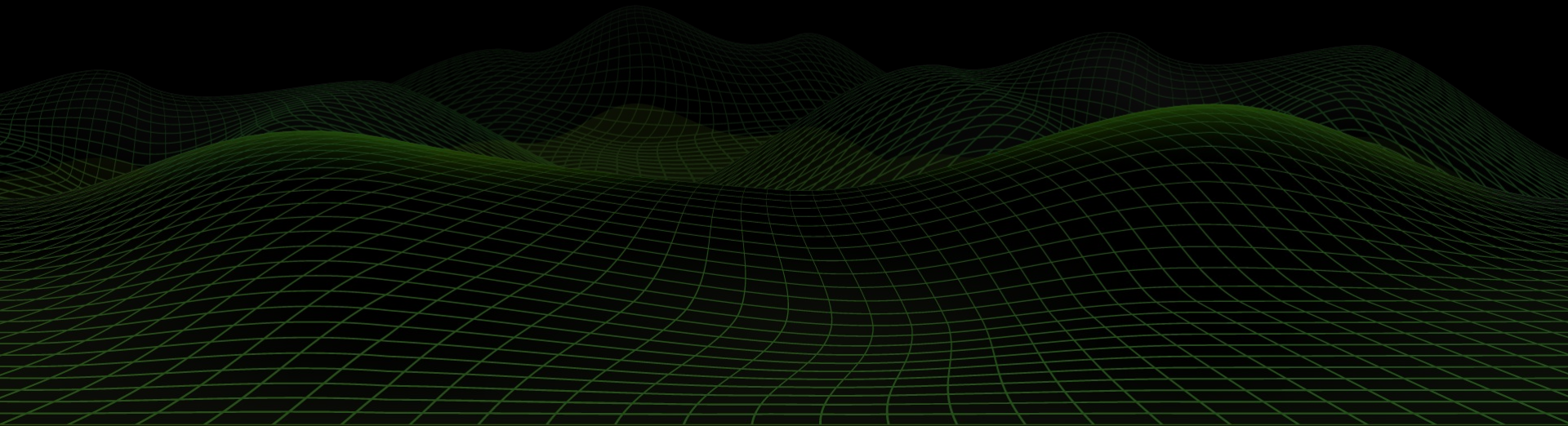


Ejercicios de Bucles



BIENVENIDOS

U.

ÍNDICE

- **1** While
- **2** Do While
- **3** For

Introducción a bucles



> While

- Al utilizar la palabra reservada “while” podemos crear un ciclo de repetición el cual necesita una **condición** para poder ejecutar repetidamente una serie de acciones. Luego de la palabra reservada se coloca paréntesis (**sentencia a evaluar**), dónde estará la expresión que será evaluada y que estará encargada de permitir o no continuar ejecutando su bloque de código.
- Luego de los paréntesis se abrirán y cerrarán las llaves {...**bloque de código**...} en donde estarán la o las acciones a ser ejecutadas en cada iteración. Al ser un **bucle condicional** se necesitará tener dentro del bloque de código a iterar algo que altere en algún momento la el valor resultante de la condición dentro de los paréntesis del ciclo **while** para que en algún momento finalice, ya que de no hacerlo estaremos cayendo en un bucle infinito el cual podría colgar el navegador en donde se esté ejecutando el código.

```
while ( sentencia a evaluar ) {  
  bloque de código  
}
```

> Do While

- “do while” funciona de la misma manera que el **while** con la diferencia de que se comienza con la palabra reservada “do” seguido de las llaves {...bloque de código...} donde estarán la o las acciones a iterar, y luego de la llave de cierre colocar la palabra reservada “while” seguido de los paréntesis con la expresión a evaluar (**expresión o condición**). Así como el **while** debe haber algo que cambie el valor de verdad de la expresión dentro de los paréntesis para que en algún momento finalice. Y como última diferencia respecto de su contraparte while, el bucle do **while** permite por lo menos ejecutar 1 vez el bloque de código.

```
do {  
  bloque de código  
} while ( sentencia a evaluar )
```

> For

- Con la palabra reservada **"for"** podemos realizar un ciclo de repetición una cantidad determinada de veces. La estructura común es la siguiente:
for (A ; B ; C) {...bloque de código...}.

A: Declaración e inicialización de la variable de control, por lo general **"let i = 0"**, esta será la variable que se utilizará como índice de la iteración.

