«Talento Tech»

Desarrollo Web 2

Clase 05

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 5 |

Temario:

- Bucle While
- Bucle Do While
- Bucle for
- Arrays
- Bucle for con arrays

Buenos Aires aprende
Agencia de Habilidades para el Futuro

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 (condición){
    //Bloque de código a repetir mientras la condición sea verdadera
}
```

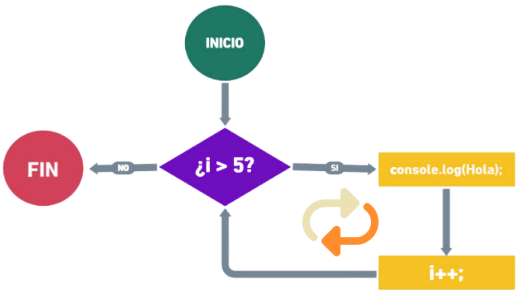
Iteración: Es repetir un conjunto de instrucciones o acciones una y otra vez hasta que se cumpla una condición específica.

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

```
console.log("Hola");
console.log("Hola");
console.log("Hola");
console.log("Hola");
console.log("Hola");
```

Resultado:
Hola
Hola
Hola
Hola
Hola

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:

```
let contador = 1; //Inicio de variable contador = 1
while (contador <= 5){ // Condición -> contador menor o igual a 5
    console.log("Hola");
    contador++; // Incrementa en 1 -> contador = contador + 1
}
```

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

Buenos Aires aprende
Agencia de Habilidades para el Futuro

	Condicion			Bloque de Codigo dentro de la condicion	
Variable contador	contador <= 5	Resultado de la Condicion	Ejecuta el Bloque de Codigo?	console.log("Hola");	contador++
1	1 <= 5	VERDADERO	Si	Hola	1 + 1 = 2
2	2 <= 5	VERDADERO	Si	Hola	2 + 1 = 3
3	3 <= 5	VERDADERO	Si	Hola	3 + 1 = 4
4	4 <= 5	VERDADERO	Si	Hola	4 + 1 = 5
5	5 <= 5	VERDADERO	Si	Hola	5 + 1 = 6
6	6 <= 5	FALSO	No	-	-

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.

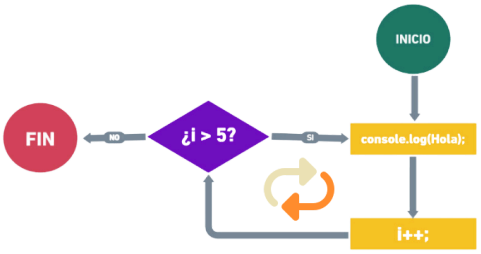
Bucle Do-While

```
do{
    //Bloque de código que se ejecuta al menos una vez si la condición
es FALSA
} 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”.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

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




```
let contador = 1; // Inicio de Variable contador = 1
do {
    console.log("Hola"); // Imprime el mensaje
    contador++; // Incrementa en 1
} while (contador <= 5); // Verifica la condición
```

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:

```
let contador = 7 // Inicio de Variables = 7
do {
    console.log("Hola"); // Imprime el mensaje
    contador++; // Incrementa en 1 -> 7 + 1 = 8
    // contador = 8
} while (contador <=5); // Verifica la condición
// 8 no es menor ni igual a 5 --> FALSO
```

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.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Bucle For

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 (inicialización; condición; actualización){
    // Código a ejecutar en cada iteración
}
```

- **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 <=; i++){
    console.log(i);
}
```

Resultado:
1
2
3
4
5

Buenos Aires aprende
Agencia de Habilidades para el Futuro

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.

```
// Declaración de array vacio
const arrayVacio = [];
// Declaración de array con números
const deUnaCifra = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0];
// Declaración de array con strings
const listaDeCompras = ["Jabon", "Detergente", "Suavizante", "Escoba",
"Trapo de piso"];
// Declaración de array con booleanos
const booleanos = [true, false, true, false];
// Declaración de array mixto
const datosVariados = [1, false, "C4", 3.14];
```

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:

```
const listaDeCompras = ["Jabón", "Detergente", "Suavizante", "Escoba",
"Trapo de piso"];
// index              = [    0    ,     1       ,      2       ,   3     ,
4      ]
```

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:

```
const listaDeCompras = ["Jabón", "Detergente", "Suavizante", "Escoba",
"Trapo de piso"];
// index              = [    0    ,     1       ,      2      ,   3     ,
4       ]
console.log(productos[0]);
console.log(productos[1]);
console.log(productos[2]);
console.log(productos[3]);
console.log(productos[4]);
```

Resultado:
Jabón
Detergente
Suavizante
Escoba
Trapo de piso

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:

.length: Devuelve la cantidad de elementos en el array.

