Materia: Web Security.
Tema: Introducción a **`canvas`**.

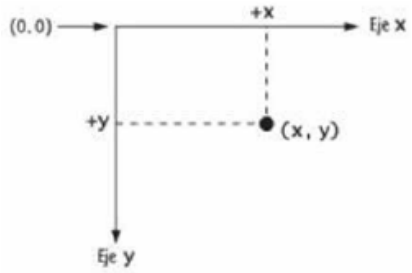
PARTE I.

1. Introducción

- El elemento `canvas` proporciona los métodos a una interfaz de programación de aplicaciones (API) de JavaScript para dibujar gráficos de mapas de bits y animaciones bidimensionales, manipular fuentes e imágenes, y también insertar imágenes y videos.
- El elemento `canvas` es soportado por todos los navegadores.
- Un beneficio clave de `canvas` es que está integrado al navegador, con lo que se elimina la necesidad de complementos como Flash y Silverlight, con lo cual se mejoran el rendimiento y la conveniencia, además de reducir costos. Al final del capítulo se crea el Juego del cañón, que antes se creaba en Flash.

2. Sistema de coordenadas de `canvas`

- El sistema de coordenados de `canvas` es un esquema para identificar rodos los puntos en un elemento `canvas`.
- De manera predeterminada, la esquina superior izquierda de un `canvas` tiene las coordenadas (0,0).
- Un par de coordenadas tiene una coordenada `x` (la coordenada horizontal) y una coordenada `y` (la coordenada vertical.
- La coordenada `x` es la distancia horizontal hacia la derecha desde el borde izquierdo de un `canvas`.
- La coordenada `y` es la distancia vertical hacia abajo desde el borde superior de un `canvas`.
- El eje `x` define cada coordenada horizontal y el eje `y` define cada coordenada vertical.
- Para posicionar el texto y las figuras en un `canvas` se especifican sus coordenadas `x` y `y`.
- Las unidades de espacio de coordenadas se miden en píxeles ("elementos de imagen"), que son las unidades más pequeñas de resolución en una pantalla.



Coordenadas del `canvas`, en pixeles.

3. Rectángulos

- Un `canvas` es un área rectangular en la que podemos dibujar.
- El elemento `canvas` tiene dos atributos: `width` y `hight`. El valor predeterminado de `width` es `300` y el valor predeterminado de `height` es `150`.
- El atributo `fillStyle` especifica el color del rectángulo.
- Para especificar las coordenadas del rectángulo, usamos `fillRect` en el formato (`x, y, w, h`), en donde `x` y `y` son las coordenadas de la esquina superior izquierda del rectángulo, `w` es la anchura del rectángulo y `h` es la altura.
- El atributo `strokeStyle` especifica el color del trazo y `lineWidth` especifica la anchura de línea.
- El método `strokeRect` especifica la ruta del trazo en el formato (`x, y, w, h`).
- Si `width` y `height` son `0`, no aparecerá el trazo. Si la anchura o la altura es `0`, el resultado será una línea y no un rectángulo.

EJEMPLO 1.
Archivo dibujarrectangulo.html:

```
<!-- Dibujo de un rectángulo en un canvas. -->
```

