CANVAS

Autor 1: Víctor Manuel muñoz Espinal
Programa ingeniería de sistemas, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: victor.munoz1@utp.edu.co

***Resumen*— Canvas (lienzo) es un elemento HTML que permite la creación de gráficos y animaciones de forma dinámica por medio de scripts. Sus aplicaciones son prácticamente inimaginables: crear juegos, interfaces, editores gráficos o efectos dinámicos, aplicaciones 3D…. Sólo la imaginación pone límites a Canvas. En este artículo, profundizamos en las posibilidades de Canvas y las ejemplificamos con un sencillo gráfico.**

***Palabras clave*— HTML, scripts,**

***Abstract*—Canvas (Canvas) is an HTML element that allows the creation of graphics and animations dynamically in the middle of scripts. Its applications are practically unimaginable: create games, interfaces, graphic editors or dynamic effects, 3D applications.... Only the imagination puts limits to a canvas. In this article, we delve into the possibilities of Canvas and exemplify them with a simple graphic.**

***Key Word* — HTML, scripts,**

I. INTRODUCCIÓN

En las siguientes páginas se expone brevemente que es y cómo funciona "canvas" se realizan también algunos ejemplos gráficos, este trabajo se realizó mediante la investigación en fuentes confiables donde se pudiera obtener la información correcta mediante la técnica de la observación.

II. CONTENIDO

A. CANVAS.

es un elemento HTML el cual puede ser usado para dibujar gráficos usando scripts (normalmente JavaScript). Este puede, por ejemplo, ser usado para dibujar gráficos, realizar composición de fotos o simples (y no tan simples) animaciones. Las imágenes a la derecha muestran algunos ejemplos de implementaciones <canvas> las cuales se verán en un futuro en este tutorial.

<canvas> fue introducido primero por Apple para el Mac OS X Dashboard y después implementado en Safari y Google Chrome. Navegadores basados en Gecko 1.8, tal como Firefox 1.5, que también soportan este elemento. El <canvas> es un elemento parte de las especificaciones de la WhatWG Web applications 1.0 mejor conocida como HTML5.

En este tutorial se describe cómo usar el elemento <canvas> para dibujar gráficos en 2D, empezando con lo básico. Los ejemplos le proveerán mayor claridad a las ideas que pueda tener referentes al canvas, así como los códigos que necesita para crear su propio contenido.

B. DIBUJANDO FORMAS.

Antes de que podamos comenzar a dibujar, debemos hablar sobre la cuadrícula del lienzo o el espacio de coordenadas . Nuestro esqueleto HTML de la página anterior tenía un elemento de lienzo de 150 píxeles de ancho y 150 píxeles de alto. A la derecha, verá este lienzo con la cuadrícula predeterminada superpuesta. Normalmente, 1 unidad en la cuadrícula corresponde a 1 píxel en el lienzo. El origen de esta cuadrícula se coloca en la esquina superior izquierda en la coordenada (0,0). Todos los elementos se colocan en relación a este origen. Así que la posición de la esquina superior izquierda del cuadrado azul se convierte en x píxeles desde la izquierda y píxeles desde la parte superior, en la coordenada (x, y). Más adelante en este tutorial veremos cómo podemos traducir el origen a una posición diferente, rotar la cuadrícula e incluso escalarla, pero por ahora nos ceñiremos a la predeterminada.

Dibujo rectángulos sección
A diferencia de SVG , <canvas>solo soporta una forma primitiva: los rectángulos. Todas las demás formas deben crearse combinando una o más rutas, listas de puntos conectados por líneas. Por suerte, tenemos una variedad de funciones de dibujo de trayectoria que permiten componer formas muy complejas.

Primero veamos el rectángulo. Hay tres funciones que dibujan rectángulos en el lienzo:

Fecha de Recepción: (Letra Times New Roman de 8 puntos)
Fecha de Aceptación: Dejar en blanco

fillRect(x, y, width, height)
Dibuja un rectángulo relleno.

strokeRect(x, y, width, height)
Dibuja un contorno rectangular.

clearRect(x, y, width, height)
Despeja el área rectangular especificada, haciéndola completamente transparente.

Cada una de estas tres funciones toma los mismos parámetros. X y y especifique la posición en el lienzo (en relación con el origen) de la esquina superior izquierda del rectángulo. Widthy height proporcionar el tamaño del rectángulo.

