

Contenido de este curso

- Conocer Canvas de HTML
- Crear funciones e iteraciones
- Interacción con el usuario (eventos)
- Animaciones
- Juego de Acertar al Blanco

Creemos nuestro tablero de Dibujo canvas en Html.

```
<canvas width="600" height="400"> </canvas>

<script>
  var pantalla = document.querySelector("canvas");
  var pincel = pantalla.getContext("2d");
</script>
```



FillRect

El fillRect() método dibuja un "filled" rectángulo. => pincel.fillRect(x, y, width, height);
El color por defecto del relleno es negro.

FillStyle

Establece o devuelve el color (aplico en "fillRect()")

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

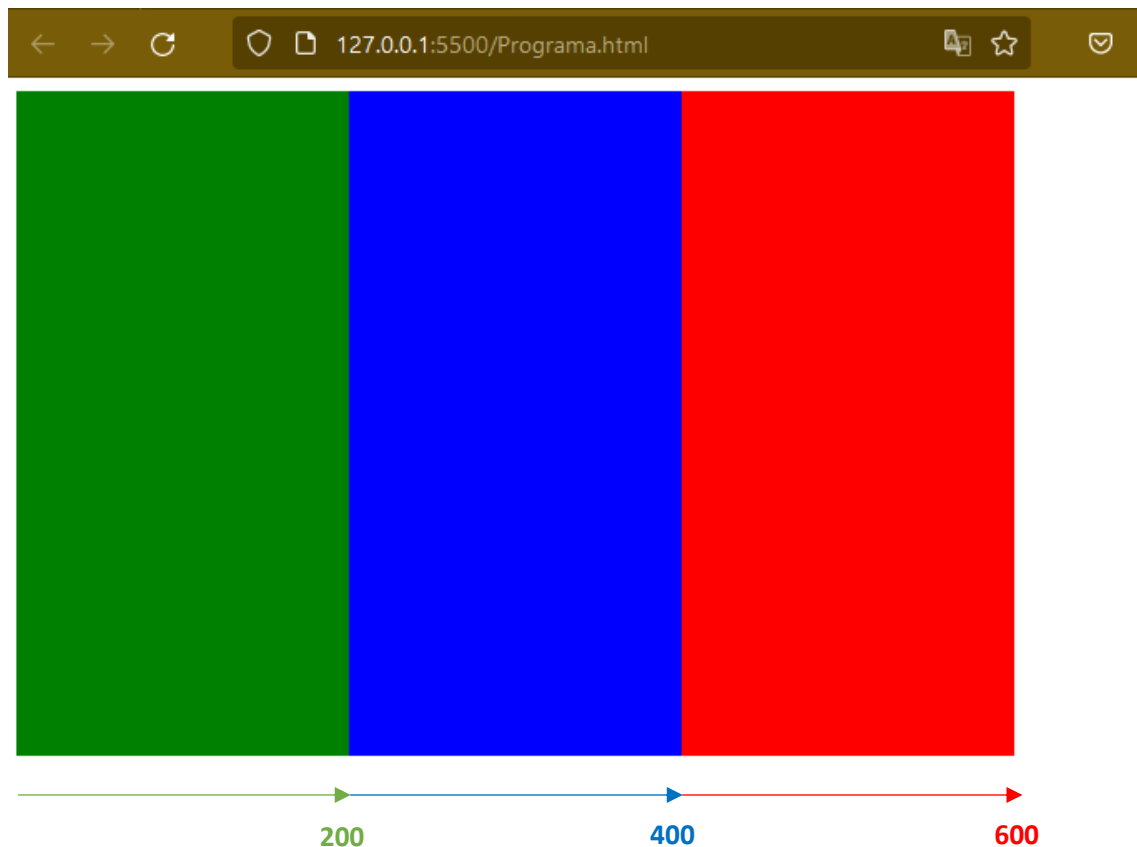
    pincel.fillStyle = "lightgrey"; //propiedad
    pincel.fillRect(0,0, 600,400); // función

    pincel.fillStyle = "green";
    pincel.fillRect(0,0, 200,400);

    pincel.fillStyle = "blue";
    pincel.fillRect(200,0, 200,400);

    pincel.fillStyle = "red";
    pincel.fillRect(400,0, 200,400);

  </script>
</body>
```

De esta manera yo puedo dibujar "pintar" con canvas usando las posiciones a través del plano cartesiano.

beginPath

El método beginPath() comienza una “nueva” ruta o restablece la ruta actual.

Si requiero más información puedo encontrarlo buscando: **2D canvas**

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle = "lightgrey"; //propiedad
    pincel.fillRect(0,0, 600,400); // función

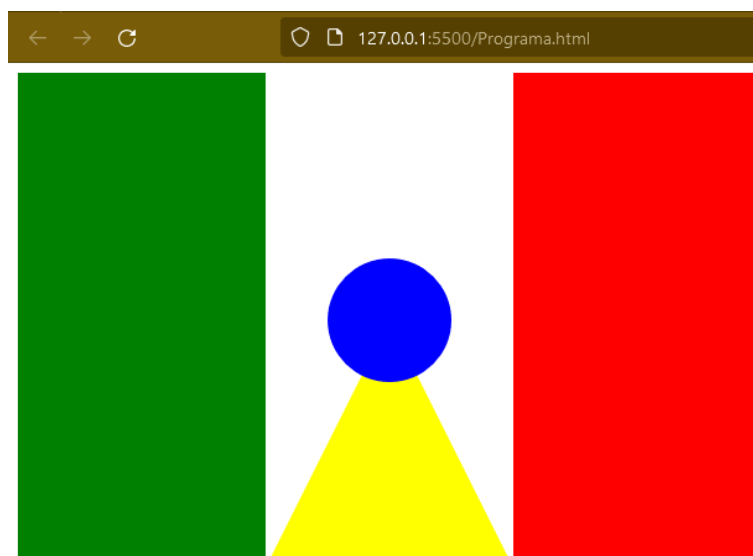
    pincel.fillStyle = "green";
    pincel.fillRect(0,0, 200,400);

    pincel.fillStyle = "white";
    pincel.fillRect(200,0, 200,400);

    pincel.fillStyle = "red";
    pincel.fillRect(400,0, 200,400);

    //Dibujar Triangulo
    pincel.fillStyle = "yellow";
    pincel.beginPath();
    pincel.moveTo(300,200)//muevo mitad de eje X y Y
    pincel.lineTo(200,400);//dibujo linea de:
    pincel.lineTo(400,400);
    pincel.fill();

    //Dibujar Circulo
    pincel.fillStyle = "blue";
    pincel.beginPath();
    //ubico mitad eje X - Y ; Radio (50) ; Angulo Inic (0) ; Ang. Final (2 veces Pi)
    pincel.arc(300,200,50,0, 2*3.14);
    pincel.fill();
  </script>
</body>
```



Nuestro pincel y sus características

Aprendimos a usar dos instrucciones que son características de nuestro objeto `pincel`, las mismas nos ayudaron a formatar nuestras figuras.

Las instrucciones usadas fueron:

```
pincel.fillStyle  
pincel.fillRect()
```

Cuál de las alternativas está usando las 2 instrucciones de forma correcta:

A

```
pincel.fillStyle("green");  
pincel.fillRect(0,0,600,400);
```

**B**

```
pincel.fillStyle = "green";  
pincel.fillRect = (0,0,600,400);
```



```
pincel.fillStyle = "green";  
pincel.fillRect(0,0,600,400);
```

¡Correcto! La instrucción `pincel.fillStyle` es una propiedad del objeto `pincel` y se le está atribuyendo el color verde. Por su parte, la instrucción `pincel.fillRect()` es una función (método) del objeto `pincel` que en nuestro código la usábamos para diseñar (dibujar) las figuras.

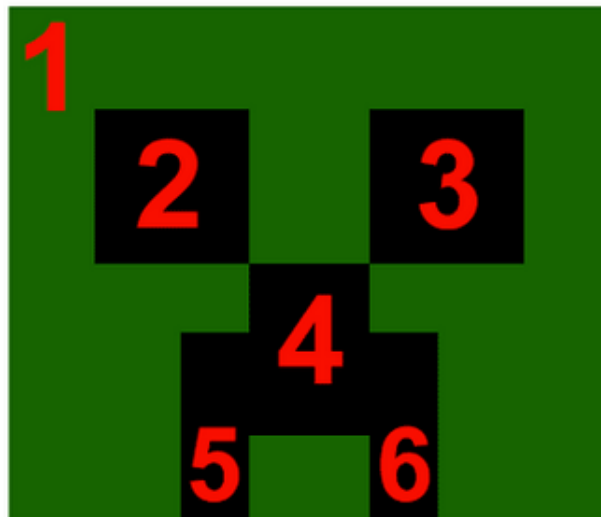
La cara de Creeper

Bernardo es un chico de 11 años y le gusta jugar Minecraft. Minecraft es un juego donde construyes un mundo con bloques, como si fuera un Lego virtual. Uno de los personajes principales del juego es Creeper, y Bernardo pidió un poster con esa caricatura.

Creeper es un monstruo que explota cuando se aproxima a un jugador y tiene más o menos la siguiente figura:



Si analizamos la figura vamos a descubrir que la misma está compuesta de una serie de rectángulos de diferentes colores:



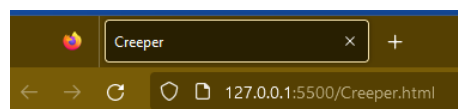
¿Será que consigues ayudar a Bernardo creando el poster con la imagen Creeper? A continuación encontrarás un paso a paso para guiarte mejor:

¡Manos a la obra!

