Unidad N°1 Animación con SVG y Javascript

Sitio: Campus Virtual - Centro de e-Learning UTN FRBA (https://campus.utnba.centrodeelearning.com)
Curso: Desarrollo web en HTML 5, CSS3 y Javascript (nivel avanzado)
Libro: Unidad N°1 Animación con SVG y Javascript
Imprimido por: Mariano Rolet
Día: Friday, 6 de January de 2023, 08:04

Tabla de contenidos

1. Presentación y objetivo

Presentación

Cuando queremos agregar imágenes a nuestro sitio, contamos con una amplia variedad de posibilidades al momento de elegir formatos. Pero, para poder entender cuál es la mejor opción en cada caso, tenemos que tener conocimiento de qué nos provee cada uno de estos formatos. Así, podremos decidir cuál utilizar.

En esta unidad aprenderemos sobre un formato relativamente nuevo que nos brinda una gama de herramientas y nos permite personalizar imágenes y aplicar distintos diseños: SVG.

Objetivo

Que el alumno conozca sobre SVG, sus atributos y su funcionamiento.

2. Conociendo al SVG

Al momento de trabajar con imágenes, tenemos que tener en cuenta el formato que estaremos utilizando.

Existen múltiples formatos de imagen. Saber qué formato utilizar, dependerá del contexto en el que nos encontremos y del conocimiento que tengamos sobre cada uno.

Por un lado, contamos con el formato **.jpg**, que es un formato de compresión de imagen. Si bien pierde calidad, puede elegirse cuánta calidad deseamos perder. Por lo tanto, es un formato ideal en el trabajo con fotografías o imágenes que no requieran de transparencia.

También se encuentra el formato **.gif** , que es un formato con muchísima compresión y poca calidad. Puede trabajarse para animaciones en medios digitales y conserva la transparencia, donde la calidad no sea lo más importante, como en la web. Sin embargo, lo ideal en el caso de trabajar con transparencia es hacerlo a través del formato **.png.**

Por otra parte, contamos con el formato **SVG**, el cual es relativamente nuevo y nos provee de mucha calidad e imágenes de poco peso.

Este tipo de formato nos brinda gran cantidad de herramientas y beneficios que veremos en los siguientes puntos.

¿Qué es un SVG?

Gráficos vectoriales escalables (o SVG) es un formato gráfico vectorial que se basa en XML, un lenguaje de estándar abierto que forma parte de HTML5 y que es utilizado para realizar dibujos vectoriales en dos dimensiones.

Este tipo de formato se encuentra recomendado por la W3C y nos permite crear gráficos creados a partir de datos.

Pero.. ¿Que significa que es un gráfico vectorial?¿A qué se refiere que se basa en XML?

Los gráficos vectoriales son imágenes digitales formados por segmentos de línea (o vectores) que pueden ser rectos o curvos y poseen atributos matemáticos que definen su forma, posición, etc. Estos segmentos se unen formando figuras geométricas que nos servirán para crear distintas imágenes.

Cuando decimos que se basa en XML, hablamos de un lenguaje de marcado que te permite crear tus propias etiquetas diseñadas para una necesidad específica.

Con todo esto, en resumen, queremos decir que (a diferencia de una imagen de formato jpg o png), las imágenes svg están formadas por etiquetas y texto que definen un conjunto de líneas y formas geométricas, dando como resultado la imagen o gráfico.

Este tipo de formatos existe antes del HTML5, pero no de una manera estandarizada. Esto hacía que no todos los navegadores lo soportaran (como por ejemplo el Internet Explorer, que tenía su propia versión: VML), complicando su uso al momento de hacer código.

Ya con la versión de HTML5, **svg** se reconoce y se implementa de forma nativa en los navegadores, incluyendo el Internet Explorer (desde su versión 9).

2.1. Conociendo un poco más sobre SVG: beneficios y limitaciones

El svg presenta la capacidad de ser escalable y pesar menos que otros formatos como png.

Al ser una imagen vectorial, nos brinda una mayor definición y calidad, que nos permite hacer zoom sin que la imagen se pixele o que necesitemos cargar otra versión de imagen .

Otro beneficio de este formato es que permite que apliquemos tanto propiedades CSS como JavaScript para modificarla y es de código abierto.