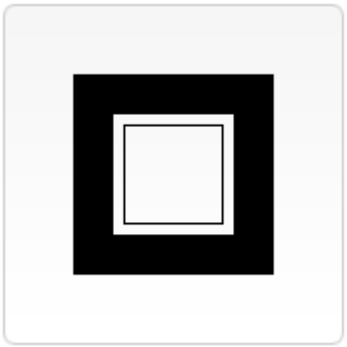
A continuación se muestra la draw () función de la página anterior, pero ahora está haciendo uso de estas tres funciones.

Ejemplo: de forma rectangular Sección

```
function draw() {
  var canvas = document.getElementById('canvas');
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d');

    ctx.fillRect(25, 25, 100, 100);
    ctx.clearRect(45, 45, 60, 60);
    ctx.strokeRect(50, 50, 50, 50);
  }
}
```

La salida de este ejemplo se muestra a continuación



La fillRect()función dibuja un gran cuadrado negro de 100 píxeles en cada lado. La clearRect()función luego borra un cuadrado de 60x60 píxeles del centro, y luego strokeRect()se llama para crear un contorno rectangular de 50x50 píxeles dentro del cuadrado borrado.

En las próximas páginas veremos dos métodos alternativos para clearRect(), y también veremos cómo cambiar el color y el estilo de trazo de las formas renderizadas.

A diferencia de las funciones de ruta que veremos en la siguiente sección, las tres funciones de rectángulo se dibujan inmediatamente en el lienzo.

Trazado de dibujo de la sección
Las únicas otras formas primitivas son los caminos. Una ruta es una lista de puntos, conectados por segmentos de líneas que pueden ser de diferentes formas, curvas o no, de diferente ancho y de diferente color. Un camino, o incluso un subpath, puede ser cerrado. Para hacer formas usando caminos toma algunos pasos adicionales:

Primero, creas el camino.
A continuación, utiliza los comandos de dibujo para dibujar en la ruta.
Una vez que se ha creado la ruta, puede trazar o rellenar la ruta para representarla.
Aquí están las funciones utilizadas para realizar estos pasos:

beginPath()
Crea un nuevo camino. Una vez creados, los futuros comandos de dibujo se dirigen a la ruta y se usan para construir la ruta hacia arriba.

Métodos de ruta
Métodos para establecer diferentes caminos para los objetos.

closePath()
Agrega una línea recta a la ruta, y va al inicio de la ruta secundaria actual.

stroke()
Dibuja la forma acariciando su contorno.

fill()
Dibuja una forma sólida rellenando el área de contenido de la ruta.

El primer paso para crear una ruta es llamar al beginPath(). Internamente, las rutas se almacenan como una lista de subrutas (líneas, arcos, etc.) que juntas forman una forma. Cada vez que se llama a este método, la lista se restablece y podemos comenzar a dibujar nuevas formas.

Nota: cuando la ruta actual está vacía, como inmediatamente después de la llamada beginPath(), o en un lienzo recién creado, el primer comando de construcción de ruta siempre se trata como a moveTo(), independientemente de lo que realmente sea. Por ese motivo, casi siempre querrá establecer específicamente su posición de inicio después de restablecer una ruta.

El segundo paso es llamar a los métodos que realmente especifican las rutas a dibujar. Los veremos en breve.

El tercero, y un paso opcional, es llamar closePath(). Este método trata de cerrar la forma dibujando una línea recta desde el punto actual hasta el inicio. Si la forma ya se ha cerrado o solo hay un punto en la lista, esta función no hace nada.

Nota: cuando llama fill(), cualquier forma abierta se cierra automáticamente, por lo que no tiene que llamar closePath(). Este no es el caso cuando llamas stroke().

Dibujando una sección triangular
Por ejemplo, el código para dibujar:

```
function draw() {
  var canvas = document.getElementById('canvas');
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d');

    ctx.beginPath();
    ctx.moveTo(75, 50);
    ctx.lineTo(100, 75);
    ctx.lineTo(100, 25);
    ctx.fill();
  }
}
```

El resultado sería el siguiente:

Moviendo la sección de la pluma

Una función muy útil, que en realidad no dibuja nada pero se convierte en parte de la lista de rutas descrita anteriormente, es la moveTo()función. Probablemente, lo mejor es pensar en esto como levantar una pluma o un lápiz de un lugar en un pedazo de papel y colocarlo en el siguiente.

moveTo(x, y)
Mueve el lápiz a las coordenadas especificadas por xy y.
Cuando el lienzo está inicializado o beginPath()se llama, normalmente querrá usar la moveTo()función para colocar el punto de partida en otro lugar. También podríamos usar moveTo()para dibujar caminos desconectados. Echa un vistazo a la cara sonriente debajo.

