Programación Creativa con p5.js

Aprende a dibujar e interactuar en la web

¿Qué es p5.js?

p5.js es una librería de JavaScript pensada para que cualquiera pueda dibujar, animar e interactuar en una página web.

Está inspirada en *Processing*, una herramienta creada en el MIT para que artistas aprendieran a programar.

¿Por qué nos gusta?

- No hay que instalar nada
- En 5 líneas ya se ve algo
- Matemáticas + arte + juego

editor.p5js.org

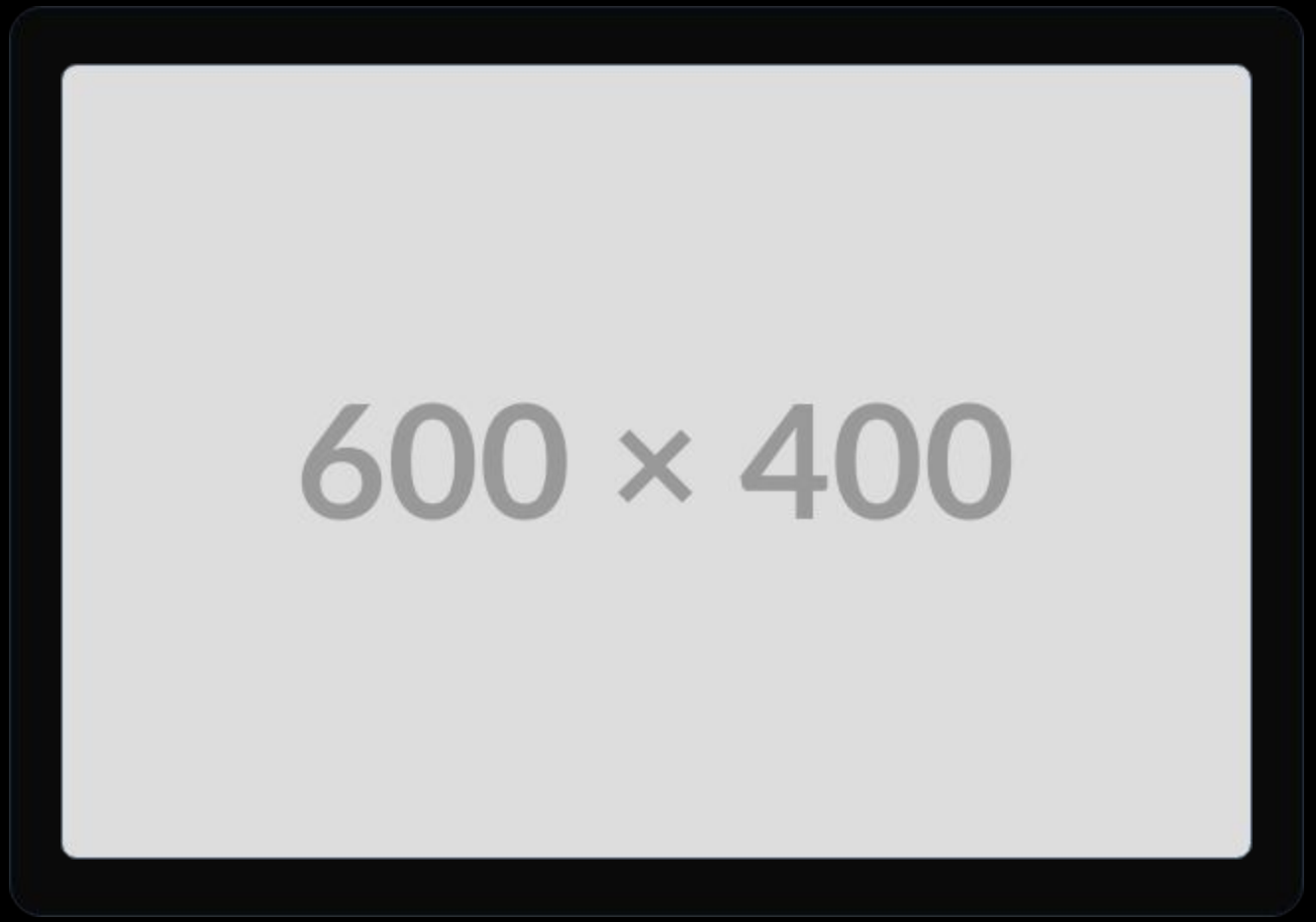
El editor online

No necesitas instalar nada. Abre el navegador y entra a:

https://editor.p5js.org

Verás tres zonas:

- **Editor de código** (a la izquierda): donde escribimos
- **Lienzo** (a la derecha): donde aparece el dibujo
- **Botón Play** (▶): ejecuta el código



Truco: cada vez que cambies algo, dale a Play.

