaxis:

<stop offset="<num>" stop-color="<color>"/>

El atributo `offset` tiene como valor un número decimal del 0 al 1. Coloca un marcador en la línea que va desde el punto de inicio (en el 0) al punto final (en el 1) del degradado. Es conveniente poner siempre como mínimo dos etiquetas `stop` una `stop offset="0"` para el color de inicio, y otra `stop offset=1` para el color final. Entre medio podemos poner otros colores mediante decimales entre el 0 y el 1.



El atributo `color-stop` indica el color que habrá en ese punto. Entre un punto y otro se produce el cambio progresivo de color.

Ya sólo nos queda incluir el degradado de color en un elemento, para ello en la propiedad o atributo `fill` haremos referencia al degradado, poniendo el siguiente valor:

`fill="url(#nombre_id)"`

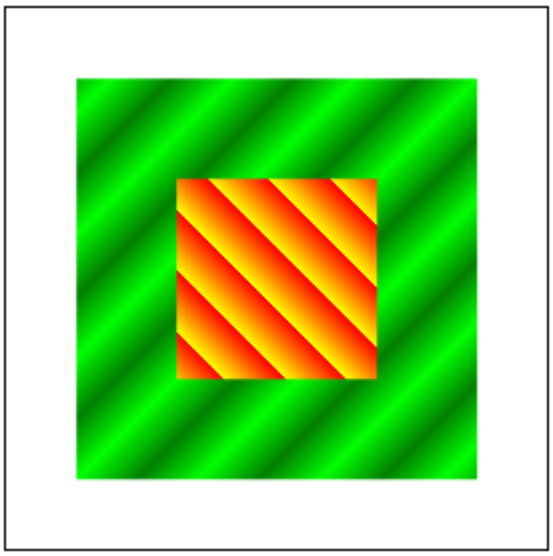
Como valor ponemos la palabra "url" y después entre paréntesis y con un signo de almohadilla delante (#) el nombre del atributo `id` que le hemos dado a la gradiente.

Vemos un ejemplo que nos da como resultado el círculo que vemos arriba a la derecha. Este es su código.

```
<svg width=250 height=250>
<defs>
<linearGradient id="grad1"
    x1="50%" y1="0%" x2="50%" y2="100%">
    <stop offset="0" stop-color="silver"/>
```

```
    <stop offset="0.2" stop-color="blue"/>
    <stop offset="0.5" stop-color="lime"/>
    <stop offset="0.8" stop-color="blue"/>
    <stop offset="1" stop-color="silver"/>
</linearGradient>
</defs>
<circle cx=125 cy=125 r=120 fill="url(#grad1)"/>
</svg>
```

Hemos puesto otro ejemplo, un poco más complicado que nos da como resultado el dibujo de la derecha. Aquí vemos la diferencia entre el atributo `spreadMethod="reflect"` que hemos aplicado al borde (stroke), y `spreadMethod="repeat"`, aplicado al relleno (fill).



Tal como vemos aquí el degradado de color puede aplicarse también a los bordes o trazos (stroke), de la misma manera que al relleno.

También puede usarse un degradado que hayamos definido anteriormente en otro contenedor SVG, siempre que éste esté en la misma página.

El código del ejemplo de la derecha es el siguiente:

```
<svg width=270 height=270>
<defs>
<linearGradient id="reflejo"
     x1="40%" y1="40%" x2="50%" y2="50%"
     spreadMethod="reflect">
     <stop offset="0" stop-color="green"/>
     <stop offset="1" stop-color="lime"/>
</linearGradient>

<linearGradient id="repite"
     x1="40%" y1="50%" x2="50%" y2="40%"
     spreadMethod="repeat">
     <stop offset="0" stop-color="yellow"/>
     <stop offset="1" stop-color="red"/>
</linearGradient>
</defs>
<rect x=10 y=10 width=250 height=250
    fill="url(#reflejo)" />
<rect x=60 y=60 width=150 height=150
    fill="url(#repite)" />
</svg>
```

Degradado radial

En el degradado radial el efecto de cambio de color se produce mediante círculos concéntricos. El cambio de color va desde un punto central hacia el exterior formando círculos.

