

Estructuras de Control de Repetición

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Bucle

Un **bucle** es una estructura de control que **repite instrucciones**.

Un bucle entonces nos permite repetir un bloque de instrucciones determinado hasta que se cumpla cierta condición.

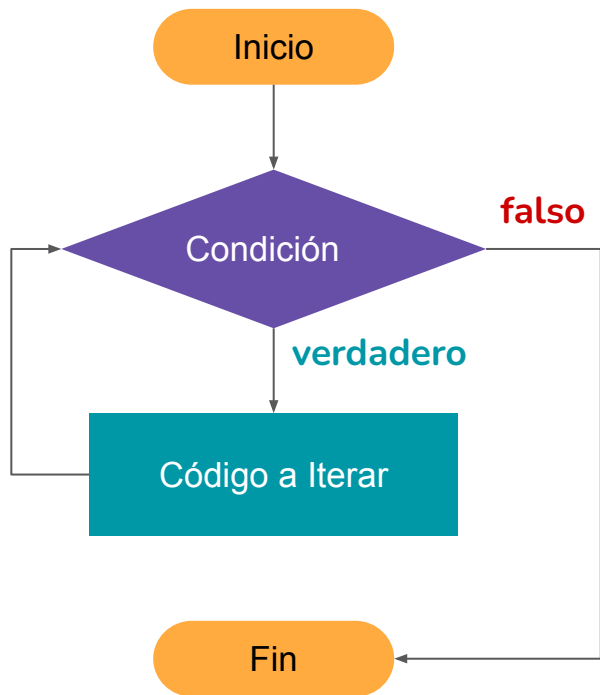
Cada repetición se suele llamar iteración.



Ciclo While

DEV.F
DESARROLLAMOS(PERSONAS);

dev



Ciclo While

La idea principal del ciclo while es: **MIENTRAS** se cumpla la condición **REALIZAR** estas acciones. Cuando la condición deje de cumplirse salimos del bucle y continúa el flujo del programa.

Muy importante: En el ciclo while la condición es lo primero que se evalúa, antes de ejecutar el código a iterar.

Sintaxis: Ciclo while

Usamos la palabra reservada **while**, seguida de la condición entre paréntesis **()** y finalmente colocamos el código que se repetirá entre llaves **{}**

```
while (condicion) {  
    // codigo a ejecutar  
}
```

Importante: Necesitamos en el código a iterar insertar una variable de control que nos permita salir eventualmente el ciclo while. En caso contrario nuestro programa se quedará ciclado “infinitamente”.

Ejemplo #1: Ciclo while

1. Imprimir los números del 1 al 10 en la consola.

```
let i = 0;

while( i <= 10 ){
  console.log(i);
  i++;
}
```

2. Imprimir la tabla de multiplicar del 5.

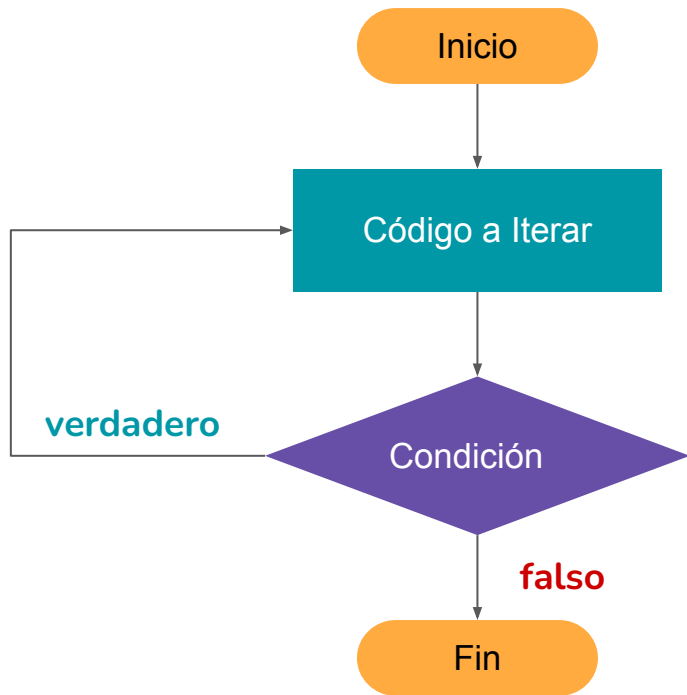
Ejemplo #2: Ciclo while

Guardar en una variable los valores introducidos. Si el usuario no introduce un valor, termina el ciclo.

```
let text = '';

let userInput;

while( !(userInput == '') ){
  userInput = prompt('Introduce un caracter');
  text += userInput;
}
```



Ciclo Do While

Variante del ciclo While puro, con la diferencia que la primera vez siempre se ejecuta el código y posteriormente evalúa la condición para ver si se vuelve a ejecutar.