También permite ser detectado por los motores de búsqueda, esto quiere decir que al ser texto , el mismo puede ser buscado por los motores de búsqueda permitiendo que el sitio figure en más búsquedas.

Es independiente a la resolución del dispositivo en que se encuentre y no solo lo soportan los navegadores sino también por gran cantidad de dispositivos .

Entre sus desventajas se encuentra el ser una tecnología relativamente nueva, que presenta herramientas limitadas que requieren de más tiempo para que se sigan desarrollando .

También en las animaciones, hace un uso intensivo del procesador de la computadora al tener que realizar múltiples cálculos cada vez que debe ejecutarlas.

3. Declarando SVG en HTML5

Ahora que ya contamos con el conocimiento de que es un SVG hablemos de cómo declararlas y cómo crear una imagen en este formato. Tenemos que tener en cuenta que contamos con varias maneras para utilizar svg: utilizando etiquetas como **<img>** , **<object>** o **<embed>** ; o colocándola directamente en el documento por medio de la etiqueta **<svg>**.

Veamos un poco más de cerca estas maneras...

3.1. Utilizando SVG por medio de la etiqueta img o embed

Esta manera de utilizar svg es la que conocemos al momento, en donde tanto la etiqueta <img> como la etiqueta <embed> utilizan su atributo “src” para acceder a un archivo externo, en este caso de formato svg.

A diferencia de la etiqueta <img> que representa una imagen en el documento, la etiqueta <embed> representa un punto de inclusión de contenido interactivo o aplicación externa, dándonos más diversidad en lo que agreguemos al documento.

Cualquiera de estas etiquetas que permiten este tipo de inclusión de imagen svg nos presenta la limitación de no poder acceder a varias de las herramientas que nos provee SVG, como aplicar propiedades CSS y JavaScript ya que trata a este documento como los temas formatos, como un archivo estático.

SINTAXIS con <img>

**

SINTAXIS con <embed>

<embed src=“ruta-al-archivo.svg” />

3.2. Utilizando SVG por medio de la etiqueta object

Esta etiqueta nos permite incluir en el documento distintos tipos de contenido y es muy similar a las etiquetas <img> y <embed>, ya que también trata al documento svg como un archivo estático haciendo complicado poder utilizar todas las herramientas que nos provee este formato.

La diferencia recae en el atributo que utiliza para acceder al archivo externo: usa el atributo “data” que le brinda la dirección de donde se encuentra dicho archivo.

SINTAXIS con <object>

<object data=“ruta-al-archivo.svg” />

3.3. Utilizando SVG por medio de la etiqueta svg

A diferencia de las anteriores, esta forma de utilizar SVG es la que nos va a permitir hacer uso de las herramientas que nos proporciona este tipo de formato. Para ello utiliza el elemento <svg> que posee tanto etiqueta de apertura como de cierre. Este elemento nos va a servir como pizarra para crear la imagen, es decir que servirá como contenedor de etiquetas que conforman la imagen.

Como toda etiqueta, contiene atributos que le permiten tanto definir su contenido como agregar funcionalidad. Detallamos las más utilizadas en el siguiente cuadro.

ATRIBUTOS	DESCRIPCION	VALORES QUE PUEDE TOMAR
x	indica la coordenada del eje "x" donde comienza la imagen	Valor numerico sin unidad
y	La coordenada del eje "y" donde comienza la imagen	Valor numerico sin unidad
xmlns:xlink	Permite referenciar archivos dentro del SVG, como imagenes,para poner enlaces y si lo dejamos de poner suele dar problemas con Mozilla	string donde figura el ruteo
viewBox(min-x min-y width height)	atributo que define la posicion y dimension de una ventana grafica SVG	valor numerico donde: min-x y min-y representan la posicion de la ventana, mientras que con los otros dos datos (width - height) definimos las dimenciones
version	atributo que indica la version de SVG utilizada	1.0 | 1.1

Paralelamente, esta etiqueta engloba a un grupo de etiquetas que definen el gráfico, es decir que nos permitirá crear los vectores o las figuras geométricas que en conjunto formarán la imagen. Ellas son:

Rectangle (<rect>)

Esta etiqueta nos permite obtener tanto un rectángulo como cualquier figura cuadrada. Puede llevar los siguientes atributos:

