U T 3 : IMPLANTACIÓN DE CONTENIDO MULTIMEDIA

GRÁFICOS

DIW – FRANCISCO CORBERA NAVAS

GRÁFICOS

Introducción

- En HTML5, dos de las formas más utilizadas para representar gráficos son mediante las etiquetas **svg** y **canvas**.
- Ambos tienen ventajas y desventajas, se elegirá uno u otro en función del uso que se le vaya a dar en la web.

GRÁFICOS

Característica	SVG	Canvas
Modo de renderización	Basado en gráficos vectoriales	Basado en píxeles
Escalado	Mantiene la calidad del gráfico, sin distorsión	El escalado puede resultar en pixelación
Manejo de eventos	Soporta eventos DOM (clics, desplazamientos, etc.)	No soporta directamente, requiere cálculos manuales
Animación	Animación con CSS y JavaScript	Actualización manual mediante JavaScript
Tamaño de archivo	Generalmente más pequeño	Depende de la complejidad de la imagen, puede ser más grande
Casos de uso	Gráficos, mapas, iconos	Juegos, visualización de datos, procesamiento de imágenes

GRÁFICOS

SVG

- Scalable Vector Graphics (svg) es un formato de imagen que puede dibujar gráficos 2D en un navegador web.
- Basado en XML, soporta algunas características interesantes como las interacciones, transiciones y animaciones.
- Es un estándar creado por el W3C.

GRÁFICOS

SVG

- Introduce una variedad de formas que pueden ser utilizadas en un amplio abanico de escenarios.
- Esas formas pueden ir combinándose para crear diseños más complejos.

```
<svg height="200" width="400">
    <rect x="100" y="50" rx="20" ry="20" width="250" height="100" fill="#1B043C" />
    <rect x="100" y="50" width="200" height="100" fill="#1B043C" />
    <circle cx="100" cy="100" r="50" fill="#472772" />
    <text fill="#D7BFF3" font-size="28" font-family="Segoe UI Light" x="160"
    y="108">CompanyName</text>
</svg>
```

GRÁFICOS

SVG - Círculo

- Para dibujar el círculo se define un punto central y el radio.

.

Attribute	Description
cx & cy	These two attributes together define the coordinates for the center of the circle. By default, the center of the circle is *(0,0)*
r	This attribute specifies the radius of the circle
fill	This attribute defines the color used for the interior of the circle
stroke	This attribute defines the color used for the border of the circle
stroke-width	This attribute defines the width of the border of the circle

GRÁFICOS

SVG - Rectángulo

- Se definen cuatro lados a partir de un punto (esquina superior-izquierda), un ancho y un alto.

Attribute	Description
x & y	These two attributes together define the coordinates for the top-left of the rectangle. By default, the top-left of the rectangle is *(0,0)*
rx & ry	These two attributes specify the radius for rounded corners on the x or y axis
width	This attribute defines the width of the rectangle
height	This attribute defines the height of the rectangle
fill	This attribute defines the color used for the interior of the rectangle
stroke	This attribute defines the color used for the border of the rectangle
stroke-width	This attribute defines the width of the border of the rectangle

GRÁFICOS

SVG - Texto

- SVG define texto a partir de situar un punto en la esquina inferior-izquierda

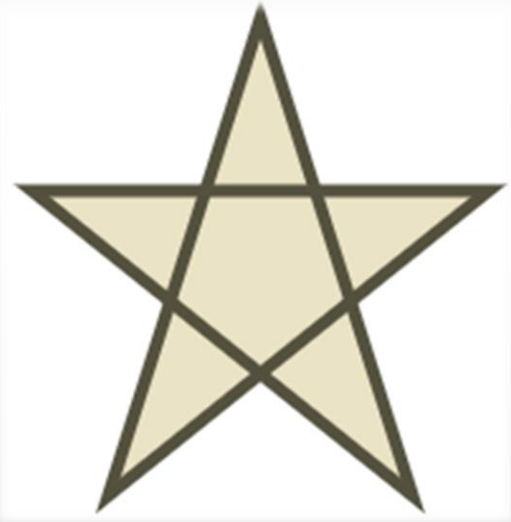
Attribute	Description
x & y	These two attributes together define the coordinates for the bottom-left of the text.
fill	This attribute defines the color used for the text
font-size	This attribute defines the size of the text font
font-family	This attribute defines the text font