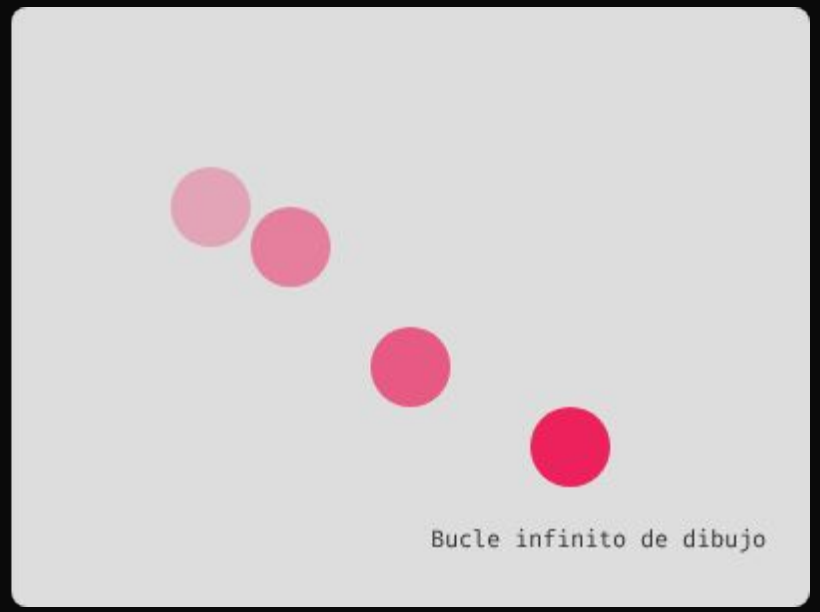
NPara guardar, crea una cuenta.

Las dos funciones mágicas

Todo sketch de p5.js tiene esta estructura:

```
function setup() {
  createCanvas(600, 400); // tamano del
lienzo
  background(220);        // color de fondo
}

function draw() {
  ellipse(mouseX, mouseY, 40, 40); // un
circulo
}
```



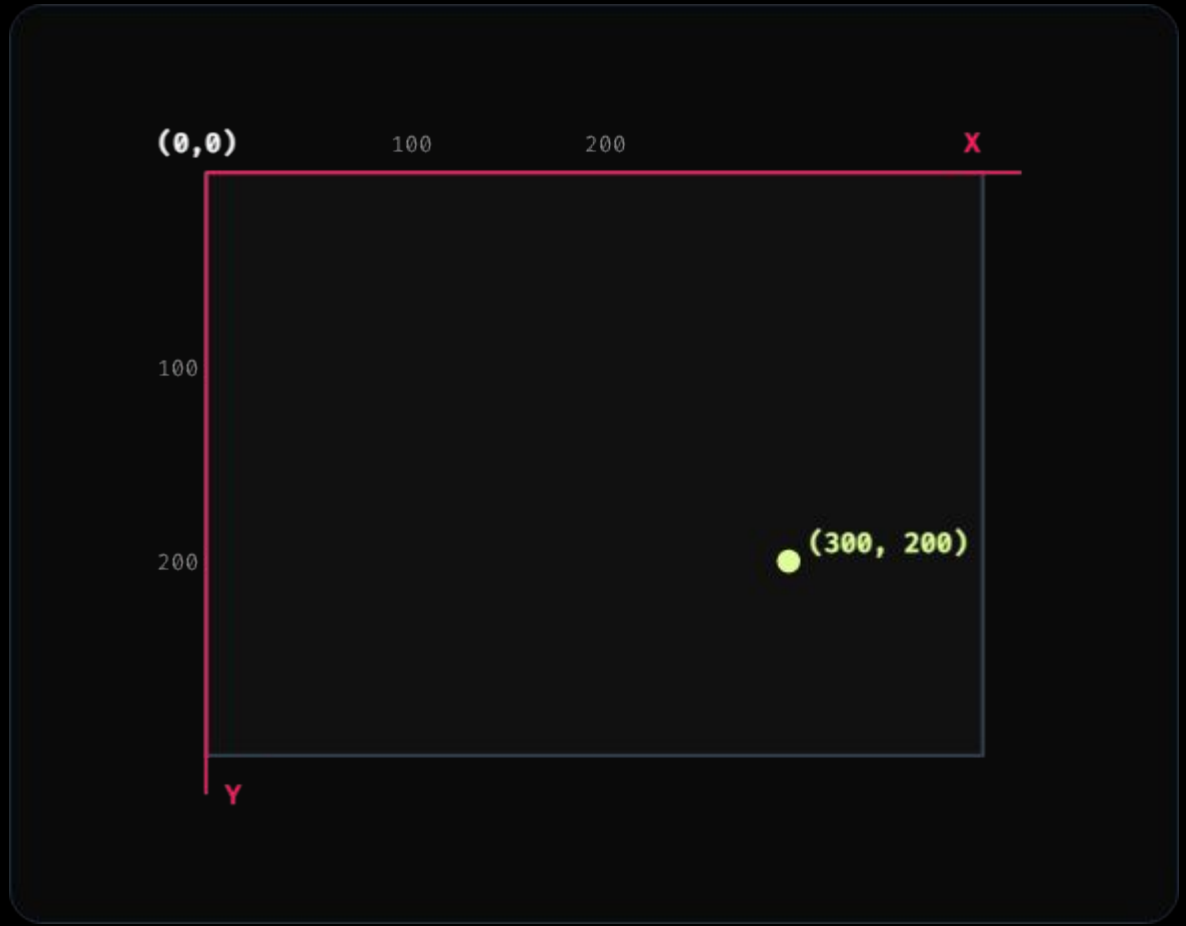
- setup() se ejecuta **una sola vez** al empezar
- draw() se ejecuta **60 veces por segundo**, sin parar

