Canvas - HTML5

Dibujo y animaciones con Canvas

Antonio Picó

Introducción

- Canvas es un elemento de HTML 5 que permite dibujar gráficos, manipular imágenes y realizar animaciones en una página web de forma dinámica:
  - El gráfico se crea en el momento en que se carga la página.
  - El gráfico se define mediante programación, usualmente Javascript.
  - Una vez creado el gráfico se pueden programar acciones para que el usuario interactúe con él.
- Al hablar de Canvas nos referimos a toda la API que incluye un conjunto de funciones para dibujar, líneas, rectángulos, círculos, transformar elementos, etc. no sólo a la etiqueta <canvas>.

Introducción

- Al definir un Canvas en una página web se crea un lienzo de dibujo o área de dibujo rectangular.
- Después de crear el dibujo lo que queda es un "mapa de bits", es decir, cada coordenada del lienzo tiene asignado un color, no queda ninguna estructura de lo que contiene el lienzo.
- Canvas no crea objetos vectoriales al estilo de otros entornos como SVG o Flash, sino mapas de bits como una imagen fotográfica.

Contexto

- Canvas frente a Flash:
- La mayoría de los gráficos, juegos y animaciones interactivas de la web se realizaban con Adobe Flash. Poco a poco está tomando fuerza la opción Canvas/HTML5.
- Canvas es un elemento estándar de HTML5 y cualquier navegador debe ser capaz de manejarlo, mientras que Flash es una tecnología de una empresa que requiere instalar en el navegador un complemeto (plug-in)
- Desde el punto de vista del creador de gráficos es más fácil y rápido trabajar con Flash que con Canvas ya que Flash dispone de un entorno de diseño gráfico avanzado (Adobe Flash CS6, y luego Adobe Animate 2017) mientras que crear gráficos Canvas con JavaScript es más arduo. Esto también va cambiando gracias a frameworks para Canvas (por ejemplo paper.js).
- Canvas es libre y abierto por lo que no hay que pagar licencia, mientras que Flash es propiedad de la empresa Adobe.
- Si tenemos que utilizar un gráfico a varias resoluciones, la tecnología vectorial de Flash es más eficiente que los mapas de bits de Canvas. En cambio, para gran parte de gráficos para la web, los gráficos Canvas se cargan más rápidamente que los de Flash.
- Flash requiere más recursos para ejecutarse por lo que en dispositivos móviles Canvas lleva ventaja.
- Otra alternativa para crear gráficos para la web es SVG que es un sistema vectorial, reconocido por W3C y soportado por los navegadores web modernos sin necesidad de instalar plug-in.

Empezando con Canvas

- El elemento <canvas> no está soportado en navegadores viejos, pero están soportado en la mayoría de las versiones más recientes de los navegadores.
- El tamaño por defecto del canvas es 300px * 150px [ancho (width) * alto (height)]. Pero se puede personalizar el tamaño usando las propiedades height y width. Con el fin de dibujar gráficos en el lienzo <canvas> se utiliza un objeto de contexto de JavaScript que crea gráficos sobre la marcha.

Empezando con Canvas

- Ejemplo:

```
<canvas id="tutorial" width="500"
height="500"></canvas>
```

Empezando con Canvas

- Ejemplos con respaldo por si o bien el navegador no puede mostrar canvas, o bien el usuario no puede verlo (meter la alternativa entre las etiquetas):
- <canvas id="stockGraph" width="150" height="150">

  current stock price: $3.15 + 0.15

  </canvas>
- <canvas id="clock" width="150" height="150">

  </canvas>

El contexto de renderizado

- Siempre vamos a empezar por:

```
var canvas = document.getElementById('miCanvas');
var ctx = canvas.getContext('2d');
```

- También podemos detectar aquí si está soportado:

