

Bootcamp Java Developer

Fase 2 - Java Web Developer
Módulo 18

Bucles

For

Vamos a suponer que queremos **recorrer un arreglo de empleados**. Veamos los ejemplos de la derecha.

Esta forma es tediosa y molesta para trabajar.

```
<script>
var empleados = ['Juan', 'Roberto', 'Maria', 'Ana']

</script>
```

```
1  var empleados = ['Juan', 'Roberto', 'Maria', 'Ana']
2  var mensaje = '';
3
4  mensaje += empleados[0] + ' || '
5  mensaje += empleados[1] + ' || '
6  mensaje += empleados[2] + ' || '
7  mensaje += empleados[3] + ' || '
8
9  console.log(mensaje);
10
```

De esta manera será mucho más sencillo:

```
1  for(var i = 0; i < 2; i++)  
2    |  console.log(i);  
3
```



El contador **i** deja de incrementar luego del primer nombre o dato, porque se ha fijado que **i nunca debe ser mayor que 2**. De esta manera, se repite un ciclo (aumentar en este caso **i**) en tanto y en cuanto el mismo no supere a 2.

```
1  var empleados = ['Juan', 'Roberto', 'Maria', 'Ana'];
2  var mensaje = '';
3
4  for (var i = 0; i < 2; i++)
5  |   console.log(empleados[i]);
6
```

Nota: La realidad es que este ejemplo carece de sentido, a menos que lo usemos, por ejemplo, para cuestiones más prácticas como **iterar un arreglo**.

El resultado mostrará el nombre de cada empleado siempre que no supere al índice del arreglo 2, es decir, hasta *Roberto*:

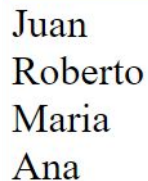
```
1  var empleados = ['Juan', 'Roberto', 'Maria', 'Ana']
2  var mensaje = '';
3
4  for (var i = 0; i < 2; i++)
5      mensaje += empleados[i] + ' || '
6
7  console.log(mensaje);
8
```

¿Qué pasaría si no conocemos la cantidad exacta de ítems a mostrar? ¿Cómo se puede resolver eso?

La propiedad `length` de cualquier *array* devuelve la cantidad actual de ítems de dicho *array*.

```
1  var empleados = ['Juan', 'Roberto', 'Maria', 'Ana']
2  var mensaje = '';
3
4  for (var i = 0; i < empleados.length; i++)
5      mensaje += empleados[i] + ' || '
6
7  console.log(mensaje);
8
```

De esta manera siempre **se adaptará la condición a la cantidad de datos que tenga en el arreglo**, por ejemplo:



Juan
Roberto
Maria
Ana

Luego se puede agregar a *Pedro*. El contenedor automáticamente **mostrará todos los datos del arreglo**, sin necesidad que se deba variar el límite:

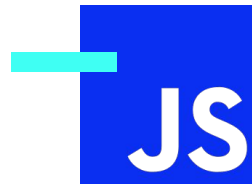
```
var empleados = ['Juan', 'Roberto', 'Maria', 'Ana', 'Pedro']  
var mensaje = ""
```


While

El bucle **while** es similar al **for**, pero tiene ciertas diferencias que son importantes establecer.

Veamos su sintaxis:

```
while (i < 10) {  
    mensaje += "El dato es " + i;  
    i++;  
}
```



Por ejemplo:

```
1  var empleados = ['Juan', 'Roberto', 'Maria', 'Ana', 'Pedro']
2  var mensaje = '';
3  var i = 0;
4
5  while (i < empleados.length) {
6      mensaje += empleados[i] + ' || '
7      i++
8  }
9
10 console.log(mensaje);
11
```

Iteración de arreglos con objetos


Para recorrer un arreglo con objetos, lo haremos de la siguiente manera:



```
1  var empleados = [  
2      { nombre: 'Juan', apellido: 'Pedro', edad: 40 },  
3      { nombre: 'Ana', apellido: 'Maria', edad: 25 },  
4      { nombre: 'María', apellido: 'Zarate', edad: 34 }  
5  ]  
6  
7  var mensaje = '';  
8  var i = 0;  
9  
10 while (i < empleados.length) {  
11     mensaje += empleados[i].nombre + ' || '  
12     i++  
13 }  
14  
15 console.log(mensaje);
```

Revisión

- Repasar el concepto de **bucle**.
- Mostrar datos de un arreglo con **for**.
- Mostrar datos de un arreglo con **while**.
- Mostrar datos de un **arreglo con objetos** que **contengan propiedades**.
- Trabajar en el **Proyecto Integrador**.
- Realizar las preguntas necesarias al/la docente antes de continuar.



```
for(...)  
while(...)
```

**¡Sigamos
trabajando!**