GRÁFICOS

SVG - Polígonos

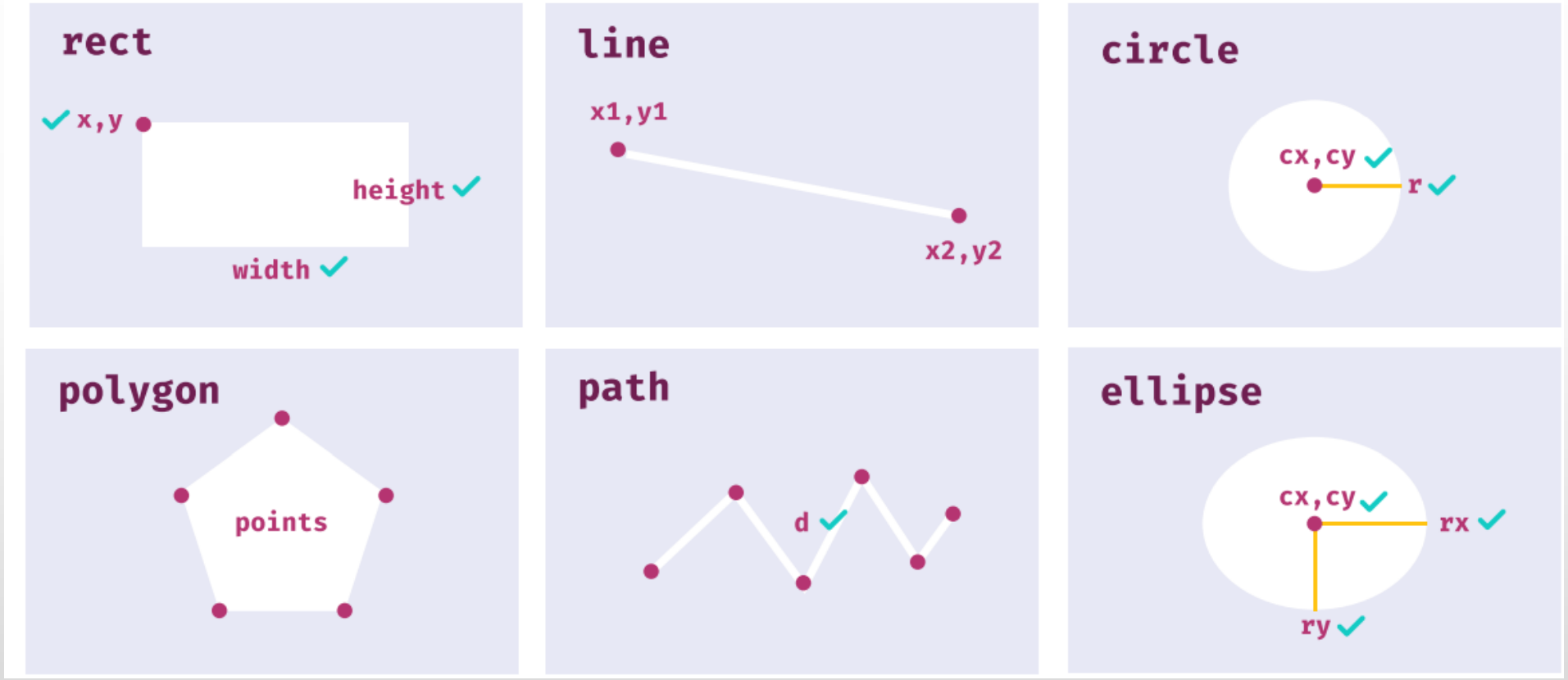
- SVG define polígonos a partir de coordenadas en un plano 2D

```
1  <svg width="300" height="300">
2      <polygon
3          points="100,10 40,198 190,78 10,78 160,198"
4          stroke="#54523F"
5          stroke-width="5"
6          fill="#EBE3C5"
7      />
8  </svg>
```