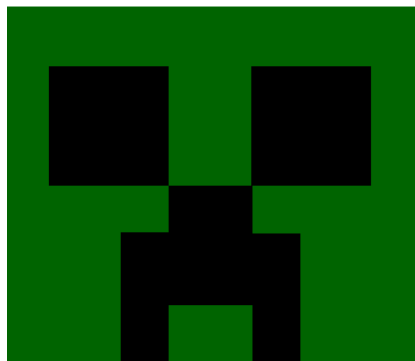
```

<body>
  <h2>CREEPER</h2>
  <canvas width="600" height="400"> </canvas>
  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    // 1: Cabeza
    pincel.fillStyle = "darkgreen";
    pincel.fillRect(0,0, 350,300);
    // 2: Ojos
    pincel.fillStyle = "black";
    pincel.fillRect(35,50, 100,100);
    // 3: Ojos
    pincel.fillStyle = "black";
    pincel.fillRect(204,50, 100,100);
    // 4: nariz
    pincel.fillStyle = "black";
    pincel.fillRect(135,150, 70,100);
    // 5: Parte de la boca
    pincel.fillStyle = "black";
    pincel.fillRect(95,189, 40,110);
    // 6: Parte de la boca
    pincel.fillStyle = "black";
    pincel.fillRect(205,190, 40,110);
  </script>
</body>

```

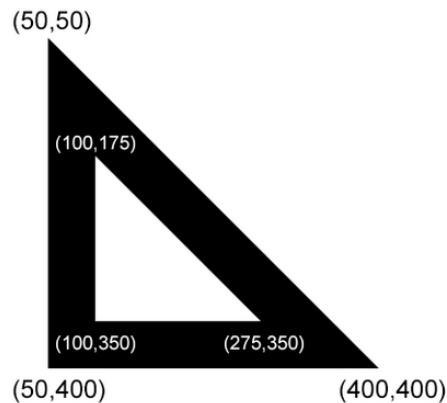


CREEPER



Diseñando una Escuadra

Ya practicamos bastante con rectángulos, es hora de diseñar otra figura. En este nuevo desafío vamos a diseñar una escuadra (plantilla especial en forma de un triángulo isósceles usada por arquitectos e ingenieros).



Si analizamos la figura, podemos percibir que una escuadra no es otra cosa que dos triángulos, uno dentro del otro. Recordando también que diseñar triángulos es diferente de diseñar rectángulos, pues es preciso diseñar línea por línea. En otras palabras podemos decir que la API es diferente.

```
<body>
  <h2>ESCUADRA</h2>
  <canvas width="600" height="400"> </canvas>
  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle = "white"; //propiedad
    pincel.fillRect(0,0, 600,400); // función

    //Dibujar Escuadra exterior
    pincel.fillStyle = "black";
    pincel.beginPath();
    pincel.moveTo(50,50)//muevo mitad de eje X y Y
    pincel.lineTo(50,400);//dibujo línea
    pincel.lineTo(400,400);
    pincel.fill();

    //Dibujar Escuadra interior
    pincel.fillStyle = "white";
    pincel.beginPath();
    pincel.moveTo(100,175)
    pincel.lineTo(100,350);
    pincel.lineTo(275,350);
    pincel.fill();

  </script>
</body>
```

Lo que aprendimos

- A usar el `Canvas` como una pizarra donde realizamos nuestros diseños.
- A diseñar figuras geométricas como rectángulos y circunferencias.
- A personalizar algunas propiedades de nuestras figuras como el color.

ACTIVIDAD

Creamos 5 cuadrados le damos color (`green`) luego a cada contorno del cuadrado le damos un color (`black`).

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle = "green";
    pincel.fillRect(0,0, 50,50);
    pincel.strokeStyle = "black";
    pincel.strokeRect(0,0, 50,50);

    pincel.fillRect(50,0, 50,50);
    pincel.strokeRect(50,0, 50,50);

    pincel.fillRect(100,0, 50,50);
    pincel.strokeRect(100,0, 50,50);

    pincel.fillRect(150,0, 50,50);
    pincel.strokeRect(150,0, 50,50);

    pincel.fillRect(200,0, 50,50);
    pincel.strokeRect(200,0, 50,50);

  </script>
</body>
```



ACTIVIDAD

Creamos una función (**dibujarcuadradoverde**) donde pasamos los valores de (**x**). Esto nos permitirá crear 3 cuadrados de forma horizontal.

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    function dibujarcuadradoverde(x){
      var pantalla = document.querySelector("canvas");
      var pincel = pantalla.getContext("2d");
      pincel.fillStyle = "green";
      pincel.fillRect(x,0, 50,50);
      pincel.strokeStyle = "black";
      pincel.strokeRect(x,0, 50,50);
    }

    dibujarcuadradoverde(0);
    dibujarcuadradoverde(50);
    dibujarcuadradoverde(100);

  </script>
</body>
```



En la función (**dibujarcuadradoverde**) pasamos los valores de (**x, y**). Esto nos permitirá crear 3 cuadrados en línea vertical.

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    function dibujarcuadradoverde(x,y){
      var pantalla = document.querySelector("canvas");
      var pincel = pantalla.getContext("2d");
      pincel.fillStyle = "green";
      pincel.fillRect(x,y, 50,50);
      pincel.strokeStyle = "black";
      pincel.strokeRect(x,y, 50,50);
    }

    dibujarcuadradoverde(0, 0);
    dibujarcuadradoverde(0, 50);
    dibujarcuadradoverde(0, 100);

  </script>
</body>
```



ACTIVIDAD

Ahora creamos los tres cuadrados en línea vertical. Que representa los colores de un semáforo.

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    function dibujarcuadradorojo(x,y){
      var pantalla = document.querySelector("canvas");
      var pincel = pantalla.getContext("2d");
      pincel.fillStyle = "red";
      pincel.fillRect(x,y, 50,50);
      pincel.strokeStyle = "black";
      pincel.strokeRect(x,y, 50,50);
    }
    function dibujarcuadradoamarillo(x,y){
      var pantalla = document.querySelector("canvas");
      var pincel = pantalla.getContext("2d");
      pincel.fillStyle = "yellow";
      pincel.fillRect(x,y, 50,50);
      pincel.strokeStyle = "black";
      pincel.strokeRect(x,y, 50,50);
    }
    function dibujarcuadradoverde(x,y){
      var pantalla = document.querySelector("canvas");
      var pincel = pantalla.getContext("2d");
      pincel.fillStyle = "green";
      pincel.fillRect(x,y, 50,50);
      pincel.strokeStyle = "black";
      pincel.strokeRect(x,y, 50,50);
    }

    dibujarcuadradorojo(0, 0);
    dibujarcuadradoamarillo(0, 50);
    dibujarcuadradoverde(0, 100);

  </script>
</body>
```

**MEJORANDO NUESTRO CÓDIGO**

En la función (**dibujarcuadrado**) pasamos los valores de (**x, y, color**). Esto nos permitirá crear 3 cuadrados en línea vertical y le asignamos el color a cada cuadro. Personalizando mejor nuestra función.

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    function dibujarcuadrado(x,y,color){
      var pantalla = document.querySelector("canvas");
      var pincel = pantalla.getContext("2d");
      pincel.fillStyle = color;
      pincel.fillRect(x,y, 50,50);
      pincel.strokeStyle = "black";
      pincel.strokeRect(x,y, 50,50);
    }

    dibujarcuadrado(0, 0, "red");
    dibujarcuadrado(0, 50, "yellow");
    dibujarcuadrado(0, 100, "green");

  </script>
</body>
```

Altura aleatoria del rectángulo

Pedro creó una función que dibuja un rectángulo que recibe 4 parámetros: la coordenada `x`, la coordenada `y`, el color y la altura del rectángulo. El único valor constante es la base del rectángulo y es igual a 50 píxeles. La función creada fue la siguiente:

```
<canvas width="600" height="400"> </canvas>

<script>
  function dibujarRectangulo(x,y,color,altura){
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = color;
    pincel.fillRect(x,y,50,altura);
    pincel.strokeStyle = "black";
    pincel.strokeRect(x,y,50,altura);
  }
</script>
```

Él desea que al momento de llamar a la función `dibujarRectangulo`, la variable altura sea un número aleatorio entre 0 y 50.

¿Cuál de las alternativas realiza en forma correcta la llamada a la función `dibujarRectangulo` considerando los requerimientos de Pedro?

A