Ejemplo:

```
const listaDeCompras = ["Jabón", "Detergente", "Suavizante", "Escoba",
"Trapo de piso"];
// index              = [    0    ,     1       ,      2      ,   3     ,
4       ]
console.log(listaDeCompras.length);
```

Resultado:
5

.push: Agrega un elemento en la última posición del array.

Ejemplo:

```
const listaDeCompras = ["Jabón", "Detergente", "Suavizante", "Escoba", "Trapo de piso"];
listaDeCompras.push("Lavandina"); // Agregamos un producto al final del array
console.log(listaDeCompras);
```

Resultado por consola:

```
["Jabón", "Detergente", "Suavizante", "Escoba", "Trapo de piso", "Lavandina"];
```

.unshift: Agrega un elemento en la primera posición del array.

Ejemplo:

```
const listaDeCompras = ["Jabón", "Detergente", "Suavizante", "Escoba", "Trapo de piso"];
listaDeCompras.unshift("Lavandina"); // Agregamos un elemento al comienzo del array
console.log(listaDeCompras);
```

Resultado por consola:

```
["Lavandina", "Jabón", "Detergente", "Suavizante", "Escoba", "Trapo de piso"];
```

.slice(): 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.

Ejemplo:

```
const productos = ["Televisores", "Celulares", "Microondas",
"Heladeras", "X-box"];
//   Indice     = [      0      ,      1     ,       2     ,       3
,     4   ]
const nuevoprod = productos.slice(2, 4);
// Se crea un nuevo array desde el indice 2 (incluido) hasta el indice
4 (no incluido)
console.log(nuevoprod);
```

Resultado por consola:

```
2() ["Microondas", "Heladeras"]
```

.pop(): Eliminar el último elemento.

Ejemplo:

```
const productos = ["Televisores", "Celulares", "Microondas",
"Heladeras", "X-box"];
//   Indice     = [      0      ,      1     ,       2     ,       3
,     4   ]
productos.pop();
console.log(productos);
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Resultado por consola:

```
4() ["Televisores", "Celulares", "Microondas", "Heladeras"]
```

.shift(): Eliminar el Primer elemento.

Ejemplo:

```
const productos = ["Televisores", "Celulares", "Microondas",
"Heladeras", "X-box"];
//   Indice   = [      0      ,      1      ,      2      ,      3    ,
4   ]
productos.shift();
console.log(productos);
```

Resultado por consola:

```
4() ["Celulares", "Microondas", "Heladeras", "X-box"]
```

.splice(): Eliminar varios elementos.

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:

Ejemplo:

```
const productos = ["Televisores", "Celulares", "Microondas",
"Heladeras", "X-box", "Tablet", "Cocina"];
//    Indice    = [      0      ,      1      ,      2      ,      3
,    4   ,     5   ,     6   ]
productos.splice(1, 3);
console.log(productos);
```

Resultado por consola:

```
(4) ["Televisores", "X-box", "Tablet", "Cocina"]
```

.indefOf: nos permite conocer la posición de un elemento, si el Elemento NO existe no devuelve -1

Ejemplo:

