Bucle For

for loop i, for loop j, for loop k

aprendé programando Virtual

SECUNDARIA del futuro

UTN F r a

GyE Gestión y Evaluación Educativa

For

Ya vimos en clases anteriores la sintaxis y utilidad del bucle for. Hay veces - pocas - en que se necesitan recorrer matrices, es decir arrays dentro de arrays (entre otras cosas) y para eso necesitamos utilizar dos bucles. Recordemos un poco cómo funcionaba el bucle for:

Nos permite crear un bucle, es decir repetir un código una y otra vez.
Se utiliza para realizar tareas que se repiten una y otra vez.

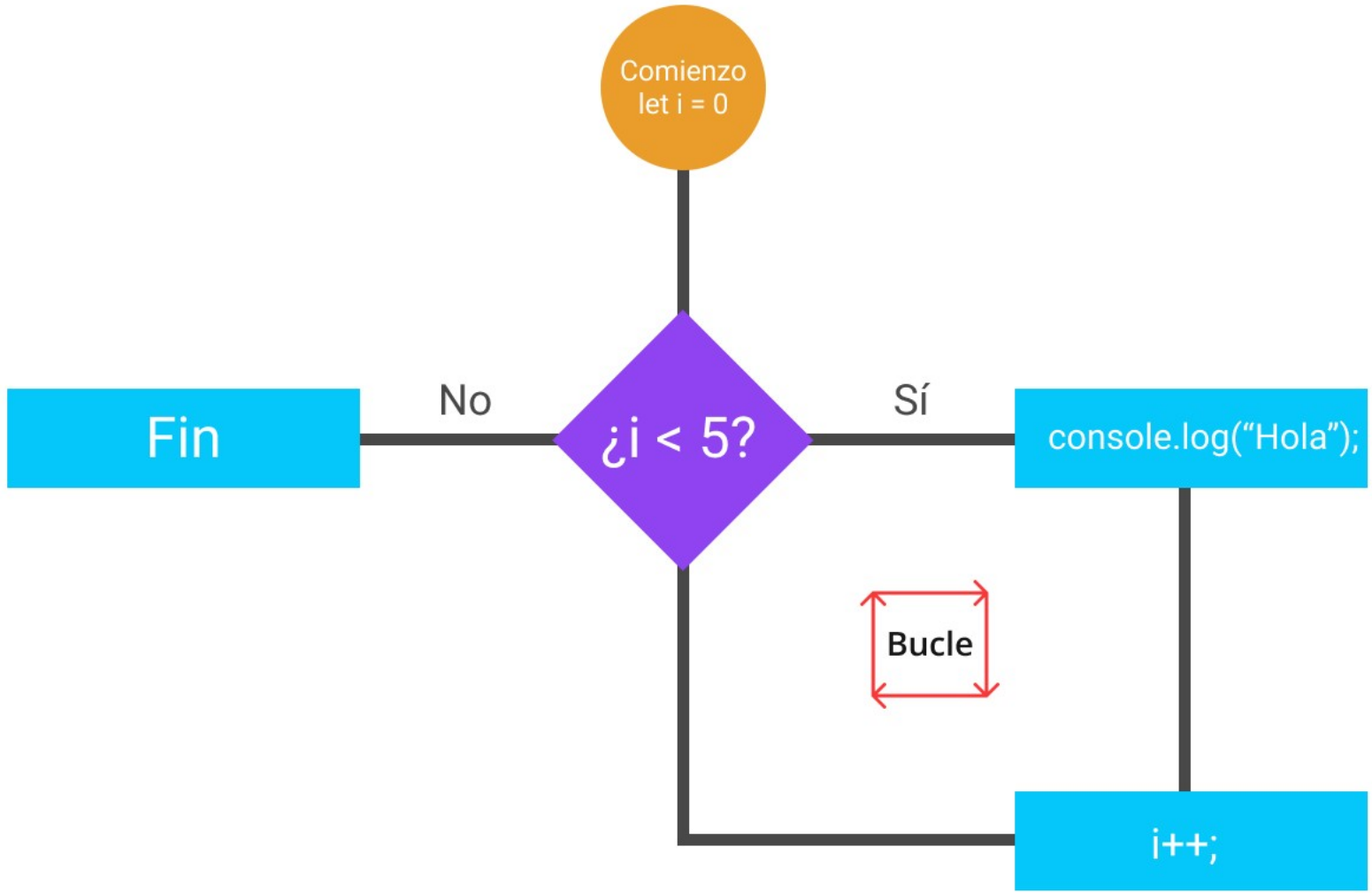
Utiliza la palabra reservada **for** y recibe 3 expresiones dentro de los paréntesis

for ([expresion-inicial]; [condicion]; [expresion-final]) { código a ejecutar }

expresión inicial: variable inicial
condición: condición de corte
expresión final: acción de incremento

For

Para entender el ciclo for, vamos a pensar primero en un diagrama:



For

Todas las expresiones juntas, escritas en código se ven de la siguiente manera:

```
for ( let i = 0; i < 5; i ++) {
    código a ejecutar
}
```

expresión inicial: definimos una variable i que comienza en 0

condición: definimos la condición de corte, en este caso se debe cumplir que i sea menor que 5

expresión final: luego de cada vuelta, se ejecuta esta parte del código, que en este caso sumamos 1 a la variable i

For y arrays

Además de utilizar el bucle for para repetir código, una de las funciones más importantes es la de poder recorrer arrays.

Tenemos el siguiente array:

```
let nombres = [“Juan”, “Mercedes”, “Sofía”, “Lucas”, “Luca”];
                0        1         2        3        4
```

y el siguiente bucle:

```
for ( let i = 0; i < 5; i ++) {
   código a ejecutar
}
```

¿Qué pasa si hacemos lo siguiente?

```
for ( let i = 0; i < 5; i ++) {
   console.log('Hola ' + nombres[i]);
}
```

Hola Juan
Hola Mercedes
Hola Sofía
Hola Lucas
Hola Luca

Veremos lo siguiente en la consola

For y arrays

Analicemos qué es lo que pasó

```
let nombres = [“Juan”, “Mercedes”, “Sofía”, “Lucas”, “Luca”];
                0         1          2        3        4
```

Viendo este for, sabemos que se va a ejecutar 5 veces.

```
for ( let i = 0; i < 5; i ++) {
   console.log(‘Hola ’ + nombres[i]);
}
```

1er Vuelta	2da Vuelta	3er Vuelta	4ta Vuelta	5ta Vuelta
i = 0	**i = 1**	**i = 2**	**i = 3**	**i = 4**
nombres[0] = “Juan”	***nombres[1] = “Mercedes”***	***nombres[2] = “Sofía”***	***nombres[3] = “Lucas”***	***nombres[4] = “Luca”***

La variable i se reemplaza por el valor que tiene en cada vuelta

For y arrays

Evaluando el array en [i], siendo **i** la variable que va cambiando, podemos recorrer un array por medio de un bucle for.

Ahora... ¿qué pasa si cambiamos la condición de corte, y en vez de i < 5, ponemos i < 6?

```
for ( let i = 0; i < 6; i ++) {
   console.log('Hola ' + nombres[i]);
}
```

1er Vuelta	2da Vuelta	3er Vuelta	4ta Vuelta	5ta Vuelta	6ta Vuelta
i = 0	**i = 1**	**i = 2**	**i = 3**	**i = 4**	**i = 5**
nombres[0] = "Juan"	***nombres[1] = "Mercedes"***	***nombres[2] = "Sofía"***	***nombres[3] = "Lucas"***	***nombres[4] = "Luca"***	***nombres[5] = undefined***

Como el array solo tenía 5 posiciones, si quiere buscar la 6ta, da como resultado "**undefined**" o que **no está definido**.

For y arrays

Para evitar este tipo de problemas, se usa la propiedad de los arrays .length que nos devuelve el número de elementos que tiene el array. De esta forma si el array cambia, el bucle queda actualizado siempre.

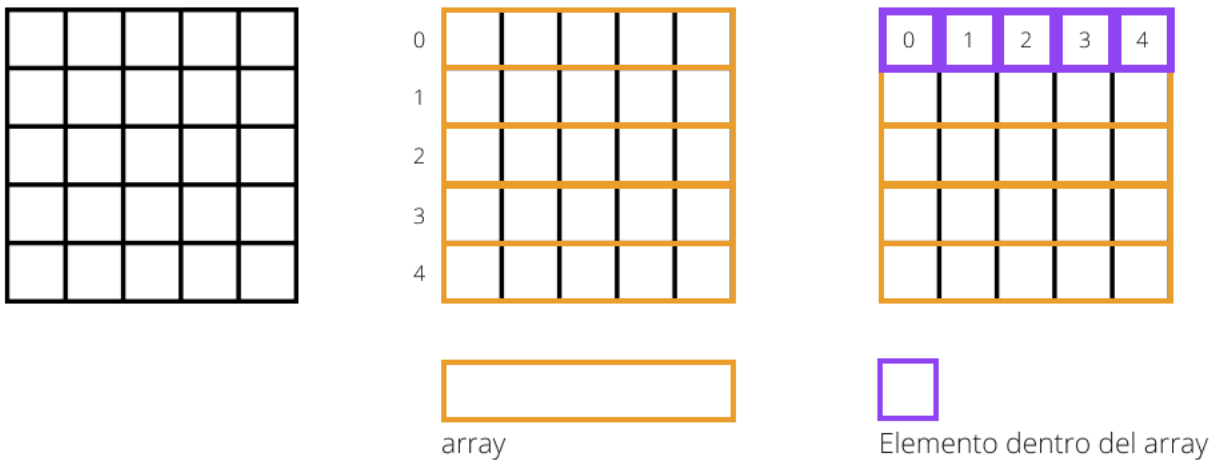
```
let nombres = [“Juan”, “Mercedes”, “Sofía”, “Lucas”, “Luca”];
                0        1          2        3        4

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

El .length va a evitar que el bucle se exceda del número de elementos, cosa que puede pasar si lo escribimos a mano.

1er Vuelta	2da Vuelta	3er Vuelta	4ta Vuelta	5ta Vuelta
i = 0	**i = 1**	**i = 2**	**i = 3**	**i = 4**
nombres[0] = “Juan”	***nombres[1] = “Mercedes”***	***nombres[2] = “Sofía”***	***nombres[3] = “Lucas”***	***nombres[4] = “Luca”***

Matrices

Una matriz no es más que un array, dentro de otro array. Es como jugar a la batalla naval en donde trabajamos con un tablero de dos dimensiones, o pensar en un cuaderno cuadriculado en donde cada fila es un array y cada cuadrado un elemento del mismo.

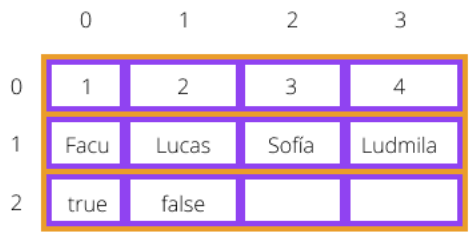


Cada fila, tiene un subíndice, pero también lo tiene (como ya vimos) cada elemento del array.

Matrices

Para crear una matriz en código, es tan simple como pensar arrays dentro de arrays:

const matriz = [[1,2,3,4], [“Facu”, “Lucas”, “Sofía”, “Ludmila”], [true,false]]



Si queremos acceder al primer array [1,2,3,4] pensamos:

1. El nombre del array
2. La posición en la que se encuentra el array que queremos (en este caso 0)

matriz[0] // **[1,2,3,4]**

---

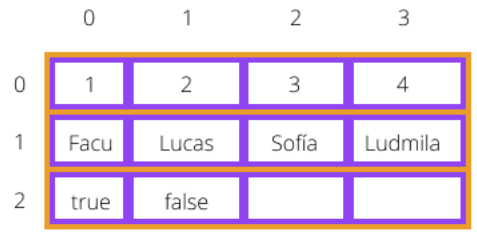
matriz[0] = [1,2,3,4]

matriz[1] = [“Facu”, “Lucas”, “Sofía”, “Ludmila”]

matriz[2] = [true,false]

Matrices

Ahora que accedimos a los arrays del primer nivel que queremos, simplemente podemos acceder a sus elementos de la manera tradicional, evaluándo en subindices.



matriz[0][1] = 2

matriz[1][3] = “Ludmila”

matriz[2][0] = true

Si pensamos en bucles for, podemos pensar en un gran for para recorrer los primeros grandes arrays, y un nuevo for para recorrer los elementos de esos arrays, es decir:

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

1er Vuelta	2da Vuelta	3er Vuelta
i = 0	**i = 1**	**i = 2**
[1,2,3,4]	***[“Facu”, “Lucas”, “Sofía”, “Ludmila”]***	***[true,false]***

Matrices

Si pensamos ahora en que por cada vuelta tenemos un array simple, podemos pensar un nuevo bucle for para recorrer ese array simple. Es como emprezar de nuevo, pensando en recorrer un solo array.

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

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

}
```

1er Vuelta con array en **i**

matriz[0] *[1,2,3,4]*
matriz[0].length = 4

Vueltas con array en j

para j = 0	**matriz[0][0] = 1**
para j = 1	**matriz[0][1] = 2**
para j = 2	**matriz[0][2] = 3**
para j = 3	**matriz[0][3] = 4**

2da Vuelta con array en **i**

matriz[1] *["Facu", "Lucas", "Sofía", "Ludmila"]*
matriz[1].length = 4

Vueltas con array en j

para j = 0	**matriz[1][0] = Facu**
para j = 1	**matriz[1][1] = Lucas**
para j = 2	**matriz[1][2] = Sofía**
para j = 3	**matriz[1][3] = Ludmila**

Matrices

Por convención, cuando anidamos bucles for, utilizamos la letra **i** para el for más genérico, la letra **j** para uno más interno, la letra **k** para uno más interno.

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

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

}
```

Si bien el uso de matrices no es lo que se presenta en el día a día, si nos puede servir para la resolución y ordenamientos de datos.