Curso de Computación: Codificando con JavaScript

Tariq Rashid
Traducción y adaptación de Mario del Solar Moraga[1]

[1] *Profesor del Colegio Homeduca de Colina.*

24 de marzo de 2024

En este curso aprenderemos a codificar, es decir, a usar computadores, tablets y celulares para crear programas. Usaremos sitios web como www.code.org y www.codeguppy.com para codificar (o sea, programar). Si estás en algún nivel avanzado en programación puedes avanzar independientemente con el objetivo de crear un video juego.

1. Introducción

En este curso aprenderás JavaScript, el idioma más usado en el mundo para programar. Primero exploraremos las actividades del sitio www.code.org. Te sugiero comenzar con esta actividad https://hourofcode.com/mchoc. Luego de que termines dicha actividad usaremos codeguppy para programar.
A continuación te presento un curso de JavaScript desde cero creado por el profesor Tariq Rashid https://codeguppy.com/download.html y adaptado por mi.

2. Comenzamos con un círculo

Entra a codeguppy, ve al botón verde que dice CODE NOW y escribe:

```
circle(300, 300, 200);
```

Modifica los valores, observa los cambios y responde:
El primer valor (300) es para
El segundo valor (300) es para
El tercer valor (200) es para

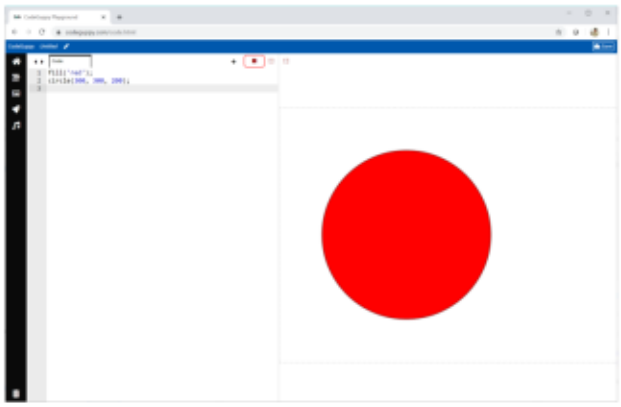

Figura 1: *Cambiando el color de una figura.*

2.1. Cambiamos el color del círculo

Escribe las siguiente líneas:

```
fill('red');
circle(300, 300, 200);
```

Puedes cambiar el color escribiendo los nombres de los colores en inglés. Por ejemplo *blue, green, black, white, grey, violet, pink.*

2.2. Dibujamos cuadrados

Escribe las siguientes líneas:

```
fill('red');
square(300, 300, 200);
```

Cambia su color, ubicación y tamaño.

2.3. Círculos y cuadrados juntos

Escribe lo siguiente:

```
fill('red');
square(300, 300, 200);
circle(300, 300, 200);
```

¿Qué sucedió? Ahora eres libre de modificar y crear nuevas líneas. Intenta con este tipo de líneas.

```
fill('red');
circle(300, 300, 200);
circle(300, 300, 150);
circle(300, 300, 100);
circle(300, 300, 50);
```

O éstas:

```
 fill('yellow');
circle(400, 300, 200);
fill('green');
circle(200, 200, 150);
fill('blue');
circle(150, 400, 100);
fill('red');
circle(300, 300, 50);
```

2.4. Cambiando de lugar las figuras

Ahora usa estas líneas como inspiración para dibujar algo que te sorprenda.

```
circle(200, 300, 200);
circle(300, 300, 200);
circle(400, 300, 200);
circle(500, 300, 200);
```

3. Números aleatorios

3.1. Tamaños aleatorios

Escribe lo siguiente:

```
circle(400, 300, 200);
circle(400, 300, 150);
circle(400, 300, 100);
circle(400, 300, 50);
```

Ahora hay cuatro círculos *concéntricos* ubicados en el centro de la tela. La diferencia de esos círculos es el *radio*. Ahora vamos a modificar un poco esas líneas para que sea el computador quien escoja *aleatoriamente* los tamaños de las figuras. Veamos:

```
noFill();
stroke('blue');
circle(400,300,randomNumber(200));
circle(400,300,randomNumber(200));
circle(400,300,randomNumber(200));
circle(400,300,randomNumber(200));
```

El comando *noFill* sirve para dejar en blanco los círculos y el comando *stroke('blue');* es para dar el color al conjunto de figuras.