GRÁFICOS

SVG



GRÁFICOS

SVG - Composiciones

GRÁFICOS

SVG - Intenta

GRÁFICOS

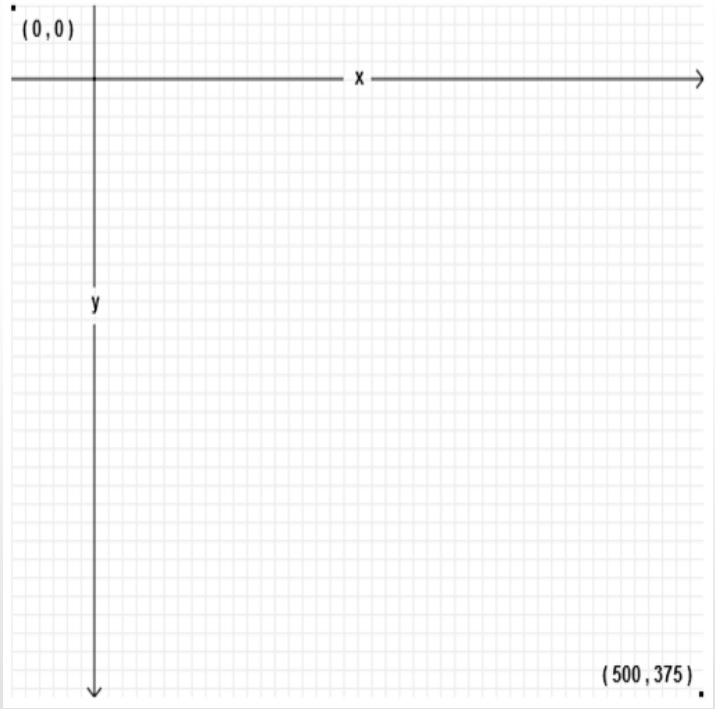
CANVAS

- Canvas (o "lienzo" traducido al español) es un elemento incorporado en HTML5 que permite la generación de gráficos dinámicamente por medio del scripting.
- El API de canvas soporta diferentes formas: líneas, rectángulos, elipses, arcos, curvas, texto, imágenes, etc..
- Soporta tanto diseños en 2D como en 3D (usando el API WebGL).
- Dispone de un canal alpha para añadir transparencia.
- Uso extendido en animaciones, efectos especiales o juegos, ya que permite animaciones fluidas y control de fotogramas.

GRÁFICOS

CANVAS

- El sistema de coordenadas usado para dibujar canvas es similar al de otras APIs de dibujo 2D.
- El punto (0,0) está situado en la esquina superior-izquierda del lienzo.



GRÁFICOS

CANVAS

- Para añadir un elemento <canvas> a la página web hay que definir el espacio que va ocupar, estableciendo una altura y anchura.

```
<canvas id="myCanvas" width=200 height=200>
    Tu browser no soporta canvas
</canvas>
```

- Por si solo no tendrá ningún efecto visual en la página, ya que por defecto el lienzo es transparente.
- Una buena práctica es mostrar un mensaje en caso que canvas no sea soportado por el navegador.

GRÁFICOS

CANVAS

- Para empezar trabajar con <canvas> lo haremos desde JavaScript, seleccionando primero el elemento.

```
var canvas = document.querySelector('#myCanvas');
var canvas = document.getElementById("myCanvas");
```

- Una vez seleccionado, obtendremos el contexto asociado (2D o 3D) para poder definir las propiedades del dibujo.

```
var ctx = canvas.getContext("2d");
```

GRÁFICOS

CANVAS - Ejemplo

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Mi primer canvas</title>
        <script src="reglas.js"></script>
        <style>
            #myCanvas{
                border: 1px solid black;
            }
        </style>
    </head>
    <body onload="init();">
        <canvas id="myCanvas" width="200" height="200">
                Tu navegador no soporta canvas
        </canvas>
    </body>
</html>
```

```javascript
var canvas, ctx;

function init() {
    // 1 - Se obtiene el canvas
    //canvas = document.getElementById('myCanvas');
    canvas = document.querySelector('#myCanvas');

    // 2 - Se obtiene el contexto
    ctx = canvas.getContext('2d');

