EBook Gratis

APRENDIZAJE SVG

Free unaffiliated eBook created from
Stack Overflow contributors.

#svg

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: svg

It is an unofficial and free SVG ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official SVG.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con SVG

Observaciones

Gráficos vectoriales escalables (SVG) es un estándar W3C para dibujar imágenes vectoriales.

Aquí hay un simple archivo SVG independiente:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <circle cx="50" cy="50" r="25" fill="blue"/>
</svg>
```

SVG también se puede incrustar en HTML, en cuyo caso el atributo `xmlns` no es obligatorio.

Otros elementos gráficos son:

- `<line>`
- `<ellipse>`
- `<path>`
- `<polygon>` y `<polyline>`
- `<text>` incluyendo elementos secundarios como `<tspan>` y `<textPath>`

CSS se usa para el estilo, aunque no todas las propiedades de CSS se aplican a SVG y SVG define algunas propiedades específicas, como el `fill` y el `stroke` que no se usan en otros lugares.

Las formas se pueden rellenar con degradados o patrones y se pueden lograr efectos de trama adicionales mediante el uso de filtros.

El recorte está disponible utilizando los elementos gráficos anteriores como rutas de recorte.

Respecto a las versiones del estándar W3C SVG:

- La versión actual es SVG 1.1 (Segunda Edición)
- El W3C está trabajando actualmente en un borrador de SVG 2

Versiones

Versión	Fecha de lanzamiento
1.0	2001-09-04
1.1 Primera edición	2003-01-14
1.2 Diminuto	2008-12-22
1.1 Segunda edición	2011-08-16

Examples

SVG en línea

Inline SVG permite el marcado SVG, escrito dentro de HTML, para generar gráficos en el navegador.

Cuando se usa SVG en línea, no se requiere estrictamente un DOCTYPE. En lugar de eso, solo será suficiente con `<svg>` abrir y cerrar etiquetas junto con los atributos viewBox o width y height:

```
<svg width="100%" height="100%">
    <!-- SVG elements go here -->
</svg>
```

El fragmento `<svg>` anterior actúa como un contenedor y como un elemento estructural. Este fragmento establece su propio sistema de coordenadas.

A continuación se muestra un ejemplo de representación de un fragmento SVG con algo de contenido. Producirá un rectángulo con "¡Hola mundo!" texto dentro de ella

```
<svg width="50%" viewBox="0 0 10 10">
    <rect x="1" y="1" width="5" height="3" fill="teal" />
    <text x="2" y="2" font-family="Palatino, Georgia, serif" font-size="3%" font-weight="bold"
fill="white">Hello World!</text>
</svg>
```

Resultado:

SVG como un

Puede representar el contenido de un archivo SVG como una imagen dentro de un documento

HTML utilizando una etiqueta `<img>` . Por ejemplo:

```
<img src="my_svg_file.svg" alt="Image description">
```

Las dimensiones de la imagen, de manera predeterminada, se mostrarán de acuerdo con las propiedades de **ancho** y **alto** especificadas en el archivo SVG al que se hace referencia en el atributo `src` .

Vale la pena señalar varias limitaciones inherentes a este enfoque:

- La compatibilidad del navegador, aunque es buena, no incluye Internet Explorer 8 y versiones anteriores, ni Android 2.3 y versiones anteriores.
- No puede aplicar estilo a los elementos individuales contenidos en el archivo SVG utilizando CSS, que es externo al archivo SVG. Todo el CSS debe estar dentro del propio archivo de imagen.
- JavaScript no se ejecutará.
- La imagen debe estar completa en un solo archivo. Por ejemplo, si el archivo SVG contiene imágenes de trama, esas imágenes internas deben codificarse como URL de datos.

SVG como imagen de fondo

Puede mostrar un archivo SVG dentro de un documento HTML, especificándolo como una imagen de fondo en CSS. Por ejemplo:

```
.element {
    background-size: 100px 100px;
    background: url(my_svg_file.svg);
    height: 100px;
    width: 100px;
}
```

Si las dimensiones especificadas en su archivo SVG son mayores que las dimensiones de su elemento HTML, puede ser conveniente especificar la propiedad de `background-size` para escalar el SVG para que se ajuste a su elemento.

Al igual que con el uso de SVG como `<img>` , vale la pena señalar algunas limitaciones con este enfoque:

- La compatibilidad del navegador no incluye Internet Explorer 8 y versiones anteriores, ni Android 2.3 y versiones anteriores.
- No puede aplicar estilo a los elementos individuales contenidos en el archivo SVG utilizando CSS, que es externo al archivo SVG. Todo el CSS debe estar dentro del propio archivo de imagen.

Lea Empezando con SVG en línea: https://riptutorial.com/es/svg/topic/963/empezando-con-svg

Capítulo 2: Animación

Observaciones

La animación SMIL a través del elemento `<animate>` se admite actualmente (julio de 2016) en los principales navegadores, con la excepción de los navegadores de Microsoft. Hay una biblioteca (fakeSMIL) que se puede usar como polyfill para la compatibilidad de Microsoft.

Chrome 45 dejó de usar SMIL en favor de las animaciones CSS y la próxima sintaxis de animación declarativa de animaciones web, que desafortunadamente, solo se implementa parcialmente en los navegadores actuales. Pero los desarrolladores de Chrome suspendieron recientemente su intención (consulte esta respuesta de StackOverflow)

Examples

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="50" y="50" height="100" width="100" stroke="black" fill="yellow">
        <animate
            attributeType="XML"
            attributeName="height"
            begin="0s"
            dur="10s"
            from="100"
            to="200"
            repeatCount="indefinite"
        />
    </rect>
</svg>
```

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="50" y="50" height="100" width="100" stroke="black" fill="yellow">
        <animateTransform
            attributeType="XML"
            attributeName="transform"
            type="rotate"
            begin="0s"
            dur="10s"
            from="0"
            to="360"
            repeatCount="indefinite"
        />
    </rect>
</svg>
```

Lea Animación en línea: https://riptutorial.com/es/svg/topic/3260/animacion

Capítulo 3: cambiar

Observaciones

`<switch>` es un atributo de procesamiento condicional. No evita que los elementos sean referenciados por otros elementos. En nuestro caso, `<switch>` evalúa el valor de **SystemLanguage** en sus elementos secundarios directos que coinciden con el idioma del usuario. Una vez que se encuentra, el niño se procesa y los otros niños se omitirán.

Si no se especifica el **systemLanguage,** se mostrará el niño, lo que nos permite especificar si falla.

Información relacionada con la Recomendación W3C

Examples

Visualización alternativa en función del idioma del usuario.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <switch>
    <text systemLanguage="en-UK" x="10" y="10">UK English</text>
    <text systemLanguage="fr" x="10" y="10">Français</text>
    <text systemLanguage="ru" x="10" y="10">Русский</text>
    <text x="10" y="20">English</text> <!-- fallback (if none of the languages match) -->
  </switch>
</svg>
```

Lea cambiar en línea: https://riptutorial.com/es/svg/topic/4702/cambiar

Capítulo 4: Caminos

Introducción

Las rutas son el elemento más flexible de SVG. Una trayectoria es una serie de curvas de Bezier cúbicas o cuadráticas, dispuestas en splines conectadas. Un camino puede estar abierto o cerrado en un bucle, o puede ser complejo con varios subcomponentes. Si una ruta no es simple, la regla de relleno es importante para decidir qué áreas están dentro o fuera de la ruta.

Las rutas normalmente serán generadas por editores automáticos. Normalmente, las rutas cuadráticas se utilizan para las fuentes y las rutas cúbicas para las ilustraciones.

Parámetros

Atributos / parámetros	Descripción
re	Define una secuencia de comandos de dibujo que crean la forma. por ejemplo, d = "M 50,60 L50,60". Los comandos de dibujo en mayúsculas designan coordenadas absolutas. Los comandos de dibujo en minúscula designan coordenadas relativas.
(...)	**Comandos de dibujo**
m / m	Mueva la posición actual del dibujo a XY d = "M XY"
l / l	Dibuja una línea a X, Y d = "L XY"
v / v	Dibuja una línea vertical para Y d = "V Y"
S.S	Dibuja una línea horizontal a X d = "H X"
Automóvil club británico	Dibuje un arco a X, Y con un radio implícito de Rx y Ry y una rotación especificada por la rotación del eje X. Las grandes banderas de arco y barrido seleccionan cuál de los 4 arcos posibles que satisfacen estas restricciones deben dibujarse. d = "A Rx Ry eje X rotación (grados) gran arco bandera (0/1) barrido bandera (0/1) X, Y".
q / q	Dibuje una curva bezier cuadrática a X, Y usando el punto de control X1Y1 d = "X1Y1 X Y"
t / t	Dibuje una curva bezier cuadrática abreviada (el punto de control se calcula como una reflexión del punto de control del comando de dibujo q / Q anterior a través de la posición de dibujo actual)
c / c	Dibuje una curva bezier cúbica a X, Y utilizando los puntos de control X1,

Atributos / parámetros	Descripción
	Y1 y X2, Y2 d = "C X1Y1, X2Y2, XY"
s / s	Dibuje una curva bezier cúbica abreviada (el primer punto de control se calcula como una reflexión del segundo punto de control del comando de dibujo c / C anterior a través de la posición de dibujo actual).
- z / z	Cierre la ruta dibujando una línea para comenzar la ruta (o la línea de ruta si se ha usado otra z anteriormente)
(...)	*(fin de la lista)*
longitud de la trayectoria	(Opcional) Permite al autor especificar una longitud de ruta nominal que se utilizará para la calibración en otros cálculos, por ejemplo, para texto a lo largo de una ruta
Parámetros de trazo	*Comunes entre todas las formas y elementos de dibujo.*
carrera	Color del camino
anchura del trazo	Ancho del camino

Observaciones

Se puede encontrar información detallada sobre el elemento de `path` SVG en la Recomendación W3C para SVG .

Examples

Dibuja una línea diagonal azul usando el comando L path

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <path d="M 10,10 L 100,50" stroke="blue" stroke-width="5" />
</svg>
```

Resultado:

Dibuja una línea naranja horizontal usando el comando de dibujo H

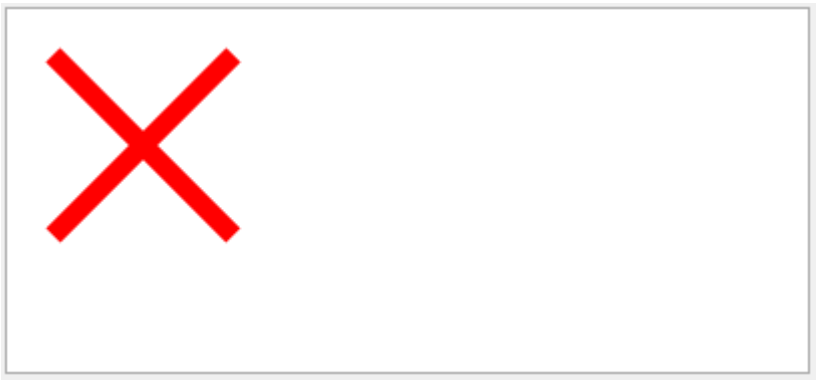
```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <path d="M 10,10 H 200" stroke="orange" stroke-width="5" />
</svg>
```

Resultado:

Dibuje una cruz roja usando l (línea relativa) comandos de ruta

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <path d="M 10,10 l 90,90 M 100,10 l -90,90" stroke="red" stroke-width="10" />
</svg>
```

Resultado:



Dibuja una línea verde vertical usando el comando V path

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <path d="M 10,10 V 200" stroke="green" stroke-width="5" />
</svg>
```

Resultado:

Lea Caminos en línea: https://riptutorial.com/es/svg/topic/2397/caminos

Capítulo 5: Circulo

Parámetros

Parámetros	Detalles
cx	Coordenada x del centro del círculo.
cy	Coordenada y del centro del círculo.
r	Radio del circulo.
carrera	Color del borde del círculo.
llenar	Color *dentro del* borde del círculo.

Observaciones

Puede encontrar información detallada sobre el elemento 'círculo' de SVG en la Recomendación W3C para SVG .

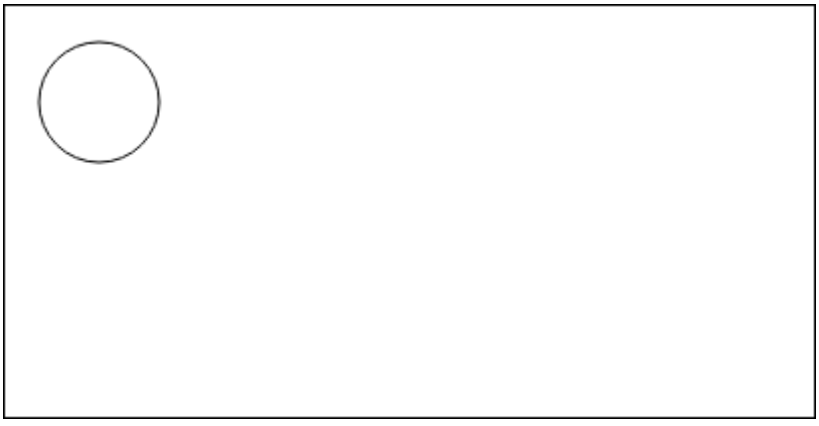
Examples

Dibuja un círculo negro sin relleno.

- Los valores `cx` y `cy` designan la ubicación del centro del círculo.
- El atributo `r` especifica el tamaño del radio del círculo.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <circle cx="40" cy="40" r="30" stroke="black" fill="none" />
</svg>
```

Resultado:



Lea Circulo en línea: https://riptutorial.com/es/svg/topic/1559/circulo

Capítulo 6: clipPath

Parámetros

Parámetro	Descripción
clipPathUnits	el sistema de coordenadas de los contenidos del patrón ya sea *objectBoundingBox* o *userSpaceOnUse*

Observaciones

Información relacionada con la Recomendación W3C

Examples

Recorte a un camino circular

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 100"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <defs>
    <clipPath id="circleClip">
      <circle cx="50" cy="60" r="20" />
    </clipPath>
  </defs>
  <image width="100" height="100" style="clip-path:url(#circleClip)"
xlink:href="https://cdn.sstatic.net/Sites/stackoverflow/company/img/logos/so/so-icon.png" />
</svg>
```

Logo original	Logo despues del recorte
	

Lea clipPath en línea: https://riptutorial.com/es/svg/topic/4840/clippath

Capítulo 7: Colores

Examples

Colores con nombre: use nombres predefinidos para los atributos de relleno y trazo

Puede encontrar una lista de nombres de palabras clave de colores reconocidos en la Recomendación W3C para SVG .

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="red" stroke="green" />
  <rect x="200" y="200" width="50" height="50" fill="yellow" stroke="blue" />
</svg>
```

Colores RGB usando notación hexadecimal

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="#ff0000" stroke="#00ff00" />
  <rect x="200" y="200" width="50" height="50" fill="#ffff00" stroke="#00ffff" />
</svg>
```

Igual que arriba usando forma hexadecimal abreviada :

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="#f00" stroke="#0f0" />
  <rect x="200" y="200" width="50" height="50" fill="#ff0" stroke="#0ff" />
</svg>
```

Colores RGB con notación funcional - valores enteros o porcentajes

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="rgb(255, 0, 0)" stroke="rgb(0, 255, 0)" />
  <rect x="200" y="200" width="50" height="50" fill="rgb(100%, 100%, 0%)" stroke="rgb(0%,
100%, 100%)" />
</svg>
```

en notación funcional, los valores RGBA también son compatibles.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <circle r="30" cx="100" cy="100" fill="rgba(255, 0, 0, 0.5)" stroke="rgba(0, 255, 0,5
0.5)" />
    <rect x="200" y="200" width="50" height="50" fill="rgba(100%, 100%, 0%, 0.5)"
stroke="rgba(0, 100%, 100%, 0.5)" />
</svg>
```

La palabra clave currentColor

`currentColor` es más útil en SVG en línea. Con esto puedes heredar el color css de los padres y usarlo en todos los colores que se usan en SVG.