3.2. Colores Aleatorios

Hagamos que nuestra computadora elija colores, no solo tamaños. Eche un vistazo a este código:

```
fill('orange');
circle(400, 300, 300);
fill('yellow');
circle(400, 300, 200);
```

Ejecute el código y verá que dibuja un pequeño círculo amarillo sobre un círculo naranja grande. Para ello necesitamos una lista de colores para elegir. Para codificar una lista de cosas las ponemos entre corchetes []. Aquí hay una lista de tres colores:

```
fill('orange');
circle(400, 300, 300);
fill( random( ['red', 'purple', 'green'] ) );
circle(400, 300, 200);
```

Ejecute el código nuevamente varias veces. ¿Puedes conseguir todos los colores?

3.3. Desafío!

Usa lo que has aprendido para dibujar cuatro círculos donde:

- la ubicación es un lugar aleatorio en el lienzo
- el tamaño es un número aleatorio
- el color se elige aleatoriamente de una lista de colores

4. Variables simples

Qué haremos En este proyecto vamos a:

- aprender cómo las variables pueden recordar números
- ver cómo las variables pueden ser útiles

4.1. Tres círculos

Comencemos el nuevo programa con tres círculos simples.

```
circle(200, 200, 25);
circle(200, 250, 25);
circle(200, 300, 25);
```

¿Puedes ver la diferencia entre estos tres círculos? La coordenada x permanece igual en 200. La coordenada y comienza en 200 y aumenta en 50 para cada nuevo círculo. Ejecute el código para ver qué sucede. ¡Ajá! Los tres círculos parecen una cadena. Están en una línea vertical porque solo cambia la coordenada y.

4.2. Dibuja la cadena en lugares aleatorios

¿Cómo dibujaríamos la línea de círculos en un lugar aleatorio?. Ya hemos utilizado la instrucción *randomNumber* para elegir una ubicación aleatoria en el lienzo.

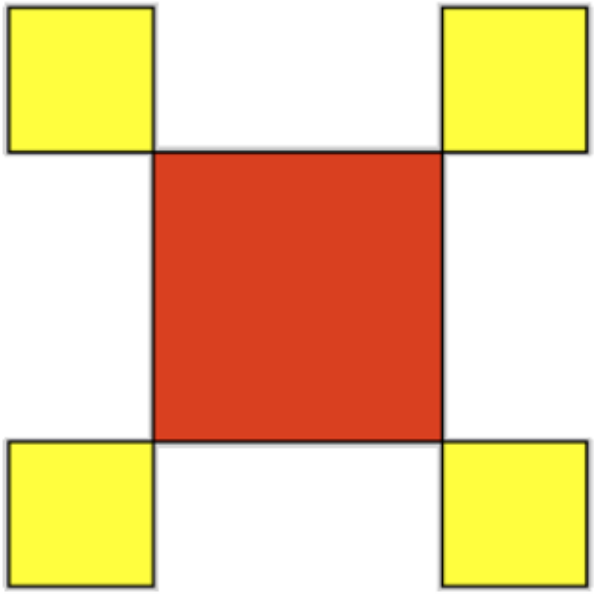

Figura 2: *Dibuja esta figura en JavaScript.*

Eche un vistazo a este código. Utiliza *randomNumber* para seleccionar números para las coordenadas *x* e *y*.

```
circle(random(800), randomNumber(600), 25);
circle(random(800), randomNumber(600), 25);
circle(random(800), randomNumber(600), 25);
```

Ejecute el código para comprobar que funciona.
Ese código no hizo lo que queríamos. ¿Puedes ver por qué?
No funciona porque todos los círculos están dibujados en ubicaciones aleatorias. Nuestro código anterior no hacía eso. Mire nuevamente ese código anterior:

```
circle(400, 300, 25);
circle(400, 350, 25);
circle(400, 400, 25);
```

Podemos ver que las coordenadas x no son totalmente aleatorias. se mantienen lo mismo para todos los círculos. También podemos ver que las coordenadas y no son totalmente aleatorias. Ellos aumentan en pasos de 50 desde el primer círculo. Eso significa que debemos elegir una ubicación aleatoria sólo para el primer círculo.
También debemos recordar esa ubicación para poder dibujar los otros dos círculos al lado. Podemos pedirle a nuestra computadora que recuerde un número usando una variable. Hablaremos de variables a continuación.

