Una de las novedades de HTML5 ha sido la posibilidad de hacer dibujos (crear imágenes con HTML) mediante las etiquetas Canvas y SVG.

INCLUIR UN ELEMENTO CANVAS

Canvas no es más que un lienzo (traducción literal del ingles) en el que podremos dibujar lo que nos parezca, básicamente como si estuviésemos utilizando el Paint de Windows aunque eso si hay que decirle cada cosa que se quiera hacer mediante una instrucción. Por lo tanto la etiqueta Canvas siempre será como la del código siguiente con los cambios lógicos del ancho y alto para cada situación y también decir que siempre hay que incluir la etiqueta id para poder pintar en él.

```
<canvas id="micanvas" width="500px" height="200px">

    Tu navegador no soporta canvas. <!-- Mensaje para mostrar si el
navegador no soporta la etiqueta canvas -->

</canvas>
```

Para pintar en la etiqueta canvas hay utilizar javaScript (no hace falta saber javaScript para pintar en canvas). El primer paso es obtener el canvas en el que se quiere pintar de hay la razón de usar un id para el canvas y en la segunda línea se obtiene el contexto 2d para poder dibujar. Estas 2 líneas siempre estarán y se pintara sobre la variable cxt.

Canvas es una etiqueta más de HTML5 por lo tanto para definir su borde, color de fondo o el estilo que se quiera poner se hace mediante CSS.

```
var c = document.getElementById("micanvas");

var cxt = c.getContext("2d");

// Aqui ira el codigo para dibujar
```

DIBUJAR UN RECTANGULO EN CANVAS

Se pueden pintar 2 tipos de rectángulos en canvas, rectángulos rellenos (de un color, un degrado o una imagen) o dibujar solo el borde del rectángulo.

```
var c = document.getElementById("micanvas");

var cxt = c.getContext("2d");



cxt.fillStyle = "#123456"; // Definimos el color para rellenar el rectangulo

cxt.fillRect(30,10,50,100); // Dibuja un rectangulo relleno -
fillRect(x,y,width,height)



cxt.strokeStyle = "red"; // Definimos el color para pintar el borde

cxt.strokeRect(70,32,100,50); // Dibuja el borde del rectangulo -
strokeRect(x,y,width,height)



cxt.fillRect(145,25,70,70); // Dibuja un rectangulo relleno del ultimo color
definido con fillStyle



cxt.fillStyle = "rgba(220,220,10,0.7)"; // Definimos otro color para
rellenar el rectangulo

cxt.fillRect(180,10,100,100); // Dibuja un rectangulo relleno



cxt.clearRect(30,10,20,20); // Borra el contenido que hubiese en el area del
rectangulo definido



cxt.strokeRect(190,20,80,80); // Dibuja el borde del rectangulo de definido
por ultima vez con strokeStyle.
```

El resultado del script anterior es el de la imagen siguiente que como puedes comprobar es una imagen como cualquiera que pudiera incluir pero con la diferencia de que esta imagen sea creado con HTML (Si usas opción inspeccionar elemento de Firefox u otra similar de otro navegador veras como la imagen es la etiqueta canvas, o simplemente viendo el código fuente de la página).

Con los comentarios se puede claramente que es lo hace cada línea, primero hay que definir el estilo con el que se pintara (fillStyle y strokeStyle) y se le pone el color que se quiera con cualquiera de los formatos que se pueden usar incluidos los colores con transparencia con rgba incluidos con CSS3.

Y para fillRect y strokeRect los parámetros que se le pasan son las coordenadas de la esquina superior izquierda (las coordenadas se toman a partir de la esquina superior izquierda del canvas que es la 0,0) y el ancho y el alto del rectángulo. Y clearRect igual que las dos anteriores pero en lugar de dibujar borra.

