

Ciclos While, Do-While y For

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Objetivos de la sesión

Sesión Teórica y Práctica

- Aprender sobre ciclos
- Aprender sobre las estructuras de control:
 - While
 - Do-While
 - For
- Utilizar estas estructuras



¡¡¡Disfruten!!!

Ciclos o Bucles

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Bucle o Ciclo

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

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

Cada repetición se suele llamar iteración.



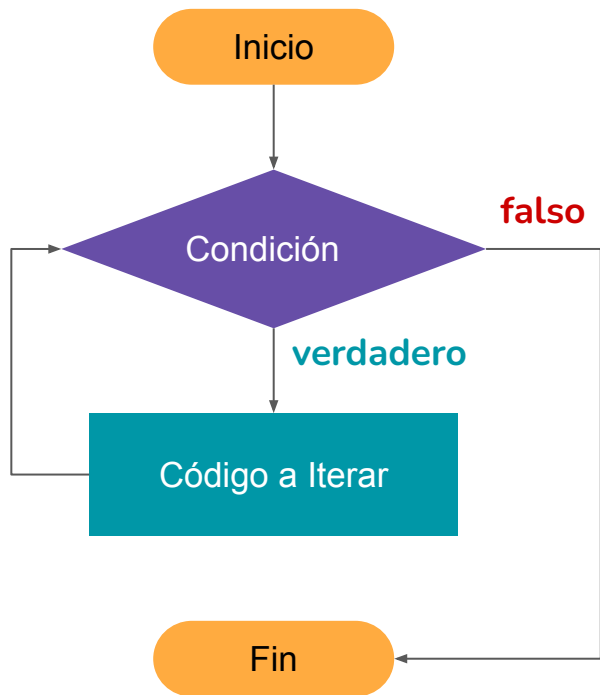
Ejemplo

```
Mientras (mi cerebritito aguante no comer garnachas){  
    yo.seguirMiDieta()  
}  
yo.romperMiDieta()
```

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

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

```
> // Ejemplo #1  
  // Imprimir números del 0 al 10 en consola.  
var index = 0;  
while(index < 11) {  
  console.log(index);  
  index++;  
};
```

Resultado:

0

1

2

3

4

5

6

7

8

9

10

Ejemplo #2: Ciclo while

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

```
> // Guarda un arreglo de valores introducidos  
// Si el usuario no introduce un valor, termina el ciclo.  
var arreglo = [];  
var userInput;  
while (!(userInput=="")) {  
  userInput = prompt("Ingresa cualquier carácter");  
  arreglo.push(userInput);  
}  
console.log("Introduciste estos valores: " + arreglo);
```

Resultado:

d

f

g

a

Introduciste estos valores: d,f,g,a,

¿Qué detalle encuentras?
¿Cómo lo solucionarías?

Ejercicio en clase

Crea un programa que solicite al usuario un día de la semana (ej: lunes, jueves, domingo, etc). El programa mostrará un mensaje personalizado para cada día de la semana por medio de un alert. Y seguirá pidiendo al usuario introducir otro día. En caso de que el día introducido sea domingo mostrar al usuario el mensaje “Ve a descansar” y terminar la captura de información.

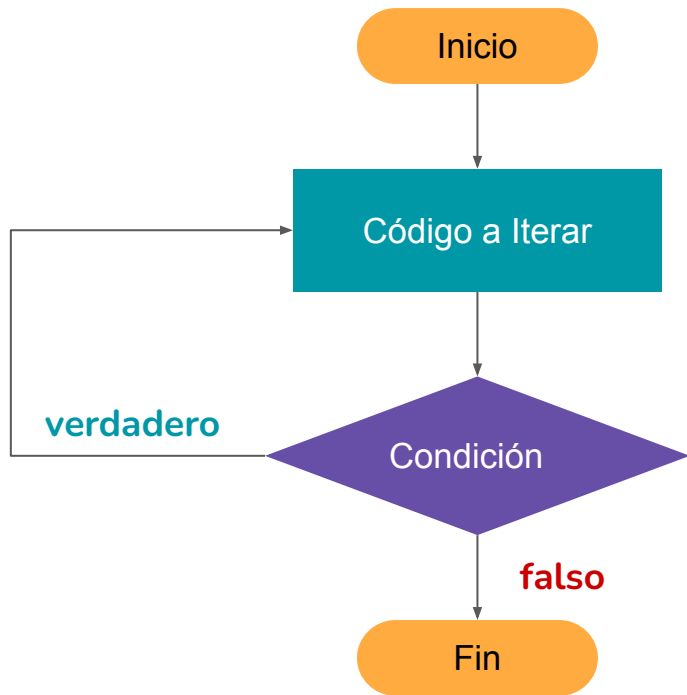


05:00

Ciclo Do-While

DEV.F
DESARROLLAMOS(PERSONAS);