En este ejemplo, el primer círculo usa el color del texto como color de relleno, y el segundo círculo lo usa como el color del trazo.

```
<html>
    <head>
        div{color:green}
    </head>
    <body>
        <div>
            some Text
            <svg width="2em" height="1em" viewBox="0 0 200 100">
                <circle cx="50" cy="50" r="45" fill="currentColor"/>
                <circle cx="150" cy="50" r="45" fill="none" stroke-width=5
stroke="currentColor"/>
            </svg>
        </div>
    </body>
</html>
```

Lea Colores en línea: https://riptutorial.com/es/svg/topic/2463/colores

Capítulo 8: Creando fuentes

Introducción

Las fuentes SVG ya no son compatibles directamente con los navegadores. Aún así, son muy convenientes para generar fuentes mediante programación, como fuentes de símbolos o fuentes de códigos de barras. Existen muchas herramientas que le permiten convertir fuentes svg a cualquier otro formato de fuente.

Observaciones

Aquí hay una lista de herramientas que puede usar con las fuentes SVG.

Convertidores

- https://github.com/fontello/svg2ttf

Examples

una fuente simple

Un ejemplo simple de una fuente svg. Algunas cosas a tener en cuenta aquí:

- el sistema de coordenadas de los glifos está en oposición al sistema de coordenadas habitual en svg. **El eje y apunta hacia arriba** . El punto 0,0 está en la esquina inferior derecha.
- Todas las rutas en una fuente deben dibujarse en sentido contrario a las agujas del reloj.
- En la mayoría de las herramientas solo se admite el atributo d del elemento glifo. Los elementos secundarios no funcionarán, aunque técnicamente están permitidos.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <font id = "myFont"
        horiz-adv-x   = "1000"
        vert-origin-x = "0"
        vert-origin-y = "0" >
    <font-face font-family  = "myFont"
               font-weight  = "normal"
               units-per-em = "1000">
      <font-face-src>
        <font-face-name name="myFont"/>
      </font-face-src>
    </font-face>`
    <glyph unicode="a" d="M0 0 H1000 L500 1000z M200 200 L500 800 L800 200z" />
    <glyph unicode="b" d="M0 0 H1000 L500 1000z M200 200 L500 800 L800 200z" />
  </font>
</svg>
```

Si tiene glifos más anchos o más estrechos, simplemente cambie el horiz-adv-x en el elemento del glifo.

```
<glyph unicode="a" horiz-adv-x="512" d="M0 0 H1000 L500 1000z M200 200 L500 800 L800 200z" />
```

selección de fuentes

La propiedad Unicode se utiliza para la selección posterior de glifos. Puede usar letras simples o puntos de código Unicode, así como ligaduras (combinación de letras o puntos de código Unicode)

- `unicode="abc"`
- `unicode="&#97;&#98;"`
- `unicode="ab&#97;&#98;"`
- `unicode="a"`
- `unicode="&#98;"`

Los glifos siempre se seleccionan por primera coincidencia, así que tenga todas las ligaduras antes de cualquier carácter individual.

los puntos de código de Unicode se pueden escribir en decimal `&#123;` o en notación hexadecimal `&#x1f` .

Ascenso, descenso y línea de base.

La propiedad units-per-em es una de las propiedades de fuente más importantes. Se utiliza para dar cualquier valor de cualquier otra propiedad cualquier significado.

La especificación de fuente CSS2 tiene una buena definición del em sqare :

> Ciertos valores, como las métricas de ancho, se expresan en unidades que son relativas a un cuadrado abstracto cuya altura es la distancia prevista entre líneas de tipo en el mismo tamaño de tipo

la **línea base por defecto está en 0** en la casilla em. Para los cálculos de altura de línea y alineación, las dos propiedades de ascenso y descenso son de la mayor importancia.

El ascenso es la distancia máxima desde la línea de base hasta el punto más alto de tu glifo más grande. En la práctica, eso es 1em, por lo que el valor que otorgó para unidades por em.

El descenso es la distancia máxima desde la línea de base hasta el punto más bajo en cualquier glifo de su fuente.

Aquí hay una fuente con glifos que representan una línea en el punto más bajo y más alto, así como en la línea de base.

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1000 1000">
  <font id = "myFont"
        horiz-adv-x   = "1000"
```

```
        vert-origin-x = "0"
        vert-origin-y = "0" >
    <font-face font-family  = "myFont"
               font-weight  = "normal"
               units-per-em = "1000"
               descent="500"
               ascent="1000">
      <font-face-src>
        <font-face-name name="myFont"/>
      </font-face-src>
    </font-face>`
    <glyph unicode = "a" d = "M0 900h1000v100h-1000z" />
    <glyph unicode = "b" d = "M0 0h1000v100h-1000z" />
    <glyph unicode = "c" d = "M0 -500h1000v100h-1000z" />
  </font>
</svg>
```

Ascenso y descenso se utilizan para determinar la altura de la línea. Las unidades por em y la línea de base se utilizan para determinar la posición y el tamaño en relación con otras fuentes utilizadas.

Lea Creando fuentes en línea: https://riptutorial.com/es/svg/topic/8147/creando-fuentes

Capítulo 9: defs

Sintaxis

- `<defs>` ... *elementos definidos* ... `</defs>`

Parámetros

Parámetro	Detalles
defs	El elemento defs no tiene parámetros.

Observaciones

El elemento `<defs>` se utiliza como un elemento contenedor para los elementos que están destinados a ser utilizados únicamente por referencia y no se representan directamente. Los elementos que normalmente se renderizarían (por ejemplo, `<rect>` , `<circle>`) que se declaran dentro de un bloque `<defs>` se tratan como si su estilo incluyera la `display:none` .

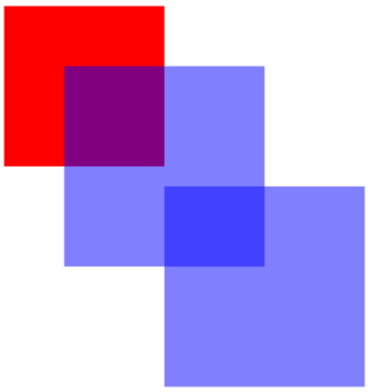
Aunque no es estrictamente necesario, la especificación SVG. recomienda colocar todas las definiciones de gradiente, filtro, patrón, máscara, símbolo y marcador dentro de un bloque `defs` .

Examples

BASIC ejemplo

```
<svg width="400px" height="400px">
<defs>
  <rect id="defrect" fill="blue" fill-opacity=".5" x="50" y="50" width="100" height="100"/>
</defs>

<rect fill="red" x="20" y="20" width="80" height="80"/>
<use xlink:href="#defrect"/>
<use xlink:href="#defrect" x="50" y="60"/>

</svg>
```

Resultado



Lea defs en línea: https://riptutorial.com/es/svg/topic/5592/defs

Capítulo 10: El elemento SVG

Examples

viewBox

El atributo viewBox define el sistema de coordenadas para un elemento `<svg>` . Esto le permite cambiar fácilmente el tamaño y la proporción relativa de un gráfico SVG sin tener que ajustar la posición y las dimensiones de cada elemento dibujado individual.

```
<!-- stretches a small icon to 60px square -->
<svg viewBox="0 0 16 16" height="60px" width="60px">
    <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.021l4.912 3.57-1.904
5.834h.021l4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

Ese código se ve así:

Sin el viewBox, se ve así:

preservar la relación de aspecto

`preserveAspectRatio` es un atributo que indica si la imagen debe escalarse uniformemente. Este atributo solo funciona si el elemento `<svg>` también tiene un `viewBox` .

El valor predeterminado es `xMidYMid` , que mantiene la relación de aspecto y centra la ruta dentro del contenedor SVG:

```
<!-- when not included `preserveAspectRatio` defaults to `xMidYMid` -->
<svg viewBox="0 0 16 16" height="60" width="120">
    <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.021l4.912 3.57-1.904
5.834h.021l4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

Cuando `preserveAspectRatio` se establece en `none` , el icono se estira para ajustarse a la caja:

```
<svg viewBox="0 0 16 16" height="60" width="120" preserveAspectRatio="none">
    <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.021 4.912 3.57-1.904
5.834h.021 4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

Hay muchos otros valores para `preserveAspectRatio` , pero estos dos son, con mucho, los más comunes.

preserveAspectRatio - cumplir y cortar atributos

El atributo `preserveAspectRatio` tiene un parámetro opcional: `meet` | `slice` El comportamiento predeterminado es `meet` que se extiende el contenido tanto en la x y la dimensión Y hasta que llena la anchura ***o*** altura de la viewBox. El `slice` alternativo conserva la relación de aspecto del contenido, pero amplía el gráfico hasta que llena ***tanto*** el ancho como el alto del cuadro de vista (recortar el contenido que desborda el viewBox).

Este es el ejemplo usando `slice`

```
<svg viewBox="0 0 16 16" height="60px" width="120px" preserveAspectRatio="xMinYMin slice">
<path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.021 4.912 3.57-1.904
5.834h.021 4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
```

que se traduce como:

y el mismo ejemplo usando `meet`

```
    <svg viewBox="0 0 16 16" height="60px" width="120px" preserveAspectRatio="xMinYMin meet">
    <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.021 4.912 3.57-1.904
5.834h.021 4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

que se traduce como:

Lea El elemento SVG en línea: https://riptutorial.com/es/svg/topic/6923/el-elemento-svg

Capítulo 11: Elipse

Parámetros

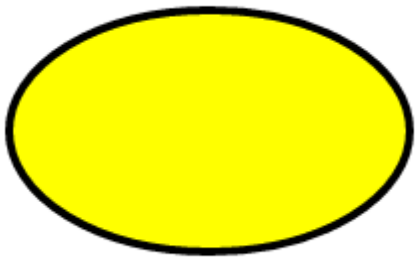
Parámetro	Detalles
`cx`	Coordenada X del centro de la elipse
`cy`	Coordenada Y del centro de la elipse.
`rx`	Radio horizontal
`ry`	Radio vertical

Examples

Elipse amarilla simple

```
<svg height="80" width="160">
  <ellipse cx="80" cy="40" rx="50" ry="30"
    style="fill:yellow; stroke:black; stroke-width:2" />
</svg>
```

Prestados:



Lea Elipse en línea: https://riptutorial.com/es/svg/topic/3993/elipse

Capítulo 12: eventos punteros

Introducción

Con la propiedad `pointer-events` , puede controlar qué parte de su dibujo reaccionará a los eventos de puntero.

Examples

ninguna

el caso de uso más común es establecer eventos de puntero en `none` para evitar que ciertas formas o todo su dibujo capturen eventos del mouse, y permitir que las formas debajo de ellos reciban los eventos.

Si se desplaza sobre el área donde el círculo rojo se superpone al círculo azul, el círculo azul seguirá recibiendo los eventos del mouse, ya que los eventos de puntero se establecen en `none`

```
<svg viewBox="0 0 150 100">
    <style>
        .target:hover{fill:green}
    </style>
    <circle class="target" cx="50" cy="50" r="50" fill="blue"/>
    <circle cx="100" cy="50" r="50" fill="red" pointer-events="none"/>
</svg>
```

llenar

La configuración de `pointer-events="fill"` permite recibir eventos del mouse en una forma, incluso si su relleno se establece en `none`

```
<svg viewBox="0 0 100 100">
    <style>
        circle:hover{fill:green}
    </style>
    <circle class="target" cx="50" cy="50" r="50" fill="none"/>
</svg>
```

Lea eventos punteros en línea: https://riptutorial.com/es/svg/topic/8166/eventos-punteros

Capítulo 13: Filtros

Sintaxis

- Declaración de filtro: `<filter id="filter-id"` > ... lista de primitivos secundarios ... `</filter>`
- Aplicar filtro a través del atributo SVG: `<elementname filter="url(#filter-id)" ... />`
- Aplicar filtro a través de la propiedad CSS: *(-prefix-*) filter: url ("# filter-id");

Parámetros

Atributos del elemento	Detalles
Filtrar región	El elemento de filtro puede definir opcionalmente la posición, las dimensiones, la resolución y las unidades para la salida de un efecto de filtro. La posición y las dimensiones de un filtro se pueden especificar utilizando los siguientes parámetros: x, y, ancho, altura. Los valores predeterminados *no* son *intuitivos* y son: x: -10% y: -10% ancho: 120% altura: 120%
Resolución del filtro	El atributo `filterRes` es un atributo opcional en SVG 1.1 que se puede usar para especificar la resolución a la que se procesa el filtro. Este atributo tenía un soporte desigual y ahora está en desuso en los navegadores actuales.
Unidades de filtro	De forma predeterminada, las unidades y el sistema de coordenadas para la región de efectos de filtro (x, y, ancho, alto) de un elemento de filtro se establecen en relación con el cuadro delimitador del elemento que hace referencia al filtro. En términos de SVG, esto se llama "objectBoundingBox". Cuando escribimos x = "50%" significa "establecer la posición inicial x de la región del filtro en el lado izquierdo del cuadro delimitador del elemento de referencia + 50% del ancho del elemento". Pero también puede especificar las unidades y las coordenadas explícitamente estableciendo la propiedad filterUnits. Las dos alternativas son "objectBoundingBox" (el valor predeterminado que acabamos de describir) o "userSpaceOnUse". userSpaceOnUse establece las unidades de filtro y el sistema de coordenadas en el lienzo del elemento de referencia, o en términos de SVG, el "userSpaceOnUse".
Unidades primitivas	Además del sistema de unidad para el filtro en sí, también puede especificar el sistema de unidad que las primitivas de filtro hijo del filtro usarán a través del atributo primitiveUnits. Una vez más, la elección es entre userSpaceOnUse y objectBoundingBox. Estos afectan las coordenadas 0,0 y los valores unitarios de las primitivas de filtro de la misma manera que para las unidades de filtro.
Espacio de	El espacio de color predeterminado para los filtros SVG es linearRGB. El

Atributos del elemento	Detalles
color	atributo opcional `color-interpolation-filters` se puede establecer en `sRGB` para cambiar el espacio de color al espacio sRGB más convencional.