5. Usando Variables

Las variables son como cuadros en los que podemos poner números. La figura 3 muestra una variable llamada x. Puedes ver que estamos poniendo el número 10 dentro. Siempre que usemos x en nuestro código, nuestra computadora buscará dentro del cuadro y usará el número 10. Eche un vistazo a este nuevo código. ¿Puedes averiguar qué hace?

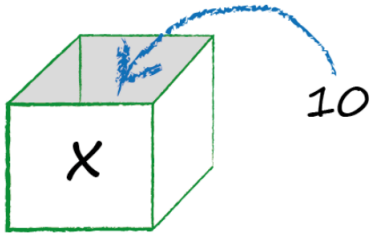

Figura 3: *Ejemplo de variable.*

```
var x = randomNumber(800);
var y = randomNumber(600);
circle(x, y, 25);
circle(x, y + 50, 25);
circle(x, y + 100, 25);
```

Estamos creando una nueva variable x y colocando un número aleatorio entre 0 y 800 dentro de ella. No podemos adivinar qué número será. Sólo sabemos que estará entre 0 y 800. Estamos creando otra variable llamada y y poniendo un número aleatorio entre 0 y 600 dentro de él.
¿Qué crees que hará el primer comando circular círculo(x, y, 25)? ¿Qué crees que hará el comando del segundo círculo círculo (x, y + 50, 25)? Pruebe el código usted mismo y vea qué sucede. Deberías obtener una cadena de tres círculos, en algún lugar del lienzo.
La primera instrucción de círculo (x, y, 25) utiliza los números aleatorios que se colocaron dentro de las variables x e y. La siguiente instrucción de círculo (x, y + 50, 25) también usa los mismos números que se colocaron dentro de x e y. Ejecute el código nuevamente. La cadena de círculos estará en un lugar diferente. Esto se debe a que la ubicación del primer círculo se elige al azar. ¡Buen trabajo por haber llegado tan lejos!

6. Progresando

6.1. Funciones simples

En este proyecto vamos a:

- aprender a empaquetar código útil como una función
- ver cómo las funciones pueden ser útiles

En el último proyecto aprendimos a usar variables para dibujar un grupo de formas en cualquier lugar del lienzo. Echa un vistazo a esta imagen que muestra una flor hecha de círculos. ¿Dónde está el centro del círculo amarillo? Puedes ver en la imagen que está en (x, y). Estamos usando letras en lugar de números. ¿Dónde está el círculo rojo inferior? Está un poco más abajo del círculo amarillo. Al mirar la imagen, puedes ver que está en (x, y + 50).
Si el centro del círculo amarillo está en (x, y), entonces podemos calcular los centros. de todos los círculos rojos:

- el círculo rojo superior está en (x, y - 50)
- el círculo rojo inferior está en (x, y + 50)
- el círculo rojo de la derecha está en (x + 50, y)
- el círculo rojo de la izquierda está en (x - 50, y)

Si configuramos x en 100 y y en 100, entonces la flor debería dibujarse cerca de la parte superior izquierda del lienzo, como antes. Si establecemos x en 400 y y en 300, entonces la flor debería dibujarse en el medio del lienzo. Podemos establecer xey en cualquier ubicación del lienzo, y la flor se dibujará allí. Aquí hay un código para dibujar esa flor en (x, y).

```
var x = 400;
var y = 300;
fill('yellow');
circle(x, y, 50);
fill('red');
circle(x, y - 50, 25);
circle(x + 50, y, 25);
circle(x, y + 50, 25);
circle(x - 50, y, 25);
```

Al comienzo del código configuramos x en 400 e y en 300. Puedes ver las instrucciones para los círculos amarillo y rojo, usa x e y. en lugar de números. Cuando nuestra computadora vea la x en el círculo (x, y, 50), tomará el número dentro de la variable x y lo usará. Lo mismo sucederá con y. Entonces el círculo (x, y, 50) se convertirá en círculo (400, 300, 50). Ejecute el código para comprobar que dibuja la flor en el medio del lienzo en (400.300).

