CANVAS

Introducción al Language de Canvas

Valentina Bonilla Bedoya
Risaralda, Universidad Tecnológica de Pereira, Colombia.
Valentina.bonilla@utp.edu.co

***Resumen*— Canvas es un elemento HTML que permite la creación de gráficos y animaciones de forma dinámica por medio de scripts. Sus aplicaciones son prácticamente inimaginables en este elemento podemos crear juegos, interfaces, editores gráficos o efectos dinámicos hasta aplicaciones 3D y todo lo que tu imaginación pueda llegar hacer.**

Canvas fue creado por Apple, pero han liberado la propiedad intelectual para englobarlo dentro de los estándares de HTML. Para el usuario, todo son ventajas. No requiere ningún plugin adicional, sólo una un navegador que soporte HTML5 y hoy en día todos los navegadores importantes (Safari, Chrome, Firefox, Opera e Internet Explorer) soportan Canvas.

***Palabras clave*— Canvas, HTML, gráficos, aplicaciones, dinámica.**

***Abstract*— Canvas is an HTML element that allows the creation of graphics and animations dynamically by means of scripts. Its applications are almost unimaginable in this element we can create games, interfaces, graphic editors or dynamic effects to 3D applications and everything your imagination can do.**

Canvas was created by Apple, but they have released intellectual property to encompass it within the HTML standards. For the user, all are advantages. It does not require any additional plugin, only a browser that supports HTML5 and today all major browsers (Safari, Chrome, Firefox, Opera and Internet Explorer) support Canvas.

I. INTRODUCCIÓN

Canvas fue introducido inicialmente por Apple para su uso dentro de su propio componente de Mac OS X Surgido en 2004, para empujar aplicaciones como widgets de Dashboard y el navegador Safari. Más tarde, en 2005, se adoptó en la versión 1.8 de los navegadores Gecko y Opera en 2006. Fue estandarizado por el Grupo de Trabajo de Tecnologías de Aplicación de hipertexto Web (WHATWG por sus siglas en inglés) sobre las nuevas especificaciones propuestas para tecnologías web de última generación.

II. CONTENIDO

El Canvas consiste en una región dibujable definida en el código HTML con atributos de altura y ancho. El código JavaScript puede acceder a la zona a través de un conjunto completo de funciones similares a las de otras APIs comunes de dibujo 2D, permitiendo así que los gráficos sean generados dinámicamente. Algunos de los usos previstos incluyen construcción de gráficos, animaciones, juegos, y la composición de imágenes.

Para dibujar necesitaremos saber las coordenadas x e y de los puntos utilizados. Por ejemplo el código para dibujar un rectángulo es rect(x,y,w,h), donde x e y son las coordenadas de la esquina izquierda arriba, w es la anchura y h es la altura del rectángulo.
En el <Canvas>: El punto cuyas coordenadas son: x=0; y=0; se corresponde con la esquina izquierda arriba del Canvas. Abajo en la esquina derecha x=Canvas.width; y=Canvas.height;.

Pase el ratón sobre el siguiente <Canvas> para ver sus coordenadas x e y:

Ejemplos:

El Canvas cuenta con varios métodos () y propiedades. A continuación, algunas de ellas.

	JavaScript	Descripción	Defecto
width	*canvas.width*	Determina (sets) o devuelve (returns) la anchura del canvas	300
height	*canvas.height*	Determina (sets) o devuelve (returns) la altura del canvas	150
getContext()	*canvas.getContext("2d");*	Devuelve un objeto que proporciona todos los métodos y propiedades para dibujar en el canvas.	
toDataURL()	*canvas.toDataURL(tipo);*	Convierte el contenido del canvas en una imágen - data:uri. El parámetro entre paréntesis indica el tipo de imágen.	image/png

Un elemento de Canvas en HTML

```
<canvas id="example" width="200" height="200">
Este texto se muestra si su navegador no soporta el lienzo (canvas) de
</canvas>
```

Utilizando JavaScript, usted puede dibujas en el lienzo (Canvas)

```
var example = document.getElementById('example');
var context = example.getContext('2d');
context.fillStyle = 'red';
context.fillRect(30, 30, 100, 100);
```

Sin embargo, el elemento <Canvas> no tiene habilidad de dibujo en sí mismo, siendo sólo un contenedor para gráficos. Por esto es importante que el <Canvas> tenga asignado un id, ya que habrá que manipularlo a través de JavaScript. Utilizaremos document.get ElementById para encontrar el Canvas.

El método getContext () devuelve un objeto (puede llamar este objeto como quiera; a continuación, lo llamaremos ctx) que proporciona métodos () y propiedades para dibujar en el Canvas.

LA SECCIÓN DE<CANVAS>, ELEMENTOS

<canvas id="tutorial" width="150" height="150"></canvas>

A primera vista, se <Canvas>parece al <img>elemento, con la única diferencia clara de que no tiene los atributos srcy alt. De hecho, el <Canvas>elemento tiene solo dos atributos, widthy height. Ambos son opcionales y también se pueden configurar usando las propiedades DOM . Cuando se especifican no y atributos, el lienzo inicialmente tendrá 300 píxeles de ancho y 150 píxeles de alto. El elemento puede ser dimensionado arbitrariamente por CSS , pero durante la representación, la imagen se ajusta para ajustarse a su tamaño de diseño: si el tamaño de CSS no respeta la proporción del lienzo inicial, aparecerá distorsionado.widthheight

Nota: si sus representaciones parecen distorsionadas, intente especificar sus atributos widthy heightexplícitamente en los <Canvas>atributos, y no use CSS.