Observaciones

La mayoría de los atributos de filtro se pueden animar a través del elemento `<animate>` , aunque debe usar la biblioteca "fakeSMIL" en IE para lograr los mismos resultados. La animación SMIL (el elemento `<animate>`) quedará en desuso en favor de la nueva especificación de animaciones web, que tiene un soporte muy limitado a partir de mediados de 2016.

Los elementos secundarios del elemento Filtro (primitivas de filtro) tienen dos atributos opcionales que especifican el espacio de color dentro del cual se realizan los cálculos de interpolación de color: interpolación de color y filtros de interpolación de color. El valor predeterminado para el primero es sRGB, y el valor predeterminado para el último es linearRGB. Las manipulaciones que invierten el espacio de color (a través de feColorMatrix o feComponentTransfer) pueden generar resultados no intuitivos, para aquellos que se utilizan para el espacio de color sRGB de CSS. Por ejemplo, una inversión de color de una imagen en escala de grises en RGB lineal resultará en un cambio pronunciado hacia tonos más blancos. Esto se puede corregir estableciendo el valor de la primitiva en sRGB. Estos atributos pueden establecerse en las primitivas de filtro individuales o heredarse del propio elemento de filtro.

Si no se especifica ninguna otra entrada, pero se requiere una, la primera primitiva de filtro dentro de un filtro tomará una versión rasterizada (en mapa de bits) del elemento referente como su entrada. Las primitivas de filtro posteriores que esperan una entrada tomarán el resultado de la primitiva de filtro inmediatamente anterior como entrada.

En filtros complejos, puede ser difícil mantener un seguimiento (y depurar) las entradas y salidas si se dejan implícitas; y es una buena práctica declarar explícitamente entradas y salidas para cada primitiva.

Los primitivos de filtro SVG se pueden dividir coloquialmente en entradas, transformaciones, efectos de iluminación y combinaciones.

Entradas:

feFlood: genera un campo de color

FeTurbulence: genera una gran variedad de efectos de ruido.

feImage: genera una imagen a partir de una referencia de imagen externa, URI de datos o referencia de objeto (las referencias de objeto no se admiten en Firefox a partir de mediados de diciembre de 2012)

Transformaciones

feColorMatrix: transforma los valores de entrada de un píxel RBGA en valores de salida

feComponentTransfer: ajusta la curva de color de un canal de color individual

feConvolveMatrix: reemplaza cada píxel con un nuevo píxel calculado a partir de los valores de píxel en un área relativa al píxel actual)

feGaussianBlur: reemplaza el píxel actual por un promedio ponderado de píxeles en un área alrededor del píxel

feDisplacementMap: mueve cada píxel desde su posición actual en función de los valores R, G o B de otro gráfico de entrada.

feMorphology: reemplaza cada píxel con un nuevo píxel calculado a partir del valor máximo o mínimo de todos los píxeles en un área rectangular alrededor de ese píxel.

feOffset: mueve la entrada desde su posición actual

Efectos de iluminación:

feSpecularLighting: proporciona un efecto de iluminación 2D o pseudo-3D "brillante"

feDiffuseLighting: proporciona un efecto de iluminación 2D o pseudo-3D "mate"

feDistantLight: proporciona una fuente de luz distante para iluminación especular o difusa

feSpotLight: proporciona una fuente de luz de sección cónica para iluminación especular o difusa

fePointLight: proporciona una fuente de luz puntual para iluminación especular o difusa

Combinaciones

feMerge: crea un simple sobre compuesto a partir de múltiples entradas (incluidas las entradas de filtro anteriores)

feBlend: combina entradas múltiples usando reglas de mezcla

feComposite: combina entradas múltiples utilizando reglas de combinación de conjuntos, teniendo en cuenta los valores alfa.

feTile: entrada de azulejos para crear un patrón de repetición

---

Otras notas

Aunque SVG es una tecnología de gráficos vectoriales, es importante enfatizar que los filtros SVG realizan operaciones a *nivel de píxeles* en todas las entradas (incluidas las formas SVG) y producen salidas rasterizadas (en mapa de bits) a un nivel específico de resolución. La aplicación de una transformación a escala 10x (por ejemplo) en una curva SVG simple que se ha filtrado a la

resolución de pantalla normal producirá bordes pixelados, ya que el filtro ha convertido a píxeles el efecto de suavizado del gráfico original y se ha ampliado. (No está claro si esto cumple con las especificaciones o solo es una limitación de las implementaciones actuales)

Recuerde que SVG es XML cuando escribe filtros, por lo que todas las etiquetas deben cerrarse y muchas propiedades y atributos deben especificarse explícitamente o el filtro no se ejecutará.

Un elemento de filtro nunca se renderiza directamente. Solo se hace referencia al uso de la propiedad de filtro en el elemento al que se aplica el filtro. Tenga en cuenta que la propiedad de visualización no se aplica al elemento de filtro y los elementos no se representan directamente, incluso si la propiedad de visualización se establece en un valor distinto de "ninguno". A la inversa, los elementos de filtro están disponibles para referencia incluso cuando la propiedad de visualización en el elemento de filtro o cualquiera de sus antepasados se establece en "ninguno".

Se puede hacer referencia a los filtros SVG a través de un filtro CSS, aunque a mediados de 2016, solo un subconjunto de primitivas se admiten a través de este mecanismo, y este mecanismo no es compatible con los navegadores de Microsoft.

Examples

Filtros de desenfoque: desenfoque feGaussiano (básico)

```
<svg width="900px" height="400px" viewBox="0 0 900 400">
   <defs>
      <filter id="basicGaussian">
         <feGaussianBlur stdDeviation="5"/>
      </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#basicGaussian)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

(Imagen fuente de *Daderot* en Wikimedia Commons)

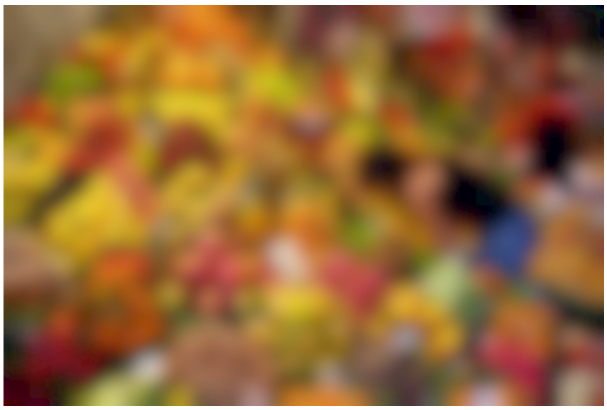
Filtros de desenfoque: feGaussianBlur (el desenfoque del eje xy el eje y se establecen por separado)

```
<svg width="900px" height="400px" viewBox="0 0 900 400">
   <defs>
      <filter id="xAxisGaussian">
         <feGaussianBlur stdDeviation="5 0"/>
      </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#xAxisGaussian)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

(Imagen fuente de *Daderot* en Wikimedia Commons)

Filtros de desenfoque: FeGaussianBlur con bordes duros y 100% de opacidad

```
<svg width="900px" height="400px" viewBox="900 400">
   <defs>
      <filter id="GaussianHardEdge" x="0%" y="0%" width="100%" height="100%">
         <feGaussianBlur stdDeviation="5"/>
         <feComponentTransfer>
           <feFuncA type="table" tableValues="1 1"/>
         </feComponentTransfer>
      </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#GaussianHardEdge)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

(Imagen fuente de *Daderot* en Wikimedia Commons)

Filtros de desenfoque: Desenfoque de caja

```
<svg width="900px" height="400px" viewBox="900 400">
   <defs>
      <filter id="GaussianHardEdge" >
         <feConvolveMatrix order="3" kernelMatrix=" 1 1 1
                                                    1 1 1
                                                    1 1 1"/>
      </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#GaussianHardEdge)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Imagen fuente de *Daderot* en Wikimedia Commons)

Filtros de desenfoque: Desenfoque de bokeh (3 capas, recortado)

```
<svg width="900px" height="400px" viewBox="0 0 900 400">
   <defs>
<filter id="BokehBlur" color-interpolation-filters="sRGB">
      <feGaussianBlur stdDeviation="2" result="blurSource"/>
      <feColorMatrix type="luminanceToAlpha"/>
      <feComponentTransfer result="brightness-mask" >
        <feFuncA type="discrete" tableValues="0 0 0 1 1"/>
```

```
    </feComponentTransfer>


 <!--bokeh Layer 1 -->
 <feTurbulence type="fractalNoise" seed="1" baseFrequency=".67" numOctaves="3"/>
<feColorMatrix type="luminanceToAlpha"/>
  <feComponentTransfer>
    <feFuncA type="discrete" tableValues="0 0 0 1"/>
 </feComponentTransfer>
  <feComposite operator="in" in="brightness-mask"/>
  <feComposite operator="in" in="blurSource"/>

  <feMorphology operator="dilate" radius="5"/>
  <feGaussianBlur stdDeviation="8"/>
  <feColorMatrix type="matrix" values="1 0 0 0 0  0 1 0 0 0  0 0 1 0 0
                                       0 0 0 9 0" />
  <feComponentTransfer result="bokeh1">
    <feFuncA type="linear" slope=".5" />
 </feComponentTransfer>



  <!--bokeh Layer 2 -->
 <feTurbulence type="fractalNoise" seed="49" baseFrequency=".67" numOctaves="3"/>
<feColorMatrix type="luminanceToAlpha"/>
  <feComponentTransfer>
    <feFuncA type="discrete" tableValues="0 0 0 1"/>
 </feComponentTransfer>
  <feComposite operator="in" in="brightness-mask"/>
  <feComposite operator="in" in="blurSource"/>

  <feMorphology operator="dilate" radius="10"/>
  <feGaussianBlur stdDeviation="12"/>
  <feColorMatrix type="matrix" values="1 0 0 0 0  0 1 0 0 0  0 0 1 0 0
                                       0 0 0 15 0" />
  <feComponentTransfer result="bokeh2">
    <feFuncA type="linear" slope=".3" />
 </feComponentTransfer>

<!--bokeh Layer 3 -->

<feTurbulence type="fractalNoise" seed="44" baseFrequency=".67" numOctaves="3"/>
<feColorMatrix type="luminanceToAlpha"/>
  <feComponentTransfer>
    <feFuncA type="discrete" tableValues="0 0 0 1"/>
 </feComponentTransfer>
  <feComposite operator="in" in="brightness-mask"/>
  <feComposite operator="in" in="blurSource"/>

  <feMorphology operator="dilate" radius="10"/>
  <feGaussianBlur stdDeviation="18"/>
  <feColorMatrix type="matrix" values="1 0 0 0 0  0 1 0 0 0  0 0 1 0 0
                                       0 0 0 15 0" />
  <feComponentTransfer result="bokeh3">
    <feFuncA type="linear" slope=".2" />
 </feComponentTransfer>

<!--Merge -->
 <feBlend mode="multiply" in="bokeh3" in2="bokeh2"/>
 <feBlend mode="lighten" in2="bokeh1"/>
```

```
      <feMorphology operator="erode" radius="0" result="bokeh"/>
      <feGaussianBlur stdDeviation="9" in="SourceGraphic"/>
      <feComposite operator="over" in="bokeh"/>
      <feComposite operator="in" in2="SourceGraphic"/>

    </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#BokehBlur)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Imagen fuente de *Daderot* en Wikimedia Commons)

Filtros de sombras: Sombra básica

```
<svg width="800px" height="600px">
<defs>
  <filter id="drop-shadow">
    <feGaussianBlur in="SourceAlpha" stdDeviation="4"/>
    <feOffset dx="5" dy="5" result="offsetblur"/>
    <feFlood flood-color="red"/>
    <feComposite in2="offsetblur" operator="in"/>
    <feMerge>
      <feMergeNode/>
      <feMergeNode in="SourceGraphic"/>
  </feMerge>
</filter>
  </defs>

  <text filter="url(#drop-shadow)" x="30" y="100" font-size="80">SVG Filters</text>

</svg>
```

Filtros de sombras: Resplandor interior

```
<svg width="800px" height="600px">
<defs>
  <filter id="inner-glow">
    <feFlood flood-color="red"/>
    <feComposite in2="SourceAlpha" operator="out"/>
```

```
    <feGaussianBlur stdDeviation="2" result="blur"/>
    <feComposite operator="atop" in2="SourceGraphic"/>
</filter>
  </defs>

  <text filter="url(#inner-glow)" x="30" y="100" font-size="80" font-family="Sans-Serif" font-
weight="bold">SVG Filters</text>

</svg>
```

Filtros de sombras: Complex Dropshadow (contorneado, ruidoso, en forma)

```
<svg width="800px" height="600px">
<defs>
<filter id="complex-shadow" color-interpolation-filters="sRGB" x="-50%" y="-50%" height="200%"
width="200%">

<!-- Take source alpha, offset it by angle/distance and blur it by size -->
<feOffset id="offset" in="SourceAlpha" dx="11" dy="6" result="SA-offset"/>
<feGaussianBlur id="blur" in="SA-offset" stdDeviation="4" result="SA-o-blur"/>

<!-- Apply a contour by using a color curve transform on the alpha and clipping the result to
the input -->

<feComponentTransfer in="SA-o-blur" result="SA-o-b-contIN">
  <feFuncA id="contour" type="table" tableValues="0 1 .3 .1 0.05 .1 .3 1 "/>
</feComponentTransfer>

<feComposite operator="in" in="SA-o-blur" in2="SA-o-b-contIN" result="SA-o-b-cont"/>

<!-- Adjust the spread by multiplying alpha by a constant factor --> <feComponentTransfer
in="SA-o-b-cont" result="SA-o-b-c-sprd">
  <feFuncA id="spread-ctrl" type="linear" slope="2.8"/>
</feComponentTransfer>

<!-- Adjust color and opacity by adding fixed offsets and an opacity multiplier -->
<feColorMatrix id="recolor" in="SA-o-b-c-sprd" type="matrix" values="0 0 0 0 0.945 0 0 0 0 0
0.137 0 0 0 0 0.137 0 0 0 0.49 0" result="SA-o-b-c-s-recolor"/>