ATRIBUTOS	DESCRIPCION	VALORES QUE PUEDE TOMAR
x	brinda la coordenada horizontal donde se situa en el <svg>.Se toma como referencia el extremo izquierdo	Numeros sin unidad
y	brinda la coordenada vertical donde se situa en el <svg>.Se toma como referencia el extremo superior	Numeros sin unidad
stroke	definel el color del contorno de la figura	string - definicion de color
strokewidth	define el grosor del contorno	Numeros sin unidad
fill	define el color de relleno de la figura	string - definicion de color
rx	define el borde redondeado	Numeros sin unidad

Disponible en el siguiente link: https://www.youtube.com/watch?v=ZPOLVynMUgg

Circle (<circle>)

Como indica su nombre, la etiqueta <circle> nos permite obtener un círculo. Al igual que <rect>, comparte algunos atributos y posee algunos propios. Estos son:

ATRIBUTOS	DESCRIPCION	VALORES QUE PUEDE TOMAR
cx	define la coordenada del eje X del centro del circulo.	Numeros sin unidad
cy	Define la coordenada del eje Y del centro del circulo.	Numeros sin unidad
stroke	definel el color del contorno de la figura	string - definicion de color
strokewidth	define el grosor del contorno	Numeros sin unidad
fill	define el color de relleno de la figura	string - definicion de color
r	Define el radio del circulo.	Numeros sin unidad

Disponible en el siguiente link: https://www.youtube.com/watch?v=g0fZ9ORxJNs

Elipse (<ellipse>)

Esta etiqueta permite crear elipses y posee los mismos atributos que <circle> con la única adición de tener dos radios en vez de uno, definidos por las propiedades “**rx**” y “**ry**”, que diferencian las elipses de los círculos.

Disponible en el siguiente link: https://www.youtube.com/watch?v=WgQvqicv5Ac

Polygon (<polygon>)

Esta etiqueta nos permite obtener formas de tres o más caras. Utiliza los mismos atributos de <rect> con el adicional del atributo “**points** “ que nos permite indicar la coordenada de cada uno de los vértices del polígono.

Disponible en el siguiente link: https://www.youtube.com/watch?v=lMJjE86q__I

Line (<line>)

Etiqueta que nos permite obtener una línea. Utiliza los atributos “**x1**” e “**y1**” que representan la coordenada donde inicia a dibujarse la línea y, “**x2**” e “**y2**” que representa la coordenada donde finaliza la línea. Con estos atributos podemos no solamente definir su largo sino también su inclinación.

Disponible en el siguiente link: https://www.youtube.com/watch?v=DTia2zvjLxg

Polyline (<polyline>)

Esta etiqueta representa una línea creada por varias líneas seguidas, es decir que nos permite unir varias líneas para crear una forma en particular. Utiliza el atributo “points”, donde define las coordenadas de inicio y fin de cada línea que lo compone.

También utiliza los mismos atributos de <rect> : **“fill" , “stoke”** y **“stokewidth”** para darle estilos.

Disponible en el siguiente link: https://www.youtube.com/watch?v=1cFfEqH83as

Ahora que tenemos conocimiento de estas etiquetas, podemos entender cómo está compuesto un archivo externo que nos permitirá ser modificado desde el mismo archivo, y también crear nuestra propia imagen SVG.

Les proponemos ver el siguiente video y hacer juntos una imagen ***svg*** utilizando lo aprendido al momento.

Disponible en el siguiente link: https://www.youtube.com/watch?v=xpfZvIUvaEs

4. Animación con SVG

Dentro de las herramientas que nos provee svg, una de ellas es la capacidad de poder animar dicha imagen. Tenemos la posibilidad de hacerlo por medio de CSS, utilizando etiquetas o por medio JavaScript haciendo uso tanto de herramientas ya conocidas como algunas nuevas.

Al momento de animar, debemos considerar el tipo de animaciones que realizaremos para evitar una mala experiencia de aquellas personas con capacidades diferentes.