```
dibujarRectangulo(50,50,"green",Math.round(Math.aleatorio()*50))
```

B

```
dibujarRectangulo(50,50,"green",Math.random())
```

C

```
dibujarRectangulo(50,50,"green",Math.round(Math.random()*50));
```

¡Correcto! La llamada a la función tiene todos los parámetros en el orden y forma correcta, el último parámetro que es el valor de la altura está usando dos fórmulas matemáticas que ya hemos visto, `Math.round()` que redondea a un entero el número y `Math.random()` que nos devuelve un número aleatorio entre 0 y 1, luego lo multiplicamos por 50 para poder obtener un número entre 0 y 50.

Iteraciones While y For

Recordemos que ambas iteraciones cumplen funciones parecidas (depende nosotros aplicarlas)

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    function dibujarcuadrado(x,y,color){
      var pantalla = document.querySelector("canvas");
      var pincel = pantalla.getContext("2d");
      pincel.fillStyle = color;
      pincel.fillRect(x,y, 50,50);
      pincel.strokeStyle = "black";
      pincel.strokeRect(x,y, 50,50);
    }

    var x = 0;
    while (x < 600){
      dibujarcuadrado(x,0, "red");
      dibujarcuadrado(x,50, "yellow");
      dibujarcuadrado(x,100, "green");

      x = x + 50;
    }

  </script>
</body>
```



```
<script>
  function dibujarcuadrado(x,y,color){
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = color;
    pincel.fillRect(x,y, 50,50);
    pincel.strokeStyle = "black";
    pincel.strokeRect(x,y, 50,50);
  }

  for(var x =0; x <600; x= x+ 50){

    dibujarcuadrado(x, 0, "red");
    dibujarcuadrado(x, 50, "yellow");
    dibujarcuadrado(x, 100, "green");
  }

</script>
```

Diseñando una flor

Tenemos el siguiente código que declara la función

`dibujarCirculo`. Esa función permite diseñar en la pantalla una circunferencia en el eje `x` y `y`, y también nos permite definir su color.

```
<canvas width="600" height="400"> </canvas>

<script>
  var pantalla = document.querySelector("canvas");
  var pincel = pantalla.getContext("2d");

  function dibujarCirculo(x,y, radio,color){

    pincel.fillStyle = color;
    pincel.beginPath();
    pincel.arc(x,y, radio,0, 2*3.14,);
    pincel.fill();
  }

</script>
```

Pero en ese código la función `dibujarCirculo` todavía no está siendo usada, haz uso de la función para diseñar una flor conforme la imagen

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    function dibujarCirculo(x,y, radio,color){

      pincel.fillStyle = color;
      pincel.beginPath();
      pincel.arc(x,y, radio,0, 2*3.14);
      pincel.fill();
    }
    dibujarCirculo(300, 220, 10, "blue");
    dibujarCirculo(300, 200, 10, "red");
    dibujarCirculo(300, 180, 10, "yellow");
    dibujarCirculo(280, 200, 10, "orange");
    dibujarCirculo(320, 200, 10, "black");

  </script>
</body>
```



pero para dejar más prolijo nuestro código vamos a dejar las cinco llamadas a la función `dibujarCirculo` dentro de otra función que vamos a llamar `dibujarFlor`.

```
<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    function dibujarCirculo(x,y, radio,color){

      pincel.fillStyle = color;
      pincel.beginPath();
      pincel.arc(x,y, radio,0, 2*3.14);
      pincel.fill();
    }
    function dibujarFlor(){
      dibujarCirculo(300, 220, 10, "blue");
      dibujarCirculo(300, 200, 10, "red");
      dibujarCirculo(300, 180, 10, "yellow");
      dibujarCirculo(280, 200, 10, "orange");
      dibujarCirculo(320, 200, 10, "black");
    }
    dibujarFlor();
  </script>
</body>
```



Nuestro código mejoró, pero puede quedar mejor aún. Imagine que quisiéramos diseñar nuestra flor más a la derecha de nuestro `canvas`, para eso, tendríamos que cambiar los valores pasados como parámetro en las llamadas a la función `dibujarCirculo` en varios lugares. Si analizamos encontramos el siguiente padrón, solo necesitamos definir las coordenadas del círculo rojo que está al centro, en nuestro actual código está ubicado en la coordenada `X=300` y `Y=200`; y los demás círculos se posicionan a 20 píxeles arriba, abajo, derecha e izquierda del círculo rojo, la distancia es siempre la misma. Siendo así, podemos indicar la posición `X` y `Y`, que sería el centro de la flor, y dentro de la misma función podemos alterar para sumar o restar los 20 píxeles según corresponda.

```

<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    function dibujarCirculo(x,y, radio,color){

      pincel.fillStyle = color;
      pincel.beginPath();
      pincel.arc(x,y, radio,0, 2*3.14);
      pincel.fill();
    }
    function dibujarFlor(x,y){
      dibujarCirculo( x,  y+20,  10, "blue");
      dibujarCirculo( x,  y,      10, "red");
      dibujarCirculo( x,  y-20,  10, "yellow");
      dibujarCirculo(x-20, y,     10, "orange");
      dibujarCirculo(x+20, y,     10, "black");
    }
    dibujarFlor(300, 200);
  </script>
</body>

```

A modo de practicar puedes pasar otras coordenadas para la función `dibujarFlor`. Inclusive puedes diseñar una o más flores llamando a la función más veces.

```

<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    function dibujarCirculo(x,y, radio,color){

      pincel.fillStyle = color;
      pincel.beginPath();
      pincel.arc(x,y, radio,0, 2*3.14);
      pincel.fill();
    }
    function dibujarFlor(x,y){
      dibujarCirculo( x,  y+20,  10, "blue");
      dibujarCirculo( x,  y,      10, "red");
      dibujarCirculo( x,  y-20,   10, "yellow");
      dibujarCirculo(x-20, y,      10, "orange");
      dibujarCirculo(x+20, y,      10, "black");
    }
    dibujarFlor(300, 200);
    dibujarFlor(200, 200);
    dibujarFlor(400, 200);

    dibujarFlor(300, 100);
    dibujarFlor(200, 100);
    dibujarFlor(400, 100);
  </script>
</body>

```

127.0.0.1:5500/Programa3.html

YouTube Traducir Outlook VirusTotal SaveFrom.net Alura Latam

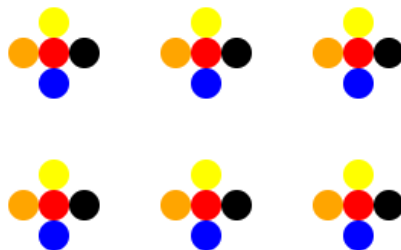


Gráfico de Barras

A través de gráficos podemos expresar visualmente datos o valores numéricos, y así facilitar la comprensión de la información que estamos presentando.

Existen varios tipos de gráficos, entre los más famosos están los gráficos de barras, que será objeto de estudio en este ejercicio.

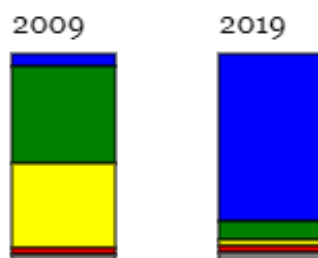
Encontramos algunos datos relevantes en internet sobre la evolución de uso de navegadores o exploradores en los últimos 10 años, el resultado fue el siguiente.

- En 2009: 6% Chrome, 47% Firefox, 41% Internet Explorer/Edge*, 3% Safari, 3% Otros.
- En 2019: 81% Chrome, 9% Firefox, 3% Internet Explorer/Edge*, 3% Safari, 4% Otros.

*Para simplificar nuestro gráfico sumamos los valores de Internet Explorer e IE Edge, los dos navegadores son de la misma empresa, Microsoft.

Como podrán ver en los datos la relevancia que ganó el navegador de Google (Chrome), teniendo una supremacía del 81% en 2019.

Existen varios tipos de gráficos de barras, en este ejemplo, vamos usar las barras verticales apiladas, nuestros datos graficados se verían así:



Tú ya aprendiste a dibujar rectángulos y ya creamos una función con ese propósito, ingresar texto dentro de nuestro `canvas` es sencillo también, a continuación sigue el código con la función `dibujarRectangulo` y `escribirTexto` :

```

<canvas width="600" height="400"></canvas>

<script>

    function dibujarRectangulo(x, y, base, altura, color) {
        var pantalla = document.querySelector("canvas");
        var pincel = pantalla.getContext("2d");

        pincel.fillStyle=color;
        pincel.fillRect(x,y, base, altura);
        pincel.strokeStyle="black";
        pincel.strokeRect(x,y, base, altura);
    }

    function escribirTexto(x , y, texto) {
        var pantalla = document.querySelector("canvas");
        var pincel = pantalla.getContext("2d");

        pincel.font="15px Georgia";
        pincel.fillStyle="black";
        pincel.fillText(texto, x, y);
    }

</script>

```

Ya vimos también cómo representar varios valores dentro de un array. Así podemos guardar los valores de los porcentuales de cada año. En el mundo de gráficos los valores son llamados de series:

```

var serie2009 = [6, 47, 41, 3, 3];
var serie2019 = [81, 9, 3, 3, 4];

```

Cada valor en el array representa un %.

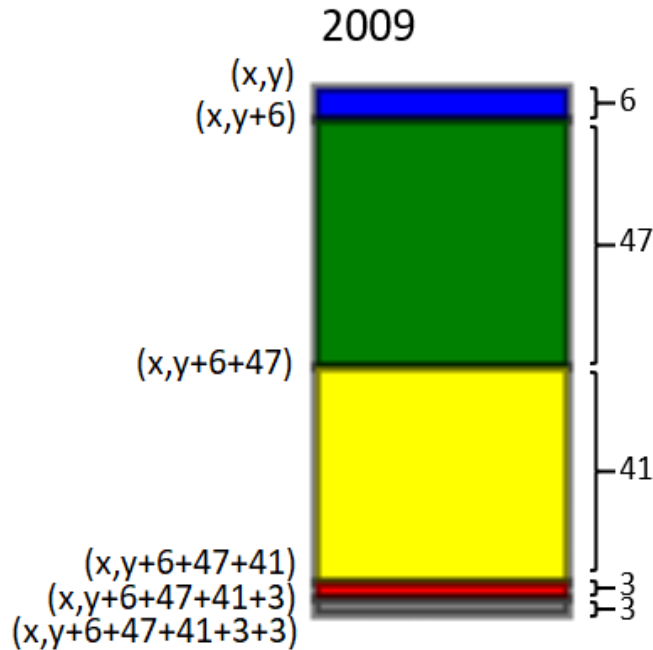
Igualmente, podemos crear otro array con los colores usados en el gráfico:

```

var colores = ["blue","green","yellow", "red","gray"];

```

var serie2009 = [6,47,41,3,3]



Ahora viene el desafío: escribe una función `dibujarBarra` que cree una barra (5 rectángulos, el primero azul, el segundo verde, etc., igual al gráfico de arriba)

Vamos a implementar la función paso a paso.

- Primero vamos a escribir el texto, que queda a 10 pixeles arriba de la barra.