<!-- Generate a grainy noise input with baseFrequency between approx .5 to 2.0. And add the
noise with k1 and k2 multipliers that sum to 1 -->
<feTurbulence result="fNoise" type="fractalNoise" numOctaves="6" baseFrequency="1.98"/>
<feColorMatrix in="fNoise" type="matrix" values="1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 7 -3"
result="clipNoise"/>
<feComposite id="noisemix" operator="arithmetic" in="SA-o-b-c-s-recolor" in2="clipNoise"
k1="0.67" k2="0.33" result="SA-o-b-c-s-r-mix"/>

<!-- Merge the shadow with the original -->
<feMerge>
  <feMergeNode in="SA-o-b-c-s-r-mix"/>
  <feMergeNode in="SourceGraphic"/>
</feMerge>
</filter>
  </defs>

  <text filter="url(#complex-shadow)" x="30" y="100" font-size="80" font-family="Sans-Serif"
font-weight="bold">SVG Filters</text>

</svg>
```

Filtros de manipulación de color: Escala de grises básica

```
<svg width="800px" height="600px">
   <defs>
     <filter id="greyscale">
     <feColorMatrix type="matrix"
             values="0.2126  0.7152  0.0722  0 0
                     0.2126  0.7152  0.0722  0 0
                     0.2126  0.7152  0.0722  0 0
                     0 0 0 1 0"/>
     </filter>
   </defs>

   <image

xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#greyscale)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Imagen fuente de *Daderot* en Wikimedia Commons)

Filtros de manipulación de color: Escala de grises (solo canal verde)

```
<svg width="800px" height="600px">
   <defs>
     <filter id="greyscale">
     <feColorMatrix type="matrix"
             values="0  1  0  0 0
                     0  1  0  0 0
                     0  1  0  0 0
                     0 0 0 1 0"/>
     </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#greyscale)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Imagen fuente de *Daderot* en Wikimedia Commons)

Filtros de manipulación de color: monótono

```
<svg width="800px" height="600px">
   <defs>
     <filter id="greyscale">
     <feColorMatrix type="matrix"
             values=".2  .2  .2  0 0
                     .6  .6  .6  0 0
                     .2  .2  .2  0 0
                     0 0 0 1 0"/>
     </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#greyscale)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Imagen fuente de *Daderot* en Wikimedia Commons)

Filtros de desenfoque: desenfoque de enfoque (gaussiano)

```
<svg width="800px" height="600px">
   <defs>
  <filter id="focus-blur" >
    <feDiffuseLighting result = "diffOut" diffuseConstant = "1" lighting-color="white">
       <feSpotLight id="spotlight" x = "500" y = "100" z = "150" pointsAtX = "500" pointsAtY =
```

```
"100" pointsAtZ = "0" specularExponent ="12" limitingConeAngle="70"/>
    </feDiffuseLighting>

     <feColorMatrix in="diffOut" result="alphaMap" type="luminanceToAlpha"/>
     <feComponentTransfer in="alphaMap" result="invertlight">
        <feFuncA type="table" tableValues="1 0 0"/>
     </feComponentTransfer>

     <feGaussianBlur in="invertlight" result="featherspot" stdDeviation="5"/>
     <feComposite operator="xor" result="infocus" in2="SourceGraphic" in="featherspot"/>
     <feGaussianBlur in="SourceGraphic" result="outfocus" stdDeviation="2"/>
     <feComposite operator="over" in="infocus" in2="outfocus"/>
    <feComposite operator="in" in2="SourceGraphic"/>
</filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#focus-blur)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Imagen fuente de *Daderot* en Wikimedia Commons)

Filtros De Manipulación De Color: Posterización

```
<svg width="800px" height="600px" >
   <defs>
     <filter id="posterize" color-interpolation-filters="sRGB">
     <feComponentTransfer>
       <feFuncR type="discrete" tableValues="0 0.25 0.75 1.0"/>
       <feFuncG type="discrete" tableValues="0 0.25 0.75 1.0"/>
       <feFuncB type="discrete" tableValues="0 0.25 0.75 1.0"/>
     </feComponentTransfer>
     </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/4/42/Andy_Warhol_1975.jpg" x="20px"
y="20px" width="300px" height="600px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#posterize)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/4/42/Andy_Warhol_1975.jpg"
x="340px" y="20px" width="300px" height="600px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

Filtros de desenfoque: resaltar desenfoque

Este filtro selecciona solo las áreas de alta luminancia de un gráfico de origen, difumina los contenidos y compone el contenido borroso sobre el original.

```
<svg width="800px" height="600px">
    <defs>
        <filter id="highlightblur" color-interpolation-filters="sRGB">
           <feColorMatrix type="luminanceToAlpha" in="SourceGraphic" result="lumMap"/>
           <feComponentTransfer in="lumMap" result="highlightMask">
              <feFuncA type="discrete" tableValues="0 0 0 0 0 0 0 1"/>
           </feComponentTransfer>
           <feComposite operator="in" in="SourceGraphic" in2="highlightMask"
result="highlights"/>
           <feGaussianBlur in="highlights" stdDeviation="3" result="highBlur"/>
           <feComposite operator="over" in="highBlur" in2="SourceGraphic" result="final"/>
        </filter>
     </defs>

    <image filter="url(#highlightblur)" x="0" y="-40" width="780" height="600"
preserveAspectRatio="true"
xlink:href="http://i554.photobucket.com/albums/jj424/allbowerpower/Christmas%202009/ChristmasTablesett

    />
</svg>
```

Lea Filtros en línea: https://riptutorial.com/es/svg/topic/3262/filtros

Capítulo 14: Gradientes

Parámetros

común	definición
gradientUnits	El sistema de coordenadas de los atributos de gradiente. Ya sea objectBoundingBox o userSpaceOnUse
gradientTransform	La transformación para aplicar a los contenidos de gradiente.
Método de propagación	Define lo que sucede fuera de los límites del gradiente. O bien el pad, reflexionar o repetir
xlink: href	enlace a otro gradiente que proporciona atributos o contenido
------	------
Gradiente lineal	Definición
------	------
x1	define el vector gradiente
x2	ver x1
y1	ver x1
y2	ver x1
------	------
Gradiente radial	Definición
------	------
cx	la coordenada x del centro del gradiente externo
cy	la coordenada y del centro del gradiente externo
r	El radio exterior del gradiente. La ubicación de una parada del 100%.
fx	La coordenada x del centro de gradiente interno. La ubicación de una parada del 0%.
fy	La ubicación y del centro de gradiente interno. La ubicación de una parada del 0%.

Observaciones

SVG distingue entre mayúsculas y minúsculas, así que recuerde usar una G mayúscula en el medio.

Examples

gradiente lineal

```
<svg>
  <defs>
        <linearGradient id='g' y1="100%" x2="100%">
            <stop offset='0%' stop-color='yellow' />
            <stop offset='100%' stop-color='green' />
        </linearGradient>
    </defs>
    <rect width='100%' height='100%' fill='url(#g)'/>
</svg>
```

RadialGradiente

```
<svg>
  <defs>
    <radialGradient id="g">
      <stop offset="10%" stop-color="green" />
      <stop offset="90%" stop-color="white" />
    </radialGradient>
  </defs>

  <rect width='100%' height='100%' fill='url(#g)'/>
</svg>
```

Lea Gradientes en línea: https://riptutorial.com/es/svg/topic/3346/gradientes

Capítulo 15: Línea

Parámetros

Atributo	Descripción
x1	Posición horizontal de inicio de línea.
y1	Posición vertical de inicio de línea.
x2	Posición horizontal de final de línea.
y2	Posición vertical de final de línea.
carrera	Color de la línea.
anchura del trazo	Ancho de línea.
opacidad al golpe	Opacidad de línea.
trazo de trazo	Patrón de guión para la línea.
trazo de línea	Cómo se renderizan los extremos de línea

Observaciones

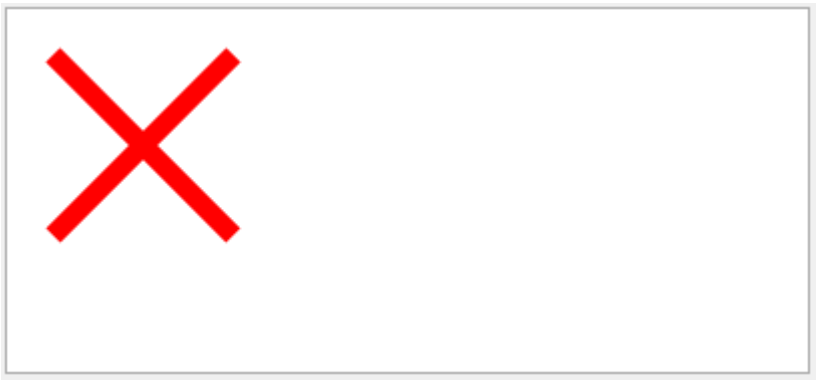
Puede encontrar información detallada sobre el elemento 'línea' de SVG en la Recomendación W3C para SVG .

Examples

Dibuja una cruz usando líneas diagonales rojas.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <line x1="10" y1="10" x2="100" y2="100" stroke="red" stroke-width="10" />
    <line x1="100" y1="10" x2="10" y2="100" stroke="red" stroke-width="10" />
</svg>
```

Resultado:



Dibujo de línea discontinua con trazo-tablero

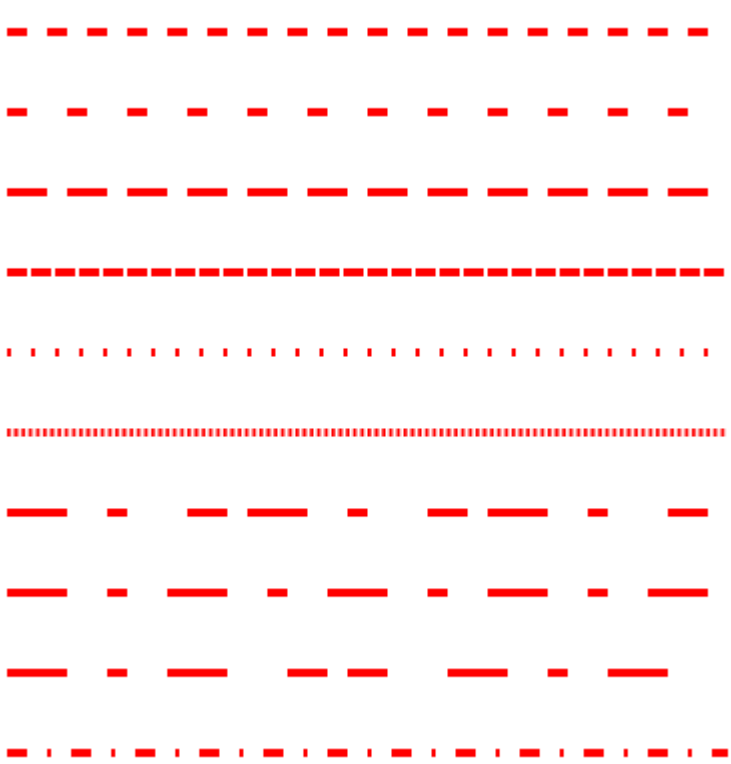
```
<svg width="400px" height="400px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <line x1="10" y1="10" x2="300" y2="10" stroke="red" stroke-width="10" stroke-
dasharray="20,2,5,2"/>
</svg>
```

Resultado

Diferentes ejemplos de trazo-tablero:

```
<svg width="200" height="200" viewBox="0 0 200 200" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <line stroke-dasharray="5, 5"              x1="10" y1="10" x2="190" y2="10" />
  <line stroke-dasharray="5, 10"             x1="10" y1="30" x2="190" y2="30" />
    <line stroke-dasharray="10, 5"             x1="10" y1="50" x2="190" y2="50" />
    <line stroke-dasharray="5, 1"              x1="10" y1="70" x2="190" y2="70" />
    <line stroke-dasharray="1, 5"              x1="10" y1="90" x2="190" y2="90" />
    <line stroke-dasharray="0.9"               x1="10" y1="110" x2="190" y2="110" />
    <line stroke-dasharray="15, 10, 5"         x1="10" y1="130" x2="190" y2="130" />
    <line stroke-dasharray="15, 10, 5, 10"     x1="10" y1="150" x2="190" y2="150" />
    <line stroke-dasharray="15, 10, 5, 10, 15" x1="10" y1="170" x2="190" y2="170" />
    <line stroke-dasharray="5, 5, 1, 5"        x1="10" y1="190" x2="190" y2="190" />
    <style><![CDATA[
    line{
        stroke: red;
        stroke-width: 2;
    }
    ]]></style>
 </svg>
```

Resultado:



Alternativas de tapón de línea usando stroke-linecap

```
<svg width="600px" height="400px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">