El id atributo no es específico del <Canvas>elemento, pero es uno de los atributos HTML globales que se pueden aplicar a cualquier elemento HTML (como class por ejemplo). Siempre es una buena idea proporcionar una, id ya que esto hace que sea mucho más fácil identificarla en un script.

El <Canvas>elemento puede ser de estilo al igual que cualquier imagen normal (margin, border, background...). Estas reglas, sin embargo, no afectan el dibujo real en el lienzo. Veremos cómo se hace esto en un capítulo dedicado de este tutorial. Cuando no se aplican reglas de estilo al lienzo, inicialmente será completamente transparente.

Contenido alternativo: El <Canvas>elemento difiere de una <img> etiqueta en que, como para <video>, <audio>o <picture>elementos, es fácil definir algún contenido alternativo, que se mostrará en los navegadores más antiguos que no lo admiten, como las versiones de Internet Explorer anteriores a la versión 9 o los navegadores de texto. Siempre debe proporcionar contenido alternativo para que se muestren en esos navegadores.

Proporcionar contenido alternativo es muy sencillo: solo inserte el contenido alternativo dentro del <Canvas>elemento. Los navegadores que no son compatibles <Canvas>ignorarán el contenedor y mostrarán el contenido de respaldo en su interior. Los navegadores que sí admiten <Canvas>ignorarán el contenido dentro del contenedor y simplemente representarán el lienzo normalmente.

Por ejemplo, podríamos proporcionar una descripción de texto del contenido del lienzo o proporcionar una imagen estática del contenido renderizado dinámicamente. Esto puede verse algo como esto:

```
<canvas id="stockGraph" width="150" height="150">
  current stock price: $3.15 + 0.15
</canvas>
<canvas id="clock" width="150" height="150">
  <img src="images/clock.png" width="150" height="150" alt=""/>
</canvas>
```

Por ejemplo, decirle al usuario que use un navegador diferente que admita lienzo no ayuda a los usuarios que no pueden leer el lienzo. Proporcionar un texto de reserva o sub DOM útil ayuda a que el lienzo sea más accesible .

Como consecuencia de la forma en que se proporciona el respaldo, a diferencia del <img>elemento, el <Canvas>elemento requiere la etiqueta de cierre (</Canvas>). Si esta etiqueta no está presente, el

resto del documento se consideraría contenido de respaldo y no se mostraría.

Si el contenido alternativo no es necesario, un simple <Canvas id="foo" ...></Canvas>es totalmente compatible con todos los navegadores que admiten lienzos.

Un simple ejemplo de CANVAS

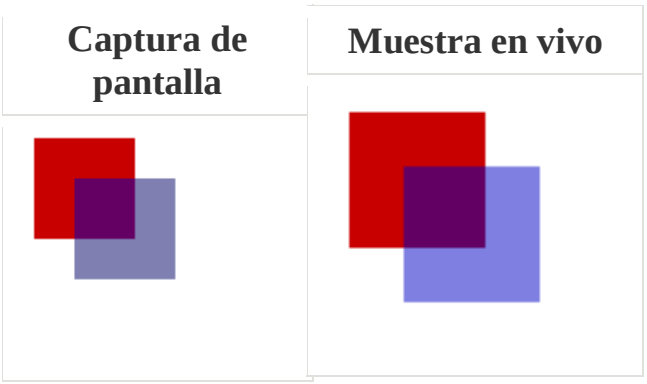
Para comenzar, echemos un vistazo a un ejemplo simple que dibuja dos rectángulos que se cruzan, uno de los cuales tiene transparencia alfa. Exploraremos cómo funciona esto con más detalle en ejemplos posteriores.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8"/>
  <script type="application/javascript">
   function draw() {
    var canvas = document.getElementById('canvas');
    if (canvas.getContext) {
     var ctx = canvas.getContext('2d');

     ctx.fillStyle = 'rgb(200, 0, 0)';
     ctx.fillRect(10, 10, 50, 50);

     ctx.fillStyle = 'rgba(0, 0, 200, 0.5)';
     ctx.fillRect(30, 30, 50, 50);
    }
   }
  </script>
 </head>
 <body onload="draw();">
  <canvas id="canvas" width="150" height="150"></canvas>
 </body>
</html>
```

Este ejemplo se ve así:



III. CONCLUSIONES

- El <Canvas>elemento crea una superficie de dibujo de tamaño fijo que expone uno o más contextos de representación .
- Los <Canvas>elemento se utilizan para crear y manipular el contenido mostrado.
- Existen diferentes contextos de renderización 2D. Otros contextos pueden proporcionar diferentes tipos de representación; por ejemplo, WebGL utiliza un contexto 3D basado en OpenGL ES .
- El lienzo está inicialmente en blanco. Para mostrar algo, una secuencia de comandos primero debe acceder al contexto de representación y dibujar en él.
- El <Canvas>elemento tiene un método llamado getContext(), que se utiliza para obtener el contexto de representación y sus funciones de dibujo. getContext()Toma un parámetro, el tipo de contexto. Para gráficos en 2D.
- La primera línea en el script recupera el nodo en el DOM que representa el <Canvas>elemento llamando al document.getElementById()método. Una vez que tenga el nodo del elemento, puede acceder al contexto de dibujo utilizando su getContext()método.
- El contenido alternativo se muestra en los navegadores que no son compatibles <canvas>. Los scripts también pueden verificar el soporte mediante programación simplemente comprobando la presencia del getContext()método.

REFERENCIAS

[1]. J. Mª Baquero García " Programador Senior de Arsys desde 2013," *Diseño web y programación.* https://www.arsys.es/blog/programacion/diseno-web/que-es-canvas/

[2]. http://w3.unpocodetodo.info/canvas/introduccion.php

[3]. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_usage.