B: Condición a ser evaluada para continuar iterando, por lo general se utiliza la variable de control, como por ejemplo **"i menor a 10"** en donde el bucle for seguirá iterando mientras que esta condición sea verdadera.

C: El paso, al realizar un incremento en la variable de control nos estamos asegurando que en cada iteración su valor cambia, por lo cual la condición en el apartado B en algún momento será falsa y finaliza la iteración.

```
for ( let i = 0; i < 10; i++ ) {  
  bloque de código  
}
```

> For of

El bucle en **for of** se utiliza para iterar sobre los valores de objetos iterables. Proporciona una forma sencilla y concisa de recorrer arrays, cadenas, nodeList, set y otros objetos iterables.

For of es menos propenso a bucles infinitos debido a su diseño específico para objetos iterables y su capacidad de finalización de manera más automática y segura

Luego de la palabra reservada **"for"**, dentro de los paréntesis se colocan, en primer lugar la declaración de variable que se utiliza para almacenar el valor actual de la iteración en cada paso del bucle. Luego se coloca la palabra reservada **"of"** y por último el elemento iterable.

```
for ( let variable of Iterable ){  
    bloque de código  
}
```


> For in

El bucle en **for in** se utiliza para iterar sobre las propiedades enumerables de un objeto. A diferencia del bucle **for of**, que itera sobre los valores de objetos iterables, el bucle **for in** itera sobre las claves (propiedades) del objeto.

For in no se considera seguro para iterar sobre array por lo que comúnmente se utiliza sobre objetos.

Luego de la palabra reservada **"for"**, dentro de los paréntesis se colocan, en primer lugar la declaración de variable que se utiliza para almacenar la clave (key) actual de la iteración en cada paso del bucle. Luego se coloca la palabra reservada **"in"** y por último el objeto que se desea iterar.

```
for ( let variable in objeto ){  
    bloque de código  
}
```

Ejercitación

Ejercitación

Bucles

1

Utilizando un bucle **for**, realizar un programa que permita el ingreso de un número a través de un **prompt** entre 1 y 100, luego que imprima por consola todos los números que se encuentran entre el número ingresado y el 0.

Supongamos que es 5 el número ingresado. Este sería el resultado.

5	bucles.js:334
4	bucles.js:334
3	bucles.js:334
2	bucles.js:334
1	bucles.js:334

2

Utilizando un bucle **for**, realizar un programa que permita el ingreso de un número a través de un **prompt** entre 1 y 10, luego se debe mostrar por consola su tabla de multiplicar (*Los primeros 10 múltiplos solamente*).

Supongamos que el número ingresado es 2. Este sería el resultado.

Tip! utiliza **backticks** para esto.

1 x 5 = 5	bucles.js:8
2 x 5 = 10	bucles.js:8
3 x 5 = 15	bucles.js:8
4 x 5 = 20	bucles.js:8
5 x 5 = 25	bucles.js:8
6 x 5 = 30	bucles.js:8
7 x 5 = 35	bucles.js:8
8 x 5 = 40	bucles.js:8
9 x 5 = 45	bucles.js:8
10 x 5 = 50	bucles.js:8

Ejercitación

Bucles

3

Utilizando un `while`, realizar un programa que permita el ingreso de números a través de un `prompt`, los cuales se tienen que ir sumando en una variable. El ingreso de números terminará cuando el usuario ingrese un 0. En ese caso se debe mostrar por consola el valor de la variable que contiene la suma de los números ingresados.

Supongamos que se ingresó 1 luego 3 y luego 0. Este sería el resultado.

Tip! Cuidado, recuerda que el `prompt` devuelve un `string`.

4

[bucles.js:42](#)

4

Utilizando un `do while` y siguiendo las mismas instrucciones que el ejercicio anterior, mostrar por consola el valor de la variable que acumula la suma de los números ingresados.

Supongamos que se ingresó 1 luego 3 y luego 0. Este sería el resultado.

4

[bucles.js:42](#)

Ejercitación

Bucles

5

Utilizando el bucle que creas correcto, vamos a utilizar la variable que guardaba la suma de los número en el **ejercicio N°3**. El programa permitirá ingresar números a través de un **prompt** hasta que el número ingresado sea igual al guardado en la variable del ejercicio n°3.

Si el valor ingresado es mayor al número de la variable, avisarle al usuario por consola, lo mismo si el valor es menor. Así sucesivamente hasta que el usuario adivine el número secreto. Por último mostrar un mensaje de felicitaciones y decirle en cuantos intentos lo ha realizado.