    <line x1="10" y1="20" x2="300" y2="20" stroke="red" stroke-width="20" stroke-
linecap="butt"/>
      <text x="320" y="20">stroke-linecap="butt" (default)</text>
    <line x1="10" y1="70" x2="300" y2="70" stroke="red" stroke-width="20" stroke-
linecap="round"/>
      <text x="320" y="70">stroke-linecap="round"</text>
    <line x1="10" y1="120" x2="300" y2="120" stroke="red" stroke-width="20" stroke-
linecap="square"/>
      <text x="320" y="120">stroke-linecap="square"</text>
</svg>
```

Resultado

Lea Línea en línea: https://riptutorial.com/es/svg/topic/3034/linea

Capítulo 16: marcador

Sintaxis

- `<marker` viewBox = " *xy width height* " refX = " *xoffset* " refY = " *yoffset* " orient = " *orientación* " ... *parámetros opcionales* >
- *... elementos dibujando el marcador ...*
- `</marker` >
- < *elementname* marker-start = "url (# *markerid*)" /> aplica un marcador al inicio de un elemento
- < *elementname* marker-mid = "url (# *markerid*)" /> aplica un marcador a la mitad de un segmento de un elemento
- < *elementname* marker-end = "url (# *markerid*)" /> aplica un marcador al final de un elemento
- Los marcadores se pueden aplicar a los elementos `<line>` , `<polyline>` , `<polygon>` y `<path>`

Parámetros

Parámetro	Detalles
viewBox	Especifica el sistema de unidades para los elementos que dibujan el marcador.
refX	Distancia del eje x del sistema de coordenadas para dibujar el marcador debe estar desplazado del punto de dibujo predeterminado. El valor predeterminado es 0.
refi	Distancia del eje y del sistema de coordenadas para dibujar el marcador debe estar desplazado del punto de dibujo predeterminado. El valor predeterminado es 0.
orientar	Los valores son `auto` o de `angle in degrees` y especifican la rotación aplicada al marcador. Se aplica después de realizar todos los demás ajustes de coordenadas (viewBox, preserveAspectRaio y refX, refY). El valor predeterminado es 0. El cálculo del ángulo para `auto` es complejo: consulte la especificación SVG para obtener más detalles.
markerUnits	`strokeWidth` o `userSpaceOnUse` . Predeterminado a `strokeWidth` .
marcador Ancho	Ancho del marcador en markerUnits. El valor predeterminado es 3.
marcadorHeight	Altura del marcador en markerUnits. El valor predeterminado es 3

Observaciones

Secuencias de comandos: los elementos marcadores representados no se exponen en el DOM, por lo que es imposible ajustar las propiedades o los elementos para marcadores representados específicos (aunque es completamente posible crear secuencias de comandos del elemento marcador definido).

La propiedad de `overflow` del elemento marcador se establece automáticamente en `hidden` . Esto es lo que recorta cualquier dibujo que desborda el marcador. Esto se puede establecer explícitamente como `visible` en CSS. A partir de julio de 2016, Chrome no admite marcadores con `overflow: visible` , pero una solución es establecer un filtro en el elemento marcador, que parece deshabilitar el recorte de desbordamiento.

Los filtros se pueden aplicar a los elementos dentro de un marcador. Aunque no está explícitamente permitido en la especificación, los filtros también parecen funcionar cuando se especifican en el elemento marcador.

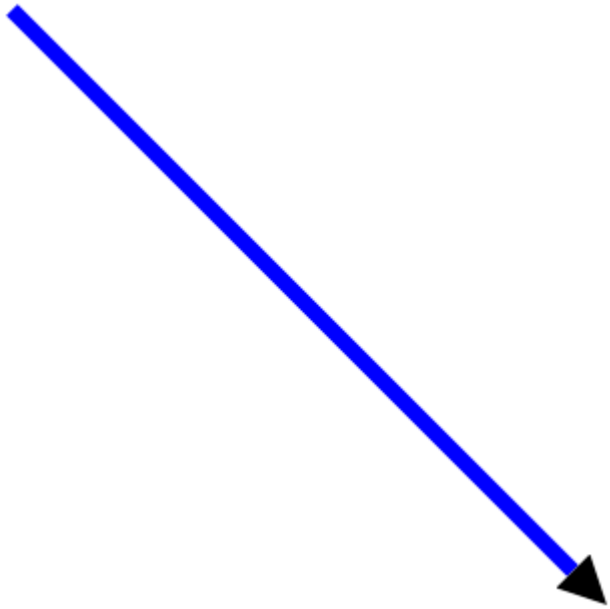
Para obtener más detalles sobre el elemento marcador, consulte la sección de marcadores en la especificación SVG 1.1 .

Examples

Marcador basico

Este es un ejemplo de un marcador final especificado con un número mínimo de parámetros. Tenga en cuenta que el marcador no hereda el color del trazo del elemento original.

```
<svg width="800px" height="600px">
<defs>
      <marker id="examplemarker"
        viewBox="0 0 10 10"
        refX="0" refY="5"
        orient="auto">
        <path d="M 0 0 L 10 5 L 0 10 z" />
      </marker>
  </defs>

  <line x1="20" y1="20" x2="300" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#examplemarker)" />
</svg>
```

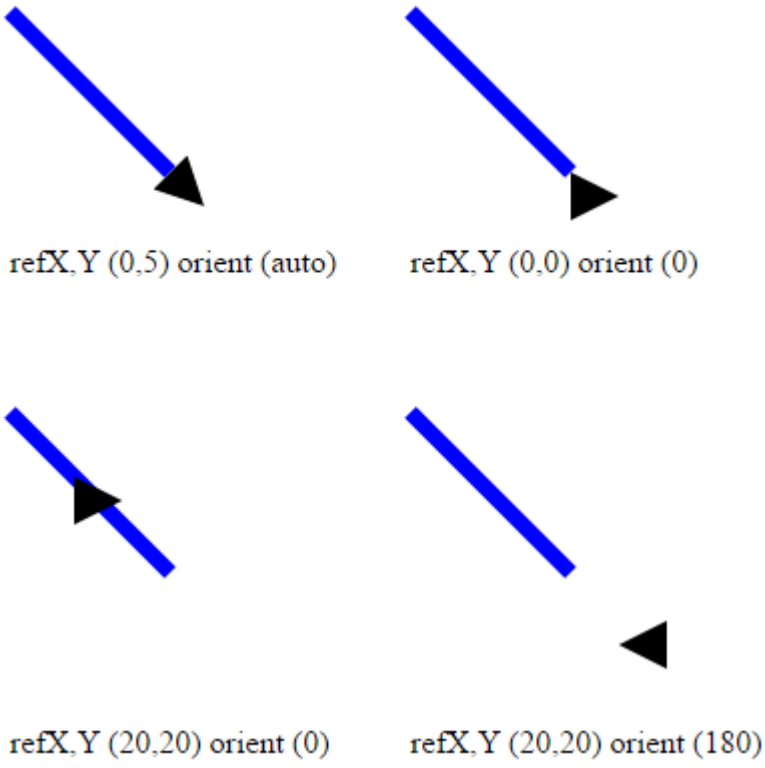


Efectos de valores alternativos para refX, refY y orient

Las compensaciones de eje para dibujar el marcador que se especifican mediante `refX` y `refY` se aplican *antes de* la rotación especificada por el parámetro `orient` . A continuación puede ver los efectos de varias combinaciones de `orient` y `refX` , `refY` para ilustrar esto.

```
<svg width="800px" height="600px">
<defs>
      <marker id="marker1"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" >
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker2"
      viewBox="0 0 10 10" refX="0" refY="0" orient="0" >
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker3"
      viewBox="0 0 10 10" refX="20" refY="20" orient="0" >
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker4"
      viewBox="0 0 10 10" refX="20" refY="20" orient="180" >
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
  </defs>

  <line  x1="20" y1="20" x2="100" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker1)" />

  <text x="20" y="150"> refX,Y  (0,5) orient (auto) </text>

  <line  x1="220" y1="20" x2="300" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker2)" />

    <text x="220" y="150"> refX,Y  (0,0) orient (0) </text>

  <line  x1="20" y1="220" x2="100" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker3)" />
```

```
      <text x="20" y="390"> refX,Y  (20,20) orient (0) </text>

  <line  x1="220" y1="220" x2="300" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker4)" />

      <text x="220" y="390"> refX,Y  (20,20) orient (180) </text>
</svg>
```

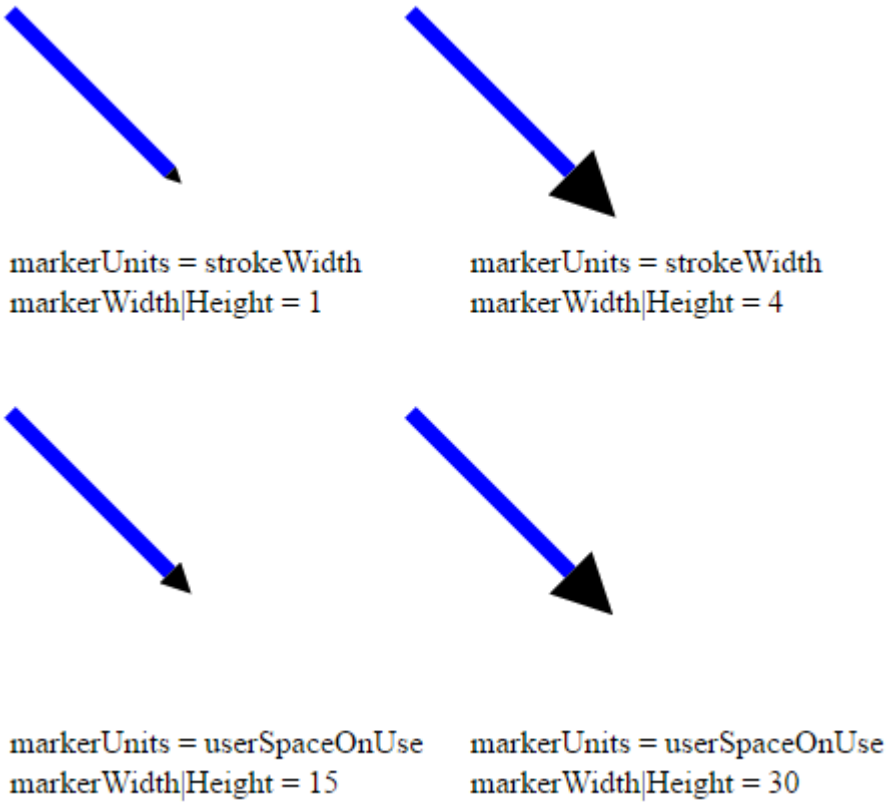


Efectos de valores alternativos para las Unidades de marcadores, Ancho de marcador, Ancho de marcador

El valor predeterminado para los marcadores de dibujo es utilizar el ancho del trazo del elemento de llamada, pero puede especificar explícitamente que los marcadores se dibujen utilizando el sistema de unidades para el elemento al que se aplica el marcador especificando `markerUnits="userSpaceOnUse"` . Los marcadores se dibujan en un cuadro de 3x3 markerUnits (3 strokewidths si no se especifican markerUnits). Pero el ancho y el alto del cuadro se pueden especificar explícitamente con `markerHeight` y `markerWidth` . Vea a continuación los efectos de varias combinaciones de `markerUnits` , `markerHeight` y `markerWidth` .

```
<svg width="800px" height="600px">
<defs>
      <marker id="marker1"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="strokeWidth"
markerWidth="1" markerHeight="1">
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                  </marker>
      <marker id="marker2"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="strokeWidth"
markerWidth="4" markerHeight="4">
```

```
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker3"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="userSpaceOnUse"
markerWidth="15" markerHeight="15">
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker4"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="userSpaceOnUse"
markerWidth="30" markerHeight="30">
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
  </defs>

  <line  x1="20" y1="20" x2="100" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker1)" />
    <text x="20" y="150"> markerUnits = strokeWidth </text>
    <text x="20" y="170"> markerWidth|Height = 1 </text>

  <line  x1="220" y1="20" x2="300" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker2)" />
    <text x="250" y="150"> markerUnits = strokeWidth </text>
    <text x="250" y="170"> markerWidth|Height = 4 </text>

  <line  x1="20" y1="220" x2="100" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker3)" />
      <text x="20" y="390"> markerUnits = userSpaceOnUse </text>
      <text x="20" y="410"> markerWidth|Height = 15 </text>

  <line  x1="220" y1="220" x2="300" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker4)" />
      <text x="250" y="390"> markerUnits = userSpaceOnUse </text>
      <text x="250" y="410"> markerWidth|Height = 30 </text>
</svg>
```

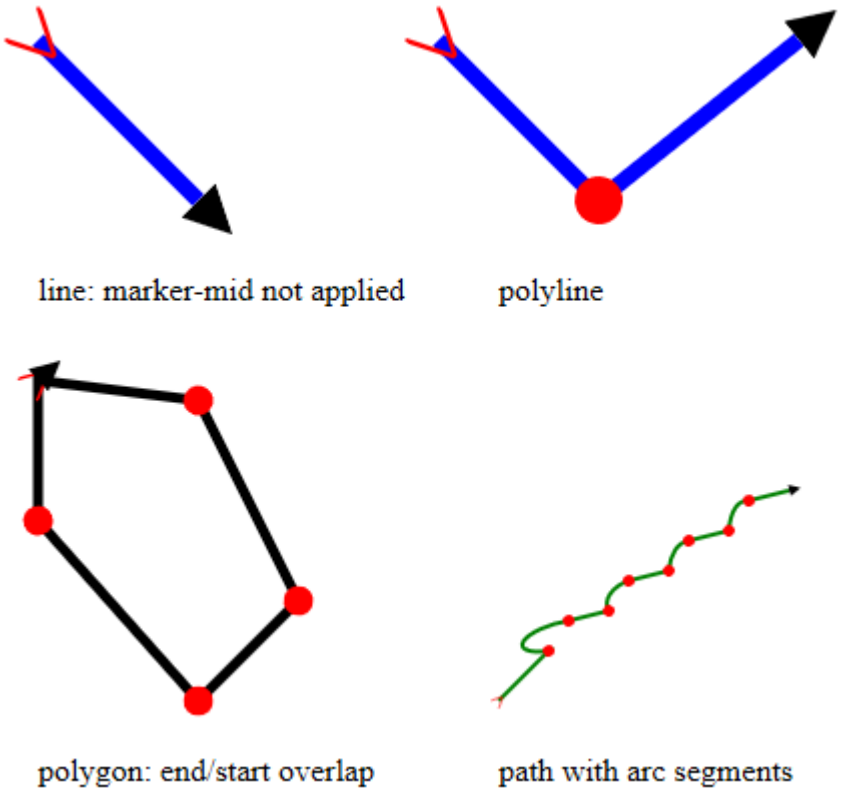


Marcadores de inicio, medio y final en línea, polilínea, polígono y elementos de trayectoria.

Los elementos pueden especificar marcadores de inicio, medio y final por separado. A continuación, se incluyen ejemplos de marcadores de inicio, medio y final para todos los elementos que se pueden marcar. Tenga en cuenta que Chrome actualmente (julio de 2016) no calcula correctamente la orientación automática para los marcadores de inicio y fin de los polígonos (error # 633012), y tampoco coloca correctamente los marcadores centrales para los segmentos de la trayectoria del arco (error # 583097)

```
<svg width="800px" height="600px">
<defs>
      <marker id="red-chevron"
      viewBox="0 0 10 10" refX="5" refY="5" orient="auto" >
         <path d="M 0 0 L 10 5 L 0 10" fill="none" stroke="red" />
      </marker>

      <marker id="black-arrow"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto">
         <path d="M 0 0 L 10 5 L 0 10 z" />
      </marker>

      <marker id="red-circle"
      viewBox="0 0 10 10" refX="5" refY="5" orient="auto" >
         <circle fill="red" cx="5" cy="5" r="5" />
      </marker>
  </defs>

  <line  x1="20" y1="20" x2="100" y2="100" stroke-width="8" stroke="blue" marker-
```

```
start="url(#red-chevron)" marker-end="url(#black-arrow)" marker-mid="url(#red-circle)" />
  <text x="20" y="150"> line: marker-mid not applied</text>

  <polyline  points="220,20 300,100 400,20" fill="none" stroke-width="8" stroke="blue" marker-
start="url(#red-chevron)" marker-end="url(#black-arrow)" marker-mid="url(#red-circle)" />
    <text x="250" y="150"> polyline </text>