dev



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 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);
```

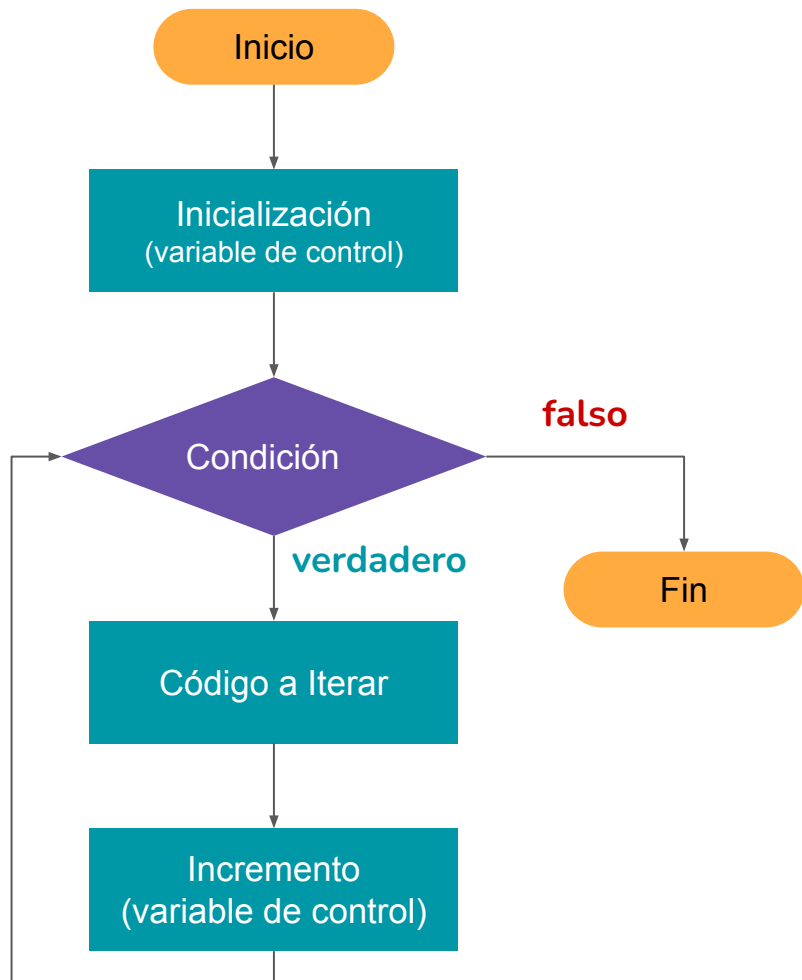
Ejemplo

```
Haz {  
    probarNuevaldea()  
    venderNuevaldea()  
} mientras (vendoSuficiente);  
abortarEmprendimiento()
```

Ciclo For

DEV.F
DESARROLLAMOS(PERSONAS);

dev



Ciclo for

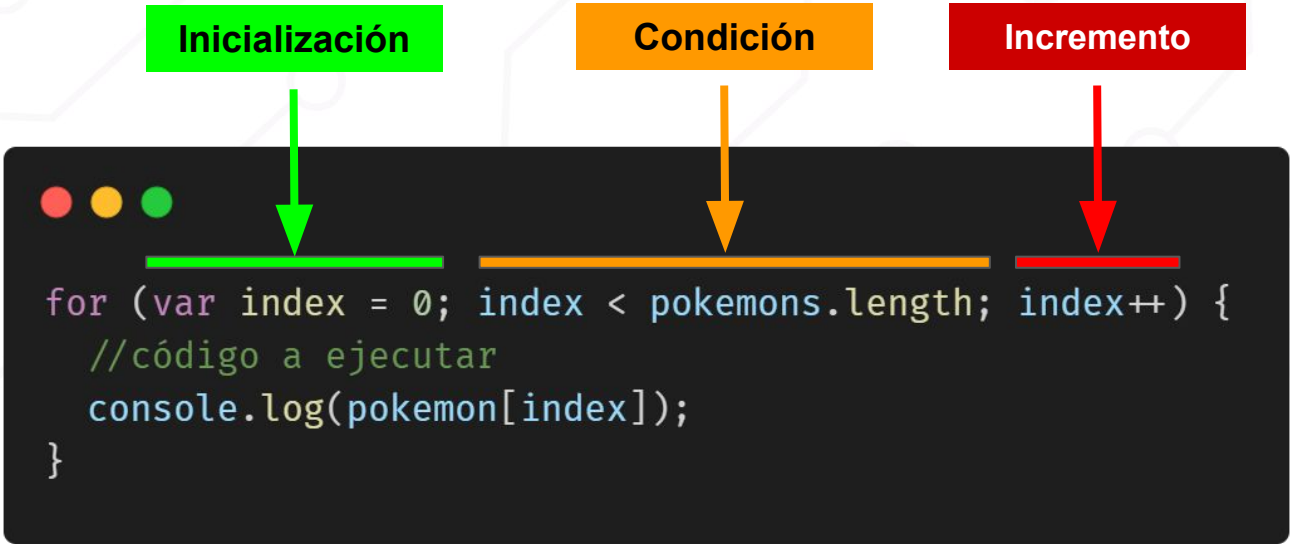
Un **bucle for** es un bucle que **repite** el bloque de instrucciones **un número predeterminado de veces**.

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 window control bar (red, yellow, green buttons) at the top left. The code is color-coded: 'var' is purple, 'index' is blue, '0' is green, ';' is white, 'index' is blue, '<' is white, 'pokemons.length' is light blue, ';' is white, 'index++' is blue, '{' is white, '//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

Ejercicio en clase

Crea un programa que recorra el arreglo:

```
[4, "dos", 8, "tres", 5, 9, 1, "cero"]
```

Y muestre en consola solo los elementos que son tipo número



05:00

¿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 la variable que nos permite leer un archivo.
- Usamos el **ciclo while** para preguntar por entradas del usuario (user input).
- Usamos el **ciclo while** cuando el incremento de valor en iteración es algún valor no estándar.

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

DEV.F.



Vamos al código

dev