



Confiabilidad, pruebas y análisis de la información

Canvas

Canvas ofrece un método con el cual se toma el contexto de renderizado y todas las funciones de dibujo y es el `getContext()`.

```
3   var canvas = document.getElementById("ejemplo");
4   var ctx = canvas.getContext();
```

Ilustración 125. Tomando el contexto canvas

Fuente: Aguilar, J. 2020

La primera línea de código recupera el nodo en el DOM que tiene la representación del canvas y una vez se tenga este nodo se pasa en la otra línea a acceder al contexto de dibujo.

A continuación, una plantilla corta que permite ver un ejemplo de canvas para poder empezar a trabajar y entender este elemento.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Canvas</title>
6      <script type="text/javascript">
7          function draw(){
8              var canvas = document.getElementById('ejemplo');
9              if(canvas.getContext){
10                 var ctx = canvas.getContext("2D");
11             }
12         }
13     </script>
14     <style type="text/css">
15         canvas { border: 1px solid black; }
16     </style>
17 </head>
18 <body>
19     <canvas id="ejemplo" width="150" height="150"></canvas>
20 </body>
21 </html>
```

Ilustración 126. Plantilla creación de canvas.

Fuente: Aguilar, J. 2020

El Script que se ve en el código anterior hace un llamado a una función `draw()` que se ejecuta una vez termine de cargarse la página, esto lo puede hacer a través de escuchar el evento `load` de la página donde está el Script.

La plantilla anterior se vería de la siguiente manera desde un navegador:

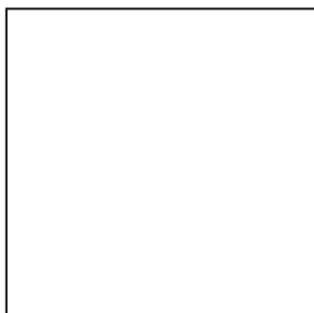


Ilustración 127. Etiqueta canvas vista desde el navegador
Fuente: Aguilar, J. 2020

Ahora, se va a revisar un simple ejemplo como parte del inicio del tema, donde se verán dos rectángulos que se intersectan y a uno de ellos se le aplica transparencia.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Canvas</title>
5 <script type="application/javascript">
6   function draw() {
7     var canvas = document.getElementById("ejemplo");
8     if (canvas.getContext) {
9       var ctx = canvas.getContext("2d");
10
11       ctx.fillStyle = "rgb(200,0,0)";
12       ctx.fillRect (10, 10, 55, 50);
13
14       ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
15       ctx.fillRect (30, 30, 55, 50);
16     }
17   }
18 </script>
19 </head>
20 <body onload="draw();">
21   <canvas id="ejemplo" width="150" height="150"></canvas>
22 </body>
```

Ilustración 128. Ejemplo de canvas.
Fuente: Aguilar, J. 2020

El anterior código como se dijo antes, muestra un ejemplo del uso canvas para graficar dos cuadros que se ven en el navegador de la siguiente manera:



Ilustración 129. Vista del ejemplo en el navegador.

Fuente: Aguilar, J. 2020

Después de ver como se configura canvas y como se hace una gráfica sencilla en el lienzo, ya se puede entrar en detalle a ver algunos de sus elementos.

Para dar inicio y entender canvas, lo primero es hablar de la cuadrícula, este concepto se explicará con la siguiente imagen:

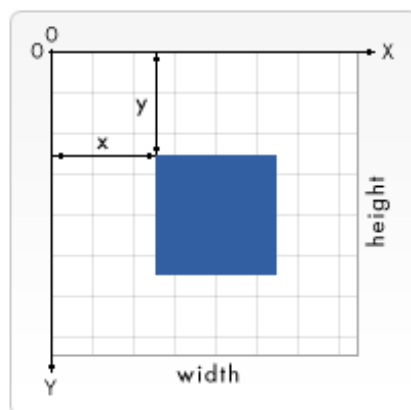


Ilustración 130. Cuadrícula canvas

Fuente: (MDN, 2019)

Como se observa en la imagen, la cuadrícula depende del ancho y el alto del lienzo de canvas, se inicia en la posición cero punto cero y cada cuadro de la cuadrícula corresponde a un pixel. Todos los elementos que se grafiquen están organizados con una posición relativa al origen de la cuadrícula. Así que la posición de la esquina superior izquierda del cuadrado azul es de 'x' pixeles desde la izquierda y 'y' pixeles desde arriba (coordenada (x,y)).

Canvas solo soporta una forma primitiva y es el rectángulo, las demás formas que se deseen crear deben ser combinaciones de varios trazos y listas de puntos conectados entre sí. Pero, canvas ofrece varias funciones que permiten trabajar estos trazos de forma sencilla.

Se iniciará indicando como se dibuja un rectángulo, existen tres funciones para hacerlo.

`fillRect (x, y, width, height).` Esta función dibuja un rectángulo relleno

`strokeRect (x, y, width, height).` Dibuja solo el contorno de un rectángulo.

`clearRect (x, y, width, height).` Se encarga de borrar la zona especificada dejándola transparente.

Las tres funciones tienen los mismos parámetros para su configuración, es decir, necesitan definir un X y Y que especifica la posición del rectángulo según el origen de la cuadrícula y un ancho y un alto que va a definir el tamaño del rectángulo.