4.1. Animación con CSS

Para animar un svg con CSS, hacemos uso de las propiedades ***animation*** y ***transform*** sobre las etiquetas que componen. Esto quiere decir que al definir un gráfico svg utilizando la etiqueta <svg> podemos acceder a las figuras que lo componen (<rect>,<circle>,<elipse>,etc) y aplicarles estas propiedades de la misma forma que a los demás elementos. Tengamos en cuenta que estas figuras que componen al svg son etiquetas con propiedades, por lo que podemos acceder desde el CSS haciendo referencia a sus selectores.

Disponible en el siguiente link: https://www.youtube.com/watch?v=7yUbvmM8hjQ

4.2. Usando la etiqueta animate

Otra manera de animar svg es el uso de la etiqueta **<animate>** que nos permite colocarla dentro de la etiqueta que da forma (<circle>, <rect>, etc.) para animar una de sus propiedades a lo largo de un tiempo. Esta etiqueta cuenta con los siguientes atributos:

ATRIBUTOS	DESCRIPCION	VALORES QUE PUEDE TOMAR
attributeName	Atributo que indica el nombre de la propiedad CSS del elemento de destino que se va a animar.	selectores
attributeType	Atributo que especifica el mapa de nombres en el que se define el atributo de destino	CSS | XML | auto
from	Atributo que indica el valor que tendra la propiedad al principio de la animacion	Valor de la propiedad que se anima
to	Atributo que indica el valor que tendra la propiedad al final de la animacion	Valor de la propiedad que se anima
dur	Atributo que indica el tiempo de durqacion de la animacion	Unidad de tiempo en segundos (s) o milisegundos (ms)
begin	Atributo que establece cuando inicia la animacion	Unidad de tiempo en segundos (s)
end	Atributo que establece cuando finaliza la animacion	Unidad de tiempo en segundos (s)
fill	Atributo que indica lo que debe hacer el elemento al acabar la animacion	"freeze" | "restore"
min | max	Atributos que indican la duracion minima y maxima de la animacion.	Numero sin unidad
restart	Atributo que indica si la animacion puede reiniciarse	"always" | "wenNotActive" | "never
repeatDur	Atributo que indica el tiempo que se le dedicara a la animacion, es similar al repearCount pero en vez de tomar como referencia la cantidad de veces toma un tiempo determinado	Unidad de tiempo en segundos (s) | "indefinite"
repeatCount	Indicael numero de veces que se repetira la animacion.	Valores numericos | "indefinite"

EJEMPLO SINTAXIS

<rect x="10" y="10" width="100" height="100">

<animate attributeType="XML" attributeName="x" from="-100" to="120" dur="10s" repeatCount="indefinite"/>

</rect>

En este ejemplo podemos ver que en la animación el rectángulo cambiará su ancho y ubicación en un plazo de 10 segundos, desde la ubicación -100 a 120.

Disponible en el siguiente link: https://www.youtube.com/watch?v=5aAg0m6q4hk

4.3. Usando la etiqueta animateMotion

Al igual que la etiqueta **<animate>**, la etiqueta **<animateMotion>** se coloca dentro de la que define una forma de la imagen (<elipse>, <rect> , etc); y nos permite definir cómo se comportará el elemento a lo largo de una ruta determinada.

Esta ruta puede estar ya creada y que el elemento la siga. También le podemos indicar al elemento qué camino seguir sin tener de antemano la ruta creada, es decir sin que esté visible el camino que va a seguir dicho elemento.

Tengamos en cuenta que **<animateMotion>** utiliza las mismas propiedades de la etiqueta **<animate>** que permiten controlar la animación. Ellas son:

