Bucle While

Es una estructura de control en programación que ejecuta un bloque de código mientras una condición dada sea verdadera. La condición se verifica antes de cada iteración.

**Estructura del Bucle While**: while(condicion){}

**Iteración:** Es repetir un conjunto de instrucciones o acciones una y otra vez hasta que se cumpla una condición específica.

Hagamos un ejemplo, supongamos que necesitamos mostrar 5 veces un mensaje “Hola”, esto podemos realizarlo escribiendo cinco **console.log(“Hola”)**. Vamos a verlo:

Algo que vemos en este ejemplo es que siempre se repite un bloque de código, el **console.log(“Hola”)**. Mejoremos este ejemplo con el **Bucle While**.

Sabemos que se debe mostrar 5 veces ,por lo tanto, debemos ir desde el **1 al 5**.

Entonces, la variable contador inicia en **1 (contador = 1)**, dentro de la condición del while debe ser que el contador debe ser menor y/o igual a 5.

Dentro del Bloque de código ponemos el **console.log(“Hola”)**y un operador de corte que será el incremento de contador **(contador++):**

**Ejemplo del código:**

Analicemos qué valores toma contador y qué ocurre en cada línea de código con el siguiente cuadro:

Cuando la condición deja de ser verdadera, deja de repetir el bloque de código y se termina la ejecución del **Bucle While.**

**Importante:** Nunca se olviden de colocar un operador de corte, sino tendríamos que enfrentarnos a un Bucle Infinito y eso nos obligaría a cerrar todas las ventanas.

**Bucle Do-While: do{}while(condicion);**

Es similar al bucle while, pero el bloque de código se ejecuta al menos una vez, incluso si la condición es falsa desde el principio. Es decir, que primero ejecuta y luego verifica con la condición, como si dijera “hago, después chequeo”.

Vamos a volver hacer el mismo ejemplo de mostrar 5 veces “Hola”con **Do-While:**

Vamos con un ejemplo en el que no se cumpla la condición, ya que habíamos hablado que aunque no se cumple la condición se repite al menos una vez:

El contador inicia en 7, luego se ejecuta el bloque de Código que está dentro del**Do**, luego imprime el mensaje e incrementa en uno contador. Por lo tanto, contador es igual a 8 pero cuando va a realizar la verificación de la Condición nos encontramos que contador **no** cumple con la condición (contador debe ser menor o igual a 5 ) y como la condición es **False**termina y deja de iterar.

El **bucle for** es una estructura de control que se utiliza para repetir un bloque de código un número específico de veces.

Es evaluada antes de cada iteración. Si la condición es verdadera, el código dentro del bucle se ejecuta; de lo contrario, el bucle se detiene.

For(x,y,z){}

* **Inicialización:** Se ejecuta una vez al comienzo del bucle y se utiliza para inicializar una variable de control.
* **Condición:** Es evaluada antes de cada iteración. Si la condición es verdadera, el código dentro del bucle se ejecuta; si el falso, el bucle se detiene.
* **Actualización:** Se ejecuta al final de cada iteración y se utiliza para actualizar la variable de control.

Si queremos imprimir del **1 al 5** sabemos que la **inicialización** va a empezar en **1** **(let i = 1)**, luego la **condición**es que los números sean menores o igual a **5 (i <= 5)** y por último la **actualización**es que se vaya sumando de**1 en 1**, es decir, que le sume **1 (i++)**

**For (let i = 1; i <= 5; i++){ console.log(i);}**

**Arrays**

Un **array** es una estructura de datos que se utiliza para almacenar una colección de elementos. Estos elementos pueden ser números, cadenas de texto, objetos, funciones u otros tipos de datos. Los **arrays** en **JavaScript**son flexibles y pueden contener una mezcla de diferentes tipos de datos.

Un array tiene una estructura determinada donde sus elementos están ordenados por el índex, cada elemento tiene un número de posición, siempre el primer elemento tiene la posición 0:

**Acceder a un Elemento de Array**

Para acceder a un solo elemento tenemos que llamar a la variable del Array y especificar la posición del elemento:

**Propiedades y métodos de un array:**

Ya sabemos que un array tiene diferentes elementos dentro y que cada elemento tiene una posición iniciando desde el 0(cero) y como acceder a ellos.

**Propiedad Length:** Devuelve la cantidad de elementos en el array.

**Agregar un elemento nuevo al array:**

* Agregar un elemento en la última posición del array: (variable).push()
* Agregar el elemento como Primer elemento del Array: (variable).unshift()

**Extraer una porción de un array y devuelve un nuevo array con los elementos seleccionados:**

En este ejemplo, productos.slice(2, 4) extrae los elementos desde el índice 2 (incluido) hasta el índice 4 (no incluido) del array productos, creando un nuevo array llamado nuevoprod. El **array** original productos no se modifica.

Const productos =[0,1,2,3,4]

Const nuevoprod = productos.slice(2,4)

Console.log(nuevoprod);

**Eliminar uno o más elementos del Array:**

* Eliminar el último element: (variable).pop()
* Eliminar el Primer element: (variable).shift()
* Eliminar varios elementos: (variable).splice()

Recibe dos parámetros: Uno es la posición donde inicia el corte y el otro es la cantidad de elementos a eliminar a partir de esa posición:

**Conocer la posición de un Elemento, si el Elemento NO existe no devuelve -1:**

Console.log(variable.indexOx(“”));

**Bucle For con Arrays:**

En la clase anterior aprendimos todo acerca del**bucle for**, y conocimos los **arrays**.

En la clase de hoy vamos a combinarlos, y ver la utilidad que les podemos dar cuando los usamos en conjunto.

Los **arrays o arreglos** los podemos entender como una gran caja, que dentro contiene distintas celdas donde podemos almacenar datos.

Vimos también que, para completar las posiciones del array, lo podemos hacer de manera manual, completando celda por celda.

**Por ejemplo:**

**let arrayNombres = [“Julieta” , “Diego”, “Facundo”, “Sara”];**

De esta forma completamos los datos de un array de nombres. Tiene 4 posiciones, y lo tenemos ordenado de la siguiente manera:

Si queremos mostrar cada posición del array, o el nombre “Diego”, debemos escribir:

Pero… y si quisiéramos mostrar todos los elementos del array mediante un bucle, ¿Cómo podríamos llevarlo a cabo?

Esto desde ya sería muy útil para ahorrarnos código, y no tener que escribir un console.log por cada posición del array. Imaginemos que nuestro array tiene 1000 posiciones, entonces, si no usamos bucles deberíamos usar 1000 console.log distintos, uno por elemento. ¿Parece un montón, no?

Entonces, para ello utilizaremos un bucle For. Iniciará desde la posición 0, hasta la última posición de nuestro array, y a medida que el iterador va aumentando vamos a imprimir en pantalla lo que se encuentra dentro de cada posición del array.  
  
En la práctica se vería de esta manera:

Por último, este array contiene en total **4** elementos **(desde el 0 al 3)**. Este array contiene muy pocos elementos, hay otros que contienen muchísimos más.  
Si no sabemos con exactitud cuántos elementos tiene nuestro array existe una propiedad en **JavaScript** que nos puede ahorrar el conteo, ésta propiedad se llama Lenghty se usa de la siguiente manera.

Como se puede ver, reemplaza al **“3”**. Y funciona de la misma manera para todos los arrays, no importa cual sea su cantidad de elementos.