Para probar esto por ti mismo, puedes usar el siguiente fragmento de código. Solo pégalo en la draw()función que vimos anteriormente.

```
function draw() {
  var canvas = document.getElementById('canvas');
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d');

    ctx.beginPath();
    ctx.arc(75, 75, 50, 0, Math.PI * 2, true); // Outer circle
    ctx.moveTo(110, 75);
    ctx.arc(75, 75, 35, 0, Math.PI, false);  // Mouth (clockwise)
    ctx.moveTo(65, 65);
    ctx.arc(60, 65, 5, 0, Math.PI * 2, true);  // Left eye
    ctx.moveTo(95, 65);
    ctx.arc(90, 65, 5, 0, Math.PI * 2, true);  // Right eye
    ctx.stroke();
  }
}
```

El resultado se ve así:

C. **COLORES**

Hasta ahora solo hemos visto métodos del contexto de dibujo. Si queremos aplicar colores a una forma, hay dos propiedades importantes que podemos usar: fillStyley strokeStyle.

fillStyle = color
Establece el estilo utilizado al rellenar formas.
strokeStyle = color
Establece el estilo para los contornos de las formas.
colores una cadena que representa un CSS <color>, un objeto degradado o un objeto de patrón. Veremos los objetos de gradiente y patrón más adelante. De forma predeterminada, el trazo y el color de relleno se establecen en negro (valor de color CSS #000000).

Nota: Cuando configura la propiedad strokeStyley / o fillStyle, el nuevo valor se convierte en el valor predeterminado para todas las formas que se dibujan a partir de ese momento. Para cada forma que desee

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

en un color diferente, deberá reasignar la propiedad fillStyleo strokeStyle.

Las cadenas válidas que puede ingresar deben ser, según la especificación, <color>valores CSS . Cada uno de los siguientes ejemplos describe el mismo color.

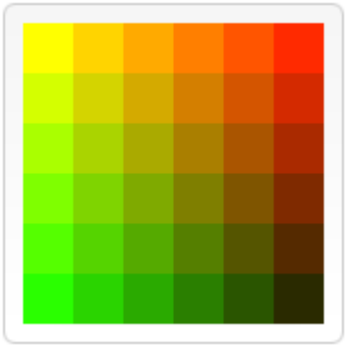
```
// these all set the fillStyle to 'orange'

ctx.fillStyle = 'orange';
ctx.fillStyle = '#FFA500';
ctx.fillStyle = 'rgb(255, 165, 0)';
ctx.fillStyle = 'rgba(255, 165, 0, 1)';
```

Una fillStyle sección de ejemplo
En este ejemplo, una vez más usamos dos forbucles para dibujar una cuadrícula de rectángulos, cad uno en un color diferente. La imagen resultante debe parecerse a la captura de pantalla. No hay nada demasiado espectacular sucediendo aquí. Usamos las dos variables iy jpara generar un color RGB único para cada cuadrado, y solo modificamos los valores rojo y verde. El canal azul tiene un valor fijo. Al modificar los canales, puedes generar todo tipo de paletas. Al aumentar los pasos, puede lograr algo que se parece a las paletas de colores que usa Photoshop.

```
function draw() {
  var ctx =
document.getElementById('canvas').getContext('2d');
  for (var i = 0; i < 6; i++) {
    for (var j = 0; j < 6; j++) {
      ctx.fillStyle = 'rgb(' + Math.floor(255 - 42.5 * i) + ', ' +
                  Math.floor(255 - 42.5 * j) + ', 0)';
      ctx.fillRect(j * 25, i * 25, 25, 25);
    }
  }
}
```

El resultado se ve así:



D. **DIBUJO DE TEXTOSECTION**

El contexto de representación de lienzo proporciona dos métodos para representar texto:

fillText(text, x, y [, maxWidth])
Rellena un texto dado en la posición dada (x, y). Opcionalmente con un ancho máximo para dibujar.
strokeText(text, x, y [, maxWidth])
Traza un texto dado en la posición dada (x, y). Opcionalmente con un ancho máximo para dibujar.
Un fillTextejemploSection
El texto se rellena utilizando el actual fillStyle.

```
function draw() {
  var ctx =
document.getElementById('canvas').getContext('
2d');
  ctx.font = '48px serif';
  ctx.fillText('Hello world', 10, 50);
}
```

CONCLUCION

Las conclusiones son obligatorias y deben ser claras. Deben expresar el balance final de la investigación o la aplicación del conocimiento.

REFERENCIAS

- https://developer.mozilla.org
- https://www.arsys.es

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos