Módulo 5 - Arrays y bucles - Introducción

Durante el desarrollo vamos a necesitar un tipo de estructura o matriz que nos permita tener una lista de elementos o variables, y una forma de poder recorrer esa lista para acceder a cada uno.

Imaginemos por un momento que estamos desarrollando un programa para anotar todas las notas de los alumnos de una clase, y su nota media. Para cada alumno necesitaremos una lista donde anotar cada una de sus notas, y una forma de recorrerla para calcular la media de todas ellas.

Javascript, al igual que la mayoría de los lenguajes, nos permite crear esta estructura de tipo lista mediante los llamados Arrays, y para recorrer la lista de forma iterativa utilizaremos Bucles.

Arrays

Los Arrays nos van a permitir mantener una lista de elementos. Utilizamos corchetes [ ] para representarlos:

```
[1, 1, 2, 3, 5, 8, 13, 21];
```

Y normalmente asignaremos las estructuras de tipo Array a variables:

```
var myArray = [1, 1, 2, 3, 5, 8, 13, 21];
```

Crear Arrays

Las siguientes declaraciones crean Arrays equivalentes:

```
var arr = new Array(element0, element1, ..., elementN);
var arr = Array(element0, element1, ..., elementN);
var arr = [element0, element1, ..., elementN];
```

Contenido de los Arrays

Los Arrays pueden almacenar cualquier tipo de variable o elemento en cada una de sus posiciones: números, cadenas, objetos javascript, funciones, incluso otros Arrays.

```
var myNumberArray = [1, 1, 2, 3, 5, 8, 13, 21];
var myStringArray = ["John", "Doe", "Will"];
var myTwoDimensionArray = [myNumberArray, myStringArray];
```

Acciones sobre Arrays

Vamos a ver algunas acciones que podemos realizar sobre los Arrays. Para cada uno de los ejemplos, partiremos de cero con el siguiente Array que contiene nombres de ciudades:

```
var ciudades = ["Bali", "Roma", "Berlín"];
```

· Eliminar elemento del final: pop

```
var ultimaCiudad = ciudades.pop();

// ciudades: ["Bali", "Roma"]
// ultimaCiudad: "Berlín"
```

· Añadir elemento al final: push

```
var longitud = ciudades.push("Río");

// ciudades: ["Bali", "Roma", "Berlín", "Río"]
// longitud: 4
```

· Eliminar elemento del principio: shift

```
var primeraCiudad = ciudades.shift();

// ciudades: ["Roma", "Berlín"]
// primeraCiudad: "Bali"
```

· Añadir elemento al inicio: unshift

```
var longitud = ciudades.unshift("Buenos Aires");

// ciudades: ["Buenos Aires", "Bali", "Roma", "Berlín"]
// longitud: 4
```

· Encontrar el índice de un elemento: indexOf

```
var pos = ciudades.indexOf("Berlín");

// pos: 2
```

· Eliminar un elemento con el índice: splice

```
var elementosEliminados = ciudades.splice(1, 1);
// ciudades: ["Bali", "Berlín"]
// elementosEliminados: ["Roma"]

// <array>.splice(pos, n);
// así es como se eliminan elementos, n define la cantidad de elementos a eliminar,
// de esa posicion(pos) en adelante hasta el final del array.

var elementosEliminados = ciudades.splice(1, 2);

// ciudades: ["Bali"]
// elementosEliminados: ["Roma", "Berlín"]
```

· Copiar un Array: slice

```
var copia = ciudades.slice();

// ciudades: ["Bali", "Roma", "Berlín"]
// copia: ["Bali", "Roma", "Berlín"]
```

Acceder al elemento contenido en una posición del Array

Podemos acceder a un elemento del Array a partir de su posición (índice o index). Los índices en los Arrays son valores numéricos, que **empiezan por 0**, y se indican utilizando corchetes sobre la variable que representa el Array:

```
myArray[0]; // accedemos al elemento contenido en la posición 0
myArray[1]; // accedemos al elemento contenido en la posición 1
// ...
myArray[n]; // accedemos al elemento contenido en la posición 'n' del array
```

Podemos saber la longitud (número de elementos del Array) por su propiedad *length*:

```
var myArray = ["John", "Doe", "Will"];
myArray.length; // 3
```

Podemos incrementar gradualmente 'n' desde 0 hasta la última posición del array (myArray.length - 1), para ir accediendo a cada uno de sus elementos de forma repetitiva (iterativa): **¡bienvenidos bucles!**

Bucles

Ya tenemos nuestros datos en un Array, ahora vamos a recorrerlo accediendo a sus elementos utilizando bucles.