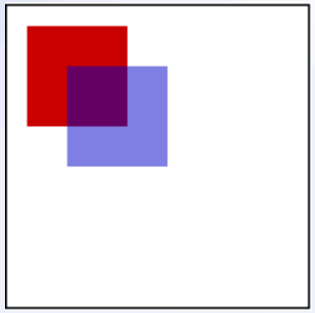
```
var canvas = document.getElementById('miCanvas');
if (canvas.getContext) {
  var ctx = canvas.getContext('2d');
  // código para dibujar aquí
} else {
  // código para canvas no soportado aquí
}
```

Ejemplo

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>Canvas tutorial</title>
    <script type="text/javascript">
      function draw() {
        var canvas = document.getElementById('tutorial');
        if (canvas.getContext) {
    var ctx = canvas.getContext('2d');
    ctx.fillStyle = 'rgb(200, 0, 0)';    //fijarse en que parametrizamos el contexto, no el objeto
    ctx.fillRect(10, 10, 50, 50);

    ctx.fillStyle = 'rgba(0, 0, 200, 0.5)';
    ctx.fillRect(30, 30, 50, 50);

        }
      }
    </script>
    <style type="text/css">
      canvas { border: 1px solid black; }
    </style>
  </head>
  <body onload="draw();">
    <canvas id="tutorial" width="150" height="150"></canvas>
  </body>
</html>
```



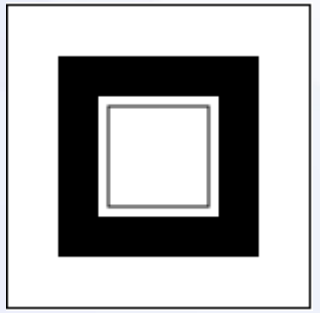
Funciones de dibujar rectángulos

- Señalamos el centro de la esquina superior izquierda, y a continuación el ancho y el alto:
- fillRect(x, y, width, height)
  - Dibuja un rectángulo relleno.
- strokeRect(x, y, width, height)
  - Dibuja un rectángulo sin relleno.
- clearRect(x, y, width, height)
  - Limpia el área rectangular especificada, haciéndola transparente.

Ejemplo

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>Canvas tutorial</title>
    <script type="text/javascript">
      function draw() {
        var canvas = document.getElementById('tutorial');
        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);

        }
      }
    </script>
    <style type="text/css">
      canvas { border: 1px solid black; }
    </style>
  </head>
  <body onload="draw();">
    <canvas id="tutorial2" width="150" height="150"></canvas>
  </body>
</html>
```



Funciones de dibujar rectas

- Señalamos el centro de la esquina superior izquierda, y a continuación el ancho y el alto:
- fillRect(x, y, width, height)
  - Dibuja un rectángulo relleno.
- strokeRect(x, y, width, height)
  - Dibuja un rectángulo sin relleno.
- clearRect(x, y, width, height)
  - Limpia el área rectangular especificada, haciéndola transparente.

Ejemplo

```
  function draw() {
    var canvas = document.getElementById('tutorial');
    if (canvas.getContext) {
   //Triángulo relleno
  ctx.beginPath();
  ctx.moveTo(25, 25);       //Centramos el origen desde donde empezaremos a dibujar
  ctx.lineTo(105, 25);      //Una línea, desde el origen hasta x,y
  ctx.lineTo(25, 105);     //Otra línea línea, desde el origen hasta x,y
  //Rellenamos con colores naranja con el método fillStyle (sólo con uno de ellos basta):
  ctx.fillStyle = 'orange';
  ctx.fillStyle = '#FFA500';
  ctx.fillStyle = 'rgb(255, 165, 0)';
  ctx.fillStyle = 'rgba(255, 165, 0, 1)';
  ctx.fill();

 //Triángulo no relleno
  ctx.beginPath();
  ctx.moveTo(125, 125);
  ctx.lineTo(125, 45);
  ctx.lineTo(45, 125);
  ctx.closePath();
  ctx.stroke();
    }
   }
```

Más funciones de dibujo:

- Arcos.
- Curvas de Bezier.
- Paths.
- Transparencias.
- Estilos y grosores de línea.
- Gradientes lineales y radiales.
- Patrones (patterns).
- Sombras
- Textos y funciones de texto
- ...