Sistema de Coordenadas

El origen **(0,0)** está en la **esquina superior izquierda**.

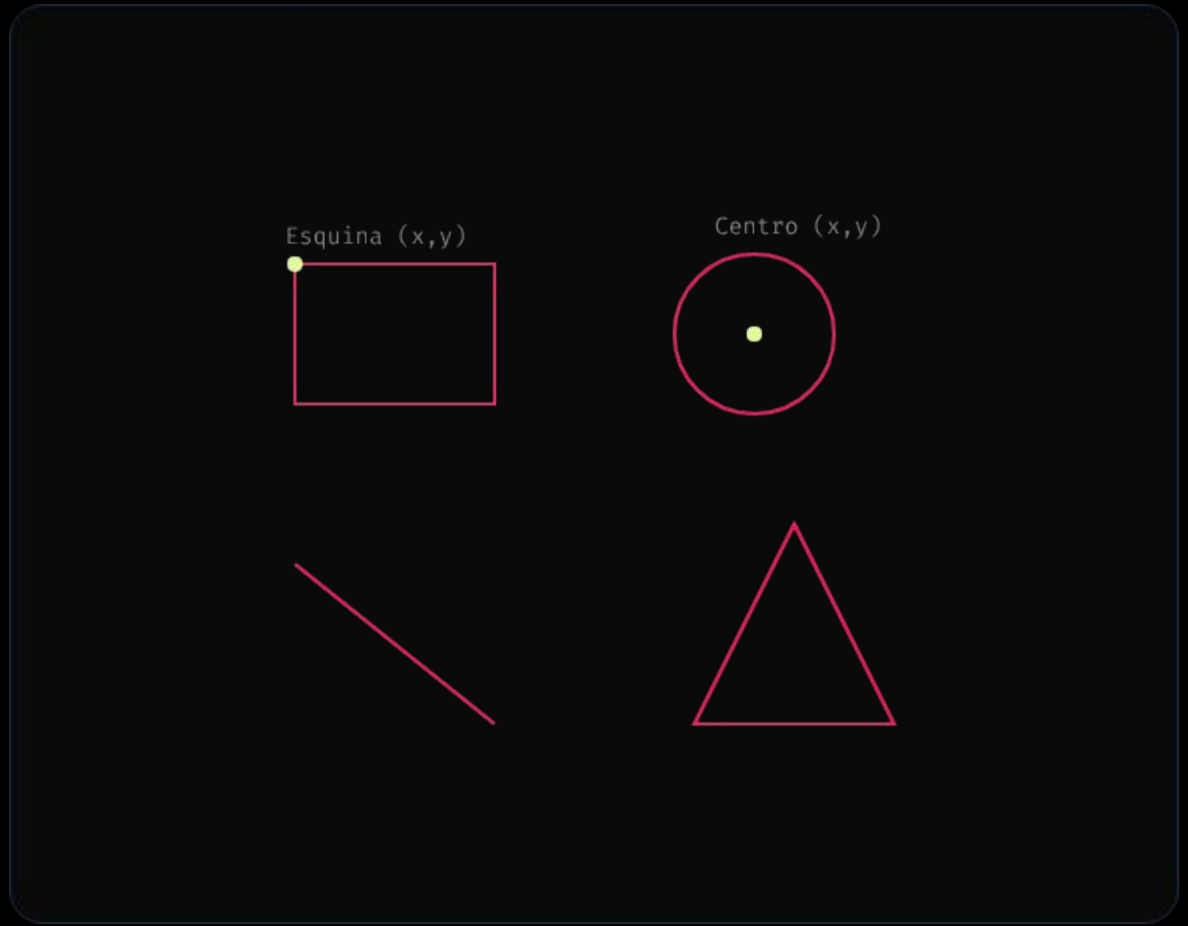
- Eje **X**: horizontal (crece a la derecha)
- Eje **Y**: vertical (crece hacia **abajo**)

Ejemplo: En un lienzo de 400x300, el centro es (200, 150).