```
function dibujarBarra(x, y, serie, colores, texto) {
    escribirTexto(x, y - 10, texto);
}
```

- Ahora, vamos a crear un Loop usando la serie. El truco aquí es que declaramos antes una variable para luego acumular las alturas dentro del Loop.

```
function dibujarBarra(x, y, serie, colores, texto) {
    escribirTexto(x, y - 10, texto);
    var sumaAltura = 0; //variable auxiliar
    for (var a = 0; i < serie.length; i++) {
    }
}
```

- Dentro del Loop vamos a capturar los valores de la serie usando el índice. Ese valor representa la altura de los rectángulos.

```
function dibujarBarra(x, y, serie, colores, texto) {
    escribirTexto(x, y - 10, texto);
    var sumaAltura = 0; //variable auxiliar
    for (var a = 0; i < serie.length; i++) {
        var altura = serie[i];
    }
}
```

- Y ahora podemos dibujar un rectángulo con nuestra función dibujarRectangulo

```
function dibujarBarra(x, y, serie, colores, texto) {
    escribirTexto(x, y - 10, texto);
    var sumaAltura = 0; //variable auxiliar
    for (var a = 0; i < serie.length; i++) {
        var altura = serie[i];
        dibujarRectangulo(x, y + sumaAltura, 50, altura, colores[i]);
    }
}
```

- Estamos usando las variables sumaAltura y Altura. Al final, no podemos olvidar de sumar las alturas:

```
function dibujarBarra(x, y, serie, colores, texto) {
    escribirTexto(x, y - 10, texto);
    var sumaAltura = 0; //variable auxiliar
    for (var a = 0; i < serie.length; i++) {
        var altura = serie[i];
        dibujarRectangulo(x, y + sumaAltura, 50, altura, colores[i]);
        sumaAltura = sumaAltura + altura;
    }
}
```

- Eso es todo lo que necesitamos para dibujar una barra, ahora, basta llamar a la función pasando las coordenadas iniciales, la serie, los colores y el texto.

```
var colores = ["blue","green","yellow", "red","gray"];
var serie2009 = [6,47,41,3,3];
var serie2019 = [81,9,3,3,4];

dibujarBarra(50, 50, serie2009, colores, "2009");
dibujarBarra(150, 50, serie2019, colores, "2019");
```

```

<canvas width="600" height="400"></canvas>

<script>

function dibujarRectangulo(x, y, base, altura, color) {
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle=color;
    pincel.fillRect(x,y, base, altura);
    pincel.strokeStyle="black";
    pincel.strokeRect(x,y, base, altura);
}

function escribirTexto(x, y, texto) {
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.font="15px Georgia";
    pincel.fillStyle="black";
    pincel.fillText(texto, x, y);
}

function dibujarBarra(x,y,serie,colores,texto){

    escribirTexto(x, y - 10, texto);

    var sumaAltura = 0; //variable auxiliar

    for (var i = 0; i < serie.length; i++) {
        var altura = serie[i];
        dibujarRectangulo(x, y + sumaAltura, 50, altura, colores[i]);
        sumaAltura = sumaAltura + altura;
    }
}

var colores = ["blue","green","yellow", "red","gray"];
var serie2009 = [6,47,41,3,3];
var serie2019 = [81,9,3,3,4];

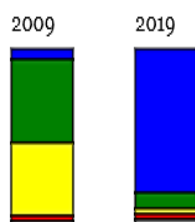
dibujarBarra(50, 50, serie2009, colores, "2009");
dibujarBarra(150, 50, serie2019, colores, "2019");

</script>

```

← → ↺ 127.0.0.1:5500/GraficoDeBarras.html

[Gmail](#) [YouTube](#) [Traducir](#) [Outlook](#) [VirusTota](#)



Lo que aprendimos

- A usar funciones para encapsular la creación de figuras.
- A utilizar ciclos *Loop* para repetir actividades cuando creamos nuestras figuras.

Actividad

Creo la función **exibirAlerta** que al momento de dar un clic en el cuadro canvas (gris) me muestra el mensaje ("Usted hizo un clic")

```
<body>
  <canvas width="600" height="400"> </canvas>
  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle = "grey";
    pincel.fillRect(0,0, 600,400);

    function exibirAlerta(){
      alert("Usted hizo un clic");
    }

    pantalla.onclick = exibirAlerta;
  </script>
</body>
```

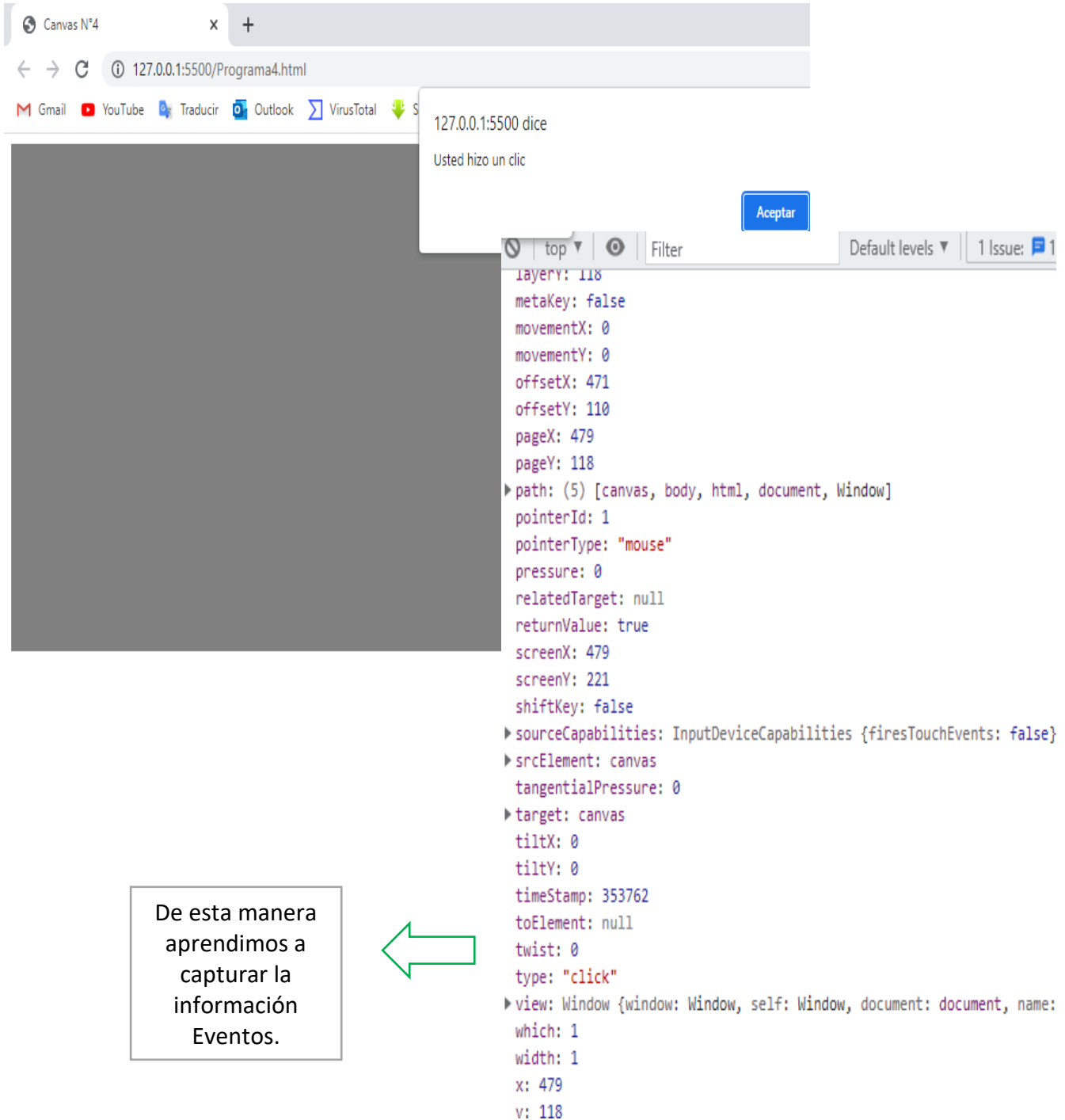
Ahora me interesa saber dónde el usuario está haciendo un clic en el cuadro canvas (gris).