    // 3 - dibujar
    dibujar();
}

function dibujar() {
    // dibujar un rectangulo rojo
    ctx.fillStyle='#FF0000';
    ctx.fillRect(10,10,180,180);
}
```

GRÁFICOS

CANVAS – Propiedades y métodos

- **fillStyle**
  - Define un color, una textura o un gradiente para el relleno de las formas.
  - Por defecto es negro.

- **fillRect (x, y, width, height)**
  - Dibuja un rectángulo relleno.
  - El relleno lo determina la última propiedad fillStyle definida.

- **clearRect (x, y, width, height)**
  - Borra una sección rectangular aplicando un color transparente.

GRÁFICOS

CANVAS – Propiedades y métodos

- **strokeStyle**
  - Define un color, una textura o un gradiente para el borde de las formas.
  - El color por defecto es el negro.

- **lineWidth**
  - Define el grosor del borde de las formas.

- **strokeRect (x, y, width, height)**
  - Dibuja el borde de un rectángulo.
  - El relleno del borde viene determinado por las propiedades strokeStyle y lineWidth.

GRÁFICOS

CANVAS – Propiedades y métodos

- **font**
  - Define la fuente y varias de sus propiedades de una sola vez (font-family, font-size, font-weight, etc..) .

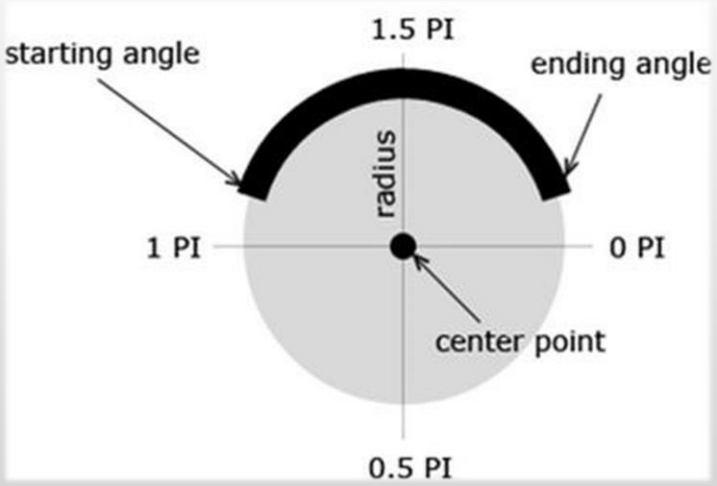
- **textAlign**
  - Sirve para alinear horizontalmente el texto (left | right | center | justify).

- **fillText (texto, x, y, [maxWidth])**
  - Define el texto y su posición en el lienzo.
  - El ancho máximo que ocupará el texto es un parámetro opcional.

GRÁFICOS

CANVAS – Propiedades y métodos

- **arc (x, y, radio, angIni, angFin, sentido)**
  - Dibuja un arco o circulo.
  - x,y define el centro del círculo. Los ángulos se definen en radianes.
  - No se visualizará hasta llamar a fill() o stroke().



GRÁFICOS

CANVAS – Propiedades y métodos

Path (camino): serie de puntos y líneas que forman una figura.

- **beginPath()**
  - Crea un nuevo trazado.
- **moveTo(x,y)**
  - Sitúa el punto de partida donde queremos iniciar el trazo.
- **lineTo(x,y)**
  - Dibuja una línea desde la posición actual hasta la posición especificada por x e y.
- **closePath(x,y)**
  - Añade una línea recta que va al inicio del trazado actual para cerrarlo.
  - Llamando a fill() cualquier forma abierta se cierra automáticamente, para dibujar solo el contorno si es necesario cerrar el path y llamar a stroke())

GRÁFICOS

CANVAS - Imágenes

- Una de las cosas más interesantes que podremos hacer cuando dibujamos en el lienzo del elemento canvas es importar y mostrar directamente el contenido de archivos gráficos externos, por ejemplo usando imágenes GIF, JPG o PNG dentro de los dibujos que realizamos con canvas.
- Para dibujar una imagen en el lienzo se utiliza el método **drawImage().**

```
contexto.drawImage(objeto_imagen, x, y);
```

- El primer parámetro es el objeto Javascript de la imagen que se desea incluir en el lienzo. Los dos siguientes son las coordenadas donde situar la imagen, siendo (x , y) el punto donde se colocará la esquina superior izquierda de la imagen.

GRÁFICOS

CANVAS - Imágenes

- El objeto imagen es uno de los objetos básicos de Javascript, pudiendo generarlo dinámicamente a través de su constructor:

```
var img = new Image();
img.src = "ruta_imagen";
```

- Hay que asignarle la URL de la imagen al atributo **src** , haciendo que en el objeto se cargue la imagen.
- Para asegurarnos que drawImage() funciona podemos usar el evento **onload** de la imagen, así solo la dibujará cuando el recurso haya terminado de descargarse.

GRÁFICOS

CANVAS - Imágenes

- Existen múltiples opciones de obtener el objeto Image:
  - Crear el propio objeto Image (visto anteriormente).
  - Acceder a una imagen que hay en la propia página.
  - A través del array de imágenes del documento.
  - Mostrar en un Canvas
    el contenido de otro Canvas.

GRÁFICOS

CANVAS - Imágenes

- Redimensionar una imagen es sencillo. Simplemente tenemos que invocar al método **drawImage()** enviando además las dimensiones de la imagen que queremos que se dibuje.
- El navegador escalará la imagen para que tenga las dimensiones que indiquemos y luego la pintará en el canvas.