ATRIBUTOS	DESCRIPCION	VALORES QUE PUEDE TOMAR
keyPoints	Atributo que indica qué tan lejos está el objeto a lo largo del camino que va tomando	Valor numerico | "none"
path	Atributo que define el pasaje la animacion	String
rotate	Atributo que define la rotacion aplicada al elemento animado a lo largo del camino .Usualmente para indicar la direccion de la animacion	aout | aotu-reverse
attributeName	Atributo que indica el nombre de la propiedad CSS del elemento de destino que se va a animar.	selectores
attributeType	Atributo que especifica el mapa de nombres en el que se define el atributo de destino	CSS | XML | auto
from	Atributo que indica el valor que tendra la propiedad al principio de la animacion	Valor de la propiedad que se anima
to	Atributo que indica el valor que tendra la propiedad al final de la animacion	Valor de la propiedad que se anima
dur	Atributo que indica el tiempo de durqacion de la animacion	Unidad de tiempo en segundos (s) o milisegundos (ms)
begin	Atributo que establece cuando inicia la animacion	Unidad de tiempo en segundos (s)
end	Atributo que establece cuando finaliza la animacion	Unidad de tiempo en segundos (s)
fill	Atributo que indica lo que debe hacer el elemento al acabar la animacion	"freeze" | "restore"
min | max	Atributos que indican la duracion minima y maxima de la animacion.	Numero sin unidad
restart	Atributo que indica si la animacion puede reiniciarse	"always" | "wenNotActive" | "never
repeatDur	Atributo que indica el tiempo que se le dedicara a la animacion, es similar al repearCount pero en vez de tomar como referencia la cantidad de veces toma un tiempo determinado	Unidad de tiempo en segundos (s) | "indefinite"
repeatCount	Indicael numero de veces que se repetira la animacion.	Valores numericos | "indefinite"

Animando con animateMotion en una ruta preexistente

Necesitamos crear una ruta que recorrerá el elemento y la definiremos por fuera de su etiqueta utilizando la etiqueta **<path>**.

Dicha ruta creará una línea para indicar el camino a seguir. Cuenta con los siguientes atributos:

ATRIBUTOS	DESCRIPCION	VALORES QUE PUEDE TOMAR
d	Atributo que define la forma del camino a seguir por la figura, que traza <path>.Estos numeros los obtiene convinando coordenadas "X" e "Y" sumando 6 tipos de letras para detallar como es la ruta.	Coordeanadas X e Y (numerico) | letra asociadas
pathLength	Atributo que permite a los autores especificar la longitud de la ruta	numerico
fill	atributo que especifica el color del relleno del trazo	string - colores
stoke	atributo que define el color del contorno del trazo	string - colores

Tener en cuenta que lo mas utili par definir un "d" es utilizando la curva de Bezier que nos da las coordenadas sin necesidad de armarlas nosotros manualmente

LETRAS ASOCIADAS	DESCRIPCION
M | m	Letras que indican a que coordenada se mueve el punto actual,no traza la linea. La M mayuscula especifica coordenadas absolutas mientras que la "m" minuscula reepresenta coordenadas relativas a la posicion actual
L | l | H | h | V | v	Letras que indican la linea atrazar de un punto actual al indicado en la coordenada.Esata al final, se vuelve el punto actual para el siguiente trazo. Las mayusculas especifican coordenadas absolutas mientras que las minusculas reepresentan coordenadas relativas a la posicion actual
C | c | S | s | Q | q | T | t	Letras que representan a la curva de Bezier
Z | z	Letras que indica que debe trazar una linea recta desde la posicion actual al punto de coordenadas definido
A | a	Letras que indican una Curva definida como una parte de una elipse . La "A" mayuscula especifica coordenadas absolutas mientras que la "a" minuscula reepresenta coordenadas relativas a la posicion actual

Una vez definida esta etiqueta, podemos agregar dentro del elemento la etiqueta **<animateMotion>** donde definiremos la duración, las veces que realiza la animación y demás.

Por último, sumamos una última etiqueta llamada **<mpath>** , junto con su atributo **xlink:href** dentro de la etiqueta **<animateMotion>** que nos indica la ruta a la cual hacer referencia.

EJEMPLO SINTAXIS

<svg >

<path id=“camino1” fill="none" stroke=“lightgrey" d="M20,50 C20,-50 180,150 180,50 C180-50 20,150 20,50 z" />

```
<circle r="5" fill="red">

<animateMotion dur="10s" repeatCount="indefinite"/>

<mpath Xlink:href="#camino1" />

</circle>

</svg>
```

En este ejemplo podemos ver un círculo que recorre una línea en forma de infinito.

Disponible en el siguiente link: https://www.youtube.com/watch?v=hKTJ5bp0n0Q

Animando con animateMotion sin una ruta preexistente

Para este tipo de casos, no requerimos de la etiqueta **<path>** sino directamente utilizamos a path como un atributo dentro de **<animateMotion>** donde le indicamos el camino que debe seguir durante la animación.