6.2. Dibuja la flor en un lugar aleatorio

Cambiemos el código para dibujar la flor en un lugar aleatorio del lienzo. Solo necesitamos cambiar los números en los que están configurados x e y. En lugar de que nosotros elijamos un número, dejamos que nuestra computadora elija uno por nosotros.

```
var x = randomNumber(800);
var y = randomNumber(600);
```

Ejecute el código para comprobar que la flor esté dibujada en otro lugar.

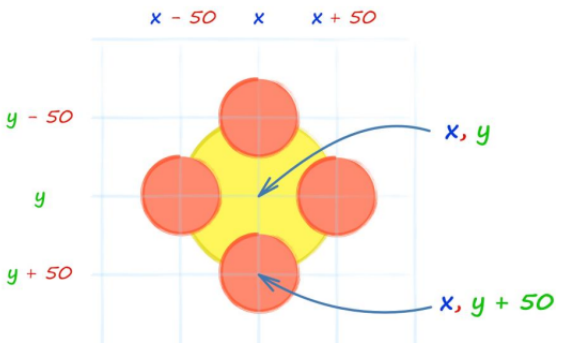

Figura 4: *Una flor hecha de círculos.*

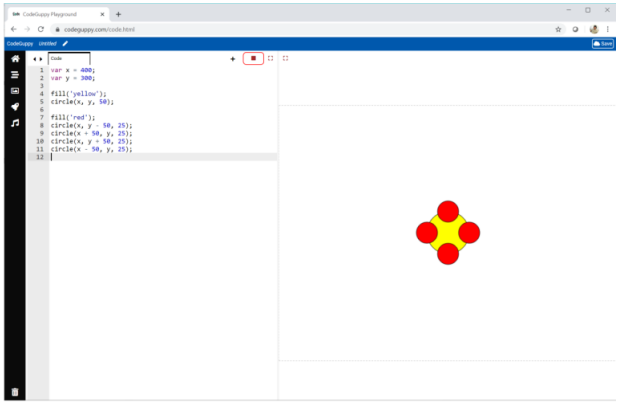

Figura 5: *Bien! Ha funcionado.*

6.3. Dibujar muchas flores

Hagamos algo nuevo y emocionante: ¡dibujemos 5 flores! ¿Cómo dibujamos 5 flores? Podríamos escribir todo ese código nuevamente 5 veces, pero sería muy largo y aburrido. Sería mejor si le enseñáramos a nuestra computadora a dibujar una flor solo una vez y luego le pidiéramos que la dibujara muchas veces. Sería como una receta de tarta de chocolate. Lo escribimos una vez y lo usamos muchas veces. También podemos escribir recetas en código. Se llaman *funciones*. Eche un vistazo a este código.

```
function my_flower()
{
var x = randomNumber(800);
var y = randomNumber(600);
fill('yellow');
circle(x, y, 50);
fill('red');
circle(x, y - 50, 25);
circle(x + 50, y, 25);
circle(x, y + 50, 25);
circle(x - 50, y, 25);
}
```

Si miras con atención, verás que es el mismo código que ya escribimos para dibujar una flor en un lugar aleatorio del lienzo. La única diferencia es que tenemos la función my_flower() { en la cima, y } en el fondo. Lo que hace este nuevo código es crear una receta o función llamada my_flower. Las instrucciones de la receta están dentro de las llaves {y}. Escriba este código para crear la función my_flower. Si ejecuta este código obtendrá un lienzo vacío.

Esto se debe a que creamos la receta my_flower, pero aún no la hemos usado. Para usarlo, simplemente escribimos el nombre de la función my_flower fuera de la función.

```
my_flower();
```

No olvide los corchetes vacíos () después del nombre de la función. Ejecute el código para comprobar que el uso de nuestra nueva instrucción my_flower realmente

funciona.
My_flower se cae del borde del lienzo. El tuyo estará en alguna parte más en el lienzo.