Dibujando una imagen:

- drawImage(image, x, y)
  - Ejemplo:

    ctx.drawImage(“imagen.jpg”, 0, 0);
- Podemos o bien crearla, o bien hacer referencia a ella si está en el HTML. Ejemplo:

Ejemplo

```
<!DOCTYPE html>
<html>  <!-- En este ejemplo la imagen no cabe en el canvas -->
 <head>
   <meta charset="utf-8"/>
   <title>Canvas tutorial</title>
   <script type="text/javascript">
     function draw() {
       var canvas = document.getElementById('tutorial');
       if (canvas.getContext) {
   var ctx = canvas.getContext('2d');
   var img = new Image();
   img.src="caracteristicas-del-cuerpo-del-tigre.jpg";
   //var img=document.getElementById("tigre");
   ctx.drawImage(img,0,0);
}
     }
   </script>
   <style type="text/css">
     canvas { border: 1px solid black; }
   </style>
 </head>
 <body onload="draw();">
   <canvas id="tutorial" width="150" height="150"></canvas>
 <img id="tigre" width="220" height="277" src="caracteristicas-del-cuerpo-del-tigre.jpg" alt="The Scream"
style="display:none">
 </body>
</html>
```

Ejemplo

```
<!DOCTYPE html>
<html><!-- EN ESTE EJEMPLO VAMOS A ESCALAR LA IMAGEN -->
  <head>
    <meta charset="utf-8"/>
    <title>Canvas tutorial</title>
    <script type="text/javascript">
 function draw() {
        var canvas = document.getElementById('tutorial');
        if (canvas.getContext) {
var ctx = canvas.getContext('2d');
var img = new Image();
img.onload = function() {          //En este caso hemos usado el evento onload en la imagen para pintar
ctx.drawImage(img,0,0, 120,65);
}
img.src="caracteristicas-del-cuerpo-del-tigre.jpg";
}

      }
    </script>
    <style type="text/css">
      canvas { border: 1px solid black; }
    </style>
  </head>
  <body onload="draw();">
    <canvas id="tutorial" width="150" height="150"></canvas>
  </body>
</html>
```

Graficar texto:

- Ejemplo:

```
lienzo.fillStyle="rgb(255,0,0)";
lienzo.font="bold 25px Arial";
lienzo.fillText("Hola Mundo",150,50);
lienzo.textAlign="center";
lienzo.fillText("Hola Mundo",150,100);
lienzo.textAlign="right";
lienzo.fillText("Hola Mundo",150,150);
var anchopx=lienzo.measureText("Hola Mundo");
lienzo.textAlign="start";
lienzo.fillText(anchopx.width,150,200);
lienzo.strokeStyle="rgb(0,0,255)";
lienzo.strokeText("Fin",150,250);
```

Grabar y recuperar el estado: save(), restore()

- Cuando tenemos que hacer dibujos complejos es muy común que necesitemos almacenar el estado de algunas propiedades del canvas para luego recuperarlas. Estas dos actividades se logran mediante los métodos:
- save()
- restore()
- Los valores que se almacenan son strokeStyle, FillStyle, globalAlpha, LineCap, LineJoin,miterLimit, shadowOffsetX, shadowOffsetY, shadowBlur, shadowColor entre otras propiedades.
- Conforman una pila. Si se llama dos veces en forma sucesiva al método save() para retornar al estado inicial debemos llamar dos veces al método restore() (cada restore hace un pop() de la pila).