Este atributo indica la ubicación del elemento a lo largo del tiempo de la misma forma que el atributo **"d"** lo hacía en la etiqueta **<path>**.

4.4. Usando la etiqueta animateTransform

Esta etiqueta, al igual que las anteriores, debe colocarse dentro del elemento a animar y nos permite animar el atributo transform del elemento, es decir que podemos hacer una transformación de un elemento a lo largo de un tiempo definido de forma animada. **<animateTransform>** cuenta con los siguientes atributos que le permiten definir la transformación a realizar, el tiempo y el estilo de la animación.

EJEMPLO SINTAXIS

```
<svg width="250" height="200">

<circle cx="20" cy="70" r="20" fill="#6ab150">

<animateTransform attributeType="XML" attributeName="transform" type="translate"
values="0,0;83,110" begin="0s; click" end="mouseover" dur="4s"
repeatCount="indefinite">
</animateTransform>

</circle>

</svg>
```

En este ejemplo hacemos que un círculo verde se traslade de un lado al otro.

Disponible en el siguiente link: https://www.youtube.com/watch?v=P_rKUMbJTMk

5. Opciones para animar con JavaScript

Como dijimos anteriormente, contamos con la posibilidad de utilizar JavaScript para poder animar el SVG. Es muy similar al manejo de elementos del DOM , donde podemos llamar a las distintas etiquetas que componen las formas o las rutas (en caso de <path>) y agregarle los atributos que requiere, teniendo en cuenta los conceptos que aprendimos anteriormente.

Por otra parte, existen distintas bibliotecas que nos permiten hacer las animaciones SVG de una manera más simple de lo que actualmente ya son. Algunas de ellas son:

- Vivus:

Librería que permite animar gráficos SVG como si estuviéramos dibujando en ese mismo instante.Para mas informacion: https://maxwellito.github.io/vivus/ (https://maxwellito.github.io/vivus/)

- Snap.svg

Librería que permite animar facilmente con JavaScript y es de simple uso.Para mas informacion: http://snapsvg.io/ (http://snapsvg.io/)

- anime.js

Librería que permite animar gráficos SVG que permite hacer que un elemento div siga una ruta SVG con solo unas pocas líneas de código .Para mas informacion: https://animejs.com/ (https://animejs.com/)

6. En resumen

En esta unidad pudimos ver las herramientas que nos provee este formato de gráfico, cómo modificar sus propiedades e incluso cómo poder animarlas.

Vimos también las distintas formas de invocarlas y la importancia que fue ganando a lo largo del tiempo.

Por último, aprendimos a combinar las herramientas vistas para poder crear nuestro propio svg y animarlo.

7. Bibliografía utilizada y sugerida

https://graffica.info/formato-svg-ventajas/ (https://graffica.info/formato-svg-ventajas/)

https://developer.mozilla.org/es/docs/Web/SVG/Element/svg (https://developer.mozilla.org/es/docs/Web/SVG/Element/svg)

https://desarrolloweb.com/articulos/que-es-svg.html (https://desarrolloweb.com/articulos/que-es-svg.html)

https://developer.mozilla.org/es/docs/Web/HTML/Element/embed (https://developer.mozilla.org/es/docs/Web/HTML/Element/embed)

http://w3.unpocodetodo.info/svg/animatemotion.php (http://w3.unpocodetodo.info/svg/animatemotion.php)

https://developer.mozilla.org/es/docs/Web/HTML/Element/object (https://developer.mozilla.org/es/docs/Web/HTML/Element/object)

https://www.w3schools.com/tags/tag_embed.asp (https://www.w3schools.com/tags/tag_embed.asp)

https://developer.mozilla.org/es/docs/Web/SVG (https://developer.mozilla.org/es/docs/Web/SVG)

https://developer.mozilla.org/en-US/docs/Web/SVG/Element/animateMotion (https://developer.mozilla.org/en-US/docs/Web/SVG/Element/animateMotion)

http://w3.unpocodetodo.info/svg/animatetransform-translate.php (http://w3.unpocodetodo.info/svg/animatetransform-translate.php)

https://developer.mozilla.org/en-US/docs/Web/SVG/Element/animateTransform (https://developer.mozilla.org/en-US/docs/Web/SVG/Element/animateTransform)