```
<body>
  <canvas width="600" height="400"> </canvas>
  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle = "grey";
    pincel.fillRect(0,0, 600,400);

    function exibirAlerta(evento){
      console.log(evento);
      alert("Usted hizo un clic");
    }

    pantalla.onclick = exibirAlerta;
  </script>
</body>
```



The screenshot shows a web browser window with a single tab titled "Canvas N°4". The address bar shows the URL "127.0.0.1:5500/Programa4.html". Below the address bar, there are several icons for services like Gmail, YouTube, Traducir, Outlook, VirusTotal, and a download icon. The main content area of the browser is a large, solid gray rectangle representing a canvas. A context menu is open over the canvas, displaying the text "127.0.0.1:5500 dice" and "Usted hizo un clic", with an "Aceptar" button. To the right of the canvas, the browser's developer console is open, showing a list of properties for the event object. A green arrow points from a text box on the left towards the console.

127.0.0.1:5500 dice
Usted hizo un clic

Aceptar

top Filter Default levels 1 Issue: 1

```

layerY: 118
metaKey: false
movementX: 0
movementY: 0
offsetX: 471
offsetY: 110
pageX: 479
pageY: 118
▶ path: (5) [canvas, body, html, document, Window]
pointerId: 1
pointerType: "mouse"
pressure: 0
relatedTarget: null
returnValue: true
screenX: 479
screenY: 221
shiftKey: false
▶ sourceCapabilities: InputDeviceCapabilities {firesTouchEvents: false}
▶ srcElement: canvas
  tangentialPressure: 0
▶ target: canvas
  tiltX: 0
  tiltY: 0
  timeStamp: 353762
  toElement: null
  twist: 0
  type: "click"
▶ view: Window {window: Window, self: Window, document: document, name:
  which: 1
  width: 1
  x: 479
  y: 118
  
```

De esta manera aprendimos a capturar la información Eventos.

Asociando eventos con funciones

Orlando está estudiando programación y recibió como desafío crear un programa usando HTML y JavaScript, el programa tiene que mostrar un mensaje con la hora actual en el siguiente formato HH:MM:SS y con las coordenadas (x,y) cuando el usuario haga clic en la pantalla.

Él ya escribió gran parte del código en forma correcta:

```
<canvas width="600" height="400"> </canvas>

<script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = "grey";
    pincel.fillRect(0,0,600,400);

    function mostrarMensaje(evento){
        var x = evento.pageX - pantalla.offsetLeft; //pos x
        var y = evento.pageY - pantalla.offsetTop; //pos y
        var d = new Date(); //crea una variable de tipo date
        var hora = checkTime(d.getHours()) + ":" + checkTime(d.getMinutes()) + ":" + checkTime(d.getSeconds());
        //Verifica si el número de las horas, minutos o segundos, tiene un solo dígito,
        //caso positivo le adiciona el cero en la frente para obtener el formato deseado 00:00:00
        function checkTime(i){
            if (i<10){
                i="0" + i;
            }
            return i;
        }

        alert("La hora es: " + hora + " y las coordenadas son: x=" + x + ", y=" + y);
    }

    // Aquí viene la llamada a la función con el evento onclick

</script>
```


Solo que Orlando no recuerda cómo asociar la función

`mostrarMensaje` con el evento `onclick`, ¿puedes ayudarlo seleccionando la alternativa correcta?

A

```
pantalla.onclick = mostrarMensaje();
```

B

```
pantalla.onclick = mostrarMensaje;
```

¡Correcto! Para asociar la función con el evento `onclick`, le pasamos la función sin los paréntesis y sin ningún parámetro. Lo que nuestro navegador está haciendo entre bastidores es cargar la función en el compilador y cada vez que el usuario hace clic en la pantalla, la función es ejecutada y recibe como parámetro algunas propiedades y características del evento que luego aprovechamos para extraer las coordenadas.

C

```
pantalla.onclick = mostrarMensaje(evento);
```

👉 **ACTIVIDAD:** Que hacer con la información que capturamos.

Cuando hagamos clic en cuadro de canvas nos indique cuales son nuestras coordenadas (x, y).

- **Evento** me captura automáticamente la información al dar clic.
- **pageX, pageY** captura las coordenadas de los ejes X y Y de toda la página.
- **pantalla.offsetLeft** resta lo que está sobrando a la izquierda de la pantalla y dentro del cuadro (gris) de canvas ahora obtengo las coordenadas (0,0)
- **pantalla.offsetTop** resta lo que está sobrando arriba de la pantalla y dentro del cuadro (gris) de canvas ahora obtengo las coordenadas (0,0).

```
<body>
  <canvas width="600" height="400"> </canvas>
  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle = "grey";
    pincel.fillRect(0,0, 600,400);

    function exhibirAlerta(evento){
      var x = evento.pageX - pantalla.offsetLeft;
      var y = evento.pageY - pantalla.offsetTop;
      console.log(evento);
      alert("Coordenadas " + " X: " +x + " " + " Y: " +y);
    }

    pantalla.onclick = exhibirAlerta;
  </script>
</body>
```

ACTIVIDAD

Ahora cuando haga clic me dibuje un círculo (no me dé coordenadas, actividad anterior).

```
<body>
  <canvas width="600" height="400"> </canvas>
  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle = "grey";
    pincel.fillRect(0,0, 600,400);

    function dibujarCirculo(evento){
      var x = evento.pageX - pantalla.offsetLeft;
      var y = evento.pageY - pantalla.offsetTop;

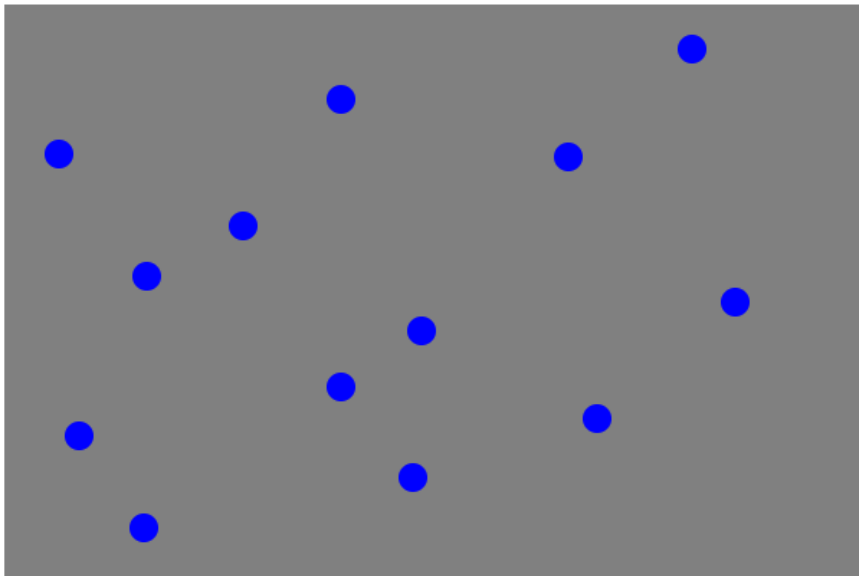
      pincel.fillStyle = "blue";
      pincel.beginPath();
      pincel.arc(x,y, 10,0,2*3.14);
      pincel.fill();

      console.log(x + "," + y);
    }

    pantalla.onclick = dibujarCirculo;
  </script>
</body>
```

← → ↻ ⓘ 127.0.0.1:5500/Programa5.html ⓘ ☆ ABB ⚙ □

📧 Gmail 📺 YouTube 🌐 Traducir 📧 Outlook 📄 VirusTotal ⬇️ SaveFrom.net 📄 Alura Latam



Aprendimos en este capítulo a interactuar con el usuario, por ejemplo, dibujando un círculo azul cada vez que hacía clic en el `canvas`, en la pantalla.

Para eso, fue necesario asociar la función `dibujarCirculo` al evento `onclick` de nuestra pantalla. Aprendimos también, que será nuestro navegador quién llamará a nuestra función al momento de identificar clics en el `canvas`. Además de llamar a la función, el propio navegador pasa un parámetro para la función. Gracias a ese parámetro, tenemos acceso a varias informaciones sobre el evento disparado, y así podemos descubrir la posición del eje `x` y `y` de la coordenada donde el usuario hizo clic.

Una forma diferente de probar un if

Tenemos el siguiente código:

```
<script>
  var aprendi = true;
  if(aprendi == true) {
    alert("El instructor queda muy feliz");
  } else {
    alert("El instructor no va a desistir hasta que el alumno sea una eminencia en programación");
  }
</script>
```

No hay ninguna novedad ¿cierto?, inclusive ya lidiamos con códigos como ese en el curso anterior. Nuestra condición `if` prueba si el valor de la variable `aprendi` es verdadera, y para ello, usamos el operador `==`.

Sin embargo, podemos simplificar el código para:

```
<script>
  var aprendi = true;
  if(aprendi) {
    alert("El instructor queda muy feliz");
  } else {
    alert("El instructor no va a desistir hasta que el alumno sea una eminencia en programación");
  }
</script>
```

Observa que no usé más el operador `==` . Puede que estés intrigado por qué funciona correctamente. Pero si analizamos la sintaxis del condicionante `if()` , él espera recibir `true` o `false` para saber si ejecuta el código dentro del `if` o dentro del `else` . Si `aprendi` ya lo estamos inicializando como `true` es redundante preguntar dentro del bloque `if` nuevamente `aprendi == true` .

Las variables booleanas ya guardan `true` o `false` y podemos usarla directamente en el `if` ahorrando algunos caracteres. Sin embargo, si te sientes más seguro con la forma anterior, puedes continuar usándola sin ningún problema.

Cambiando de color

Ejercicio

tengo una propuesta con el objetivo que apliques los conocimientos adquiridos hasta aquí, lo más importante es que desarrolles tu lógica de programación. Intenta resolverlo. Bueno el desafío es el siguiente, vamos a liberar que el usuario pueda alterar el color de los círculos que son diseñados en la pantalla. Los colores que liberaremos serán azul, rojo y verde (`blue` , `red` y `green`). Esa lista de colores te debe recordar algo que ya vimos, los `arrays` en el anterior curso.

¿Cómo dejaremos al usuario escoger el color? A cada clic del botón **DERECHO** del mouse, el color padrón que es `blue` , deberá cambiarse a `red` . Si el usuario realiza otro clic del botón **DERECHO**, el color del círculo se cambiará a `green` , debes respetar el orden de alteración de los colores (`blue` , `red` y `green`). En caso de que el usuario vuelva a presionar el botón **DERECHO** el color debe volver a ser `blue` .

Para que las circunferencias aparezcan seguimos manteniendo la misma lógica, ellas aparecerán con cada clic **IZQUIERDO** del mouse.

La instrucción para capturar el evento cuando el usuario hace clic en el botón derecho todavía no fue enseñada, pero es fácil de ser implementada, el comando a ser usado es `oncontextmenu` . La sintaxis es igual al evento `onclick` que usamos para capturar el clic del botón izquierdo del mouse, de todas formas te muestro a continuación cómo puedes usar esa instrucción:

```

<canvas width="600" height="400"> </canvas>

<script>

    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = "grey";
    pincel.fillRect(0,0,600,400);

    function dibujarCirculo(evento){
        var x = evento.pageX - pantalla.offsetLeft;
        var y = evento.pageY - pantalla.offsetTop;
        pincel.fillStyle = "blue";
        pincel.beginPath();
        pincel.arc(x,y,10,0,2*3.14);
        pincel.fill();
        console.log(x + ", " + y);
    }

    pantalla.onclick = dibujarCirculo;

    function alterarColor() {
        alert("Funcionó");
        return false;
    }

    pantalla.oncontextmenu = alterarColor;

</script>

```

Ejecuta ese programa y prueba realizar clic en el botón **DERECHO** en la pantalla. El mensaje **"Funcionó"** será exhibido. La instrucción `return false` es importante para que el menú contextual padrón de `canvas` no sea exhibido, o sea, queremos apenas alterar el color con el clic del botón y no exhibir un menú para el usuario.

Recuerda que no existe código perfecto, preocúpate por resolver el desafío con todas las herramientas y conocimientos que adquiriste hasta aquí. Si tienes dudas puedes acudir al foro. Y al final compara tu resultado con el resultado del instructor.

El primer paso será crear dos variables. La primera un `array` de los colores, la segunda el índice del color actual seleccionado.

```
<canvas width="600" height="400"> </canvas>

<script>
  var pantalla = document.querySelector("canvas");
  var pincel = pantalla.getContext("2d");

  pincel.fillStyle = "grey";
  pincel.fillRect(0,0, 600,400);
  var colores = ["blue", "red", "green"];
  var indiceColorActual = 0;

  function dibujarCirculo(evento){
    var x = evento.pageX - pantalla.offsetLeft;
    var y = evento.pageY - pantalla.offsetTop;

    pincel.fillStyle = colores[indiceColorActual];
    pincel.beginPath();
    pincel.arc(x,y, 10,0,2*3.14);
    pincel.fill();
    console.log(x +", " +y);
  }
  pantalla.onclick = dibujarCirculo;

  function alterarColor(){
    alert("Funciono");
    return false;
  }
  pantalla.oncontextmenu = alterarColor;

</script>
```

Toma en cuenta que dentro de la función `dibujarCirculo` definimos el color pasado para `pincel.fillStyle`, usando el índice y nuestro `array` de colores. Para obtener un color del `array` necesitamos de su índice (posición), la primera vez que el usuario abre la aplicación el valor del `indiceColorActual` es 0, que es el índice para el color `blue`. Necesitamos ahora implementar el cambio de color a través del clic derecho del mouse. Para eso vamos a modificar la función `alterarColor`, que ya fue asociada al evento `oncontextmenu` de nuestra pantalla.

```
function alterarColor() {
    indiceColorActual++;
    alert(indiceColorActual);
    return false; //menú contextual padrón de `canvas` no sea exhibido
}

pantalla.oncontextmenu = alterarColor;
```

Nota que en cada clic del botón derecho, incrementamos la variable `indiceColorActual`, por lo tanto, si hacemos 10 veces clic derecho, su valor final será 9 y eso no es correcto porque su valor máximo no puede pasar de 2, pues es el último índice para acceder al último elemento del array. Entonces, necesitamos comprobar con un `if` cada vez que `alterarColor` sea llamada y verificar si el `indiceColorActual` es mayor o igual a `colores.length`. Si fuera, alteramos el índice para 0, reiniciamos el índice y volvemos al color `blue`.

```
function alterarColor() {
    indiceColorActual++;
    if(indiceColorActual >= colores.length){
        indiceColorActual = 0; //vuelve para el primer color, blue
    }

    return false; //menú contextual padrón de `canvas` no sea exhibido
}

pantalla.oncontextmenu = alterarColor;
```


Resultado Final

```

<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle = "grey";
    pincel.fillRect(0,0, 600,400);
    var colores = ["blue", "red", "green"];
    var indiceColorActual = 0; // comienza con blue

    function dibujarCirculo(evento){
      var x = evento.pageX - pantalla.offsetLeft;
      var y = evento.pageY - pantalla.offsetTop;

      pincel.fillStyle = colores[indiceColorActual];
      pincel.beginPath();
      pincel.arc(x,y, 10,0,2*3.14);
      pincel.fill();
      console.log(x + "," + y);
    }
    pantalla.onclick = dibujarCirculo;

    function alterarColor() {

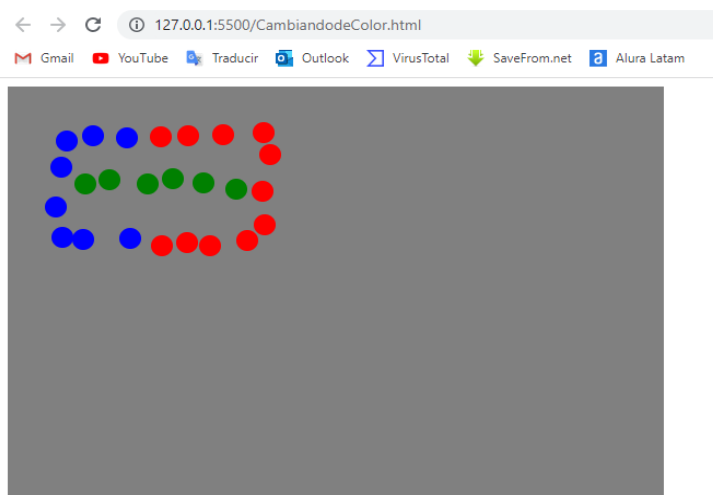
      indiceColorActual++;
      if(indiceColorActual >= colores.length){
        indiceColorActual = 0; //vuelve para el primer color, blue
      }

      return false; //menú contextual padrón de `canvas` no sea exhibido
    }

    pantalla.oncontextmenu = alterarColor;

  </script>
</body>

```



Lo que aprendimos

- A interactuar con el usuario a través de los eventos.
- A asociar funciones con los eventos y a capturar las propiedades del evento para pasarlas como parámetro.

ACTIVIDAD

```
<body>
  <canvas width="600" height="400"> </canvas>
  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");

    pincel.fillStyle = "lightgrey";
    pincel.fillRect(0,0, 600,400);

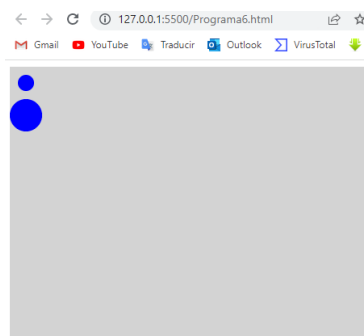
    function disenharCirculo(x,y,radio){

      pincel.fillStyle = "blue";
      pincel.beginPath();
      pincel.arc(x,y, radio,0,2*Math.PI);
      pincel.fill();

    }

    disenharCirculo(20,20, 10);
    disenharCirculo(20,60, 20);

  </script>
</body>
```



ACTIVIDAD

El círculo azul ahora recorra “línea horizontal” por si sola hasta llegar al final del cuadro de canvas.

- La función **limpiarPantalla** limpia la pantalla canvas, cada vez que dibuja un nuevo círculo.
- **SetInterval** llama repetidamente a una función, con un retraso de tiempo fijo entre cada llamada. (milisegundos “1000”)