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Dibujar un rect&aacute;ngulo</title>
	</head>
	<body>
		<canvas id = "dibujarRectangulo" width = "300" height = "100"
			style = "border: 1px solid black;">
			Su navegador no soporta el elemento canvas.
		</canvas>
		<script>
			var canvas = document.getElementById("dibujarRectangulo");
			var contexto = canvas.getContext("2d");
			contexto.fillStyle = "yellow";
			contexto. fillRect( 5, 10, 200, 75 );
			contexto.strokeStyle = "royalblue";
			contexto.lineWidth = 6;
			contexto.strokeRect( 4, 9, 201, 76);
		</script>
	</body>
</html>
```

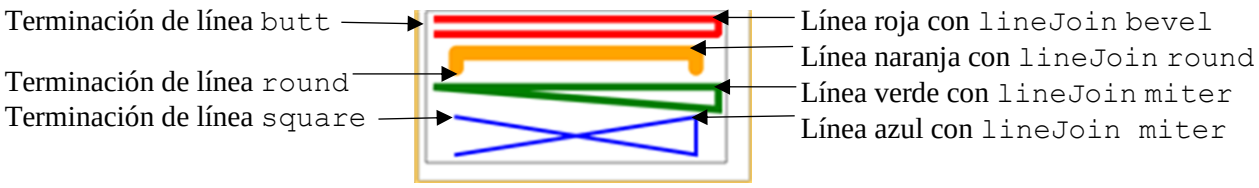
4. Uso de rutas para dibujar líneas

- El método `beginPath` comienza la ruta.
- El método `moveTo` establece bs coordenadas `x` y `y` del origen de la ruta.
- Desde el punto de origen, usamos el método `lineTo` para especificar los destinos de la ruta.
- El atributo `lineWidth` se utiliza para modificar el grosor de la línea. El valor predeterminado de `lineWidth` es `1.0`.
- El atributo `lineJoin` especifica el estilo de las esquinas en donde se unen dos líneas, tiene tres valores posibles: `bevel`, `round` y `miter`.
- El valor `bevel` de `lineJoin` proporciona esquinas inclinadas a la ruta.
- El atributo `lineCap` define el estilo de las terminaciones de línea. Existen tres posibles valores: `butt`, `round` y `square`.
- El valor `butt` de `lineCap` especifica que las terminaciones de las líneas tienen bordes perpendiculares a la dirección de la línea y sin terminación adicional.
- El atributo `strokeStyle` especifica el color de línea.
- El método `stroke` dibuja líneas en un `canvas`. El color de trazo predeterminado es `black`.
- El valor round de `lineJoin` crea esquinas redondeadas. Después, el valor `round` de `lineCap` agrega una terminación semicircular a los extremos de la ruta. El diámetro de la terminación agregada es igual a la anchura de la línea.
- El método `closePath` cierra la ruta al dibujar una línea desde el último destino especificado de regreso al punto del origen de la ruta.
- El valor `miter` de `lineJoin` bisela las líneas a un ángulo en donde se unen. Por ejemplo, las líneas que se unen a un ángulo de 90 grados tienen bordes biselados con ángulos de 45 grados en donde se unen.
- El valor `square` de `lineCap` agrega una terminación rectangular a los extremos de una línea. La longitud de la terminación es igual a la anchura de la línea, y la anchura de la terminación es igual a la mitad de la anchura de la línea. El borde de una `lineCap` tipo `square` es perpendicular a la dirección de la línea.

EJEMPLO 2.
Archivo lineas.html:

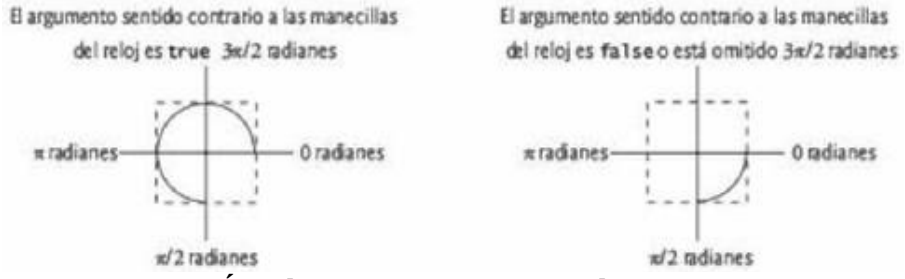
```
<!-- Dibujo de lineas en un canvas. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Dibujar l&iacute;neas</title>
	</head>
	<body>
		<canvas id = "dibujarLineas" width = "400" height = "200"
			style = "border: 1px solid black;">
		</canvas>
		<script>
			var canvas = document.getElementById("dibujarLineas");
			var contexto = canvas.getContext("2d");
			// lineas rojas sin una ruta cerrada
			contexto.beginPath(); // comenzar una nueva ruta
			contexto.moveTo(10, 10 ); // origen de la ruta
			contexto.lineTo(390, 10);
			contexto.lineTo(390, 30);
			contexto.lineTo(10, 30);
			contexto.lineWidth = 10; // anchura de la linea
			contexto.lineJoin = "bevel" // estilo de unión de la linea
			contexto.lineCap = "butt"; // estilo de terminación de la linea
			contexto.strokeStyle = "red" // color de la linea
			contexto.stroke(); // dibujar ruta
			// líneas naranjas sin ruta cerrada
			contexto.beginPath(); // comenzar una nueva ruta
			contexto.moveTo(40, 75); // origen de la ruta
			contexto.lineTo(40, 55);
			contexto.lineTo(360, 55);
			contexto.lineTo(360, 75);
			contexto.lineWidth = 20; // ancho de la línea
			contexto.lineJoin = "round" // estilo de unión de la línea
			contexto.lineCap = "round"; // estilo de terminación de la linea
			contexto.strokeStyle = "orange"; // color de la linea
			contexto.stroke(); // dibujar ruta
			// líneas verdes con una ruta cerrada
			contexto.beginPath(); // comenzar una nueva ruta
			contexto.moveTo(10, 100); // origen de la ruta
			contexto.lineTo(390, 100);
			contexto.lineTo(390, 130);
			contexto.closePath(); // cerrar ruta
			contexto.lineWidth = 10; // anchura de la linea
			contexto.lineJoin = "miter"; // estilo de unión de la linea
			contexto.strokeStyle = "green"; // color de la linea
			contexto.stroke(); // dibujar ruta
			// lineas azules sin una ruta cerrada
			contexto.beginPath(); // comienza una nueva ruta
			contexto.moveTo(40, 140); // origen de la ruta
			contexto.lineTo(360, 190);
			contexto.lineTo(360, 140);
			contexto.lineTo(40, 190);
			contexto.lineWidth = 5; // anchura de la linea
			contexto.lineCap = "square"; // estilo de terminación de la línea
```