6.4. Iterando funciones

¿Qué crees que pasará si escribimos cinco instrucciones my_flower una tras otra?

```
my_flower();
my_flower();
my_flower();
my_flower();
my_flower();
```

Inténtalo!

6.5. Desafío

Intenta dibujar 10 flores de círculos de manera que los colores sean aleatorios.

7. Repitiendo instrucciones

En este proyecto vamos a:

- aprender a repetir instrucciones
- Vea cómo la repetición puede hacer mucho trabajo sin escribir mucho código.

7.1. Comenzando con una Función simple

Comencemos este proyecto con una función realmente simple que dibuja un pequeño círculo lleno de un color elegido al azar de una lista. Eche un vistazo a este código. Hablaremos de ello a continuación.

```
function bubble()
{ var x = randomNumber(200, 600);
var y = randomNumber(200, 400);
var r = randomNumber(15, 50);
fill( random(['pink', 'yellow', 'lightgreen']) );
circle(x, y, r);
}
```

Puedes ver que hemos elegido números aleatorios para x e y. Ésta será la ubicación del círculo.
¿Viste que la instrucción RandomNumber tiene dos números entre paréntesis? Antes sólo tenía uno. La instrucción ahora elige un número aleatorio que se encuentra entre esos dos números. Entonces RandomNumber(200, 600) elige un número entre 200 y 600. También puedes ver que r se establece en un número aleatorio entre 15 y 50. Este será el radio del círculo. También elegimos un color aleatorio de una lista de rosa, amarillo y verde claro. Llamemos a esta función de burbuja cinco veces. Llamar a una función es lo que dicen los codificadores cuando usan una función.

Ejecute el código para ver cinco burbujas de colores.

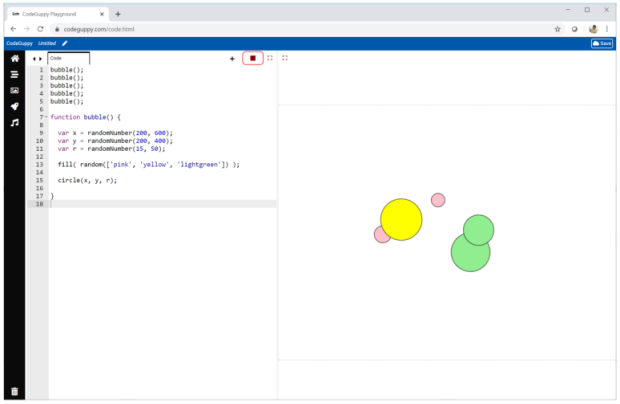

Figura 6: *Estas burbujas parecen muy dulces!*

7.2. Dibujando 20 burbujas

Ese dibujo necesita más burbujas. Dibujemos 20. Podríamos repetir la instrucción de la burbuja 20 veces, pero eso resultaría agotador. Debe haber una mejor manera. ¡Hay una mejor manera! Las computadoras son muy buenas para repetir cosas y no se aburren. Eche un vistazo a este nuevo código:

```
repeat(20, bubble);
```

¿Puedes adivinar qué hace? La instrucción de repetición repite una función. Le decimos qué función repetir. Aquí le hemos dicho que repita la función de burbuja. Ese número 20 le dice a repetir que llame a la función burbuja 20 veces. Reemplace sus cinco llamadas a la burbuja con esta única instrucción de repetición.
Cambia tu código para dibujar 200 burbujas. ¡Sí, 200! Cambié la función de burbuja para hacerlas más pequeñas. También aumenté el rango de números entre los que se eligen x e y, de modo que se use más lienzo para dibujar.

```
var x = randomNumber(100, 700);
var y = randomNumber(100, 500);
var size = randomNumber(10, 25);
```

Esto es lo que dibuja mi código:

Figura 7: *Burbujas aleatorias.*

7.3. Desafío!

Si miras de cerca la figura 8 puedes ver que está formado por muchas pequeñas gotas. ¡Hay 200 gotas, pero no es necesario contarlas! Cada gota tiene un pequeño círculo amarillo encima de un círculo más grande rojo, verde o azul. ¿Puedes escribir código para hacer un dibujo similar?