HTML

```
</script><canvas id="rectangulos" width="300px" height="120px">
 Tu navegador no soporta canvas. <!-- Mensaje para mostrar si el navegador no soporta la etiqueta canvas -->
</canvas>
```

CSS

```
body {
  background: rgb(29, 31, 32);
}
#rectangulos {
  margin: 20px auto;
  display: block;
}
```

JS

```
var c = document.getElementById("rectangulos");
var cxt = c.getContext("2d");

cxt.fillStyle = "#123456"; // Definimos el color para rellenar el rectangulo
cxt.fillRect(30,10,50,100); // Dibuja un rectangulo relleno -
fillRect(x,y,width,height)

cxt.strokeStyle = "red"; // Definimos el color para pintar el borde
cxt.strokeRect(70,32,100,50); // Dibuja el borde del rectangulo -
strokeRect(x,y,width,height)

cxt.fillRect(145,25,70,70); // Dibuja un rectangulo relleno del ultimo color
definido con fillStyle

cxt.fillStyle = "rgba(220,220,10,0.7)"; // Definimos otro color para
rellenar el rectangulo
cxt.fillRect(180,10,100,100); // Dibuja un rectangulo relleno

cxt.clearRect(30,10,20,20); // Borra el contenido que hubiese en el area del
rectangulo definido

cxt.strokeRect(190,20,80,80); // Dibuja el borde del rectangulo de definido
por ultima vez con strokeStyle.
```

Para acabar con los rectángulos decir que cada elemento se pinta encima de los que están definidos anteriormente aunque ya están colocados en el ejemplo para que se vea bien, aunque es lógico que sea así lo digo que por escribir 2 líneas no pasa nada.

DIBUJAR LÍNEAS RECTAS EN CANVAS

En esta apartado veremos como pintar líneas sueltas y hacer figuras con y sin relleno. Pero empecemos por el principio viendo como se dibuja una línea en canvas.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        body {
            background: white;
        }
        #lineas {
            margin: 20px auto;
            display: block;
        }
    </style>
</head>
<body>
    <canvas id="lineas" width="300px" height="120px">
        Tu navegador no soporta canvas.
        <!-- Mensaje para mostrar si el navegador no soporta la etiqueta
canvas -->
    </canvas>
    <script>
        var c = document.getElementById("lineas");
        var cxt = c.getContext("2d");
        cxt.lineWidth = 4;  // Cambia el grosor de la linea
        cxt.lineJoin = "round"; // Hace que la union entre lineas sea
redondeada
        cxt.moveTo(0, 0); // Punto de inicio de la linea
        cxt.lineTo(300, 120); // Punto final de la linea e inicio de la
siguiente linea
        cxt.lineTo(300, 0); // Punto final de la linea
        cxt.moveTo(300, 0); // Punto inicial de la linea
        cxt.lineTo(0, 120); // Punto final de la linea
        cxt.stroke();  // Pinta las lineas
    </script>
</body>

</html>
```

@mihifidem 01/2024

El ejemplo anterior es el más simple posible y como se puede ver para dibujar una línea hacen falta 3 instrucciones:

moveTo(x,y): Sirve para indicar las coordenadas en las que comenzara la línea.

lineTo(x,y): Indica las coordenadas hasta las que se pinta la línea (Desde el moveTo o lineTo anterior).

stroke(): Pinta las lineas.

Otras funciones:

lineWidth: Esta función sirve para cambiar el grosor de las líneas cuando se usa stroke tanto para líneas como para cualquier otra figura como los rectángulos.

lineJoin: Con esta función se puede definir como será la unión entre las líneas. Puede tomar 3 valores miter (con esquina), round (redondeado) y bevel (esquinas recortadas)

Con las instrucciones anteriores ya se pueden dibujar las líneas sueltas pero para hacer dibujos con líneas además de las anteriores existen:

beginPath(): Con esta función se indica que vamos a comenzar a dibujar una figura mediante lineas.

closePath(): Cierra la figura que se esté haciendo pintando la línea desde el ultimo punto definido hasta el primero.

fill(): Como en los rectángulos existe la posibilidad de usar stroke o fill, para usar fill la figura debe de estar cerrada o que solo quede una línea para cerrar y entonces esta función creara la línea restante como closePath y rellanara la figura resultante.

Veamos estas funciones en acción con un dibujito de un castillo con una banderita:

DIBUJAR CIRCULOS Y ARCOS EN CANVAS

No hay una función exclusiva para dibujar círculos sino que la función que hay es para dibujar arcos y si se quiere hacer un circulo pues habrá que usar un arco de 2π (es decir se usan radianes).