  <polygon  points="20,190 100,200 150,300 100,350 20,260" marker-start="url(#red-chevron)"
marker-end="url(#black-arrow)" marker-mid="url(#red-circle)" fill="none" stroke-width="5"
stroke="black" />
   <text x="20" y="390"> polygon: end/start overlap </text>

  <path d="M250,350 l 25,-25
        a15,5 -16 0,1 10,-15 l 20,-5
        a15,10 -16 0,1 10,-15 l 20,-5
        a15,25 -16 0,1 10,-15 l 20,-5
        a15,35 -16 0,1 10,-15 l 20,-5"
        fill="none" stroke="green" stroke-width="2" marker-start="url(#red-chevron)" marker-
end="url(#black-arrow)" marker-mid="url(#red-circle)"  />
      <text x="250" y="390"> path with arc segments </text>
</svg>
```



Lea marcador en línea: https://riptutorial.com/es/svg/topic/4839/marcador

Capítulo 17: máscara

Introducción

El elemento de `mask` permite "recortar" con bordes suaves. Puede componer máscaras a partir de múltiples elementos, incluido el texto. Todo lo de una máscara que sea blanca será completamente opaco. Todo lo que sea negro será completamente transparente. Los valores entre blanco y negro serán semitransparentes.

Observaciones

Tenga en cuenta que las máscaras son una operación costosa computacional. El cálculo debe realizarse para cada píxel en el área de la máscara. Así que mantén el área de tu máscara lo más pequeña posible.

Examples

mascara basica

Un rectángulo verde con un orificio redondo en el centro que muestra el fondo de la imagen debajo.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <mask id="myMask">
        <rect x="0" y="0" width="100" height="100" fill="white"/>
        <circle cx="50" cy="50" r="45" fill="black"/>
    </mask>
    <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
    <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

Ejemplo complejo con texto y formas.

Un rectángulo verde con una máscara compleja aplicada que muestra la imagen de fondo.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <mask id="myMask0">
        <circle cx="50" cy="50" r="30" fill="white"/>
    </mask>
    <mask id="myMask">
        <rect x="0" y="0" width="100" height="100" fill="white"/>
        <text x="5" y="60" font-size="40">Mask</text>
        <circle cx="50" cy="50" r="30" fill="black"/>
        <text x="5" y="60" font-size="40" mask="url(#myMask0)" fill="white">Mask</text>
    </mask>
```

```
    <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
    <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

semi transparencia

un rect verde (de nuevo ...) con 4 orificios creados con 4 valores de escala de grises que dan como resultado 4 opacidades diferentes.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <mask id="myMask">
        <rect x="0" y="0" width="100" height="100" fill="white"/>
        <circle cx="25" cy="25" r="20" fill="black"/>
        <circle cx="75" cy="25" r="20" fill="#333"/>
        <circle cx="25" cy="75" r="20" fill="#666"/>
        <circle cx="75" cy="75" r="20" fill="#999"/>
    </mask>
    <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
    <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

una máscara con un degradado

Un rect verde con un agujero en el medio, con bordes suaves.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <radialGradient id="rg">
      <stop offset="0" stop-color="black"/>
      <stop offset="1" stop-color="white"/>
    </radialGradient>
    <mask id="myMask">
        <rect x="0" y="0" width="100" height="100" fill="white"/>
        <circle cx="50" cy="50" r="45" fill="url(#rg)"/>
    </mask>
    <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
    <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

Lea máscara en línea: https://riptutorial.com/es/svg/topic/8143/mascara

Capítulo 18: Patrones

Parámetros

parámetro	descripción
Unidades de patrón	el sistema de coordenadas de los atributos de patrón, ya sea objectBoundingBox (predeterminado) o userSpaceOnUse
patternContentUnits	el sistema de coordenadas de los contenidos del patrón ya sea objectBoundingBox o userSpaceOnUse (predeterminado)
patternTransform	La transformación a aplicar a los contenidos del patrón.
X	el desplazamiento x del patrón (el valor predeterminado es cero)
y	el desplazamiento y del patrón (el valor predeterminado es cero)
anchura	el ancho del patrón (requerido)
altura	la altura del patrón (requerido)
xlink: href	Enlace a otro patrón que proporciona algunos atributos o contenido.
preservar la relación de aspecto	si la relación de aspecto del patrón debe ser preservada

Observaciones

De forma predeterminada, el patrón se creará en mosaico al establecer el centro de la unidad de patrón en la esquina superior izquierda de la forma.

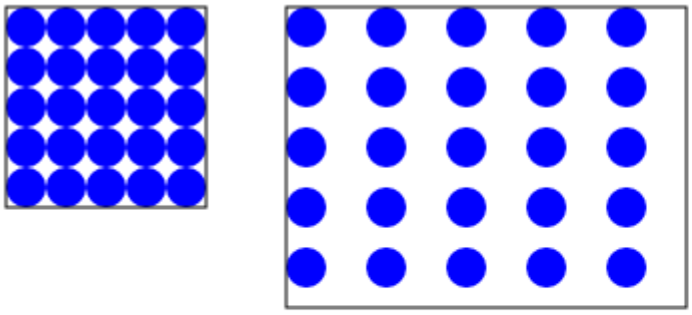
Examples

Ejemplo de patrón con unidades objectBoundingBox

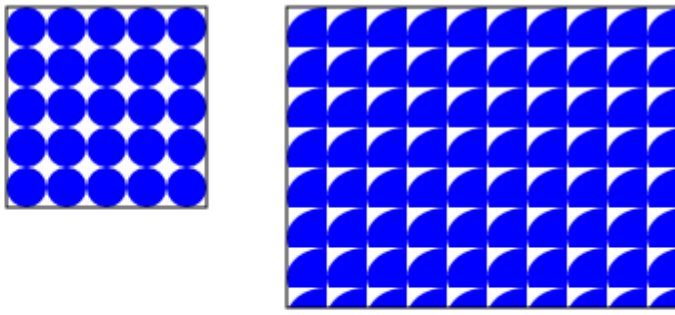
```
<svg width="400" height="400">
<defs>
  <pattern id="pattern1" width="0.2" height="0.2" patternUnits="objectBoundingBox">
      <circle cx="10" cy="10" r="10" fill="#0000ff" />
  </pattern>
</defs>

<rect x="10" y="10" width="100" height="100" stroke="black" fill="url(#pattern1)"/>
</svg>
```

Cobertura de patrones con combinaciones de Unidades de patrones y Unidades de contenido de patrones

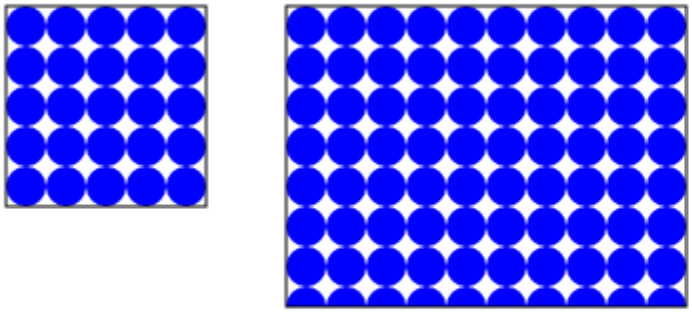
Los patrones SVG se comportan de manera significativamente diferente a las imágenes de fondo CSS al rellenar formas equivalentes. Esto puede llevar a sorpresas significativas para los recién llegados a SVG. A continuación se muestran ejemplos de un patrón definido en todas las combinaciones posibles de patternUnits y patternContentUnits, que muestran cómo estas configuraciones afectan el comportamiento de relleno.



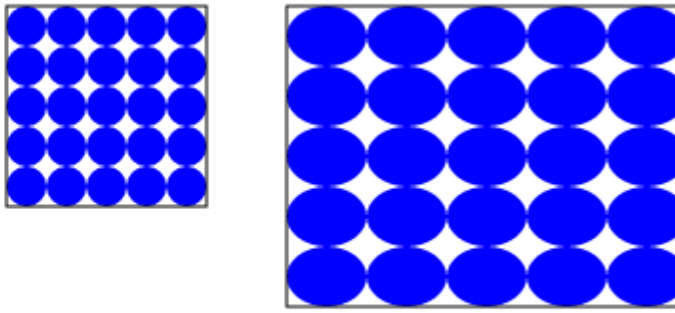
patternUnits="objectBoundingBox" (20% of shape)
patternContentUnits="userSpaceOnUse" (20px circle)
(Units used by default)



patternUnits="userSpaceOnUse" (10px square box)
patternContentUnits="objectBoundingBox"(radius=



patternUnits="userSpaceOnUse" (10px square box)
patternContentUnits="userSpaceOnUse" (20px circle)



patternUnits="objectBoundingBox" (20% of shape)
patternContentUnits="objectBoundingBox"(radius=

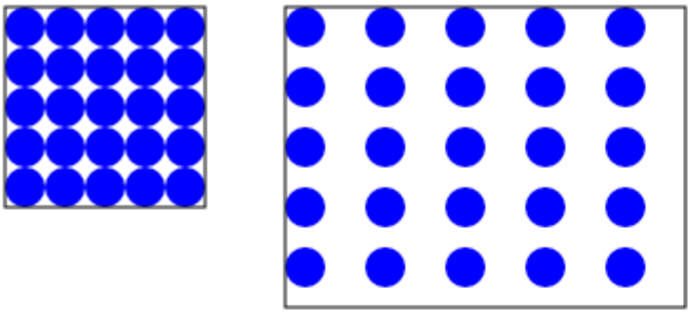
```
<svg width="800px" height="800px">
<defs>
<pattern id="pattern1" x="0" y="0" width="0.2" height="0.2"  patternUnits="objectBoundingBox"
patternContentUnits="userSpaceOnUse">
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

    <pattern id="pattern2" x="10" y="10" width="20" height="20"  patternUnits="userSpaceOnUse"
patternContentUnits="objectBoundingBox">
      <circle cx=".1" cy=".1" r="0.1" fill="blue" />
  </pattern>

    <pattern id="pattern3" x="10" y="10" width="20" height="20"  patternUnits="userSpaceOnUse"
patternContentUnits="userSpaceOnUse">
```

```
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

    <pattern id="pattern4" x="0" y="0" width="0.2" height="0.2"
patternUnits="objectBoundingBox" patternContentUnits="objectBoundingBox">
      <circle cx=".1" cy=".1" r="0.1" fill="blue" />
  </pattern>
</defs>

<rect x="10" y="10" width="100" height="100" stroke="black" fill="url(#pattern1)"/>
<rect x="150" y="10" width="200" height="150" stroke="black" fill="url(#pattern1)"/>
  <text x="10" y="200">patternUnits="objectBoundingBox" (20% of shape)</text>
  <text x="10" y="220">patternContentUnits="userSpaceOnUse" (20px circle) </text>
  <text x="10" y="240" stroke="blue" stroke-width="1">(Units used by default)</text>

<rect x="10" y="310" width="100" height="100" stroke="black" fill="url(#pattern3)"/>
<rect x="150" y="310" width="200" height="150" stroke="black" fill="url(#pattern3)"/>
  <text x="10" y="500">patternUnits="userSpaceOnUse" (10px square box)</text>
  <text x="10" y="520">patternContentUnits="userSpaceOnUse" (20px circle) </text>

<rect x="410" y="10" width="100" height="100" stroke="black" fill="url(#pattern2)"/>
<rect x="550" y="10" width="200" height="150" stroke="black" fill="url(#pattern2)"/>
  <text x="410" y="200">patternUnits="userSpaceOnUse" (10px square box)</text>
  <text x="410" y="220">patternContentUnits="objectBoundingBox"(radius="10%") </text>

<rect x="410" y="310" width="100" height="100" stroke="black" fill="url(#pattern4)"/>
<rect x="550" y="310" width="200" height="150" stroke="black" fill="url(#pattern4)"/>
  <text x="410" y="500">patternUnits="objectBoundingBox" (20% of shape)</text>
  <text x="410" y="520">patternContentUnits="objectBoundingBox"(radius="10%")  </text>

</svg>
```

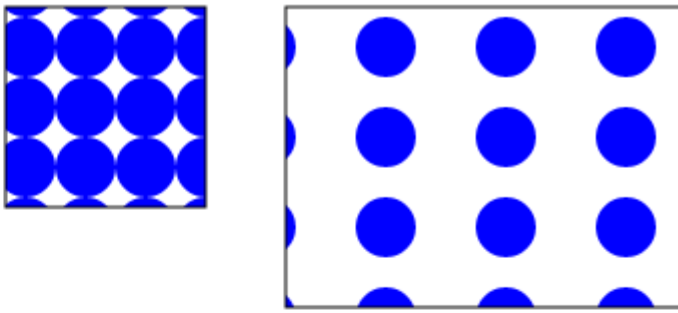
ejemplos de patrones de transformación

```
<svg width="800px" height="800px">
<defs>
<pattern id="pattern1" x="0" y="0" width="0.2" height="0.2" >
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

  <pattern id="pattern2" x="0" y="0" width="0.2" height="0.2" patternTransform="scale(1.5)">
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

  <pattern id="pattern3" x="0" y="0" width="0.2" height="0.2"   patternTransform="skewX(45)">
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

  <pattern id="pattern4" x="0" y="0" width="0.2" height="0.2"  patternTransform="matrix(1.5,-
.70,.10,1.1,-30,10)">
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>


</defs>

<rect x="10" y="10" width="100" height="100" stroke="black" fill="url(#pattern1)"/>
<rect x="150" y="10" width="200" height="150" stroke="black" fill="url(#pattern1)"/>
  <text x="10" y="200">Original</text>
```

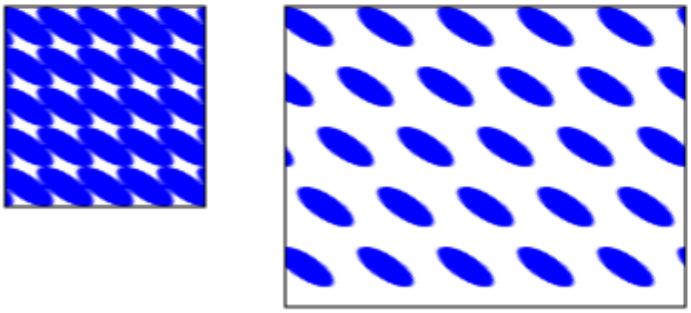
```
<rect x="410" y="10" width="100" height="100" stroke="black" fill="url(#pattern2)"/>
<rect x="550" y="10" width="200" height="150" stroke="black" fill="url(#pattern2)"/>
  <text x="410" y="200">patternTransform="scale(1.5)"</text>

<rect x="10" y="310" width="100" height="100" stroke="black" fill="url(#pattern3)"/>
<rect x="150" y="310" width="200" height="150" stroke="black" fill="url(#pattern3)"/>
  <text x="10" y="500">patternTransform="skewX(45)"</text>