Sintaxis: Ciclo do while

Usamos la palabra reservada **do**, seguido del el código que se repetirá entre llaves **{}**, seguido de la palabra reservada **while** y finalmente la condición a evaluar en cada iteración entre paréntesis **()**.

```
do {  
    // código a ejecutar  
}  
while (condicion);
```

Ejemplo: Ciclo do while

Guardar en una variable los valores introducidos. Si el usuario no introduce un valor, termina el ciclo.

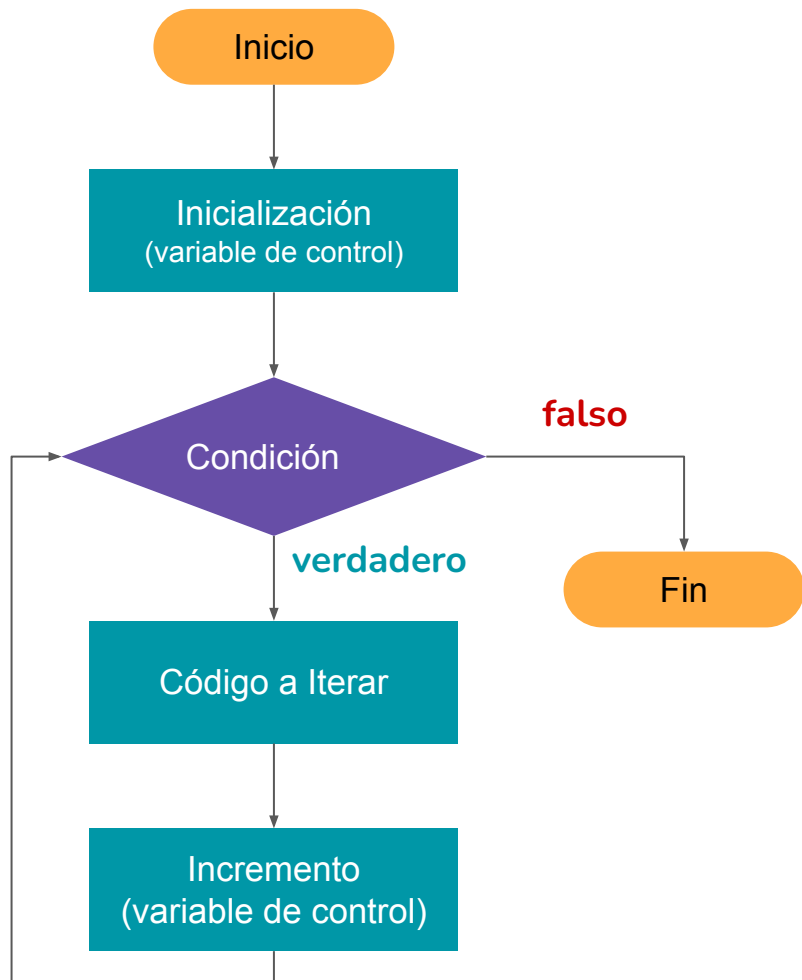
```
let contador = 0;

do{
  contador++;
  console.log('Conteo: ' + contador);
}while( contador < 10 )
```

Ciclo For

DEV.F
DESARROLLAMOS(PERSONAS);

dev



Ciclo for

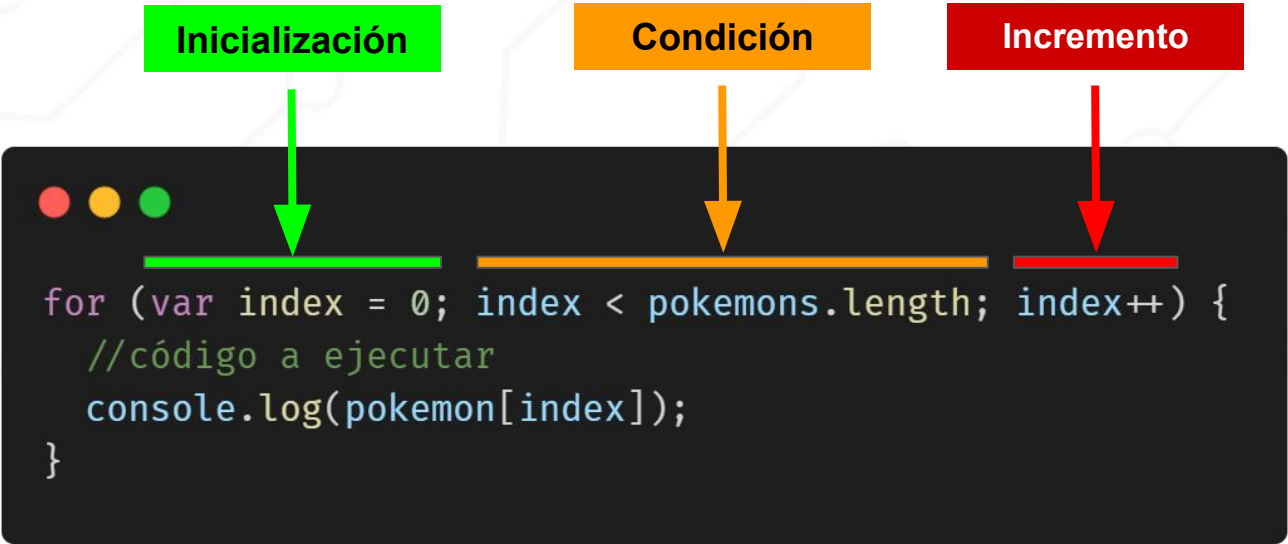
Un **bucle for** es un bucle que **repite** el bloque de instrucciones **un número predeterminado de veces**.