```
var c = document.getElementById("micanvas");
var cxt = c.getContext("2d");

cxt.beginPath();
cxt.arc(60,60,40,0,Math.PI*2,true);
//arc(x,y,radio,angulo_inicio,angulo_final,sentido[antihorario-true / horario-false])
cxt.closePath;
cxt.stroke();

cxt.beginPath();
cxt.arc(140,60,40,0,Math.PI,true); // Un arco de medio circulo en sentido antihorario
cxt.closePath;
cxt.stroke();

cxt.beginPath();
cxt.arc(220,60,40,0,Math.PI*2,false); // Dibuja el circulo en sentido horario
cxt.closePath;
cxt.fill(); // Un circulo relleno

cxt.beginPath();
cxt.arc(140,60,30,Math.PI,Math.PI*1.5); // Un arco de medio circulo en sentido horario
cxt.closePath;
cxt.fill();

cxt.beginPath();
cxt.arc(140,60,30,0,Math.PI*0.5); // Un arco de medio circulo en sentido horario
cxt.lineTo(140,60);
cxt.closePath;
cxt.fill();
```

Esta función es fácil de entender y cualquiera puede hacer un circulo sin ningún problema pero para los arcos la gente que es un poco negada en matemáticas quizás se pierda entre ángulos radianes y demás. Pero la función tampoco da para mucha explicación hay que pasarle las coordenadas del centro del circulo, el radio, el inicio y el fin del arco y para terminar el sentido en el que se pinta true para el sentido anti horario y false para el horario (Si se omite este parámetro por defecto es false por lo que pinta en sentido horario).

Aunque no es la finalidad de este post enseñar matemáticas pues recordemos para los que no lo tengan muy claro o lo tengan olvidado que una circunferencia tiene 2π radianes y que coincide con la posición de los 0 radianes que es la mas a la derecha del, 90 grados son 0.5π radianes (lo más bajo del circulo), 180 grados π radianes (lo más a la izquierda del circulo) y 270 grados pues 1.5π radianes (lo más alto del circulo).

PINTAR CURVAS EN CANVAS

En el apartado anterior hemos visto un tipo de curvas, los arcos pero hay otro tipo de curvas las curvas de Bézier (pongo un enlace para la gente que no sepa muy bien de que va porque esto ya habrá mucha gente que no lo habrá estudiado, aunque muchos si que las habrán usado en programas como photoshop). Y hay 2 funciones para pintar curvas de Bézier la función quadraticCurveTo para pintar curvas de Bézier cuadráticas y la función bezierCurveTo para pintar curvas cubicas de Bézier.

```
var c = document.getElementById("micanvas");
var cxt = c.getContext("2d");

// Un jarron dibujado con curvas cuadraticas de Bezier
cxt.beginPath();
cxt.moveTo(40,10);
cxt.quadraticCurveTo(100,40,60,60);
cxt.quadraticCurveTo(20,80,60,100);
cxt.lineTo(100,100);
cxt.quadraticCurveTo(140,80,100,60);
cxt.quadraticCurveTo(60,40,120,10);
cxt.closePath();
cxt.fill();

// Un dibujo con curvas cubicas de Bezier
cxt.beginPath();
cxt.moveTo(200,20);
cxt.bezierCurveTo(100,100,85,35,200,50);
cxt.bezierCurveTo(200,100,320,35,260,100);
cxt.stroke();