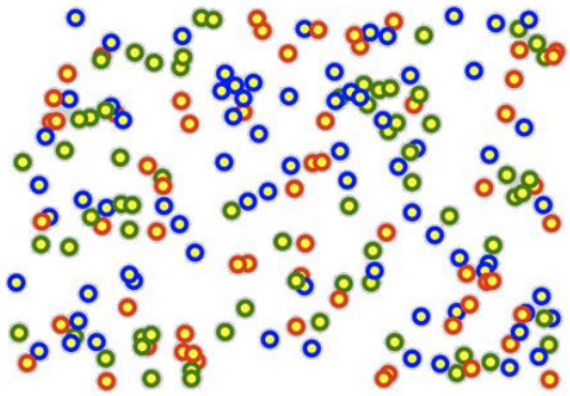

Figura 8: *Desafío en JavaScript.*

8. Más Funciones

En este proyecto vamos a:

- aprender a pasar información a funciones
- vea cómo esto hace que las funciones sean aún más útiles

8.1. Dibujando flores en el lugar que queremos

Ya hemos aprendido cómo escribir una función-código que podemos usar una y otra vez. Aquí está la función my_flower que escribimos para dibujar una flor.

```
function my_flower()
{ var x = randomNumber(800);
var y = randomNumber(600);
fill('yellow');
circle(x, y, 500);
fill('red');
circle(x, y - 50, 25);
circle(x + 50, y, 25);
circle(x, y + 50, 25);
circle(x - 50, y, 25);
}
```

No sabemos dónde dibujará una flor. Esto se debe a que el centro de la flor (x, y) se elige al azar.
Sería bueno si pudiéramos decirle a nuestra función exactamente dónde dibujar la flor.
Eso significa que necesitamos una manera de decirle a nuestra función my_flower cuáles deberían ser x e y. Eche un vistazo a este nuevo código. Es igual que antes pero con algunos pequeños cambios.
¿Puedes ver las diferencias?

```
function my_flower(x,y)
{
fill('yellow');
circle(x, y, 50);
fill('red');
circle(x, y - 50, 25);
circle(x + 50, y, 25);
circle(x, y + 50, 25);
circle(x - 50, y, 25);
}
```

Hay dos diferencias:

- el nombre de la función ahora tiene (x, y) entre paréntesis.
- Eliminamos el código para elegir x e y al azar.

Cambiar el nombre de my_flower() a my_flower(x, y) significa que Ahora es necesario indicarle a la función qué x e y usar. Así es como le decimos a my_flower qué queremos que sean x e y:

```
my_flower(100, 200);
```

Esto llama a my_flower, como antes, pero pasa el número 100 a la función para usarla como x. También pasa 200 para usarse como y.
¡Vamos a intentarlo! Así es como se ve mi código y el resultado, una flor dibujada en (100, 200). Le hemos

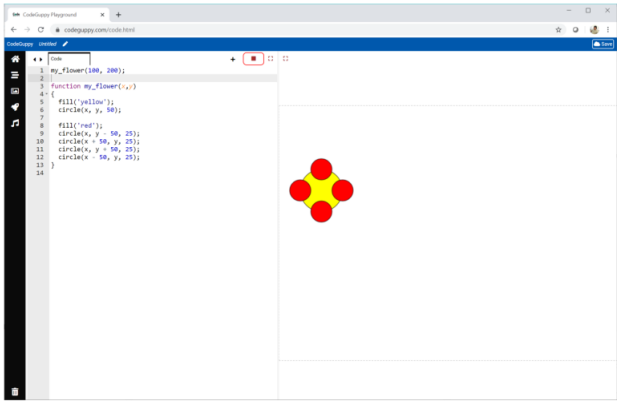

Figura 9: *Flor dibujada en (100,200)*

dicho a la función dónde dibujar la flor pasándole información. Esto se llama pasar parámetros.
La palabra *parámetros* significa la información que le das a una función cuando la usas. Podemos decir que la función my_flower(x, y) toma 2 parámetros, x e y.
Pruebe su nueva función my_flower(x, y) con diferentes parámetros. Intenta dibujar varias flores con diferentes parámetros para cada una. Mira si puedes hacer un patrón de flores. Aquí hay un patrón que hice.
(Fin de la primera parte)