```
drawImage(objeto_imagen, posX, posY, anchura, altura);
```

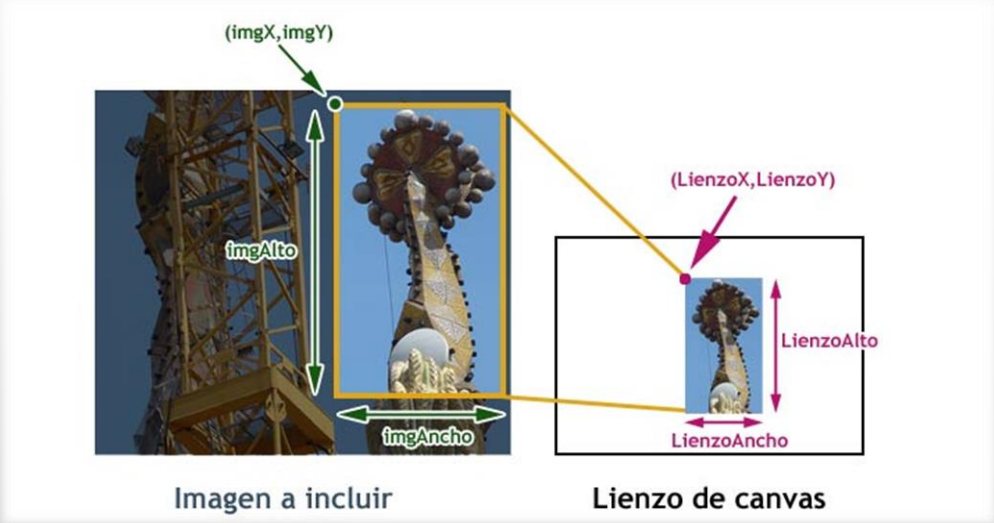
- Las dimensiones pueden no ser proporcionales, el navegador estirará o achatará la imagen para adaptarla a la anchura y altura que hayamos indicado.

GRÁFICOS

CANVAS - Imágenes

- Si además de escalar queremos recortar una imagen, el modo de invocar a **drawImage()** es un poco más complejo. La llamada tendrá los siguientes parámetros:

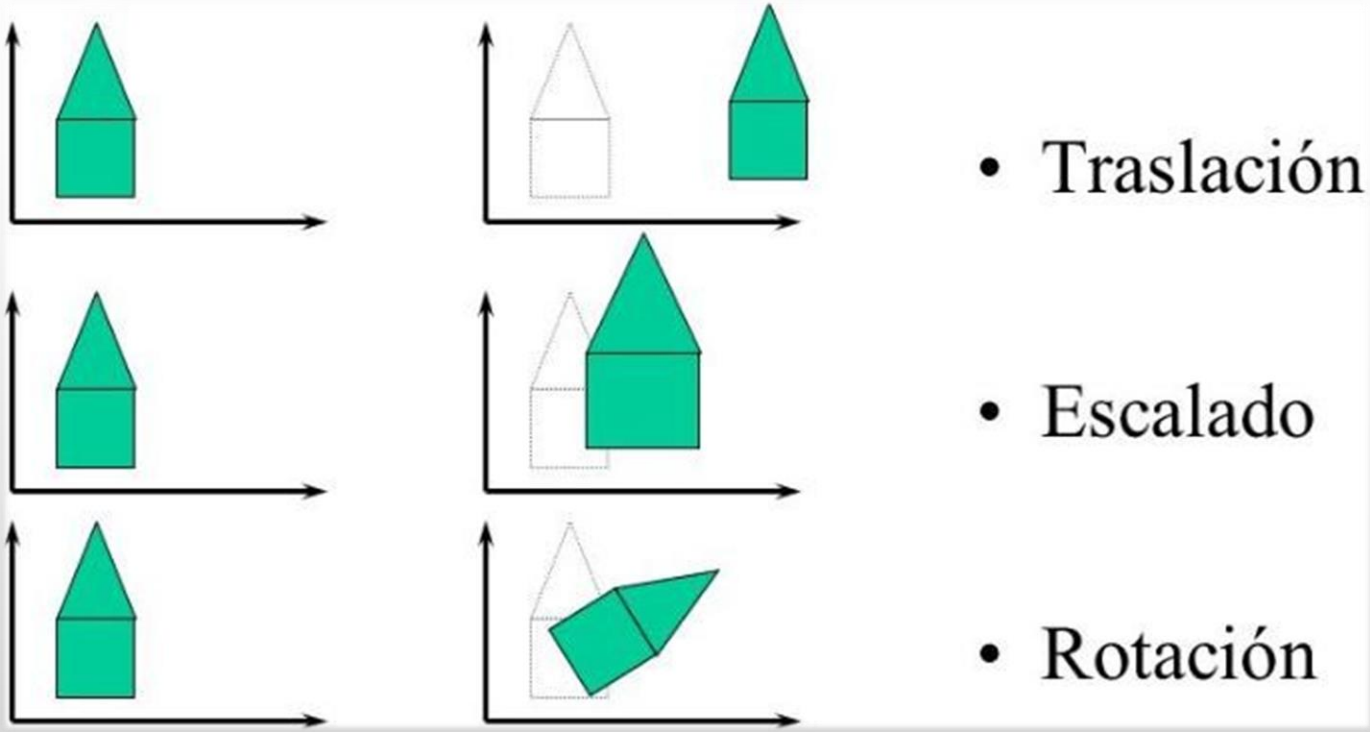
```
drawImage(objeto_imagen, imgX, imgY, imgAncho, imgAlto, lienzoX, lienzoY, LienzoAncho, LienzoAlto)
```



GRÁFICOS

CANVAS - Transformaciones

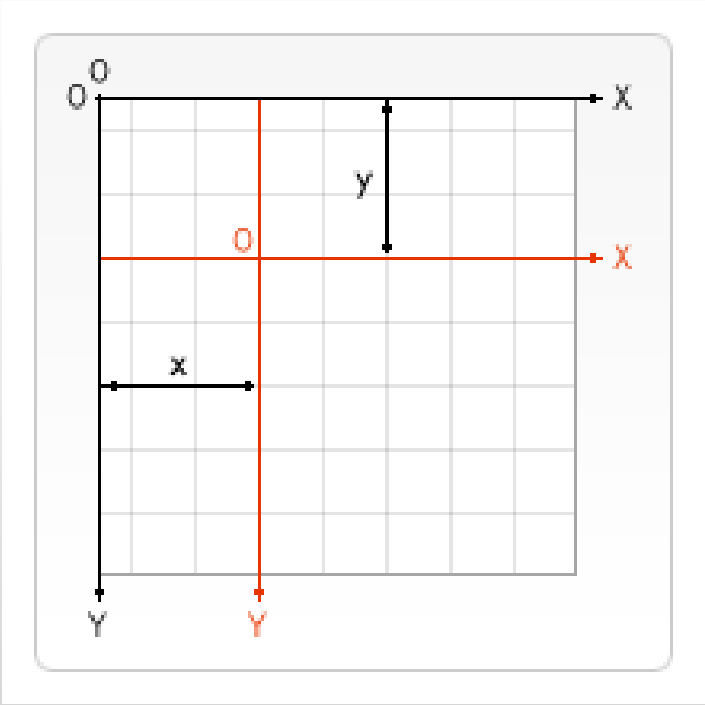
- Utilizando Canvas se pueden aplicar transformaciones a las formas:



GRÁFICOS

CANVAS – Transformaciones

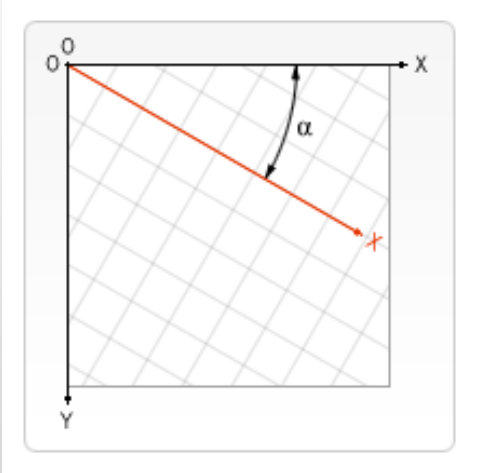
- **ctx.translate(x, y)**
  - Se especifica la coordenada a la que se tiene que trasladar la forma, modificando el origen de coordenadas del canvas.
  - Si se aplica una transformación translate(4, 5) la referencia se moverá 4px a la derecha y 5px hacia abajo.