<rect x="410" y="310" width="100" height="100" stroke="black" fill="url(#pattern4)"/>
<rect x="550" y="310" width="200" height="150" stroke="black" fill="url(#pattern4)"/>
  <text x="410" y="500">patternUnits="matrix(1.5,-.70,.10,1.1,-30,10)"</text>

</svg>
```



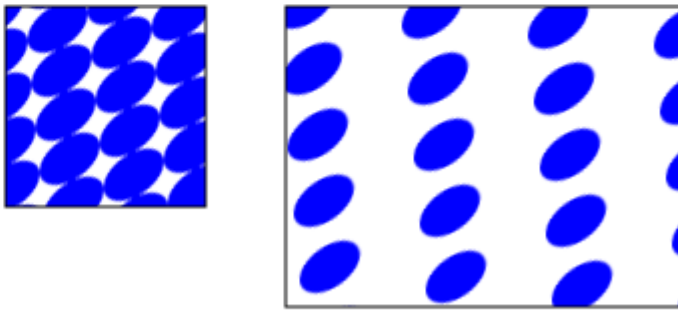
Original



patternTransform="scale(1.5)"



patternTransform="skewX(45)"



patternUnits="matrix(1.5,-.70,.10,1.1,-30,10)"

Lea Patrones en línea: https://riptutorial.com/es/svg/topic/3251/patrones

Capítulo 19: Polilínea

Sintaxis

- `<polyline points="10,5 25,15 20,10" />`

Parámetros

Parámetro	Detalles
puntos	El atributo de puntos define una lista de puntos. Cada punto está definido por las coordenadas ax y ay en el sistema de coordenadas del usuario.
anchura del trazo	Ancho de carrera
opacidad al golpe	Opacidad del trazo
trazo de trazo	(Opcional) Especifica el patrón de guión para el trazo
trazo de línea	(Opcional) Especifica si el final de la línea debe estar al ras, redondo o cuadrado ("tope" (predeterminado) / "redondo" / "cuadrado")
línea de trazo unido	(Opcional) Especifica cómo se deben unir los segmentos de línea: a inglete, redondeado o biselado ("mitre" (predeterminado) / "redondo" / "biselado")
golpe-miterlimit	(Opcional) Especifica la dimensión máxima de una mitra. Las combinaciones de ingletes que superan este límite se convierten en una combinación de bisel. Predeterminado = "4"

Examples

SVG incluyendo una polilínea

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <polyline points="10,5 25,15 20,10" />
</svg>
```

Polilíneas con líneas alternativas, casquetes y miterlimits

```
<svg width="600px" height="600px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">

    <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
```

```
stroke="red" stroke-width="10" />

      <text x="320" y="20">Default drawing stroke</text>

      <g transform="translate(0,150)">
      <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" stroke-linecap="butt" stroke-linejoin="miter" stroke-
miterlimit="2"/>

      <text x="320" y="20">stroke-linecap="butt" (default)</text>
      <text x="320" y="40">stroke-linejoin="miter"(default)</text>
      <text x="320" y="60">stroke-miterlimit="2"</text>
  </g>

  <g transform="translate(0,300)">
      <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" stroke-linecap="round" stroke-linejoin="round" />

      <text x="320" y="20">stroke-linecap="round" </text>
      <text x="320" y="40">stroke-linejoin="round" </text>

  </g>
  <g transform="translate(0,450)">
      <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" stroke-linecap="square" stroke-linejoin="bevel"/>

      <text x="320" y="20">stroke-linecap="square"</text>
      <text x="320" y="40">stroke-linejoin="bevel"</text>
  </g>

</svg>
```

Resultado

Default drawing stroke

stroke-linecap="butt" (default)
stroke-linejoin="miter"(default)
stroke-miterlimit="2"

stroke-linecap="round"
stroke-linejoin="round"

stroke-linecap="square"
stroke-linejoin="bevel"

Lea Polilínea en línea: https://riptutorial.com/es/svg/topic/3842/polilinea

Capítulo 20: Rectángulo

Parámetros

Atributo	Descripción
X	Posición horizontal del rectángulo desde el margen izquierdo.
y	Posición vertical del rectángulo desde el margen superior.
anchura	Ancho del rectángulo.
altura	Altura del rectángulo.
rx	Radio horizontal de elipse usado para redondear esquinas de rectángulo
ry	Radio vertical de elipse usado para redondear esquinas de rectángulo
carrera	Color del borde del rectángulo.
anchura del trazo	Ancho del borde del rectángulo.
llenar	Color *dentro del* borde del rectángulo.

Observaciones

Se puede encontrar información detallada sobre el elemento 'rect' de SVG en la Recomendación W3C para SVG .

Examples

Dibuja un rectángulo negro sin relleno.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="10" y="10" width="50" height="100" stroke="black" stroke-width="5" fill="none" />
</svg>
```

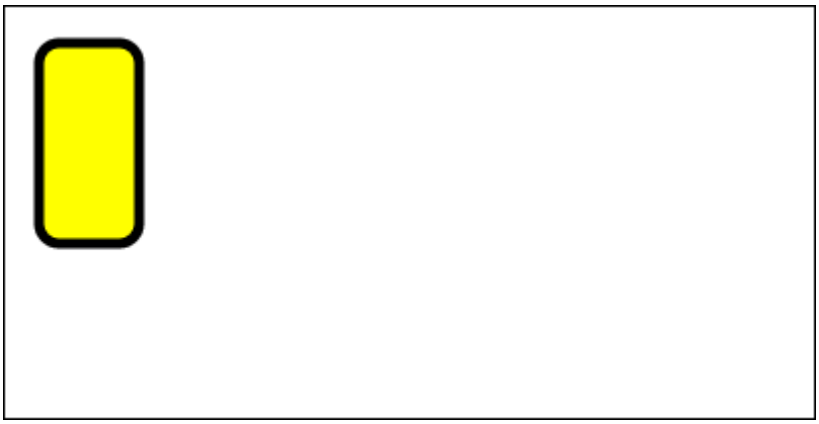
Resultado:

Dibuja un rectángulo negro con relleno amarillo y esquinas redondeadas

- Los atributos de `width` y `height` designan las dimensiones del rectángulo. Estos valores están en píxeles por defecto
- El valor de `fill` establece el color para el rectángulo. Si no se especifica ningún valor para el `fill` , el negro se utiliza de forma predeterminada

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="10" y="10" width="50" height="100" rx="10" ry="10" stroke="black" stroke-
width="5" fill="yellow" />
</svg>
```

Resultado:



Lea Rectángulo en línea: https://riptutorial.com/es/svg/topic/2993/rectangulo

Capítulo 21: Scripting

Observaciones

Las secuencias de comandos SVG que utilizan las interfaces DOM nativas se encuentran actualmente (2016) en un estado de leve cambio. El estándar SVG actual (1.1) se implementa bien en la mayoría de los principales navegadores web. Sin embargo, como el estándar SVG 2.0 está en desarrollo, algunos navegadores han comenzado a eliminar las funciones SVG 1.1 que quedarán obsoletas en 2.0. Puede ver una lista completa de los cambios propuestos desde SVG 1.1 a SVG 2.0 en el Apéndice L de SVG 2.0 .

Reemplazo de `pathSegList` y otro uso de `SVGPathSeg`

En SVG 1.1, los elementos `<path>` están definidos para tener una propiedad `pathSegList` que le da acceso a una representación nativa de todos los comandos de ruta . Google Chrome v48 eliminó esta propiedad a finales de 2015, en preparación para un reemplazo propuesto en SVG 2.0 . Hasta que se agregue la compatibilidad con SVG 2.0, debe usar un polyfill para recuperar la funcionalidad 1.1 o para implementar la API 2.0 propuesta .

Reemplazo de `getTransformToElement()`

Chrome v48 también eliminó el método `SVGGraphicsElement.getTransformToElement()` . Existe un polyfill simple para implementar el método antiguo .

Examples

Creando un elemento

La forma más sencilla de comprender cómo crear y modificar elementos SVG es operar sobre los elementos utilizando las interfaces DOM Level 2 Core , como lo haría con HTML o XML.

Es imperativo que los *elementos* creados a partir de JavaScript se creen en el mismo espacio de nombres declarado en el elemento SVG, en este ejemplo: " http://www.w3.org/2000/svg ". Sin embargo, casi todos los *atributos* de los elementos SVG no están en ningún espacio de nombres. **No se** debe colocar en el espacio de nombres SVG.

Aquí mostramos SVG alojado dentro de HTML, ya que este es un caso común:

```
<!doctype HTML>
<html><title>Creating an Element</title>
<body>
  <svg xmlns="http://www.w3.org/2000/svg"
       width="100%" height="100%"
       viewBox="0 0 400 300"></svg>

  <script>
```

```
    var svgNS = "http://www.w3.org/2000/svg";

    // Create a circle element (not part of the DOM yet)
    var circle = document.createElementNS(svgNS,'circle'); // Creates a <circle/>
    circle.setAttribute('fill','red'); // Note: NOT setAttributeNS()
    circle.setAttribute('cx',150);     // setAttribute turns 150 into a string
    circle.setAttribute('cy','80');    // using a string works, too
    circle.setAttribute('r',35);       // give the circle a radius so we can see it

    // Now, add the circle to the SVG document so we can see it
    var svg = document.querySelector('svg'); // the root <svg> element
    svg.appendChild(circle);
  </script>
</body></html>
```

Hay algunos atributos que deben crearse en un espacio de nombres particular. Son los que figuran con dos puntos en sus nombres en el Índice de Atributos SVG . Específicamente, son: `xlink:actuate` `xlink:arcrole` , `xlink:arcrole` , `xlink:href` , `xlink:role` , `xlink:show` , `xlink:title` , `xlink:type` , `xml:base` , `xml:lang` , y `xml:space` . Establezca estos atributos utilizando `setAttributeNS()` :

```
var svgNS   = "http://www.w3.org/2000/svg";
var xlinkNS = "http://www.w3.org/1999/xlink";
var img = document.createElementNS( svgNS, 'image' );
img.setAttributeNS( xlinkNS, 'href', 'my.png' );
```

Si crea elementos con frecuencia, especialmente con muchos atributos, una función auxiliar como la siguiente puede ahorrarle la escritura, evitar errores y hacer que su código sea más fácil de leer:

```
<!doctype HTML>
<html><title>Creating an Element</title>
<body>
  <svg xmlns="http://www.w3.org/2000/svg"></svg>
  <script>
     var svg = document.querySelector('svg');
     var circle = createOn( svg, 'circle', {fill:'red', cx:150, cy:80, r:35} );

    // Create an SVG element on another node with a set of attributes.
    // Attributes with colons in the name (e.g. 'xlink:href') will automatically
    // find the appropriate namespace URI from the SVG element.
    // Optionally specify text to create as a child node, for example
    //   createOn(someGroup,'text',{x:100,'text-anchor':'middle'},"Hello World!");
    function createOn(parentEl,name,attrs,text){
      var doc=parentEl.ownerDocument, svg=parentEl;
      while (svg && svg.tagName!='svg') svg=svg.parentNode;
      var el = doc.createElementNS(svg.namespaceURI,name);
      for (var a in attrs){
        if (!attrs.hasOwnProperty(a)) continue;
        var p = a.split(':');
        if (p[1]) el.setAttributeNS(svg.getAttribute('xmlns:'+p[0]),p[1],attrs[a]);
        else      el.setAttribute(a,attrs[a]);
      }
      if (text) el.appendChild(doc.createTextNode(text));
      return parentEl.appendChild(el);
    }
  </script>
```

```
</body></html>
```

Atributos de lectura / escritura

Puede usar los métodos de DOM Level 2 Core `getAttribute()` , `getAttributeNS()` , `setAttribute()` y `setAttributeNS()` para leer y escribir valores de elementos SVG, o puede usar las propiedades personalizadas y los métodos especificados en la IDL SVG 1.1 (Interfaz Lenguaje de definición).

Atributos Numéricos Simples

Por ejemplo, si tiene un elemento de círculo SVG:

```
<circle id="circ" cx="10" cy="20" r="15" />
```

puede usar métodos DOM para leer y escribir los atributos:

```
var circ = document.querySelector('#circ');
var x = circ.getAttribute('cx') * 1; // Use *1 to convert from string to number value
circ.setAttribute('cy', 25);
```

... o puede usar las propiedades `cx` , `cy` y `r` definidas para `SVGCircleElement` . Tenga en cuenta que estos no son números directos, sino que, como sucede con muchos accesores en SVG 1.1, permiten el acceso a valores animados. Estas propiedades son de tipo `SVGAnimatedLength` . Sin tener en cuenta las unidades de animación y de longitud, puede utilizar atributos como:

```
var x = circ.cx.baseVal.value; // this is a number, not a string
circ.cy.baseVal.value = 25;
```

Transformaciones

Los grupos SVG se pueden usar para mover, rotar, escalar y, de lo contrario, transformar múltiples elementos gráficos en conjunto. (Para detalles sobre traducciones de SVG, vea el Capítulo 7). Aquí hay un grupo que hace una carita feliz en la que puede ajustar el tamaño, la rotación y la ubicación ajustando la `transform` :

```
<g id="smiley" transform="translate(120,120) scale(5) rotate(30)">
  <circle r="20" fill="yellow" stroke-width="2"/>
  <path fill="none" d="M-10,5 a 5 3 0 0 0 20,0" stroke-width="2"/>
  <circle cx="-6" cy="-5" r="2" fill="#000"/>
  <circle cx="6"  cy="-5" r="2" fill="#000"/>
</g>
```

El uso de scripts para ajustar la escala de esto a través de los métodos DOM requiere manipular todo el atributo de `transform` como una cadena:

```
var face = document.querySelector('#smiley');
```

```
// Find the full string value of the attribute
var xform = face.getAttribute('transform');

// Use a Regular Expression to replace the existing scale with 'scale(3)'
xform = xform.replace( /scale\s*\([^)]+\)/, 'scale(3)' );

// Set the attribute to the new string.
face.setAttribute('transform',xform);
```

Con el SVG DOM se pueden recorrer las transformaciones específicas de la lista, encontrar la deseada y modificar los valores:

```
var face = document.querySelector('#smiley');

// Get the SVGTransformList, ignoring animation
var xforms = face.transform.baseVal;

// Find the scale transform (pretending we don't know its index)
for (var i=0; i<xforms.numberOfItems; ++i){
  // Get this part as an SVGTransform
  var xform = xforms.getItem(i);
  if (xform.type == SVGTransform.SVG_TRANSFORM_SCALE){
    // Set the scale; both X and Y scales are required
    xform.setScale(3,3);
    break;
  }
}
```

- Para recorrer y manipular el número de transformaciones, consulte `SVGTransformList` .
- Para manipular transformaciones individuales, vea `SVGTransform` .