Sintaxis Ciclo for

Inicialización

Condición

Incremento



```
for (var index = 0; index < pokemons.length; index++) {  
  //código a ejecutar  
  console.log(pokemon[index]);  
}
```

The diagram illustrates the three components of a for loop syntax. Above the code, three colored boxes are labeled: 'Inicialización' (green), 'Condición' (orange), and 'Incremento' (red). Arrows point from these boxes to the corresponding parts of the code: a green arrow points to 'var index = 0', an orange arrow points to 'index < pokemons.length', and a red arrow points to 'index++'. The code itself is displayed in a dark-themed editor window with a title bar showing red, yellow, and green window control buttons. The code is color-coded: 'var' is purple, 'index' is green, '0' is white, ';' is white, 'index' is blue, '<' is white, 'pokemons.length' is light blue, ';' is white, 'index++' is green, '{' is white, the comment '//código a ejecutar' is green, 'console.log' is light blue, 'pokemon[index]' is light blue, and '}' is white.

Inicialización: De la variable que llevará el conteo de cuantas veces se iterara.

Condición: Mientras la condición se cumpla, se ejecutará el código dentro de las llaves {}.

Incremento: Se ejecuta después de cada iteración, normalmente se coloca un **contador** que incremente en 1 la variable de inicialización.

Contadores y acumuladores

En muchos programas se necesitan variables que cuenten cuántas veces ha ocurrido algo (contadores) o que acumulen valores (acumuladores).



Contador

Se entiende por contador una variable que lleva la cuenta del número de veces que se ha cumplido una condición.

```
> // Del 1 al 10 ¿Cuántos números son múltiplos de 2?  
var contador = 0;  
for (var index = 1; index <= 10; index++) {  
    if (index % 2 == 0) {  
        contador = contador + 1;  
        console.log(`${index} es múltiplo de 2`);  
    }  
}  
console.log(`De 0 a 10 existen ${contador} múltiplos de 2`);
```

2 es múltiplo de 2

4 es múltiplo de 2

6 es múltiplo de 2

8 es múltiplo de 2

10 es múltiplo de 2

De 0 a 10 existen 5 múltiplos de 2

Acumulador

Se entiende por acumulador:

Una **variable** que **acumula** el resultado de una operación.

```
> var acumulador = 0;  
  for (var index = 0; index <= 4; index++) {  
    acumulador = acumulador + index;  
    console.log(acumulador);  
  }
```

0

1

3

6

10

Continue

Termina la ejecución de las sentencias de la iteración actual del bucle actual y continua la ejecución del bucle con la próxima iteración.

```
i = 0;
n = 0;
while (i < 5) {
    i++;
    if (i == 3)
        continue;
    n += i;
}
```

Break

Termina el bucle actual, sentencia switch o label y transfiere el control del programa a la siguiente sentencia a la sentencia de terminación de éstos elementos.

```
for (i=0;i<10;i++){  
    document.write (i)  
    escribe = prompt("dime si continuo preguntando...", "si")  
    if (escribe == "no")  
        break  
}
```

For

VS

while

¿Cuándo usar *While* y cuándo *For*?

No existen reglas fijas, pero una buena recomendación para escoger entre ambas es el caso de si conozco o no el número de iteraciones que voy a realizar:

- Usamos el **ciclo for** para iterar un **arreglo**.
- Usamos el **ciclo for** cuando sabemos que el código a iterar debería ejecutarse **n veces**.
- Usamos el **ciclo while** para que su código se ejecute indefinidamente.
- Usamos el **ciclo while** para ejecutar código hasta que se cumpla una condición
- Usamos el **ciclo while** cuando se quiere que el usuario controle cuándo debe detenerse el código

También es importante mencionar que conforme adquiramos más habilidades podríamos usar estructuras de iteración más avanzadas diferentes a for y while.

Actividad 4:

Estructuras de Control

1. Leer el cap 2 del libro Eloquent JavaScript: Estructura del programa.
2. Usando **while**, crea un programa que pregunte al usuario un número. Mostrar los números que son múltiplos de 5 desde 1 hasta el número introducido por el usuario.
3. Crea el mismo problema 1, ahora usando **Do While**
4. Usando **for**, crea un programa que imprima en consola los números impares del 1 al 50.
5. Haciendo uso del for loop, imprimir una lista de números del 1 al 100 y excluir un rango de 10 números, el inicio y fin del rango lo determinará el usuario.