TEMA 19
URL Encode y HTML Canvas

¿Qué es el URL Encoding?

Es el proceso de convertir caracteres en una URL en un formato que se pueda transmitir de manera segura a través de Internet. Se utiliza para manejar caracteres especiales, espacios y otros símbolos que podrían causar problemas en una URL.

Para poder incluir caracteres especiales o reservados en una URL, se usa el URL Encoding, que solamente es una transformación de caracteres especiales en otro formato que es legible por el navegador.

¿Cómo funciona?

Los caracteres se reemplazan por una secuencia especial que comienza con %, seguida de un código hexadecimal que representa el carácter.

- Por ejemplo:
- Espacio ( ) → %20
- & → %26
- / → %2F
- = → %3D

Ejemplo:

Si pones en el navegador la siguiente dirección:

https://www.google.com/search?q=hola mundo

Veréis como al poner la dirección en el navegador, el propio navegador hace una transformación de esta, porque los espacios en blanco no están permitidos en las URL's, ya que puede generar errores.

La transformación sería:

https://www.google.com/search?q=hola%20mundo

Formato del URL Encoding

En el ejemplo anterior habéis podido comprobar como el espacio en blanco ha sido reemplazado por el código **%20**. Significa que el URL Encoding reemplaza los caracteres no permitidos con una secuencia de % seguido por su código ASCII en hexadecimal.

Ejemplo:

Carácter " " (espacio)

Su valor ASCII decimal es: 32.

En hexadecimal es 20.

Si representamos en URL Encoding su valor es %20.

Como ejercicio para el alumno:

Si quisiéramos por ejemplo ver el código URL Encoding del carácter "&", cual sería. Represéntalo en ASCII, en Hexadecimal y en URL Encoding.

Para ver el código ASCII completo, tenéis el siguiente enlace.

https://elcodigoascii.com.ar/

Podéis utilizar la calculadora de Windows en modo programador para hacer una simple transformación a Hexadecimal.

Otros ejemplos:

Carácter	ASCII (decimal)	Hexadecimal	URL Encoding
@	64	40	%40
/	47	2F	%2F
?	63	3F	%3F

Principales caracteres codificados

Los caracteres en URLs se dividen en:

- **Caracteres reservados:** Se utilizan en la sintaxis de URLs.
- **Caracteres no reservados:** Se pueden codificar, pero no es obligatorio.

Caracteres reservados:

Carácter	Uso en URL	Codificado
`:`	Separador entre esquema y dominio (ej: `https://`)	`%3A`
`/`	Separador de rutas (ej: `/productos/1`)	`%2F`
`?`	Inicia parámetros de consulta (ej: `?id=5`)	`%3F`
`&`	Separador de parámetros (ej: `?id=5&name=Juan`)	`%26`
`=`	Asigna valores en parámetros (ej: `name=Juan`)	`%3D`
`+`	Espacio en algunos contextos	`%2B`

Diferencias entre URL Encoding y otras codificaciones

URL Encoding: para codificar caracteres especiales en las URL's. con su valor hexadecimal.

Base64: Para codificar cualquier dato (texto,binario) en caracteres seguros.

UTF-8: Codificación estándar para texto en la web. Es una codificación de caracteres utilizada para representar texto en Unicode, con 1 a 4 bytes por carácter.

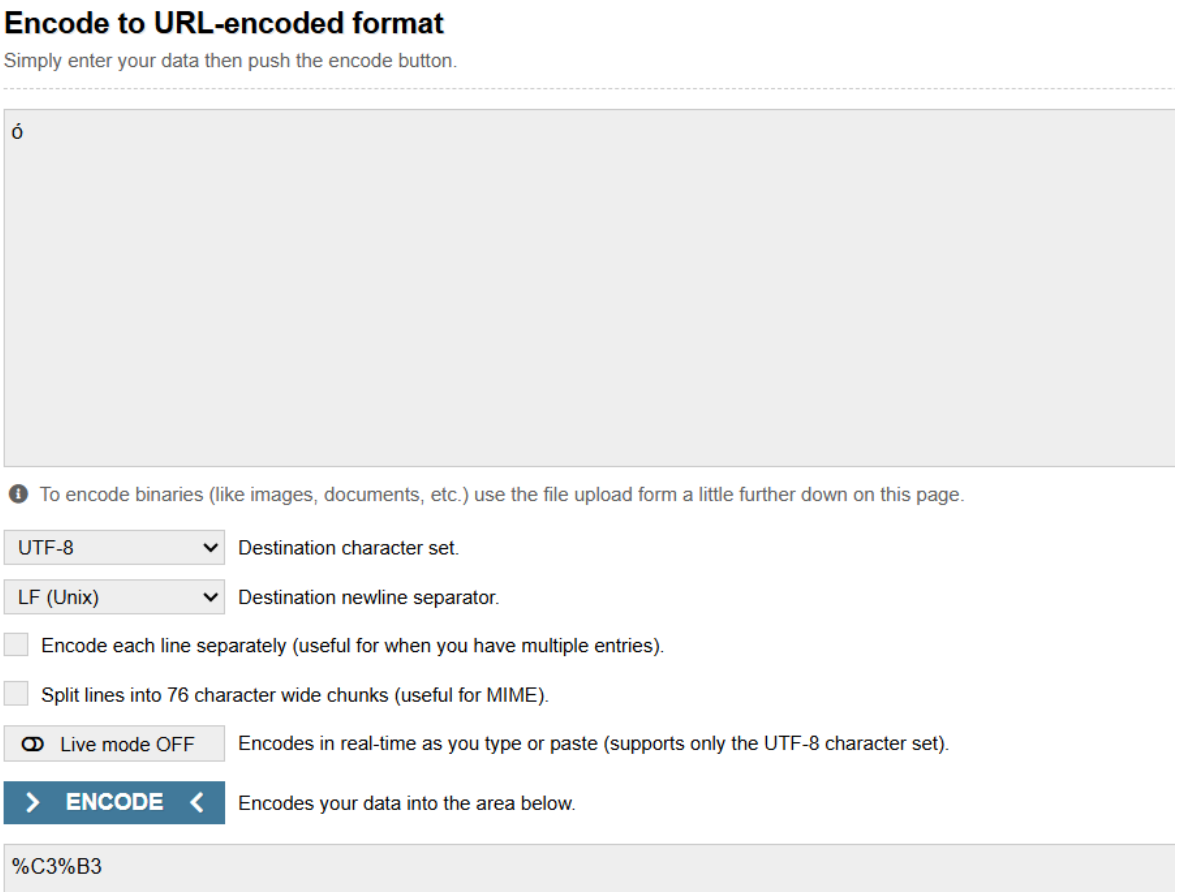
Ejemplo:

El texto "Hola mundo" en distintas codificaciones:

1. **URL Encoding** → "Hola%20mundo"
2. **Base64** → "SG9sYSBtdW5kbw=="
3. **UTF-8 (sin cambios, pero requerido en caracteres especiales)**

Para decodificar o codificar una URL podéis utilizar esta página web:

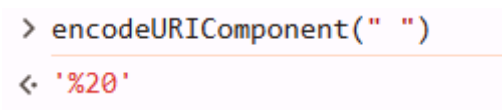
https://www.urlencoder.org/



Pero también podéis utilizar esta función de javascript en la consola de vuestro navegador:

encodeURIComponent("ó")

Como podéis ver da el mismo resultado. Podéis probar los elementos que hemos visto antes como el espacio en blanco, el símbolo &,?.



¿Qué es HTML Canvas?

El elemento HTML <canvas> se utiliza para **dibujar gráficos**, sobre la marcha, a través de JavaScript.

El elemento <canvas> es solo un contenedor para gráficos. Debe utilizar JavaScript para dibujar los gráficos.

Canvas tiene varios métodos para dibujar rutas, cuadros, círculos, texto y agregar imágenes.

Canvas es compatible con todos los navegadores principales.

Para continuar con el tema vamos a crear una página denominada: Tema19.html:

La creamos con lo básico, creamos su hoja de estilo:

tema19.html > html > body

```
1   <!DOCTYPE html>
2   <html lang="es">
3
4   <head>
5       <meta charset="UTF-8">
6       <!-- ETIQUETA PARA HACER RESPONSIVE LA PÁGINA, ADAPTACIÓN DE ESCALAS -->
7       <meta name="viewport" content="width=device-width, initial-scale=1.0">
8       <title>Tema 19 html</title>
9       <!-- ICONO DE VENTANA -->
10      <link rel="icon" type="image/x-icon" href="imagenes/favicon.ico">
11      <!-- HOJA DE ESTILOS EXTERNA-->
12      <link rel="stylesheet" href="estilos/estilostema19.css">
13  </head>
```

Estilostema19.css:

```
estilos > # estilostema19.css > ...
1   body {
2       font-family: Arial, sans-serif;
3       margin-left: 10px;
4       padding: 5px;
5   }
6
7   h2 {
8       color: #4d77ca;
9       text-align: center;
10  }
11
12
```

Ahora pondremos nuestro primer canvas:

```
<body>
    <h2>EJEMPLO CANVAS</h2>
    <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;"></canvas>

</body>
```

Debería de aparecer el siguiente marco, le hemos puesto estilo pues por defecto, al igual que las tablas no aparecen los bordes:

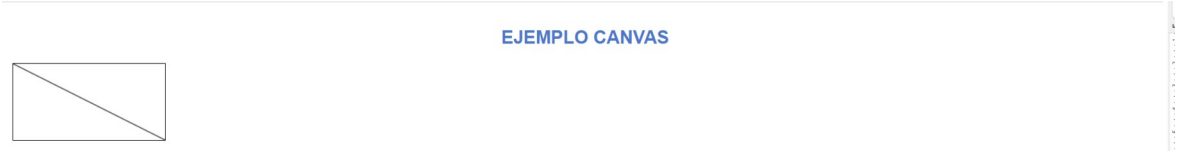
EJEMPLO CANVAS

Para dibujar vamos a ver algo de código en javascript:

Ponemos el siguiente código en nuestro archivo:

```
<body>
    <h2>EJEMPLO CANVAS</h2>
    <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;"></canvas>

    <script>
        //Recogemos el elemento canvas por su ID:
        var c = document.getElementById("myCanvas");
        //Preparamos el contexto de dibujo (en 2 dimensiones):
        var ctx = c.getContext("2d");
        //Se situa en la coordenada 0,0 del canvas:
        ctx.moveTo(0, 0);
        //Se mueve hasta la posición 200, 100 del canvas:
        ctx.lineTo(200, 100);
        //sirve para dibujar el camino definido en el canvas:
        ctx.stroke();
    </script>
</body>
```

Y ahora el resultado será:



Podemos configurar el estilo del trazo. Por ejemplo para cambiar el color de la línea y el grosor de la misma:

ctx.strokeStyle = "blue"; // Color del trazo

ctx.lineWidth = 5; // Grosor de la línea

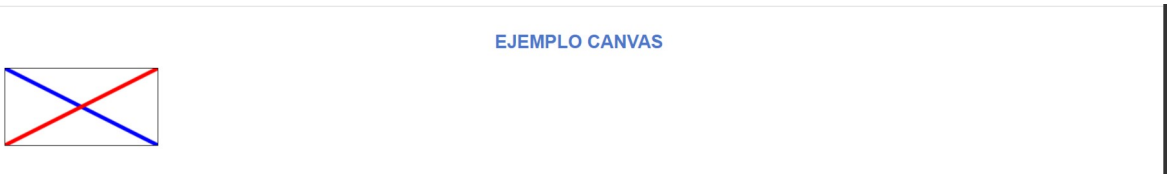
Vamos a añadirlo a nuestro archivo:

```
    <script>
        //Recogemos el elemento canvas por su ID:
        var c = document.getElementById("myCanvas");
        //Preparamos el contexto de dibujo (en 2 dimensiones):
        var ctx = c.getContext("2d");
        // Configurar el estilo del trazo
        ctx.strokeStyle = "blue";  // Color del trazo
        ctx.lineWidth = 5;         // Grosor de la línea
        //Se situa en la coordenada 0,0 del canvas:
        ctx.moveTo(0, 0);
        //Se mueve hasta la posición 200, 100 del canvas:
        ctx.lineTo(200, 100);
        //sirve para dibujar el camino definido en el canvas:
        ctx.stroke();
    </script>
</body>
```

El resultado sería:

Para iniciar un nuevo camino y que se mezclen lineas, se utiliza **beginPath**().

(*) Como ejercicio el alumno deberá de dibujar la siguiente línea de color rojo y con un grosor también de 5:



```
    //LÍNEA AZUL
    ctx.beginPath(); // Iniciar un nuevo camino
    // Configurar el estilo del trazo
    ctx.strokeStyle = "blue";  // Color del trazo
    ctx.lineWidth = 5;         // Grosor de la línea
    //Se situa en la coordenada 0,0 del canvas:
    ctx.moveTo(0, 0);
    //Se mueve hasta la posición 200, 100 del canvas:
    ctx.lineTo(200, 100);
    //sirve para dibujar el camino definido en el canvas:
    ctx.stroke();
```

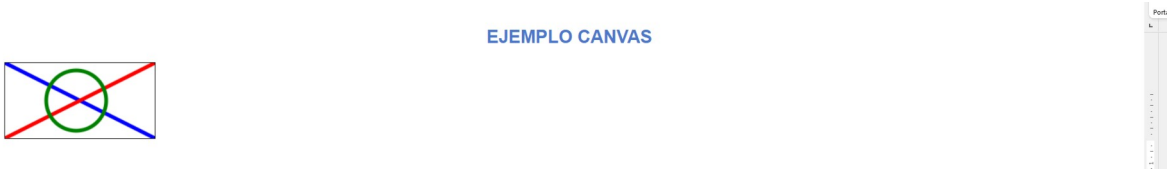
```
    //LÍNEA ROJA
    ctx.beginPath(); // Iniciar un nuevo camino
    // Configurar el estilo del trazo
    ctx.strokeStyle = "red";  // Color del trazo
    ctx.lineWidth = 5;         // Grosor de la línea
    //Dibujar línea de color rojo:
    //Se situa en la coordenada 200,0 del canvas:
    ctx.moveTo(200, 0);
    //Se mueve hasta la posición 0, 100 del canvas:
    ctx.lineTo(0, 100);
    //sirve para dibujar el camino definido en el canvas:
    ctx.stroke();

</script>
```

Con estos cierres de trazos ahora si podemos independizar las características de cada línea.

A continuación, vamos a dibujar un círculo:

```
//DIBUJO DE UN CÍRCULO DE COLOR VERDE:
ctx.beginPath();
ctx.strokeStyle = "green";  // Color del trazo
ctx.lineWidth = 5;          // Grosor de la línea
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
```

Deberá salir la siguiente imagen:



Explicamos los parámetros de arc: ctx.arc(95, 50, 40, 0, 2 * Math.PI);

x (95): Coordenada **X** del centro del círculo.

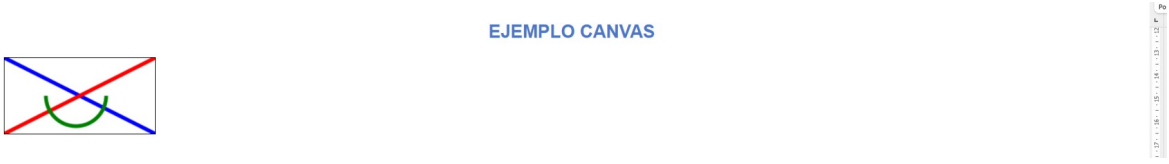
y (50): Coordenada **Y** del centro del círculo.

radio (40): Distancia desde el centro hasta el borde del círculo.

anguloInicial (0): Punto de inicio del arco en radianes (0 radianes = posición de las 3 en un reloj).

anguloFinal (2 * Math.PI): Punto final del arco en radianes. 2 * Math.PI equivale a 360°, lo que forma un círculo completo.

Si cambiáramos el punto final a 1, podemos observar que serían 180º de circunferencia: ctx.arc(95, 50, 40, 0, 1 * Math.PI);



A continuación vamos a escribir texto:

Creamos otro canvas:

```
<body>
    <h2>EJEMPLO CANVAS</h2>
    <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;"></canvas>
    <canvas id="myCanvas2" width="200" height="100" style="border:1px solid #000000;"></canvas>
```

Y a continuación, debajo al final del anterior canvas:

```
//A continuación creamos otro canvas y dibujamos texto:
//Recogemos el elemento canvas por su ID:
var c2 = document.getElementById("myCanvas2");
//Preparamos el contexto de dibujo (en 2 dimensiones):
var ctx2 = c2.getContext("2d");
ctx2.font = "30px Arial";
ctx2.fillStyle = "red";    // Color del texto
ctx2.fillText("Hello World", 10, 50);
```

El resultado sería:

Como habéis apreciado el canvas es inline, cuando se crea uno no salta de línea.

fillText(texto, x,y madwith)

texto: texto a dibujar

x,y: posición donde comienza a dibujarse el texto

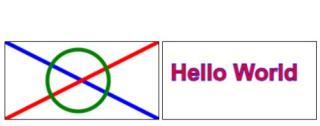
madWith (opcional): El ancho máximo del texto, si excede, el texto se ajusta.

Hemos visto como se rellena el texto de color, ahora veamos como podemos añadir color y ancho a los bordes del texto:

- fillText: dibuja un texto relleno dentro del área del texto
- strokeText: dibuja el contorno del texto, sin rellenarlo.

```
    //A continuación creamos otro canvas y dibujamos texto:
    //Recogemos el elemento canvas por su ID:
    var c2 = document.getElementById("myCanvas2");
    //Preparamos el contexto de dibujo (en 2 dimensiones):
    var ctx2 = c2.getContext("2d");
    ctx2.font = "bold 30px Arial";
    ctx2.fillStyle = "red";    // Color del texto
    ctx2.fillText("Hello World", 10, 50);

    // Texto con contorno (strokeText)
    ctx2.strokeStyle = "blue"; // Color del contorno
    ctx2.lineWidth = 1; // Grosor de la línea del contorno
    ctx2.strokeText("Hello World", 10, 50);
</script>
```

Esto saldría:



Hemos puesto el estilo de relleno “bold” para que se pueda apreciar más, pues no hay un ancho definido para el relleno, viene implícito con el tipo de fuente seleccionada.

Veamos el siguiente ejemplo:

```
    //Rellenamos texto y bordes con fuente de letra grande:
     //A continuación creamos otro canvas y dibujamos texto:
    //Recogemos el elemento canvas por su ID:
    var c3 = document.getElementById("myCanvas3");
    //Preparamos el contexto de dibujo (en 2 dimensiones):
    var ctx3 = c3.getContext("2d");
    ctx3.font = "bold 100px Arial";
    ctx3.fillStyle = "green";    // Color del texto
    ctx3.fillText("HOLA", 130, 150);

    // Texto con contorno (strokeText)
    ctx3.strokeStyle = "blue"; // Color del contorno
    ctx3.lineWidth = 1; // Grosor de la línea del contorno
    ctx3.strokeText("HOLA", 130, 150);

</script>
```

A continuación, veremos como dibujamos una línea gradiante.

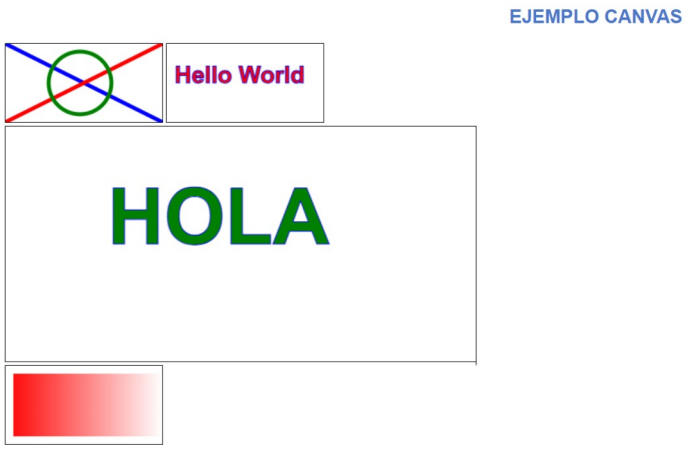
Para ello creamos otro nuevo canvas:

```
dy>
 <h2>EJEMPLO CANVAS</h2>
 <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;"></canvas>
 <canvas id="myCanvas2" width="200" height="100" style="border:1px solid #000000;"></canvas>
 <br>
 <canvas id="myCanvas3" width="600" height="300" style="border:1px solid #000000;"></canvas>
 <br>
 <canvas id="myCanvas4" width="200" height="100" style="border:1px solid #000000;"></canvas>
 <script>
    //Recogemos el elemento canvas por su ID:
    var c = document.getElementById("myCanvas");
```

Y ahora añadimos al final de nuestro código javascript:

```
// gradientes: lineal
var c4 = document.getElementById("myCanvas4");
var ctx4 = c4.getContext("2d");

// Creación gradiente:
// Gradiente lineal:
var grd = ctx4.createLinearGradient(0, 0, 200, 0);

grd.addColorStop(0, "red");
grd.addColorStop(1, "white");

// relleno del gradiante
ctx4.fillStyle = grd;
ctx4.fillRect(10, 10, 190, 80);
```

