Bucles

Los bucles nos van a permitir iterar por listas, para recorrerlas y realizar operativa sobre cada uno de sus items.

Bucle 'for'

Es el bucle más básico y funcional, lo vimos en la sesión anterior. Recordad que entre paréntesis lleva la inicialización, la condición de ruptura y el incremento (separados por `;`):

```
for (var index = 0; index < array.length; index++) {
  var element = array[index];
}
```

Bucle 'foreach'

Podemos construir un bucle for, pero con una estructura más **funcional** gracias al `foreach`. En ocasiones puede sernos muy útil generar una función para ejecutarla por cada vuelta de bucle:

```
var myArray = ['Alberto', 'Mario', 'Jose', 'Juan'];

myArray.forEach(
  function (element) {
    console.log(element);
  }
);
```

```
/* Otro ejemplo */

var myArray = ['Alberto', 'Mario', 'Jose', 'Juan'];

var imprime = function(element) {
  console.log(element);
}

myArray.forEach(imprime);
```

Bucle 'for..of'

El bucle **for-of** es un bucle que automáticamente nos devuelve los `valores`. Este bucle lo que nos permite es iterar a través de los **elementos de objetos iterables** como, por ejemplo, **String, Array, Set, Map,** etc.

```
for (value of iterable_obj) { ... }
```

Vamos a ver unos ejemplos con algunos de los tipos de objetos que hemos comentado antes que permite iterar:

```
// Iteración sobre Array
```

```
var justiceLeague = ['Batman', 'Aquaman', 'Superman'];
for (var justice of justiceLeague) {
  console.log(justice);
}

// Iteración sobre String

var clark = 'Superman';
for (var who of clark) {
  console.log(who);
}

// Iteración sobre Arguments (los argumentos/parámetros de una función) 🤯

function crazyContainer() {
  for (var value of arguments) {
    console.log(value);
  }
}
crazyContainer(justiceLeague, 8, clark);
```

Bucle 'for...in'

Para poder recorrer las `claves` de un objeto JavaScript nos ofrece la función **for in**: Mediante la cual recorremos todos los índices del objeto, de manera que podemos ir accediendo a cada una de sus propiedades.

```
// Definimos un objeto con los datos de una spiderman
var spiderman = {
    nombre: "Peter",
    apellidos: "Parker",
    pais: "USA",
    profesion: "Student"
}

for (var key in spiderman) {
    console.log("Spiderman tiene " + key + " con valor: " + spiderman[key]);
}
```

'for...of' vs. 'for...in'

Una de las diferencias es que **for-of** solamente puede iterar en objetos iterables, en cambio, **for-in** puede iterar en cualquier tipo de objeto. Otra diferencia, **es que for-in devuelve las claves y for-of los valores.**

Vamos a hacer una prueba del uso de **for-of** y de **for-in** seguro que os sorprende los resultados:

```
var dieHardArray = [1, 2,'Simon', 'John McClane', 'Zeus Carver'];

var dieHardObj = {
  name: 'John',
  surname: 'McClane',
  age: 37
};

// Iterar un Array
for (value of dieHardArray) {
  console.log(value);
}

for (key in dieHardArray) {
  console.log(key);
}


// Iterar un Objeto

for (key in dieHardObj) {
  console.log(key);
}

for (value of dieHardObj) {
  console.log(value);
}
```

'for...of' vs. 'for...each'

La principal diferencia es que **for-of** puede iterar en cualquier tipo de objeto iterable, en cambio, **.forEach** solamente puede en arrays.

Lo vemos con un ejemplo:

```
var backToTheFutureArray = [21, 10, 2015, 'Delorean'];
var backToTheFutureString = 'Dr.Emmett Brown';

// Iterar un Array
for (value of backToTheFutureArray) {
  console.log(value);
}

backToTheFutureArray.forEach(function(value, index) {
  // podemos acceder al índice
  console.log(value, index);
});


// Iterar un String
for (value of backToTheFutureString) {
  console.log(value);
}

backToTheFutureString.forEach(function(value, index) {
  console.log(value, index);
});
```

U Resumen de bucles