    Por tanto si dibujas un punto de coordenadas (7, 9), aparecerá a 11px (7 del punto +4 del desplazamiento) del lado izquierdo del canvas y a 14px (9 del punto + 5 del desplazamiento) del lado superior.
  - Las traslaciones se acumulan.



GRÁFICOS

CANVAS – Transformaciones

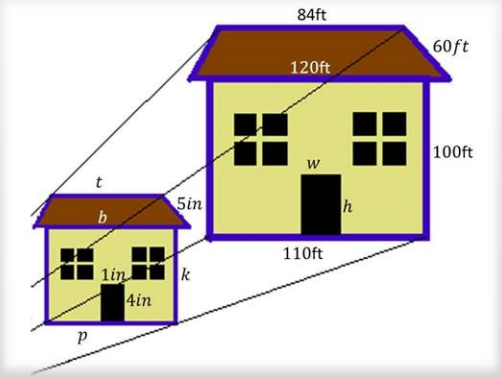
- **ctx.rotate(angle)**
  - Sirve para rotar una forma especificando un ángulo de rotación. El centro de giro es el origen de coordenadas.
  - El ángulo de rotación se expresa en radianes. Si usas un argumento negativo el giro se hace hacia la izquierda.
  - Las rotaciones también se acumulan.
  - Usaremos la fórmula
    grados * Math.PI / 180
    para calcular un radián a partir de un grado.



GRÁFICOS

CANVAS – Transformaciones

- **ctx.scale(ex, ey)**
  - De forma predeterminada, una unidad en el lienzo es exactamente un píxel. Una transformación de escala modifica este comportamiento.
  - Este cambio afecta por separado al ancho y al alto, con lo cual se puede deformar el dibujo y, por ejemplo, convertir una circunferencia en una elipse.
  - Puedes utilizar scale(-1, 1) para voltear el contexto horizontalmente y scale(1, -1) verticalmente.



GRÁFICOS

CANVAS – Transformaciones