```
<canvas width="600" height="400"> </canvas>
<script>
  var pantalla = document.querySelector("canvas");
  var pincel = pantalla.getContext("2d");

  pincel.fillStyle = "lightgrey";
  pincel.fillRect(0,0, 600,400);

  function disenharCirculo(x,y,radio){

    pincel.fillStyle = "blue";
    pincel.beginPath();
    pincel.arc(x,y, radio,0,2*Math.PI);
    pincel.fill();
  }
  function limpiarPantalla(){

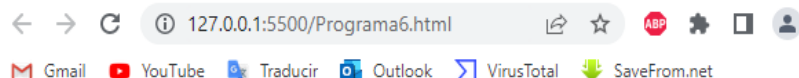
    pincel.clearRect(0,0, 600,400);
  }

  var x = 0;
  function actualizarPantalla(){

    limpiarPantalla();
    disenharCirculo(x,20, 10);
    x++;
  }

  setInterval(actualizarPantalla, 50);

</script>
```



Ayudando a nuestra compañera con su código

Gabriela creó su código siguiendo todos los pasos que fueron presentados durante los videos del aula, sólo que al momento de ejecutar su código el círculo azul no se mueve y se queda estático en el borde izquierdo de la pantalla. Ella está segura que ejecutó todo y que entendió cada detalle explicado, enseguida te mostramos el código de Gabriela:

```
<canvas width="600" height="400"> </canvas>

<script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = "lightgrey";
    pincel.fillRect(0,0,600,400);

    function disenharCircunferencia(x,y,radio){
        pincel.fillStyle = "blue";
        pincel.beginPath();
        pincel.arc(x,y,radio,0,2*Math.PI);
        pincel.fill();
    }

    function limpiarPantalla(){
        pincel.clearRect(0,0,600,400);
    }

    var x = 0

    function actualizarPantalla(){
        limpiarPantalla();
        disenharCircunferencia(x,20,10);
        x++;
    }

    setInterval(actualizarPantalla(),100);
</script>
```

¿Cuál de las alternativas tiene el código correcto del programa de Gabriela?

A

```

<canvas width="600" height="400"> </canvas>

<script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = "lightgrey";
    pincel.fillRect(0,0,600,400);

    function disenharCircunferencia(x,y,radio){
        pincel.fillStyle = "blue";
        pincel.beginPath();
        pincel.arc(x,y,radio,0,2*Math.PI);
        pincel.fill();
    }

    function limpiarPantalla(){
        pincel.clearRect(0,0,600,400);
    }

    var x = 0

    function actualizarPantalla(){
        limpiarPantalla();
        disenharCircunferencia(x,20,10);
        x++;
    }

    setInterval(limpiarPantalla,100);
</script>

```

B

```

<canvas width="600" height="400"> </canvas>

<script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = "lightgrey";
    pincel.fillRect(0,0,600,400);

    function disenharCircunferencia(x,y,radio){
        pincel.fillStyle = "blue";
        pincel.beginPath();
        pincel.arc(x,y,radio,0,2*Math.PI);
        pincel.fill();
    }

    function limpiarPantalla(){
        pincel.clearRect(0,0,600,400);
    }

    var x = 0

    function actualizarPantalla(){
        limpiarPantalla();
        disenharCircunferencia(x,20,10);
        x++;
    }

    setInterval(actualizarPantalla,100);
</script>

```

¡Correcto! El error cometido por Gabriela era que había colocado los paréntesis al momento de llamar a la función `actualizarPantalla`, entonces, en su código la función `actualizarPantalla` estaba siendo ejecutada al momento de inicializar el programa, no estaba siendo accionada por la función `setInterval`. Gabriela necesita eliminar los paréntesis dejando la llamada a la función así:

```
setInterval(actualizarPantalla,100) .
```

Este mismo concepto ya lo habíamos visto anteriormente cuando usamos el evento `onclick` del mouse.

C

```
<canvas width="600" height="400"> </canvas>

<script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = "lightgrey";
    pincel.fillRect(0,0,600,400);

    function disenharCircunferencia(x,y,radio){
        pincel.fillStyle = "blue";
        pincel.beginPath();
        pincel.arc(x,y,radio,0,2*Math.PI);
        pincel.fill();
    }

    function limpiarPantalla(){
        pincel.clearRect(0,0,600,400);
    }

    var x = 0

    function actualizarPantalla(){
        limpiarPantalla();
        disenharCircunferencia(x,20,10);
        x++;
    }

    setInterval(actualizarPantalla(100),100);

</script>
```

¡Ya que va, que vuelva!

En esta aula aprendimos a animar un círculo que se movía de izquierda a derecha, hasta dejar el área visible del `canvas`, nuestra pantalla.

DESAFIO

Altera el código para que el círculo ni bien llegue al extremo derecho de la pantalla vuelva en sentido contrario (de derecha a izquierda), y cuando retorne al lugar inicial (lado izquierdo), nuevamente se mueva de izquierda a derecha, y así sucesivamente, yendo y volviendo eternamente.

👉 Paso a Paso

Para saber si el círculo va de izquierda a derecha o de derecha a izquierda, necesitamos descubrir su sentido. La circunferencia comienza de izquierda a derecha, pero cuando su posición `x` sea superior a 600, necesitamos invertir su sentido. La misma lógica cuando esté retornando de derecha para izquierda y la posición sea menor que 0, necesitamos invertir el sentido nuevamente.

👉 Siendo así, vamos a crear una variable `sentido`. Vamos a inicializarla con el valor de 1, indicando que nuestra variable `x` debe ser incrementada de 1 en 1. Cuando sea mayor que 600, `sentido` recibirá el valor de -1, lo que hará el de decremento en la variable `x`.

```
var x = 0
var sentido = 1;
```

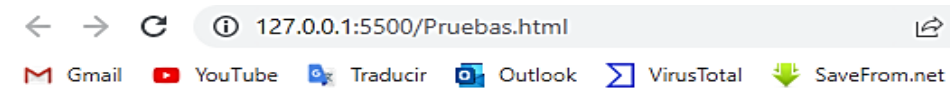
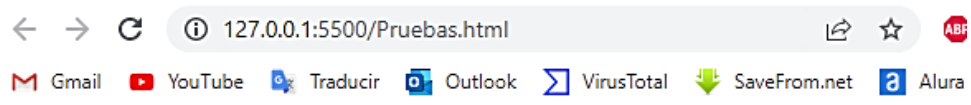
👉 Ahora, dentro de la función `actualizarPantalla`, antes de llamar a `disenharCircunferencia` vamos a verificar si necesitamos alterar el valor de `sentido`.

```
limpiarPantalla();
if(x > 600){
    sentido = -1;
}else if(x < 0){
    sentido = 1;
}
disenharCircunferencia(x,20,10);
x++;
```

👉 Ya que verificamos si `sentido` necesita ser alterado, ahora, después de llamar a la función `disenharCircunferencia`, vamos a usar el valor de nuestra variable para realizar el incremento de `x`.

```
disenharCircunferencia(x,20,10);
x = x + sentido;
```

👉 Resultado



CÓDIGO COMPLETO

```

<body>
  <canvas width="600" height="400"> </canvas>

  <script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = "lightgrey";
    pincel.fillRect(0,0,600,400);

    function disenharCircunferencia(x,y,radio){
      pincel.fillStyle = "blue";
      pincel.beginPath();
      pincel.arc(x,y,radio,0,2*Math.PI);
      pincel.fill();
    }

    function limpiarPantalla(){
      pincel.clearRect(0,0,600,400);
    }

    var x = 0
    var sentido = 1;

    function actualizarPantalla(){
      limpiarPantalla();
      if(x > 600){
        sentido = -1;
      }else if(x < 0){
        sentido = 1;
      }

      disenharCircunferencia(x,20,10);
      x = x + sentido;
    }

    setInterval(actualizarPantalla,10);

  </script>
</body>

```

Lo que aprendimos

- A realizar animaciones simples.
- A limpiar la pantalla usando `clearRect()`.
- A llamar a funciones a cada cierto intervalo de tiempo con `setInterval()`.

Creando un Objetivo Aleatorio

- La función **desenharObjetivo**(*x*, *y*) recibirá los parámetros *x* , *y* y la variable (**var** *radio* = 10) permitirá a parte del radio "círculo" le sume 10 o 20, adicional otro parámetro que recibirá es el color ("red", "white").

De esta manera dibujara el círculo correspondiente.



- La función **sortearPosicion**(*maximo*) recibirá como máximo valor los valores de:
var *x*Aleatorio = **sortearPosicion**(600);
var *y*Aleatorio = **sortearPosicion**(400);
- De esta manera me retornara (**return**) los valores de manera aleatoria con ayuda de: **Math.random()** * *maximo* y **Math.floor** permite redondear para abajo (Ejm: 400,05 -> 400) no redondear para arriba.