6

Utilizando el bucle que creas correcto, realizar un programa que reciba un número a través de un **prompt** y muestre por consola todos sus divisores. Por ejemplo $12 / 4$, resultado es 3 y sobra 0. Cuando un número que divide a otro produce un resto 0, se dice que es divisor del número dividido. Supongamos que se ingresó 50. Este sería el resultado.

Tip! recuerda el operador %

El numero ingresado es menor que [bucles.js:77](#)
el secreto

El numero ingresado es mayor que [bucles.js:80](#)
el secreto

Acertaste! el número secreto era: [bucles.js:85](#)
45, realizaste 2 intentos

50	bucles.js:155
25	bucles.js:155
10	bucles.js:155
5	bucles.js:155
2	bucles.js:155
1	bucles.js:155

Ejercitación

Bucles

7

Utilizando un bucle `for of`, crea un array de strings con colores y realizar un programa que recorra ese array para poder mostrar por consola con cada uno de los elementos del array.

En la consola debería verse de esta manera

Rojo	bucles.js:170
Verde	bucles.js:170
Azul	bucles.js:170
Violeta	bucles.js:170
Amarillo	bucles.js:170
Celeste	bucles.js:170
Narajas	bucles.js:170
Rosa	bucles.js:170

8

Utilizando un bucle `for of`, crear un array con 5 números y realizar un programa que recorra ese array para poder mostrar por consola el doble de cada uno de los elementos..

En la consola debería verse de esta manera

Tip! utiliza backticks para esto.

El número es 5 y su doble es 10	bucles.js:170
El número es 7 y su doble es 14	bucles.js:170
El número es 10 y su doble es 20	bucles.js:170
El número es 13 y su doble es 26	bucles.js:170
El número es 17 y su doble es 34	bucles.js:170

Ejercitación


Bucles

9

Utilizando el bucle que creas correcto, crea un array con al menos 4 objetos con 4 propiedades cada uno que representen un miembro de un grupo familiar (*puede ser tu familia inclusive*). Luego realizar un programa que muestre en consola un mensaje de presentación por cada elemento del array.

En la consola debería verse como en el ejemplo.

Tip! utiliza backticks para esto.

```
Modelo de objeto bucles.js:172  
{nombre: 'Juan', apellido: 'Perez', edad: 25, inte  
grante: 'Padre'}   
  apellido: "Perez"  
  edad: 25  
  integrante: "Padre"  
  nombre: "Juan"
```

```
Hola soy Juan Pérez (Padre) y tengo bucles.js:198  
25 años
```

```
Hola soy María González (Madre) y bucles.js:198  
tengo 25 años
```

```
Hola soy Julián Perez González (Hijo) bucles.js:198  
y tengo 1 años
```

```
Hola soy Chocolate Perez González bucles.js:198  
(Mascota) y tengo 4 años
```

Ejercitación

Bucles

10

Utilizando un bucle `for in`, crear un objeto con al menos 5 propiedades, realizar un programa que recorra dicho objeto y solo muestre todas las keys de sus propiedades.

En la consola debería verse de esta manera

nombre	bucles.js:405
apellido	bucles.js:405
esColombiano	bucles.js:405
edad	bucles.js:405
estudios	bucles.js:405

11

Utilizando un bucle `for in`, realizar un programa que recorra el objeto creado en el [ejercicio n°10](#) y solo muestre los valores de cada una de las keys.

En la consola debería verse de esta manera

Juan	bucles.js:406
Perez	bucles.js:406
true	bucles.js:406
28	bucles.js:406
	bucles.js:406

```
► (3) ['primario', 'secundario', 'universitario']
```


Ejercitación

Bucles

12

Utilizando el bucle que creas correcto, realizar un programa que permita la entrada de números a través de un **prompt** y calcule la suma de los números pares por un lado y los impares por otro, el ingreso de datos finaliza cuando el usuario ingresa un 0. Mostrar por consola por un lado los pares y por otro los impares.

En la consola debería verse de esta manera

```
Pares: 30 bucles.js:231  
Impares: 38 bucles.js:232
```

13

Utilizando el bucle que creas correcto, cree un array de 10 números y realizar un programa que imprima por pantalla el número más grande de dicho array.

¡MUCHAS GRACIAS!

**MIND
HUB.**