El resultado sería:



grd=createLinearGradient(x1, y1, x2, y2) define un **degradado lineal** que comienza en (x1, y1) y termina en (x2, y2)

addColorStop(posición, color): define puntos de transición de color dentro del degradado.

fillStyle = grd: establece el degradado como color de relleno.

fillRect(10, 10, 190, 80); dibuja un rectángulo en (10,10) con ancho 190 y alto 80, rellenado con el degradado.

Ahora vamos a hacer el mismo gradiente, pero radial, en grados:

Para ello, creamos otro canvas:

```
dy>
  <h2>EJEMPLO CANVAS</h2>
  <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;"></canvas>
  <canvas id="myCanvas2" width="200" height="100" style="border:1px solid #000000;"></canvas>
  <br>
  <canvas id="myCanvas3" width="600" height="300" style="border:1px solid #000000;"></canvas>
  <br>
  <canvas id="myCanvas4" width="200" height="100" style="border:1px solid #000000;"></canvas>
  <br>
  <canvas id="myCanvas5" width="200" height="100" style="border:1px solid #000000;"></canvas>
  <script>

    //Recogemos el elemento canvas por su ID:
```

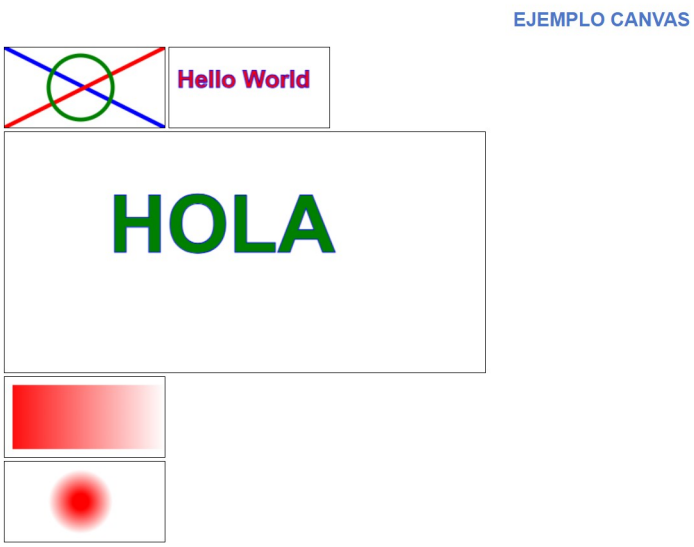
Al final de nuestro código javascript:

```
// gradientes: radial
var c5 = document.getElementById("myCanvas5");
var ctx5 = c5.getContext("2d");

// Creación gradiente:
// Gradiente Radial:
var grd = ctx5.createRadialGradient(95, 50, 10, 95, 50, 40);

grd.addColorStop(0, "red");
grd.addColorStop(1, "white");

// relleno del gradiente
ctx5.fillStyle = grd;
ctx5.fillRect(10, 10, 190, 80);
```

createRadialGradient(x1, y1, r1, x2, y2, r2) define un degradado **circular** entre dos círculos:

- (x1, y1, r1): Centro y radio del **círculo interno**.
- (x2, y2, r2): Centro y radio del **círculo externo**

El resultado sería:



Como ejercicio final el alumno deberá crear en un canvas la bandera de España, debería salir así:

El código sería el siguiente:

Y el script:

```
//Ejemplo de la bandera de España:
var canvas = document.getElementById("myCanvasBandera");
var ctxb = canvas.getContext("2d");

// Colores de la bandera
var rojo = "#AA151B";
var amarillo = "#F1BF00";

// Dibujar la franja superior (roja)
ctxb.fillStyle = rojo;
ctxb.fillRect(0, 0, 300, 60);

// Dibujar la franja central (amarilla)
ctxb.fillStyle = amarillo;
ctxb.fillRect(0, 60, 300, 80);

// Dibujar la franja inferior (roja)
ctxb.fillStyle = rojo;
ctxb.fillRect(0, 140, 300, 60);
```

Con esto, tenemos las nociones básicas de uso de los canvas para dibujar en javascript.