- **ctx.tansform(ex, sy, sx, ey, tx, ty)**
  - ex, ey : son factores de escala, igual que se definen en scale(ex,ey).
  - sx, sy: deforman la cuadrícula del canvas en horizontal (sx) y en vertical, sy. Aplicados a un rectángulo sx inclinaría los lados verticales y sy inclinaría los lados superior e inferior. Funcionan como un desplazamiento.
  - tx, ty: trasladan el origen en horizontal y vertical, de forma similar al método translate(tx, ty).
  - Es un método acumulativo. Si queremos que no se acumule las transformaciones usaremos **setTransform(),** que resetea a los valores por defecto (igual que hace el método resetTransform() ex: 1, sy: 0, sx: 0, ey: 1 , tx: 0, ty: 0) y luego aplica transform() con los valores que se le pasan como argumento.

GRÁFICOS

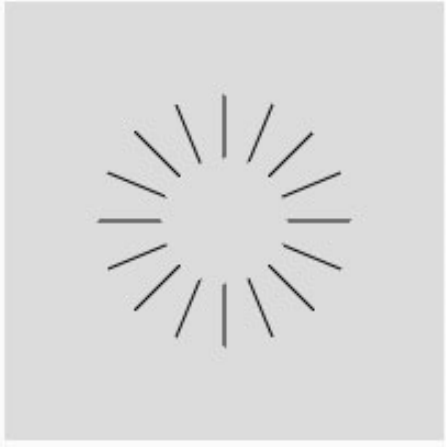
CANVAS – Estado

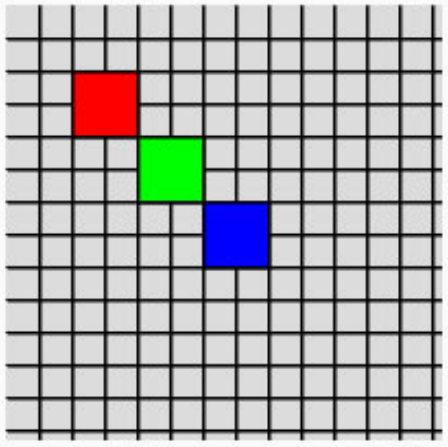
- Es una práctica recomendada utilizar el guardado y restauración del contexto al dibujar formas.
- Esto es posible gracias a los métodos **save()**, que guarda el estado actual del contexto en una pila interna, y **restore()**, que restaura el estado más reciente que se haya guardado con save().