// Una esquina redondeada con la funcion arcTo
cxt.beginPath();
cxt.lineTo(10,10);
cxt.arcTo(40,10,40,20,20);  // Dibuja una esquina redondeada
arcTo(controlX,controlY,endX,endY,radio en grados);
cxt.lineTo(40,40);
cxt.stroke();
```

Una pequeña explicación de cómo se usan estas funciones:

quadraticCurveTo(pax,pay,x,y): En esta función al igual que lineTo el punto de inicio no se es un parámetro sino que la curva comenzara donde se halla dejado el «cursor» con la función moveTo o con cualquier otra. Los parámetros que se le pasan son las coordenadas del punto de anclaje (que fuerza a que se cree una curva que partiendo del inicio pase por este punto y llegue hasta el punto final) y del punto final de la línea.

bezierCurveTo(pa1x,pa1y,pa2x,pa2y,x,y): De forma análoga a la función anterior se le pasan las coordenadas de los 2 puntos de anclaje y del final de la línea.

arcTo(pax,pay,x,y,radio): Con esta función se pueden hacer curvas como por ejemplo esquinas redondeadas.

Como se puede ver en el ejemplo con las curvas cuadráticas se pueden hacer cosas bastante interesantes y con las cubicas cosas realmente complejas.

INSERTAR IMÁGENES EN UN CANVAS

También se pueden insertar imágenes en un canvas de la siguiente forma:

```
var c = document.getElementById("micanvas");
var cxt = c.getContext("2d");

var imagen = new Image(); // Crea una imagen para luego poder mostrarla
imagen.src = 'ruta/imagen.jpg'; // Se define la ruta de la imagen
imagen.onload = function(){ // Mediante el evento onload se espera a que se
cargue la imagen
    cxt.drawImage(imagen, 50, 50); // Dibuja la imagen en la posicion
indicada
}

// Otra alternativa
var img = document.getElementById("imagen"); // Recupera una imagen del html
mediante su id
cxt.drawImage(img, 50, 50, 200, 140); // Dibuja la imagen en la posicion
indicada y con un tamaño de 200x140

// Rellenar un objeto con una imagen
var patron = context.createPattern(imagen, repeat-x);
cxt.fillStyle = patron;
cxt.fillRect(10,10,200,200);
```

Como se muestra en el ejemplo para incluir una imagen en canvas además de la función de canvas necesaria para incluir la imagen es necesaria la función de javaScript para crear imágenes y por su puesto hay que definir la ruta de la imagen. Además es necesario usar el evento onload sobre la imagen para que la imagen se dibuje cuando ya este cargada porque en caso contrario se ejecutaría la función antes de que estuviese la imagen completamente cargada con lo que fallaría. Como esta puesto en el ejemplo hay otra alternativa para cargar la imagen y es que esta este en el HTML y que la recuperemos mediante su id.

También se puede rellenar cualquier objeto creando un patrón y luego eligiéndolo como estilo de relleno. La función createPattern(imagen, repetición) que recibe una imagen y como segundo parámetro como queremos que se repita (repeat, repeat-x, repeat-y y no-repeat).

INSERTAR TEXTO EN UN CANVAS

Como no podía ser vemos en canvas también se pueden insertar textos, y como sucede en el resto de figuras el texto puede estar relleno o solo los contornos.

```
var c = document.getElementById("micanvas");
var cxt = c.getContext("2d");

cxt.font = "40px Calibri, Arial";  // Fuente para el texto
cxt.fillText("Un texto",50,50);  // Texto relleno

cxt.font = "40px Book Antiqua, Sans-serif";
cxt.strokeText("Otro texto",50,100); // Texto contornos
```

Poco hay que explicar de fillText y strokeText, pues solo escriben el texto pasado como parámetro empezando en las coordenadas que se le indiquen (las coordenadas de la esquina inferior izquierda del texto). Y con la función font pues indicamos la fuente con la que se escribirá (como ocurre en CSS se pueden indicar varias fuentes o familias de fuentes por si el usuario no tiene alguna).

HACER DEGRADADOS EN UN CANVAS

Y para terminar con los canvas pues vamos a ver como hacer degradados para luego rellenar cualquier figura o texto creado. Hay 2 tipos de degradados, degradados lineales y degradados circulares. Con los degradados lineales pues se pueden hacer degradados horizontales, verticales y diagonales y con los circulares pues degradados circulares.