Formas básicas

```
circle(x, y, d);          // x,y: centro
rect(x, y, ancho, alto);// x,y: esquina
line(x1, y1, x2, y2);
triangle(x1,y1, x2,y2, x3,y3);
point(x, y);
```



Alineación por defecto:

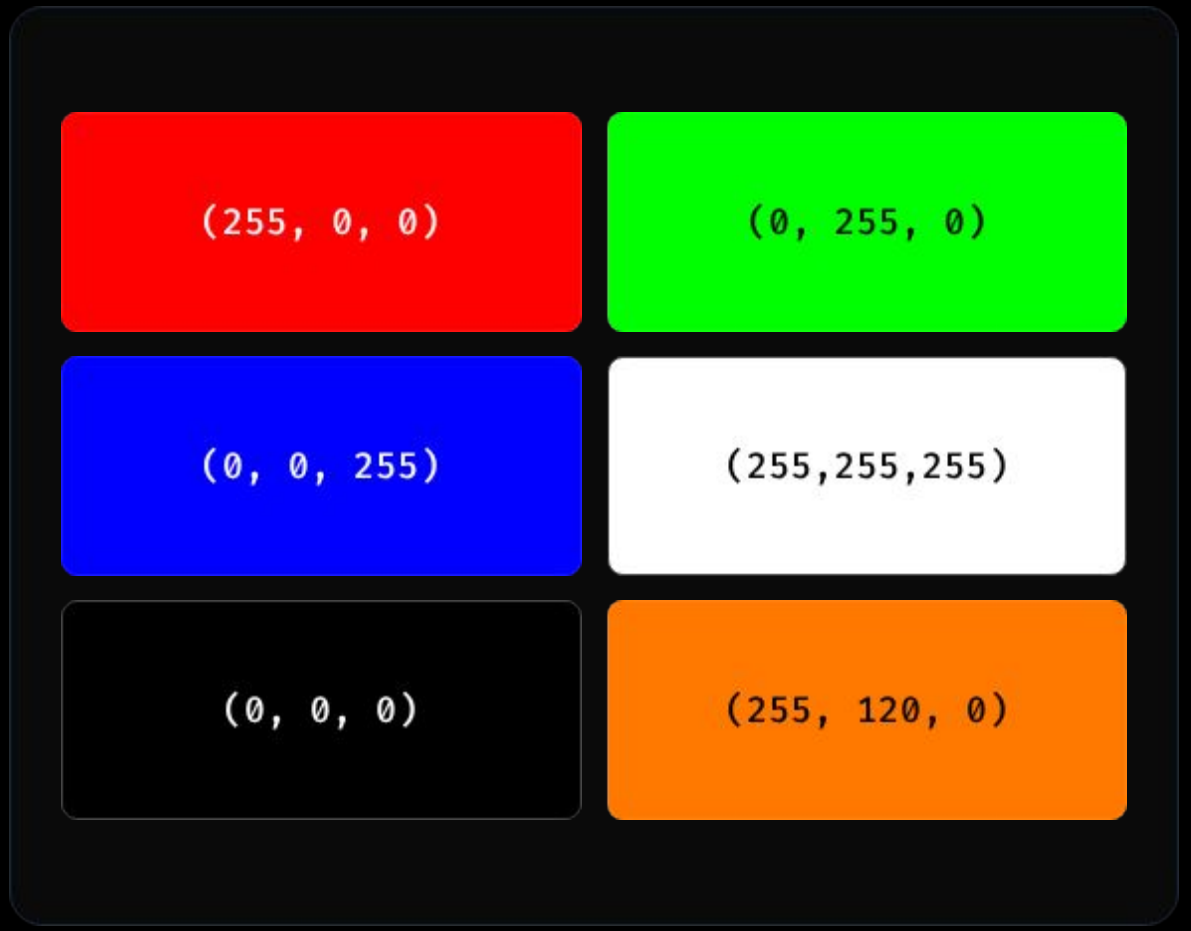
- rect se dibuja desde su esquina sup. izq.
- circle se dibuja desde su centro.

Espacio de color RGB

Cada color se define mediante tres componentes **(R,G,B)**:

```
fill(255, 120, 0); // relleno naranja
circle(200, 150, 80);
```

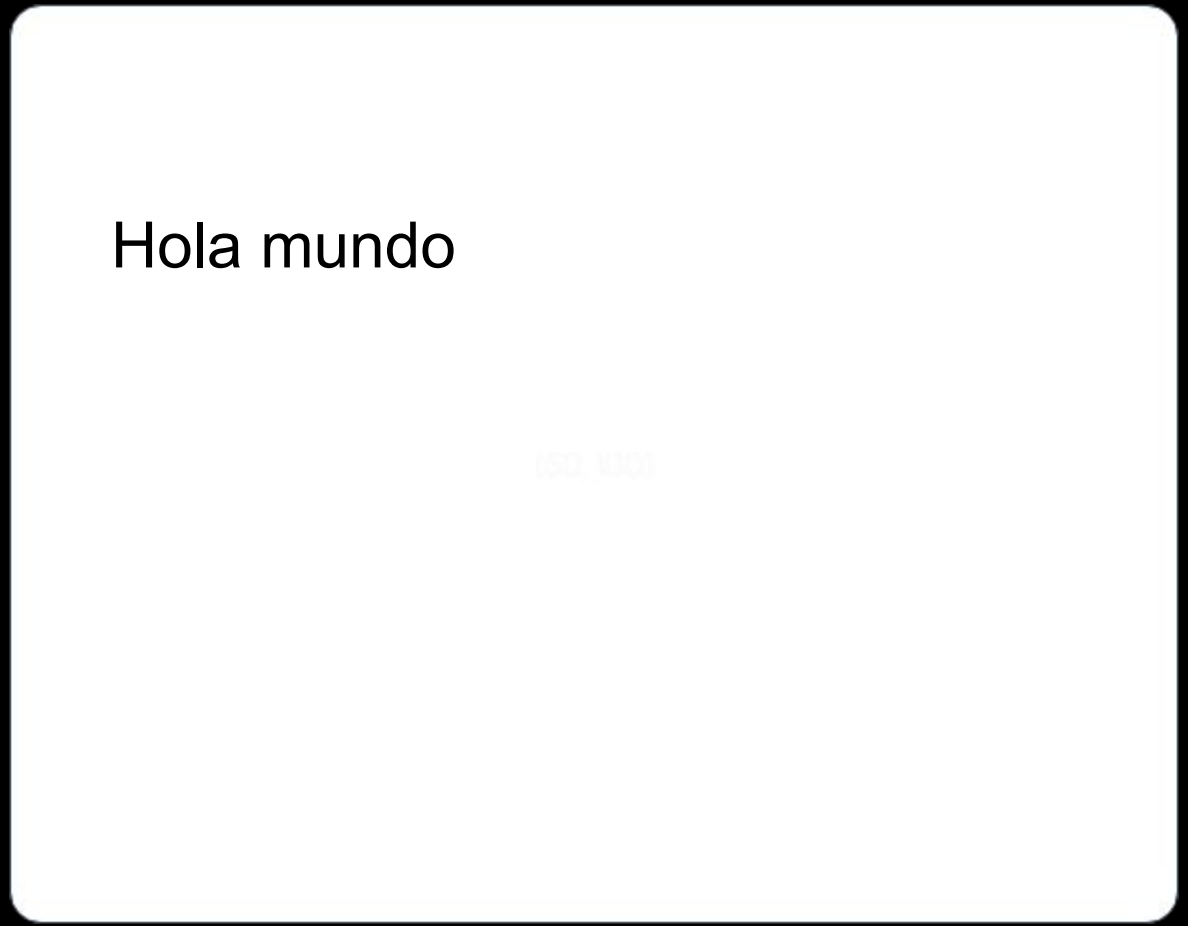
Rango: de 0 a 255 para cada canal.



Escribir texto

La función text() dibuja cadenas de caracteres.

```
background(255);
textSize(32);
fill(0);
text("Hola mundo", 50, 100);
```



Funciones de formato:

- textSize(px): Define el tamaño de la letra.
- textAlign(Horizontal, Vertical): Alinea el texto (CENTER, LEFT...).
- textFont(font): Cambia el tipo de letra.

Variables y Tipos de Datos

Las variables son cajas con nombre. Almacenan información para usarla después. En JS, usamos let:

```
let x = 100;         // Número
let nombre = "Arte"; // Cadena
let activo = true;   // Booleano
```

Las variables nos permiten controlar el tamaño, posición y color de forma dinámica.

Truco: Usa ***let*** para declarar tus variables antes de ***setup()*** para que sean globales.

Variables: Ámbito (Scope)

Variables Locales (viven dentro de una función)

```
function setup() {
  let radio = 50;
}
console.log(radio);
// Imprime: undefined
```

¿Por qué undefined?
Una variable local sólo existe dentro de las llaves { } donde se definió. Al llamarla fuera, el programa no sabe qué es.

Variables Globales (viven en todo el código)

```
let ancho = 600;
let alto = 400;

function setup() {
  createCanvas(ancho, alto);
}

function draw() {
  // ancho es visible aquí también
}
```

Declaradas fuera de cualquier función. Son visibles en todo el programa.

Variables Integradas

Ratón: mouseX, mouseY

Posición actual del cursor en el lienzo.

Lienzo: width, height

Dimensiones actuales definidas en createCanvas.

Tiempo: frameCount

Contador de fotogramas desde el inicio.

Click: mouseIsPressed

true si mantienes pulsado el ratón.

Aleatoriedad: random(a, b)

Devuelve un número al azar entre a y b.

Transparencia: fill(R, G, B, Alpha)

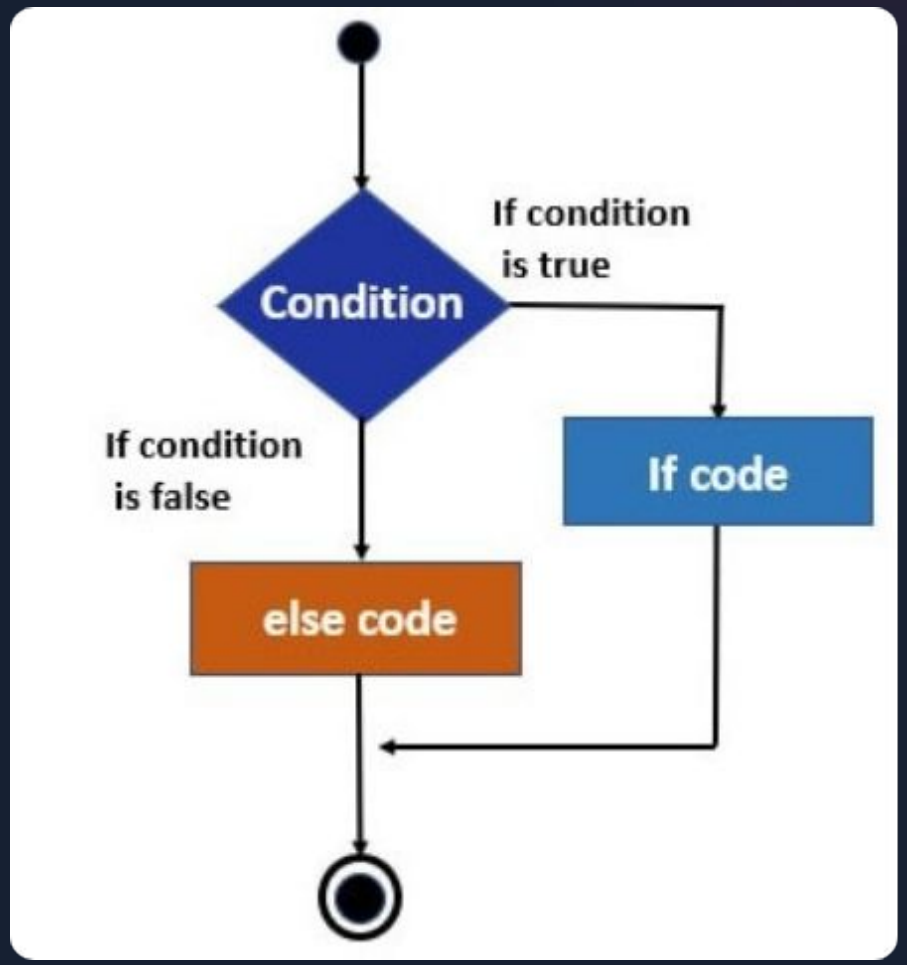
El 4o valor es opacidad (0 invisible, 255 sólido).

Lógica: La Condición if

Permite tomar decisiones. Si la condición es **verdadera**, se ejecuta el código.

```
if (x_pos > width) {
  x_pos = 0; // Teletransporte al inicio
} else {
  x_pos += 5; // Seguir avanzando
}
```

Símbolos clave: == (igual), != (distinto), >, <, && (Y), || (O).



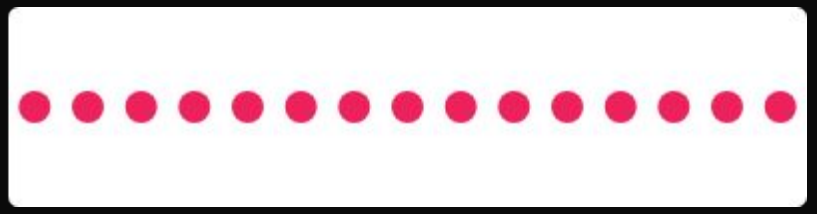
Control de flujo lógico.

Bucles for

Los bucles permiten repetir instrucciones.

```
function draw() {
  background(255);
  for (let x = 20; x ⩽ 580; x += 40) {
    circle(x, 200, 25);
  }
}
```

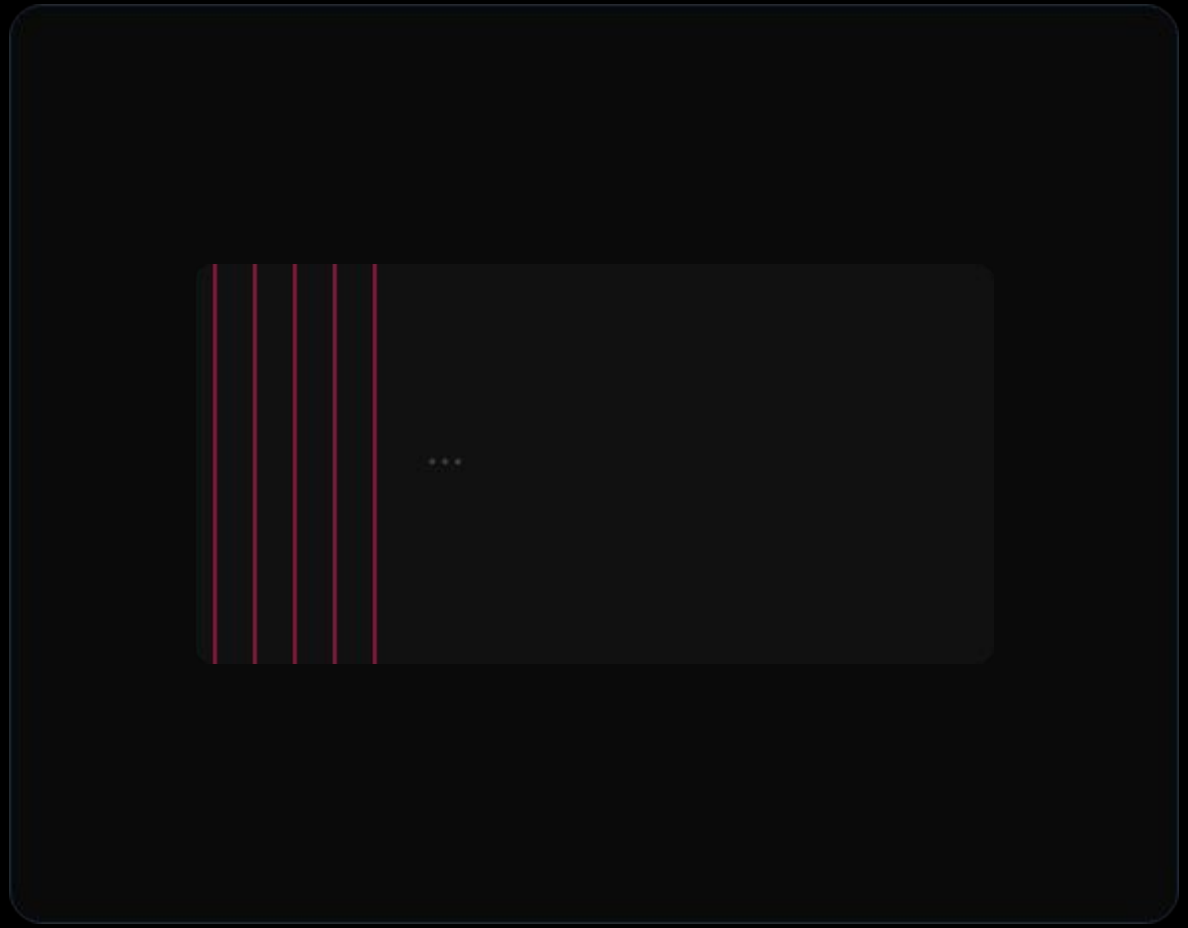
Una sola instrucción genera muchos círculos.



Bucle while

Se repite **mientras** la condición sea verdadera.

```
let x = 0;
while (x < width) {
  line(x, 0, x, height);
  x += 20;
}
```