Transformaciones - Translate

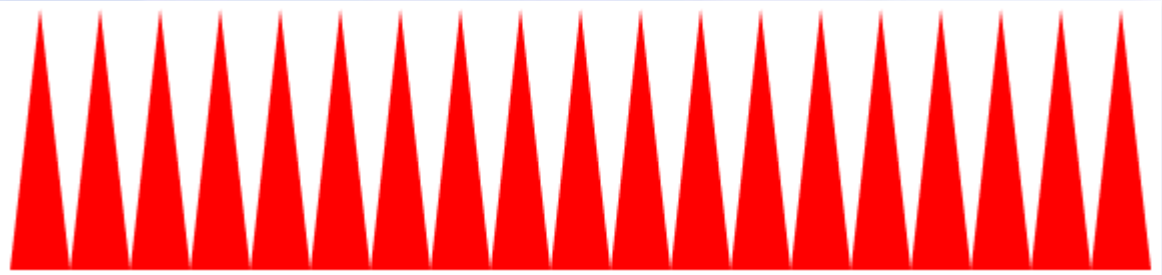
- Para mover el sistema de coordenadas disponemos del método:

  translate(x,y)

- Luego de llamar a este método la coordenada (0,0) corresponderá al par de valores indicados en los parámetros.

Ejemplo

```
<!DOCTYPE html>
<html><!-- EN ESTE EJEMPLO DIBUJAMOS UN TRIÁNGULO -->
 <head>
<script type="text/javascript">
function retornarLienzo(x) {
var canvas = document.getElementById(x);
if (canvas.getContext) {
var lienzo = canvas.getContext("2d");
return lienzo;
} else
return false;
}
function dibujarTriangulo(lienzo,x,y,base,altura) {
lienzo.save(); //Guardamos estado
lienzo.translate(x,y);
lienzo.fillStyle="rgb(255,0,0)";
lienzo.beginPath();
lienzo.moveTo(base/2,0);
lienzo.lineTo(0,altura);
lienzo.lineTo(base,altura);
lienzo.lineTo(base/2,0);
lienzo.fill();
lienzo.restore(); //Volvemos a coordenadas ori
}
function dibujar() {
var lienzo=retornarLienzo("lienzo1");
if (lienzo) {
for(var col=0;col<550;col+=30)
dibujarTriangulo(lienzo,col,10,30,130);
}
}
</script>
</head>
<body onLoad="dibujar()">
<canvas id="lienzo1" width="600" height="600">
Su navegador no permite utilizar canvas.
</canvas>
</body>
</html>
```



Transformaciones - Rotate

- La segunda transformación posible utilizando el canvas es la rotación:

  rotate(grados)

- Los grados de rotación se indican en radianes y la rotación es en sentido horario y con respecto al punto de origen (0,0), por ello es muy común que primero utilicemos el método translate y luego rotemos.

Ejemplo

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function retornarLienzo(x) {
var canvas = document.getElementById(x);
if (canvas.getContext) {
var lienzo = canvas.getContext("2d");
return lienzo;
} else
return false;
}
function dibujar() {
var lienzo=retornarLienzo("lienzo1");
if (lienzo) {
lienzo.clearRect(0,0,600,600);
lienzo.save();
lienzo.fillStyle="rgb(255,0,0)";
lienzo.translate(300,300);  //centro del canvas
lienzo.rotate(avance);
lienzo.fillRect(-100,-100,200,200);  //números negativos porque es respecto al centro
lienzo.restore();
avance+=0.025;
if (avance>Math.PI*2)
avance=0;
}
}
var avance=0;
function inicio() {
setInterval(dibujar,50);
}
</script>
</head>
<body onLoad="inicio()">
<canvas id="lienzo1" width="600" height="600">
Su navegador no permite utilizar canvas.
</canvas>
</body>
</html>
```

Ejercicio 1

- Mostrar la hora segundo a segundo en la parte superior del canvas centrado.
  - Pista: habrá que borrar lo pintado para pintar nuevo. Puedes repintar el lienzo entero con un clearRect.

Ejercicio 2

- Importar una imagen y hacer que gire indefinidamente en un mismo sentido mientras se va escalando hasta un punto y volviendo.
  - Funciones a utilizar:
    - lienzo.rotate(avance);
    - lienzo.scale(tamx,tamy);