```
const productos = ["Televisores", "Celulares", "Microondas",
"Heladeras", "X-box", "Tablet", "Cocina"];
//    Indice    = [      0      ,      1     ,       2     ,       3
,    4   ,     5   ,     6   ]
console.log(productos.indexOf("Microondas"));
// Nos imprime el numero de index que corresponde al elemento
"Microondas"
```

Resultado por consola:

2

Bucle For con Arrays:

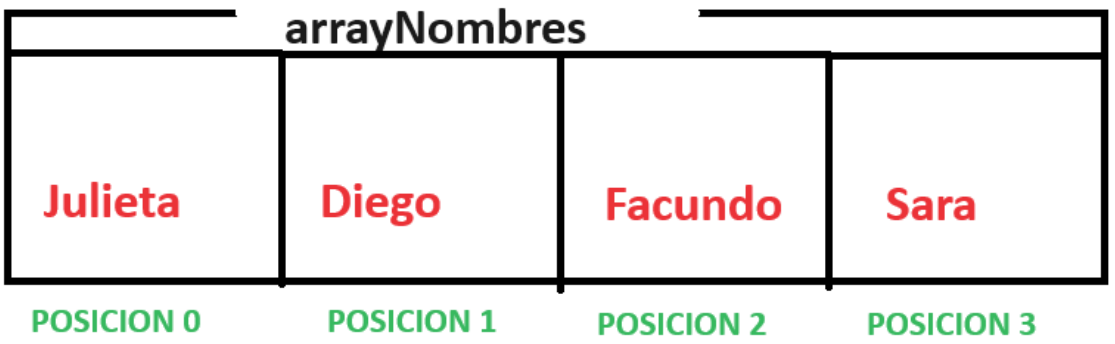
En la clase anterior aprendimos todo acerca del **bucle for**, y conocimos los **arrays**.
En la clase de hoy vamos a combinarlos, y ver la utilidad que les podemos dar cuando los usamos en conjunto.
Los **arrays o arreglos** los podemos entender como una gran caja, que dentro contiene distintas celdas 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:

```
let arrayNombres = ["Julieta" , "Diego", "Facundo", "Sara"];

console.log(arrayNombres[1]);
```

Resultado:
Diego

Pero… y si quisiéramos mostrar todos los elementos del array mediante un bucle, como 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:

```
let arrayNombres = ["Julieta" , "Diego", "Facundo", "Sara"];

for(let i=0; i<=3; i++){

    console.log(arrayNombres[i]);

}
```

Resultado:
Julieta
Diego
Facundo
Sara

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.

```
let arrayNombres = ["Julieta" , "Diego", "Facundo", "Sara"];

for(let i=0; i<=arrayNombres.length; i++){

    console.log(arrayNombres[i]);

}
```

Resultado:
Julieta
Diego
Facundo
Sara

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.

❓ Pregunta de reflexión o de llamado de atención

Pensando en la gestión de productos en un e-commerce;

Buenos Aires aprende
Agencia de Habilidades para el Futuro

¿Cómo podría ser útil el uso de bucles para mostrar productos, realizar ventas o actualizar el stock?

Considerando el uso de arrays y bucles;
¿Puedes imaginar una situación en la que estos conceptos serían esenciales para organizar y manipular datos de manera efectiva en un programa?

Buenos Aires aprende
Agencia de Habilidades para el Futuro

📝Desafío #5

En este desafío, practicaremos el uso de **arrays** y **bucles** en JavaScript para gestionar productos en un **e-commerce**

Crear un ARRAY de productos.

Mostrar el contenido del ARRAY haciendo uso de un BUCLE FOR para recorrer el mismo e imprimir por consola cada uno de sus valores

Vamos a empezar creando un **Array de Productos**:

```
// Definir el array de productos

const productos = ["Producto1", "Producto2", "Producto3", "Producto4", "Producto5"];
```

Para mostrar el contenido del array vamos a utilizar un “**bucle for**” para recorrer el mismo e imprimir en la consola cada uno de los productos.

```
// Recorrer el array de productos e imprimir en la consola cada uno de los productos

for (let i = 0; i < productos.length; i++) {

    console.log("Producto " + (i + 1) + ":");

}
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ahora vamos a actualizar el stock:

Utilizaremos otro “**bucle for**” para simular la venta de productos. Reduce el stock de productos eliminando el último con la propiedad “**pop**” e imprime en la consola el nuevo stock después de la venta.

```
// Eliminar el último producto del array (Producto5)

productos.pop();

// Imprimir en la consola el nuevo stock después de la venta

console.log("Nuevo stock despues de la venta: ");

for (let i = 0; i < productos.length; i++) {

    console.log("Producto " + (i + 1) + ":");

}
```


Buenos Aires
aprende
Buenos
Aires
Ciudad