```
                contexto.strokeStyle = "blue" // color de la linea
                contexto.stroke(); // dibujar ruta
            </script>
      </body>
</html>
```



5. Dibujo de arcos y círculos

- Los arcos son partes de la circunferencia de un círculo. Para dibujar un arco especificamos el ángulo inicial y el ángulo final que se miden en radianes, la relación de la longitud del arco con su radio.
- El método `arc` dibuja el círculo usando cinco argumentos. Los primeros dos argumentos representan las coordenadas `x` y `y` del centro del círculo. El tercer argumento es el radio del círculo. Los argumentos cuarto y quinto son los ángulos, inicial y final, del arco en radianes.
- El sexto argumento es opcional y especifica la dirección en la que se dibuja la ruta del arco. De manera predeterminada el sexto argumento es `false` para indicar que el arco se dibuja en sentido de las manecillas del reloj. Si el argumento es `true`, el arco se dibuja en sentido contrario a las manecillas del reloj.
- La constante `Math.PI` es la representación de JavaScript de la constante matemática π, la relación de la circunferencia de un círculo con su diámetro. 2π radianes representan un arco de 360 grados, π radianes son 180 grados y π/2 radianes son 90 grados.

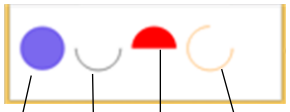


Ángulos positivos y negativos de `arc`.

EJEMPLO 3.
Archivo dibujararcos.html:

```
<!-- Dibujo de arcos y círculos en un canvas. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Arcos y c&iacute;rculos</title>
      </head>
      <body>
            <canvas id = "dibujarArcos" width = "315" height = "100">
            </canvas>
            <script>
                  var canvas = document.getElementById("dibujarArcos");
                  var contexto = canvas.getContext("2d");
                  // dibujar un circulo
                  contexto.beginPath();
                  contexto.arc(35, 50, 30, 0, Math.PI*2);
                  contexto.fillStyle = "mediumslateblue";
```

```
                contexto.fill();
                // dibujar un arco en sentido contrario a las manecillas del reloj
                contexto.beginPath();
                contexto.arc( 110, 50, 30, 0, Math.PI, false);
                contexto.stroke();
                // dibujar medio círculo en sentido de las manecillas del reloj
                contexto.beginPath();
                contexto.arc(185, 50, 30, 0, Math.PI, true);
                contexto.fillStyle = "red";
                contexto.fill();
                // dibujar un arco en sentido contrario a las manecillas del reloj
                contexto.beginPath();
                contexto.arc(260, 50, 30, 0, 3*Math.PI/2);
                contexto.strokeStyle = "darkorange";
                contexto.stroke();
            </script>
        </body>
</html>
```



Círculo azul pizarra mediano

Arco negro contrario al reloj

Arco rojo contrario al reloj

Arco naranja sentido del reloj