```
<canvas width="600" height="400"> </canvas>

<script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = "lightgrey";
    pincel.fillRect(0,0,600,400);

    var radio = 10;

    function desenharCircunferencia(x,y,radio, color){
        pincel.fillStyle = color;
        pincel.beginPath();
        pincel.arc(x,y,radio,0,2*Math.PI);
        pincel.fill();
    }

    function limpiarPantalla(){
        pincel.clearRect(0,0,600,400);
    }

    var x = 0

    function actualizarPantalla(){
        limpiarPantalla();
        desenharCircunferencia(x,20,10);
        x++;
    }

    function desenharObjetivo(x, y){
        desenharCircunferencia(x,y, radio + 20, "red");
        desenharCircunferencia(x,y, radio +10, "white");
        desenharCircunferencia(x,y, radio, "red");
    }

```

```
function sortearPosicion(maximo){
    return Math.floor(Math.random()* maximo);
}

var xAleatorio = sortearPosicion(600);
var yAleatorio = sortearPosicion(400);

diseñarObjetivo(xAleatorio,yAleatorio);

</script>
```

Disparando contra el objetivo

En la siguiente actividad cuando de clic en el círculo rojo pequeño.
Me muestre un mensaje ("Tiro Certero")



- Comienzo a crear la función disparar que cuando De clic me muestre el mensaje ("Tiro Certero")

```
function disparar(evento){
    alert("Tiro Certero");
}

pantalla.onclick = disparar;
```

- Creo variable (var x, var y) que me permite capturar la posición x, y cuando de Clic

```
var x = evento.pageX - pantalla.offsetLeft;
var y = evento.pageY - pantalla.offsetTop;
```

- Creo la lógica **If** Que me permita comparar las variables (var x, var y) con (xAleatorio, yAleatorio)

```
xAleatorio = sortearPosicion(600);
yAleatorio = sortearPosicion(400);
```

De esta manera cuando acierte dando clic Dentro del círculo rojo pequeño me mostrara el mensaje ("Tiro Certero")

```
function disparar(evento){

    var x = evento.pageX - pantalla.offsetLeft;
    var y = evento.pageY - pantalla.offsetTop;

    if ( (x < xAleatorio + radio) &&
        (x > xAleatorio - radio) &&
        (y < yAleatorio + radio) &&
        (y > yAleatorio - radio))
    {
        alert("Tiro Certero");
    }
}
```

```
<canvas width="600" height="400"> </canvas>

<script>
    var pantalla = document.querySelector("canvas");
    var pincel = pantalla.getContext("2d");
    pincel.fillStyle = "lightgrey";
    pincel.fillRect(0,0,600,400);

    var radio = 10;
    var x = 0;

    var xAleatorio;
    var yAleatorio;

    function disenharCircunferencia(x,y,radio, color){
        pincel.fillStyle = color;
        pincel.beginPath();
        pincel.arc(x,y,radio,0,2*Math.PI);
        pincel.fill();
    }

    function limpiarPantalla(){

        pincel.clearRect(0,0,600,400);

    }

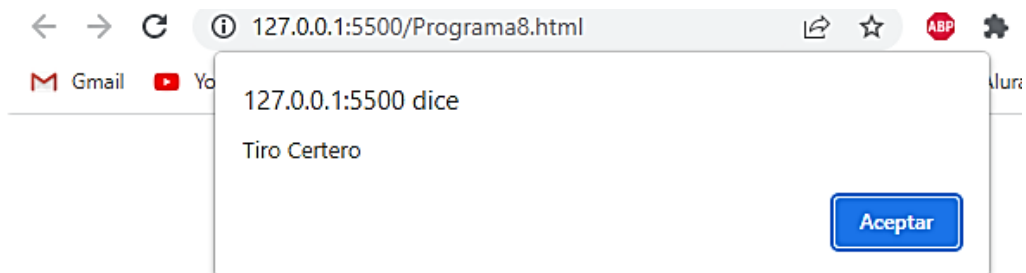
    function disenharObjetivo(x, y){

        disenharCircunferencia(x,y, radio + 20, "red");
        disenharCircunferencia(x,y, radio +10, "white");
        disenharCircunferencia(x,y, radio, "red");

    }

    function sortearPosicion(maximo){
        return Math.floor(Math.random()* maximo);
    }
</script>
```

```
function actualizarPantalla(){  
  
    limpiarPantalla();  
    xAleatorio = sortearPosicion(600);  
    yAleatorio = sortearPosicion(400);  
    disenharObjetivo(xAleatorio,yAleatorio);  
    x++;  
}  
  
setInterval(actualizarPantalla, 1000);  
  
function disparar(evento){  
  
    var x = evento.pageX - pantalla.offsetLeft;  
    var y = evento.pageY - pantalla.offsetTop;  
  
    if ( (x < xAleatorio + radio) &&  
        (x > xAleatorio - radio) &&  
        (y < yAleatorio + radio) &&  
        (y > yAleatorio - radio))  
    {  
        alert("Tiro Certero");  
    }  
}  
  
pantalla.onclick = disparar;  
  
</script>
```



Redondeo de valores

A lo largo del curso aprendimos dos formas de realizar redondeo de valores numéricos para eliminar posiciones decimales, las fórmulas matemáticas aprendidas fueron `Math.round` y `Math.floor`.

Tenemos las siguientes afirmaciones sobre estas dos funciones:

- A) `Math.round` retorna el valor de un número redondeado al entero más cercano.
- B) `Math.floor` la usamos cuando queremos redondear un número al máximo entero menor.
- C) `Math.round` y `Math.floor` realizan la misma acción, no existe diferencias entre ellas.

De las afirmaciones de arriba selecciona la alternativa correcta:



Solo la opción C es falsa.

¡Correcto! Las opciones A y B son verdaderas, y la opción C es falsa, pues si bien tanto `Math.round` como `Math.floor` las usamos para redondear valores, `Math.round` la usamos cuando queremos retornar el valor de un número redondeado al entero más cercano, y `Math.floor` cuando queremos redondear un número al máximo entero menor. Por ejemplo si quisiéramos redondear el número `27.83` con `Math.round(27.87)` el resultado sería `28` y con `Math.floor(27.87)` sería `27`.

Diseñando con el mouse

El siguiente código permite dibujar círculos en la pantalla mientras el botón izquierdo se encuentra presionado, eso significa que mientras no soltemos el botón izquierdo, diseñaremos un círculo a lado de otro, que dará un efecto como si estuviéramos pasando un pincel por la pantalla. Si soltamos el botón izquierdo, la acción de mover el mouse sobre la pantalla no tendría que diseñar nada. Al final tendremos un efecto visto en herramientas como Paint de Windows.

```
<canvas width="600" height="400"></canvas>

<script>
  var pantalla = document.querySelector('canvas');
  var pincel = pantalla.getContext('2d');

  pincel.fillStyle = 'grey';
  pincel.fillRect(0, 0, 600, 400);

  var puedoDibujar = false;

  function dibujarCirculo(evento) {
    if(puedoDibujar) {
      var x = evento.pageX - pantalla.offsetLeft;
      var y = evento.pageY - pantalla.offsetTop;
      pincel.fillStyle = 'blue';
      pincel.beginPath();
      pincel.arc(x, y, 5, 0, 2 * 3.14);
      pincel.fill();
    }
  }

  pantalla.onmousemove = dibujarCirculo;

  function habilitarDibujar() {
    puedoDibujar = true;
  }

  function deshabilitarDibujar() {
    puedoDibujar = false;
  }

  pantalla.onmousedown = habilitarDibujar;
  pantalla.onmouseup = deshabilitarDibujar;

</script>
```

Para desarrollar este programa reutilizamos todo el código que ya teníamos para dibujar circunferencias, y solo adicionamos la lógica nueva de transformar nuestro mouse en un pincel.

Considera que la parte esencial está en saber si diseñamos o no en la pantalla mientras pasamos el mouse sobre la misma. Y sabemos que la condición está asociada a si el botón izquierdo del mouse está o no siendo presionado.

Siendo así declaramos una variable booleana llamada `puedoDibujar` que comienza como `false`. Esa variable será utilizada por la función `dibujarCirculo` para saber si debe o no diseñar.

En el código habrás visto tres eventos nuevos que estamos usando, ellos son `onmousemove`, `onmousedown` y `onmouseup`, donde el primero permite capturar el movimiento del mouse, el segundo sirve para ejecutar un código cuando el mouse está presionado y el tercero cuando el botón del mouse es soltado. Entonces, asociamos el primer evento a la función para dibujar circulo: `pantalla.onmousemove = dibujarCirculo`. Además, creamos dos funciones `habilitarDibujar` y `deshabilitarDibujar` y asociamos respectivamente cada función con los eventos del mouse `onmousedown` y `onmouseup`. Dentro de las funciones, lo que hacemos es atribuir la variable `puedoDibujar` para `true` cuando queremos `habilitarDibujar` y vuelve para `false` cuando queremos `deshabilitarDibujar`.

experimenta su funcionalidad, abre la consola de desarrollador en tu navegador para evaluar y analizar el comportamiento de las variables y funciones que creamos.

Ahora vamos al desafío:

El desafío de este ejercicio es poder cambiar el color del pincel, haciendo clic en una paleta de colores que vamos a tener en el extremo izquierdo superior de nuestro `canvas`, vamos a disponibilizar 3 colores para que el usuario pueda escoger el color que quiera en su pincel, los colores que usaremos serán el rojo, verde y azul (`red`, `green` y `blue`). El usuario tiene que visualizar algo así:



Por lo tanto, cuando el usuario haga clic en el cuadrado verde, el pincel diseñará círculos verdes, cuando haga clic en el rojo, diseñará círculos rojos, y así sucesivamente, recordando que el color inicial (default) es el azul. Un punto importante a tomar en cuenta es que debemos restringir el área de la paleta para no poder diseñar nada encima de ella.

Consejos antes del ejercicio

Ya lo dijo el emperador romano Julio Cesar "Divide y reinarás", divide el problema en problemas menores e intenta resolverlo por etapas: