Bucles

Recordando arrays

Ya sabemos qué son los arrays, como declararlos y llenarlos de valores. También sabemos como acceder a distintos arrays.

Si tenemos el array

```
const nombres = ["Juan", "Sofía", "Danilo", "Luca", "Marco"];
```

podemos acceder a los distintos valores escribiendo el nombre del array y la posición del elemento que queremos. Si queremos acceder a Danilo, lo que hacemos es:

```
nombres[2];
```

porque el string “Danilo” está en la posición (empezando a contar desde cero)

Saludos múltiples

Supongamos que queremos hacer un programa que simplemente muestre un mensaje en pantalla, diciendo “Hola Juan” y luego “Hola Sofía” y luego “Hola Danilo” y así por cada uno de los elementos dentro del array. Este programa lo podríamos resolver de varias maneras

Resolución #1 – poco eficiente

Hacer console.log por cada posición:

```
console.log(nombres[0]);
console.log(nombres[1]);
console.log(nombres[2]);
console.log(nombres[3]);
console.log(nombres[4]);
```

Si bien este programa funciona, esto nos genera 5 líneas de código y en programación existe un término que llama DRY (Don’t repeat yourself) que significa “no te repitas a ti mismo”. Siempre que veamos que tenemos muchas líneas de código similares… seguro algo mejor vamos a poder hacer.

Resolución #2 - Eficiente

Utilizar un bucle for, recorrer de manera automática cada posición y ejecutar el console.log

```
for(let i = 0; i < nombres.length; i++) {
  console.log(nombres[i]);
}
```

Y con esto damos introducción al bucle for.

Bucle for

Un for nos permite ejecutar código de manera repetida las veces que nosotros querramos. Sirve, justamente, para poder realizar tareas repetidas una, otra y otra vez.

La palabra for es una palabra reservada en JavaScript y supone el comienzo de un bucle. Se escribe la palabra for, y luego se abren y cierran paréntesis y abren llaves con el código a ejecutar:

```
for() { }
```

Dentro de los paréntesis se agregan 3 expresiones:

1. condición inicial
2. condición de corte
3. condición a ejecutar al final de cada bucle

Vamos a pensar en ejemplos:

Quiero ejecutar un **console.log("Hola");** 5 veces: como sabemos que es una tarea repetitiva, usamos la palabra reservada "for", con su estructura.

```
for() { }
```

Lo que vamos a hacer es pensar en **una variable que empiece en cero**, por cada vuelta que se ejecute vamos a **sumarle uno** hasta que llegue a 5. **Cuando llegue a 5, cortamos.**

1. La condición inicial será una variable que empieza en cero
   ```
   for(let i = 0) { }
   ```
2. Separamos esta condición inicial por el uso de punto y coma
   ```
   for(let i = 0;) { }
   ```
3. Agregamos la condición de corte. Si queremos que se ejecute 5 veces, entonces vamos a cortar el loop si la variable **i** es menor que 5 (recordemos que empieza en 0)
   ```
   for(let i = 0; i < 5) { }
   ```
4. Separamos esta condición de corte por el uso de punto y coma
   ```
   for(let i = 0; i < 5;) { }
   ```
5. Agregamos la acción a ejecutarse al final de cada bucle que en nuestro caso es sumarle 1.
   ```
   for(let i = 0; i < 5; i++) { }
   ```
6. Agregamos el código a ejecutar todas estas veces
   ```
   for(let i = 0; i < 5; i++) {
     console.log("Hola");
   }
   ```

En otras palabras lo que estamos haciendo acá es decir:

Empezamos con una variable i, en cero. Si la variable i es menor que 5, entonces ejecuto lo que está dentro de las llaves. Finalmente cuando termino de ejecutar lo que está dentro de las llaves, le sumo 1 a la variable i.

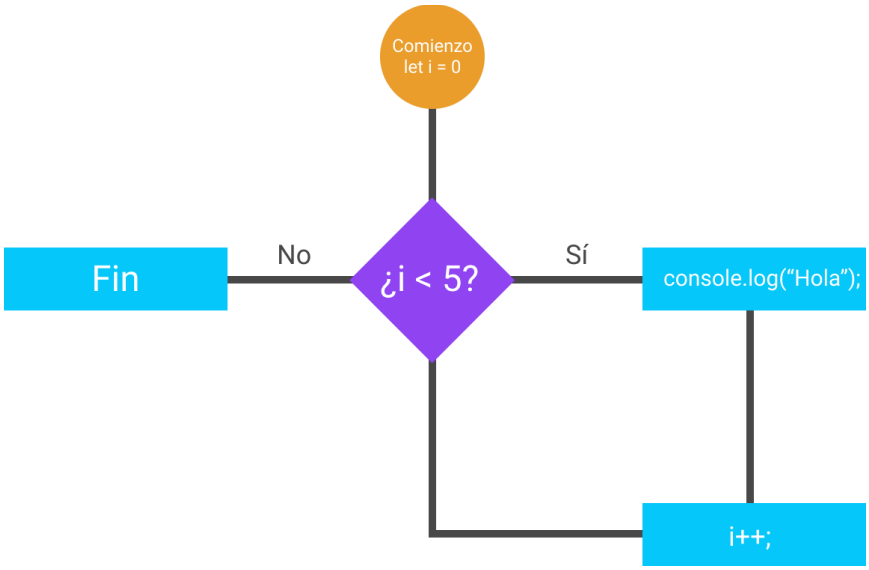
Vamos a ver por cada vuelta cuál es el valor de i

Número de vuelta	¿Cuánto vale i?	¿Se cumple la condición i < 5?	¿Ejecuta el console.log("Hola")?

Primera vuelta	0	Sí	Sí
Segunda vuelta	1	Sí	Sí
Tercera vuelta	2	Sí	Sí
Cuarta vuelta	3	Sí	Sí
Quinta vuelta	4	Sí	Sí
Sexta vuelta	5	No	No

Diagrama

Otra manera de verlo, es pensarlo en diagramas de flujo:



Analicemos las vueltas:

1. Variable i es 0. ¿0 es menor que 5?. Sí
    a. Ejecuto console.log("Hola");
    b. Le sumo 1 a i
2. Variable i es 1. ¿1 es menor que 5?. Sí
    a. Ejecuto console.log("Hola");
    b. Le sumo 1 a i
3. Variable i es 2. ¿2 es menor que 5?. Sí
    a. Ejecuto console.log("Hola");
    b. Le sumo 1 a i
4. Variable i es 3. ¿4 es menor que 5?. Sí
    a. Ejecuto console.log("Hola");
    b. Le sumo 1 a i
5. Variable i es 4. ¿4 es menor que 5?. Sí
    a. Ejecuto console.log("Hola");
    b. Le sumo 1 a i
6. Variable i es 5. ¿5 es menor que 5?. No, es igual.
    a. Fin del bucle.

Práctica

El concepto de los bucles es un concepto complicado, por eso te recomendamos hacer mucha práctica porque es solo cuestión de práctica lo que hace al entendimiento.

Mirá los ejercicios resueltos en el repositorio de Github leyendo en detenimiento los enunciados y los resultados paso a paso, para ir comprendiendo cómo armar distintos loops.