```
var c = document.getElementById("micanvas");
var cxt = c.getContext("2d");

// Gradiente vertical
var grt = cxt.createLinearGradient(0,10,0,50);
grt.addColorStop(0,"red");  // Color de inicio del gradiente
grt.addColorStop(1,"yellow"); // Color de fin del gradiente

cxt.strokeStyle = grt; // Pone el gradiente como estilo paa pintar lineas
cxt.font = "40px Book Antiqua, Sans-serif";
cxt.strokeText("Degradados",50,40); // Texto contornos

// Gradiente horizontal
var grt2 = cxt.createLinearGradient(50,0,150,0);
grt2.addColorStop(0,"red");  // Color de inicio del gradiente
grt2.addColorStop(0.3,"yellow"); // Color intermedio
grt2.addColorStop(0.6,"green"); // Color intermedio
grt2.addColorStop(1,"blue"); // Color de fin del gradiente
cxt.fillStyle = grt2;
cxt.fillRect(50,50, 100, 50);

// Gradiente en diagonal
var grt3 = cxt.createLinearGradient(160,50,200,150);
grt3.addColorStop(0,"white");  // Color de inicio del gradiente
grt3.addColorStop(0.2,"black"); // Color intermedio
grt3.addColorStop(0.4,"white"); // Color intermedio
grt3.addColorStop(0.6,"black"); // Color de fin del gradiente
cxt.fillStyle = grt3;
cxt.fillRect(160,50, 100, 50);

//Gradiente circular
var grt4 = cxt.createRadialGradient(205,75,2,210,80,35);
grt4.addColorStop(0,"red");  // Color de inicio del gradiente
grt4.addColorStop(0.3,"rgba(0,0,255,0.3)");  // Color con transparencia en
el gradiente
grt4.addColorStop(1,"#ffff00"); // Color de fin del gradiente
cxt.fillStyle = grt4;
cxt.fillRect(180,60, 60, 40)
```

En este ejemplo se ven en acción los 2 tipos de degradados, aplicados en distintos ángulos y sobre distintos elementos y como se ve en el ejemplo para pintar un objeto con un gradiente se hace como si el gradiente fuese un color (strokeStyle o fillStyle = gradiente). Y para terminar la explicación de las funciones para hacer degradados:

createLinearGradient(ix,iy,fx,fy): Con esta función se crean degradados lineales pasando las coordenadas del comienzo y del final del gradiente. Para hacer un degradado vertical se dejan las coordenadas iy y fy a 0 y con las coordenadas ix y fx se indica desde donde hasta donde ira el degradado, para un degrado horizontal se hace de forma análoga pero dejando ix y fx a 0 y dando valores a iy y fy, y para un degradado en diagonal pues se dan los valores que se quieran dependiendo del ángulo deseado.

createRadialGradient(ix,iy,radio_inicial,fx,fy,radio_final): Con ix e iy se define el inicio del gradiente y con fx y fy el final, pero además es necesario indicar el radio inicial y el final (radio del color inicial y radio del degradado completo).

addColorStop([0-1],color): El primer parámetro indica el porcentaje (el valor tiene que estar entre 0 y 1) en el que se colocara el color que se le pasa como segundo parámetro. Esta función se puede usar tantas veces como colores se quieran añadir al degradado y se pueden usar todas las etiquetas de color como en el resto de elementos de canvas incluidos los colores con transparencia. Esta función se aplica sobre el degradado y no sobre el contexto como el resto.