Arrastrando elementos SVG

El uso del mouse para arrastrar un elemento SVG (o grupo de elementos) se puede lograr mediante:

1. Agregar el controlador `mousedown` para iniciar el arrastre: agregar una traducción en el elemento para usar durante el arrastre (si es necesario), realizar un seguimiento de los eventos del `mousemove` y agregar un controlador `mouseup` para finalizar el arrastre.
2. Durante el `mousemove` , transforme la posición del mouse de las coordenadas de la pantalla en las coordenadas locales para el objeto que está arrastrando y actualice la traducción según corresponda.
3. Durante `mouseup` , se eliminan los `mousemove` y `mouseup` .

```
// Makes an element in an SVG document draggable.
// Fires custom `dragstart`, `drag`, and `dragend` events on the
// element with the `detail` property of the event carrying XY
// coordinates for the location of the element.
function makeDraggable(el){
  if (!el) return console.error('makeDraggable() needs an element');
  var svg = el;
  while (svg && svg.tagName!='svg') svg=svg.parentNode;
  if (!svg) return console.error(el,'must be inside an SVG wrapper');
  var pt=svg.createSVGPoint(), doc=svg.ownerDocument;
```

```
  var root = doc.rootElement || doc.body || svg;
  var xlate, txStartX, txStartY, mouseStart;
  var xforms = el.transform.baseVal;

  el.addEventListener('mousedown',startMove,false);

  function startMove(evt){
    // We listen for mousemove/up on the root-most
    // element in case the mouse is not over el.
    root.addEventListener('mousemove',handleMove,false);
    root.addEventListener('mouseup',  finishMove,false);

    // Ensure that the first transform is a translate()
    xlate = xforms.numberOfItems>0 && xforms.getItem(0);
    if (!xlate || xlate.type != SVGTransform.SVG_TRANSFORM_TRANSLATE){
      xlate = xforms.createSVGTransformFromMatrix( svg.createSVGMatrix() );
      xforms.insertItemBefore( xlate, 0 );
    }
    txStartX=xlate.matrix.e;
    txStartY=xlate.matrix.f;
    mouseStart = inElementSpace(evt);
    fireEvent('dragstart');
  }

  function handleMove(evt){
    var point = inElementSpace(evt);
    xlate.setTranslate(
      txStartX + point.x - mouseStart.x,
      txStartY + point.y - mouseStart.y
    );
    fireEvent('drag');
  }

  function finishMove(evt){
    root.removeEventListener('mousemove',handleMove,false);
    root.removeEventListener('mouseup',  finishMove,false);
    fireEvent('dragend');
  }

  function fireEvent(eventName){
    var event = new Event(eventName);
    event.detail = { x:xlate.matrix.e, y:xlate.matrix.f };
    return el.dispatchEvent(event);
  }

  // Convert mouse position from screen space to coordinates of el
  function inElementSpace(evt){
    pt.x=evt.clientX; pt.y=evt.clientY;
    return pt.matrixTransform(el.parentNode.getScreenCTM().inverse());
  }
}
```

Lea Scripting en línea: https://riptutorial.com/es/svg/topic/5021/scripting

Capítulo 22: Texto

Parámetros

`<text>`	Detalles
X	La posición x del texto.
y	La posición y del texto.
dx	Cambio relativo en la posición x.
dy	Cambio relativo en posición y.
girar	Especifica el desplazamiento angular de los glifos de texto.
textLength	Se adapta al texto en la longitud dada.
longitud Ajustar	Especifica si solo el kerning o kerning y los glifos se comprimen / estiran para ajustar el texto en textLength especificado. Valores: espaciado o espaciado y glifos
-	**Parámetros comunes a todos los elementos de fragmentación de texto (text, tref, textPath, tspan)**
ancla de texto	Especifica la alineación horizontal. Valores: inicio, medio, final.
cambio de línea de base	Desplaza la línea de base del texto en función de los valores proporcionados por la tabla de fuentes para el posicionamiento del superíndice o subíndice (sub, super) o por un% o longitud positivo o negativo. Valores: sub, super,% o longitud.

Observaciones

baseline-shift no es compatible con las versiones más actuales de los navegadores Firefox y Microsoft a partir de julio de 2016.

Examples

Dibujar texto

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text x="40" y="60" font-size="28">Hello World!</text>
</svg>
```

Las coordenadas x e y especifican la posición de la esquina inferior izquierda del texto (a menos que se haya modificado el anclaje de texto).

Superíndice y subíndice

Usando el parámetro de cambio de línea de base, puede especificar super o subíndice. Pero esto no es compatible con todos los principales navegadores.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text x="10" y="20">x<tspan baseline-shift="super">2</tspan></text>
    <text x="10" y="60">f<tspan baseline-shift="sub">x</tspan></text>
</svg>
```

Para una solución de navegador cruzado, puede usar dy, dx y el tamaño de fuente relativo.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text x="10" y="40">x<tspan dy="-7" font-size=".7em">2</tspan></text>
    <text x="10" y="80">f<tspan dy="3" font-size=".7em">x</tspan></text>
</svg>
```

Rotar texto

La propiedad rotar gira cada carácter según el ángulo especificado.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text x="10" y="20" rotate="30">Each character is rotated</text>
</svg>
```

Para rotar todo el elemento de texto, debe usar la propiedad de transformación.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text transform="translate(10, 60) rotate(30)">The whole text is rotated</text>
</svg>
```

Posicionamiento individual de letras con matrices de valores X e Y

```
<svg width="400px" height="200px">
 <text x="1em, 2em, 3em, 4em, 5em" y="3em, 4em, 5em">
    Individually Spaced Text
  </text>
</svg>
```

El elemento Texto admite la colocación individual de letras al aceptar una matriz de valores para x e y.

Lea Texto en línea: https://riptutorial.com/es/svg/topic/3033/texto

Capítulo 23: Transformación

Observaciones

Los elementos gráficos se pueden alterar agregando un atributo de *transformación* .

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="0" width="30" height="30" transform="translate(10, 10)" />
</svg>
```

En lugar de representar el origen superior izquierdo en las coordenadas (0, 0), se mostrará hacia abajo y hacia la derecha, desde el punto (10, 10).

Se pueden transformar grupos enteros de elementos, y las transformaciones pueden sumarse entre sí.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <g transform="rotate(45)">
        <rect x="0" y="0" width="30" height="30" />
        <circle cx="5" cy="5" r="5" transform="scale(3)" />
    </g>
</svg>
```

Primero, cada punto del círculo se escalará con el factor 3 en relación con el origen, llevando el centro del círculo al centro del rectángulo en (15, 15) y su radio a 15. Luego, el rectángulo y la El círculo girará 45 grados en sentido horario alrededor del origen.

Examples

traducir

Mueve un rectángulo 10 unidades a la derecha y 20 unidades hacia abajo:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="0" width="30" height="30" transform="translate(10, 20)" />
</svg>
```

Muevalo 0 unidades horizontalmente y 20 unidades arriba:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="20" width="30" height="30" transform="translate(0, -20)" />
</svg>
```

Muévalo 10 unidades a la izquierda y 0 unidades verticalmente:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="10" y="0" width="30" height="30" transform="translate(-10)" />
```

```
</svg>
```

escala

Escala un rectángulo horizontalmente por el factor 2 y verticalmente por el factor 0.5:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="10" y="10" width="40" height="40" transform="scale(2, 0.5)" />
</svg>
```

El resultado es equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="20" y="5" width="80" height="20" />
</svg>
```

Refleja un rectángulo horizontalmente:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="0" width="20" height="40" transform="scale(-1, 1)" />
</svg>
```

La escala opera en relación con el origen, por lo que es equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="-20" y="0" width="20" height="40" />
</svg>
```

girar

Gire un polígono hacia la derecha 90 grados alrededor del origen:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 15,20" transform="rotate(90)" />
</svg>
```

El resultado es equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 0,30 -20,15" />
</svg>
```

El centro de rotación se puede dar explícitamente:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 15,20" transform="rotate(90, 15, 15)" />
</svg>
```

El resultado es equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="30,0 30,30 10,15" />
</svg>
```

skewX, skewY

sesgar un polígono horizontalmente en 45 grados:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 30,30 0,30" transform="skewX(45)" />
</svg>
```

El resultado es equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 60,30 30,30" />
</svg>
```

Los valores de x se calculan como `x + y * tan(angle)` , y los valores de y se mantienen sin cambios

sesgar un polígono verticalmente en 30 grados:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 30,30 0,30" transform="skewY(30)" />
</svg>
```

El resultado es equivalente a (permitiendo el redondeo)

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,17.32 30,47.32 0,30" />
</svg>
```

los valores de x permanecen sin cambios, los valores de y se calculan como `y + x * tan(angle)`

matriz

Aplicar una matriz de transformación a un polígono:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 30,30 0,30" transform="matrix(1,0.6,-1.2,1,40,10)" />
</svg>
```

Cada punto (x, y) se transformará aplicando la matriz (a, b, c, d, e, f) de la siguiente manera:

```
┌ x_new ┐     ┌ a  c  e ┐ * ┌ x_old ┐    ┌ x_old * a + y_old * c + e ┐
└ y_new ┘ =   └ b  d  f ┘   │ y_old │ = └ x_old * b + y_old * d + f ┘
                            └   1   ┘
```

El resultado es equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="40,10 70,28 34,58 4,40" />
</svg>
```

Transformaciones multiples

Las transformaciones se pueden concatenar y se aplican de **derecha a izquierda.**

Gire un rectángulo 90 grados y luego muévalo hacia abajo 20 unidades y hacia la derecha 20 unidades:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="-10" y="-20" width="20" height="40"
        transform="translate(20 20) rotate(90)" />
</svg>
```

El resultado es equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="10" width="40" height="20" />
</svg>
```

Lea Transformación en línea: https://riptutorial.com/es/svg/topic/3249/transformacion

Capítulo 24: Transformación

Sintaxis

- transform = " *[funciones]* * "
- traducir (x [, y])
- rotar (θ [, x, y])
- escala (x [, y])
- skewX (θ)
- skewY (θ)
- matriz (a, b, c, d, e, f)

Examples

Aplicando Transformaciones

Las transformaciones se pueden aplicar a los elementos agregando un atributo de `transform` :

```
<circle cx="0" cy="0" r="50" transform="translate(50,50)"/>
```

O a grupos de elementos incluidos en las etiquetas `<g>` :

```
<g transform="translate(50,50)">
<circle cx="0" cy="0" r="50"/>
<circle cx="0" cy="0" r="25" fill="white"/>
</g>
```

Se pueden aplicar más transformaciones al mismo elemento al agregarlas separadas por espacios:

```
<circle cx="0" cy="0" r="50" transform="translate(50,50) scale(.5)"/>
```

Funciones de transformacion

Traducir

`translate` mueve gráficos a lo largo de vectores especificados:

```
<circle cx="0" cy="0" r="50" transform="translate(50,50)"/>
```

El primer valor es la traducción x, y el segundo la y. Si se omite la y, se establecerá de forma predeterminada en 0.

Escala

`scale` cambia el tamaño de los elementos por razones específicas:

```
<circle cx="50" cy="50" r="25" transform="scale(.5,2)"/>
```

Al igual que `translate` , los argumentos son x, luego y. Sin embargo, en la `scale` , si se omite la y, se establecerá por defecto el valor de x; en otras palabras, escala el elemento sin cambiar la relación de aspecto.

Girar

`rotate` gira los elementos según los ángulos especificados.

```
<!-- <rect> used for this example because circles can't be rotated -->
<rect width="100" height="5" transform="rotate(90,50,50)"/>
```

El primer valor es el ángulo, en grados. La transformación se aplica en sentido horario. Los otros dos valores representan el punto a rotar, por defecto al origen.

Lea Transformación en línea: https://riptutorial.com/es/svg/topic/7100/transformacion

Capítulo 25: utilizar

Parámetros

Parámetro	Detalles
X	coordenada del eje x de la esquina superior izquierda
y	coordenada del eje y de la esquina superior izquierda
anchura	ancho del elemento `<use>`
altura	altura del elemento `<use>`
xlink: href	identificador de recursos (se refiere al ID de otro elemento) SVG 2 propone desaprobar esto y reemplazarlo con un atributo href simple

Observaciones

Los detalles se pueden encontrar en la Recomendación W3C para SVG , así como en la nueva Recomendación de candidato para SVG2

Examples

Usando un icono

El elemento `<use>` se usa a menudo para iconos reutilizables, en colaboración con el elemento `<symbol>` . Eso se parece a esto:

```
<svg>
    <symbol viewBox="0 0 16 16" id="icon-star">
        <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.02l4.912 3.57-1.904
5.834h.02l4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
    </symbol>
</svg>
```

Y el elemento `<use>` :

```
<svg>
    <use xlink:href="#icon-star"/>
</svg>
```

El elemento `<use>` copia el `<symbol>` y lo muestra. También puede anular los estilos en `<symbol>` en elementos `<use>` individuales, por ejemplo,

```
<style>
```

```
    .red {
        fill: red;
    }
</style>

<svg>
    <use class="red" xlink:href="#icon-star"/>
</svg>
```

Lea utilizar en línea: https://riptutorial.com/es/svg/topic/6904/utilizar

Creditos

S. No	Capítulos	Contributors
1	Empezando con SVG	almcd, Community, Robert Longson, Timothy Miller, uruloke, w5m, web-tiki
2	Animación	Joachim Schirrmacher, Michael Mullany
3	cambiar	lodz
4	Caminos	Malcolm McLean, Michael Mullany, w5m
5	Circulo	almcd, Kake_Fisk, w5m
6	clipPath	Danny_ds, lodz
7	Colores	Danny_ds, Holger Will, Joachim Schirrmacher, Michael Mullany, w5m
8	Creando fuentes	Holger Will
9	defs	Michael Mullany
10	El elemento SVG	Michael Mullany, Timothy Miller
11	Elipse	adius
12	eventos punteros	Holger Will
13	Filtros	Anko, Michael Mullany, RamenChef
14	Gradientes	Robert Longson
15	Línea	Deni Spasovski, Michael Mullany, w5m
16	marcador	Michael Mullany
17	máscara	Holger Will
18	Patrones	Michael Mullany, Robert Longson
19	Polilínea	adius, Michael Mullany
20	Rectángulo	almcd, w5m
21	Scripting	Michael Mullany, Phrogz

22	Texto	Kake_Fisk, Michael Mullany
23	Transformación	ccprog, Michael Mullany, Stephen Leppik
24	utilizar	Robert Longson, Timothy Miller