```
ctx.fillStyle = "green";
ctx.fillRect(10, 10, 50, 50);

// Save the state
ctx.save();

// Draw new
ctx.fillStyle = "red";
ctx.fillRect(100, 10, 50, 50);

// Restore saved state
ctx.restore();

// Draw new
ctx.fillRect(200, 10, 50, 50);
```

GRÁFICOS

CANVAS – Practica





GRÁFICOS

CANVAS – Fabric.js

- Fabric.js es una biblioteca de JavaScript para trabajar con el elemento canvas.
- Potencia las funcionalidades que ya tiene la etiqueta canvas, ayudando a administrar los elementos que colocas dentro como objetos.
- Puedes modificar sus atributos e interactuar por medio de vértices, entre muchas otras opciones que simplifican este complejo proceso y mejoran la experiencia del usuario.

GRÁFICOS

CANVAS – Fabric.js

- Comenzaremos agregando la biblioteca a nuestro proyecto a través de un CDN (red de distribución de contenido).

```
<script src="https://cdn.jsdelivr.net/npm/fabric@latest/dist/index.min.js"></script>
```

- Crearemos un objeto canvas instanciando la clase fabric.Canvas, a la que habría que pasarle el id que tiene el elemento canvas en el documento o generarlo directamente.
- Dibujaremos la forma que queramos incorporar y ya solamente quedará agregarla al lienzo a través del método add().

GRÁFICOS

CANVAS – Fabric.js – agregar imágenes

- Vamos a permitir que el usuario agregue una imagen, creando un input para tal fin (*type=file*).
- Asociaremos una función al evento *onchange* del input, para cuando se seleccione un archivo la función se ejecute automáticamente.
- Usamos la función *URL.createObjectURL()* para representar el archivo seleccionado.
- Creamos el nodo imagen para cargar la visualización, usando el objeto Image(), al que asociaremos la URL generada anteriormente.
- Al igual que sucedía con el método drawImage(), usaremos el evento onload de la imagen para asegurarnos que el recurso haya terminado de descargarse.
- Ya solo quedaría generar una instancia con fabric.Image y agregarla a nuestro canvas.

GRÁFICOS

CANVAS – Fabric.js – generar imágenes

- Podemos incorporar una opción que permita descargar la imagen modificada en el lienzo.
- Agregaremos un botón para que al pulsarlo descargue el archivo.
- Obtendremos el contenido del lienzo a través del método toDataURL(), donde especificaremos el formato de la imagen generada (por defecto es PNG).
- Se crea dinámicamente un enlace <a> que se usará para descargar la imagen.
- Se puede configurar la calidad de la imagen descargada (toDataURL("image/jpeg", 0.5);) o el nombre del archivo descargado (a.download = "imagen_personalizada.png).

GRÁFICOS

CANVAS – Fabric.js – frameworks

GRÁFICOS

CANVAS - Animaciones

GRÁFICOS

CANVAS - Animaciones

- Cualquier animación en canvas seguiría los siguientes pasos:
  1. Limpiar el contexto de canvas.

     Se puede usar el método **ctx.clearRect(0, 0, canvas.width, canvas.height).**
  2. Dibujar las formas de la animación.
  3. Aplicar transformaciones (traslación, rotación, escalado), cambiar el color, bordes, etc..
  4. Volver al primer paso.

GRÁFICOS

CANVAS - Animaciones

GRÁFICOS

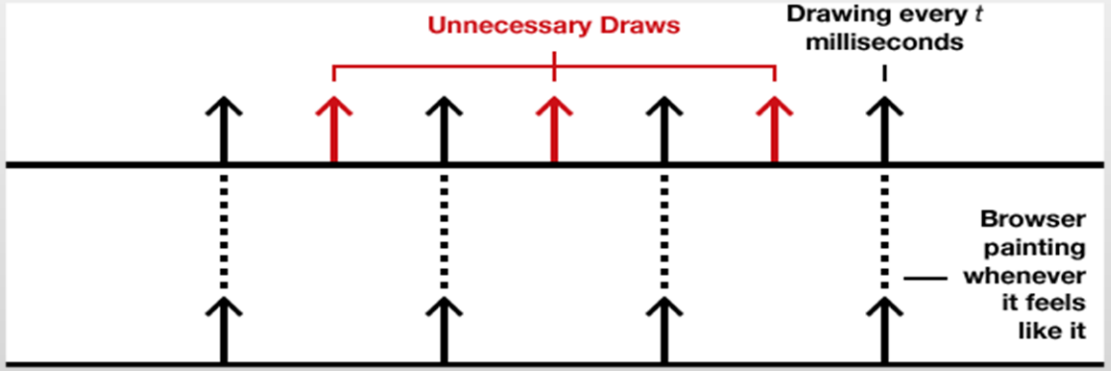
CANVAS - Animaciones

- Esta técnica es como crear dibujos animados utilizando una hoja que hay que borrar en cada fotograma.
- En JavaScript se utiliza el método **setInterval(función, ms)** para ejecutar una función o un fragmento de código repetidamente con un intervalo fijo de tiempo entre cada ejecución (especificado en milisegundos).
- El intervalo incluye el tiempo que tarda la función en ejecutarse. Si la ejecución toma más tiempo que el intervalo especificado, las llamadas no se solaparán, dando lugar a situaciones impredecibles.
- La ejecución continuará indefinidamente hasta que se detenga explícitamente con **clearInterval(id).**
- Si solo quiero ejecutar una función una sola vez después de un intervalo de tiempo específico puede usarse la función **setTimeout(función, delay).**

GRÁFICOS

CANVAS - Animaciones

- **requestAnimationFrame()** es el nuevo estándar que reemplazará a ambos.
- Permite que el navegador regule las animaciones en función de los recursos disponibles, ejecutándolas de manera más eficiente y fluida. Por lo general, 60 FPS, pero varia según el dispositivo y el entorno.
- Cuando un usuario sale del marco, la animación puede ralentizar o detenerse por completo para evitar el uso de recursos innecesarios.



GRÁFICOS

CANVAS - Animaciones

- Para iniciar una animación con **requestAnimationFrame** primero debemos definir la función que queremos que se repita en cada frame de la animación.
- Aquí se muestra un ejemplo sencillo de cómo hacer que un elemento se mueva de izquierda a derecha en la pantalla:

```
let elemento = document.getElementById('miElemento');
let posX = 0;

function animar() {
  posX += 1;  // Incrementa la posición en 1px
  elemento.style.left = `${posX}px`;  // Actualiza la posición del elemento

  if (posX < 500) {  // Condición de parada
    requestAnimationFrame(animar);  // Repite la animación
  }
}

requestAnimationFrame(animar);
```