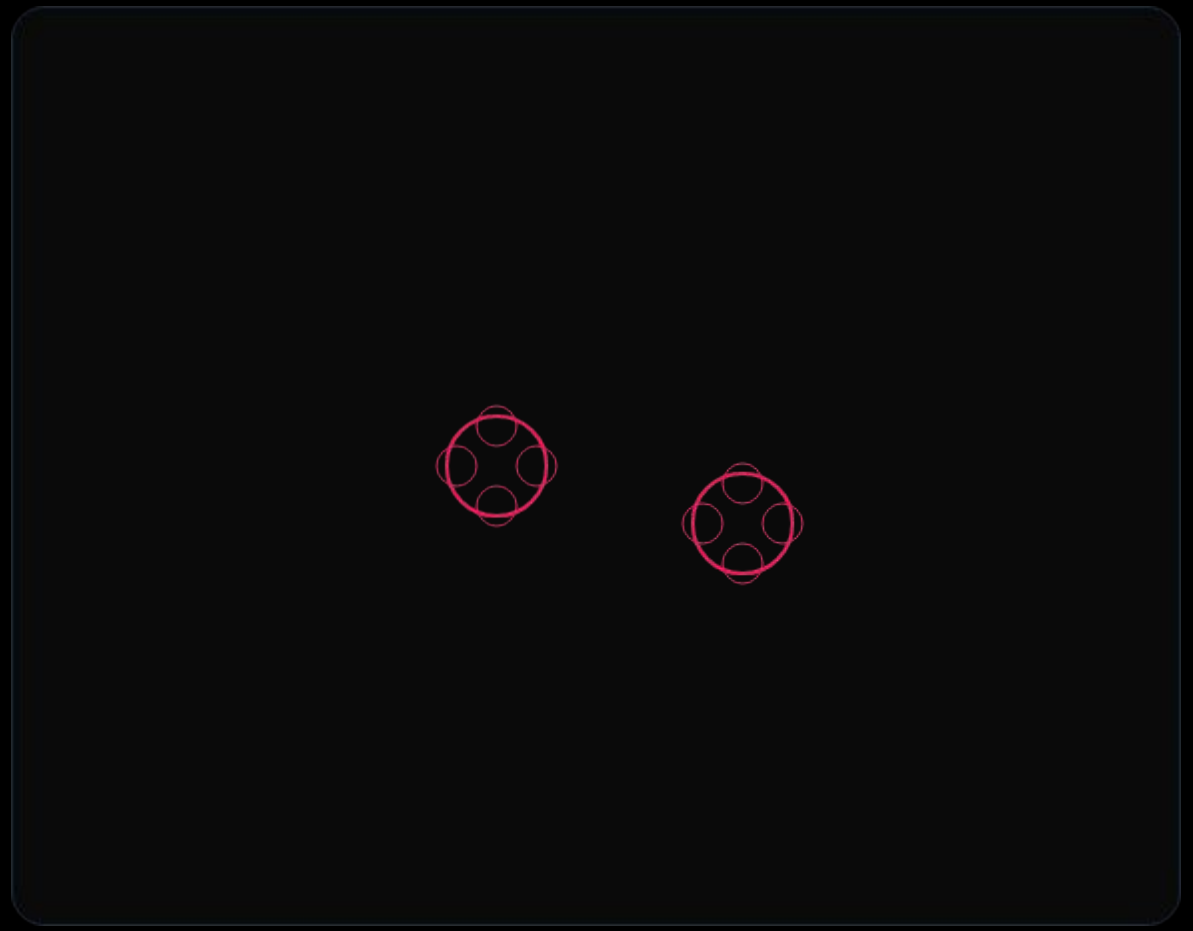
Cuidado: Si la condición siempre es verdadera, el programa se colgará.

Tus propias funciones

Las funciones permiten reutilizar código y organizar mejor tus ideas.

```
function flor(x, y) {
  circle(x, y, 50);
  circle(x-20, y, 20);
  circle(x+20, y, 20);
  circle(x, y-20, 20);
  circle(x, y+20, 20);
}

function draw() {
  background(255);
  flor(150, 150);
  flor(300, 220);
}
```



Arrays

Los arrays almacenan varios valores en una sola lista.

```
let xs = [50, 120, 250, 330];

function draw() {
  background(255);
  for (let x of xs) {
    circle(x, 200, 30);
  }
}
```

Métodos básicos con Arrays

Cómo manipular tus listas de datos:

- push(valor): Añade un elemento al **final** de la lista.
- pop(): Elimina el **último** elemento de la lista.
- length: Propiedad que te dice **cuántos** elementos hay en total.

```
let lista = [10, 20];
lista.push(30);   // [10, 20, 30]
lista.pop();      // [10, 20]
let n = lista.length; // n vale 2
```

```
.push( ) → entra al final

.pop( ) ← sale el último

.unshift( ) ← sale el primero
```

Métodos útiles de arrays

Métodos útiles para procesar datos:

```
let xs = [20, 60, 100, 140];

xs.forEach(function(x){
  circle(x, 100, 20);
});

let dobles = xs.map(x ⇒ 2*x);
let grandes = xs.filter(x ⇒ x > 80);
```

dobles:: [40, 120, 200, 280]

grandes:: [100, 140]

Image Sources

https://assets.streamlinehq.com/image/private/w_300,h_300,ar_1/f_auto/v1/icons/2/p5js-lwf4foxvens4rx28s7l75p.png/p5js-vwr8u89mec5rd5p2h7ble.png?_a=DATAiZAAZAA0

Source: www.streamlinehq.com