Utilizamos para ello la etiqueta `radialGradient`

La utilización de esta etiqueta es parecida a la etiqueta `linearGradient`.

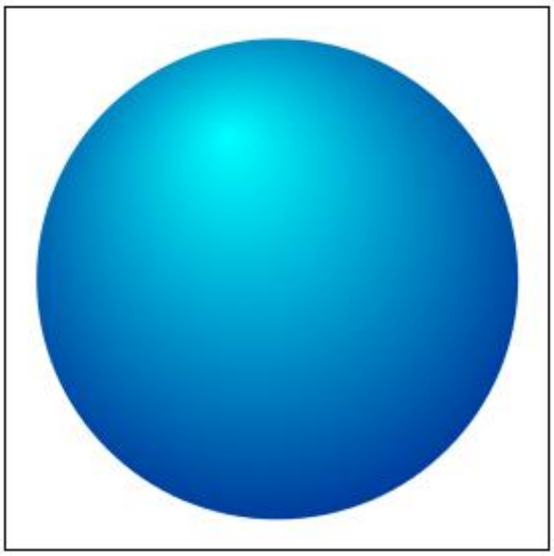
Debemos poner el atributo `id` para dar un nombre que identifique al degradado.

Una serie de atributos definen la posición del degradado y sus puntos principales. Estos son:

- `cx="<medida>" cy="<medida>"` : Coordendas "x" e "y" del centro del círculo de degradado. En estos atributos y los siguientes podemos indicar una medida, un número (píxeles) o un porcentaje.
- `r="<medida>"` : Indica el radio dentro del cual tendrá alcance el degradado.
- `fx="<medida>" fy="<medida>"` : indica el foco del degradado. El foco es el punto desde el cual se inicia el degradado. Éste se extiende desde el foco hacia el resto del círculo marcado. El foco no tiene porqué ser el mismo punto que el centro del círculo. En caso de serlo podemos omitir estos atributos.

El resto de atributos son los mismos que para `linearGradient` y con los mismos valores, es decir `gradientUnits` y `spreadMethod`.

Al igual que con `linearGradient` definimos aquí también los colores y transparencias con las etiquetas `<stop offset . . .>` que se emplean exactamente igual. En el atributo `offset` el número 0 será el punto del foco y el número 1 será el exterior del radio.



La inclusión del degradado en un elemento se hace de igual manera a lo visto anteriormente, es decir mediante `fill="url(#nombre_id)"` para incluirlo en el relleno, o `stroke="url(#nombre_id)"`

`para incluirlo en el borde o trazo.`

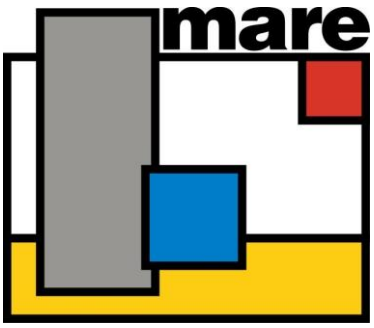
La figura que hemos puesto aquí a la derecha es un círculo con un degradado radial. Su código es el siguiente.

```
<svg width=270 height=270 >
<defs>
<radialGradient id="rad1"
   cx="50%" cy="50%" r="75%" fx="40%" fy="20%">
   <stop offset="0" stop-color="aqua"/>
   <stop offset="1" stop-color="navy"/>
</radialGradient>
</defs>
<circle cx=135 cy=135 r=120 fill="url(#rad1)"/>
</svg>
```

http://www.tutosytips.com/hablemos-de-los-degradados-en-css-los-gradient/

Practice 1T3:

A. Realiza el código en svg para reproducir el logo de la empresa pública de residuos de Cantabria MARE, y ponlo en un cuadro de 200x200 pixeles en un documento de HTML5 llamado Practica2T3Mare.html.



B. En el mismo documento y a continuación del logo anterior colocar la siguiente figura a través de su código svg.