El siguiente código muestra un ejemplo del uso de cada una de las funciones anteriores.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Canvas</title>
5 <script type="application/javascript">
6   function draw() {
7     var canvas = document.getElementById('ejemplo');
8     if (canvas.getContext) {
9       var ctx = canvas.getContext('2d');
10
11       ctx.fillRect(25,25,150,100);
12       ctx.clearRect(42.5,45,90,60);
13       ctx.strokeRect(48.5,50,75,50);
14     }
15   }
16 </script>
17 </head>
18 <body onload="draw();">
19   <canvas id="ejemplo" width="150" height="150"></canvas>
20 </body>
21 </html>

```

Ilustración 131. Funciones de dibujar rectángulos.

Fuente: Aguilar, J. 2020

La línea 11 muestra cómo se dibuja un rectángulo relleno con una posición en X y Y de 25 píxeles y la figura a dibujar tendrá un ancho de 150 píxeles y un alto de 100 píxeles. Después viene la función que elimina y deja transparente según lo que ocupe y tiene una posición en X de 42.5 píxeles y en Y de 45 píxeles.

Por último, la línea 13 muestra la figura que dibuja solo el borde del rectángulo y también tiene sus parámetros de posición en X y Y con su respectivo ancho y alto.



Ilustración 132. Rectángulos graficados con cada función

Fuente: Aguilar, J. 2020

Con estas funciones vistas se pueden dibujar rectángulos o cuadrados, pero las demás formas toca hacerlas usando trazos.

Para hacer trazos dentro de canvas se deben seguir unos pasos

- Se crea el trazo en primera medida
- Después, se usan comandos de dibujo, para dibujar dentro del trazo
- Se cierra el trazo
- Cuando ya se cierra el trazo, se le puede dar contorno o relleno para renderizar.

Las funciones que se utilizan para los pasos anteriores son:

`beginPath()`. Es la encargada de iniciar el trazo y así poder empezar a usar los comandos de dibujos dentro de él.

`closePath()`. Cierra el trazo lo que causa que todos los comandos de dibujos posteriores se apliquen al contexto principal.

`stroke()`. Dibuja el contorno de una forma.

`fill()`. Dibuja una forma sólida relleno de la forma de trazo.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Canvas</title>
5   <script type="application/javascript">
6     function draw() {
7       var canvas = document.getElementById('ejemplo');
8       if (canvas.getContext){
9         var ctx = canvas.getContext('2d');|
10        ctx.beginPath();
11        ctx.moveTo(75,50);
12        ctx.lineTo(100,75);
13        ctx.lineTo(100,25);
14        ctx.closePath();
15        ctx.fill();
16      }
17    }
18  </script>
19 </head>
20 <body onload="draw();">
21   <canvas id="ejemplo" width="150" height="150"></canvas>
22 </body>

```

Ilustración 133. Dibujando un triángulo.

Fuente: Aguilar, J. 2020

Después de ver las funciones para graficar trazos se puede explicar el código del dibujo que se hizo.

En la línea 10 se inicia el trazo, la función principal que no grafica nada, pero sirve para poner la pluma o el cursor en el punto iniciar es `moveTo()`. Esta función lo que hace es mover el cursor desde el origen hasta el punto X, Y dado por el programador.

Después las líneas 12 y 13 realizan el trazo desde el nuevo punto de origen de la figura que es (75, 50) y traza una línea hasta el punto (100, 75), la línea 13 traza una línea desde el punto de origen hasta el punto (100, 25). Teniendo esas dos líneas trazadas con la función `closePath()` de la línea 14 se cierra el trazo y por ultimo con la función `fill()` se rellena el trazo o la figura creada.

Este trazo creado visto desde el navegador se ve de la siguiente manera.



*Ilustración 134, Triangulo creado a través de trazos.
Fuente: Aguilar, J. 2020*

Con los trazos dados se crea y rellena un triángulo.

Otro ejemplo con trazos es la siguiente figura:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Canvas</title>
5 <script type="application/javascript">
6   function draw() {
7     var canvas = document.getElementById('ejemplo');
8     if (canvas.getContext){
9       var ctx = canvas.getContext('2d');
10      ctx.beginPath();
11      ctx.arc(75,75,50,0,Math.PI*2,true); // Circulo externo
12      ctx.moveTo(110,75);
13      ctx.arc(75,75,35,0,Math.PI,false); // Boca (contra reloj)
14      ctx.moveTo(65,65);
15      ctx.arc(60,65,5,0,Math.PI*2,true); // Ojo izquierdo
16      ctx.moveTo(95,65);
17      ctx.arc(90,65,5,0,Math.PI*2,true); // Ojo derecho
18      ctx.stroke();
19    }
20  }
21 </script>
22 </head>

```

*Ilustración 135. Creando figura con trazos.
Fuente: Aguilar, J. 2020*

El código anterior con la función `arc()`, crea un arco

`arc(x, y, radius, startAngle, endAngle, anticlockwise)`. Dibuja un arco de acuerdo a los parámetros dados. X y Y ya son parámetros reconocidos. Radius es un parámetro que por sí solo ya se conoce en una circunferencia, `startAngle` y `endAngle` definen el inicio y fin del arco en radianes a lo largo de toda la circunferencia. Por último, `anticlockwise` define el sentido en el que va a dibujar, si es `true` dibuja al contrario de las manecillas del reloj y si es `false` dibuja según las manecillas del reloj.

Según la explicación, todo el código anterior se ve de la siguiente manera en el navegador.



Ilustración 136. Dibujando arcos con canvas.

Fuente: Aguilar, J. 2020

Con canvas no solo se pueden dibujar triángulos, círculos y rectángulos, sino que con la combinación de trazos se puede graficar curvas de Bézier, curvas cúbicas y cuadráticas Bézier, etc.

Para más información puede consultar la página de w3schools o el siguiente enlace <https://devcode.la/tutoriales/introduccion-a-canvas-de-html5/>.