Concepto

Los bucles nos ofrecen una manera rápida y sencilla de hacer algo de manera repetitiva (iteraciones). Con los Arrays, los utilizaremos para recorrer sus elementos.

Tipos

Con javascript disponemos de varios tipos de bucles:

- for - recorre un bloque de código varias veces.
- for/in - recorre las propiedades de un objeto.
- for/of - recorre los valores de un objeto iterable.
- while - recorre un bloque de código mientras una condición específica es verdadera.
- do/while - también recorre un bloque de código mientras una condición específica es verdadera, pero la computa al final de cada iteración.

Sentencia for

Los bucles *for* tienen la siguiente estructura:

```
for (<inicialización>; <condición>; <post-ejecución>) {
  <bloque de código>
}
```

- **Inicialización** - se ejecuta **antes de comenzar el bucle**. Normalmente se utiliza para inicializar la variable que se tendrá en cuenta en la condición de finalización.
- **Condición** - se ejecuta **antes de cada iteración** del bucle, y determina si se debe continuar o salir del mismo.
- **Post-ejecución** - se ejecuta **después de cada iteración**. Normalmente se utiliza para modificar las variables que se tienen en cuenta en la condición del bucle.
- **Bloque de código** - bloque de código que se ejecutará en cada iteración.

En el siguiente ejemplo, el bucle se ejecutará 5 veces. Se ejecutará el bloque de código (mostrar el valor de *i* por consola) desde que la variable *i* vale 1 (inicialización) mientras que tenga valor menor o igual a 5 (condición), aumentando una unidad por cada iteración (post-ejecución)):

```
for (var i = 1; i <= 5; i++) {
  console.log(i);
}
```

Podemos utilizar este tipo de Bucles para acceder a los elementos de un Array, iterando por sus posiciones:

- **Inicialización**: variable con valor 0 (recordemos que los índices de los Array comienzan con 0).
- **Condición**: la variable de iteración sea menor que la longitud del Array.
- **Post-ejecución**: aumentar una unidad la variable de iteración.
- **Bloque de código**: mostrar por consola el valor de la posición correspondiente del Array.

```
var ciudades = ["Bali", "Roma", "Tokio", "Montevideo"];
for (var i = 0; i < ciudades.length; i++) {
  console.log("Estamos visitando " + ciudades[i]);
}
```

Igualmente, podemos recorrer el mismo Array en sentido inverso:

```
var ciudades = ["Bali", "Roma", "Tokio", "Montevideo"];
for (var i = ciudades.length - 1; i >= 0; i--) {
  console.log("Estamos visitando " + ciudades[i]);
}
```

Ejemplo: vamos a calcular el total de todas las reservas de un hotel a partir de un listado de reservas:

```
var reserva1 = { id: 1, price: 24.5, client: "Caroline" };
var reserva2 = { id: 2, price: 61.5, client: "Liam" };
var reserva3 = { id: 3, price: 49.85, client: "Andrew" };
var reserva4 = { id: 4, price: 37.1, client: "John" };

var reservas = [reserva1, reserva2, reserva3, reserva4];

//Ejemplo de bucle:

var total = 0;
for (var i = 0; i < reservas.length; i++) {
  total += reservas[i].price;
}

// total: 172.95
```

Sentencia for...in

Este tipo de bucle recorre las propiedades de un objeto. En cada iteración tendremos disponible una de las propiedades del objeto que estamos recorriendo.

Un bucle for...in se escribe de la siguiente forma:

```
for (var variable in objeto) {
  // sentencias
}
```

Por ejemplo, para recorrer los atributos de un objeto User, con campos: id, name, age, registered:

```
var user = { id: 1, name: "John", age: 20, registered: true };
for (var field in user) {
  console.log(field + ": " + user[field]);
}

/*
id: 1
name: John
age: 20
registered: true
*/
```

Al aplicar un bucle for...in sobre un array accedemos a sus indices:

```
var ciudades = ["Bali", "Roma", "Tokio", "Montevideo"];
for (var i in ciudades) {
  console.log("Estamos visitando " + ciudades[i]);
}

/*
"Estamos visitando Bali"
"Estamos visitando Roma"
"Estamos visitando Tokio"
"Estamos visitando Montevideo"
*/
```

Ejemplo: vamos a calcular el total de todas las reservas de un hotel a partir de un listado de reservas:

```
var reserva1 = { id: 1, price: 24.5, client: "Caroline" };
var reserva2 = { id: 2, price: 61.5, client: "Liam" };
var reserva3 = { id: 3, price: 49.85, client: "Andrew" };
var reserva4 = { id: 4, price: 37.1, client: "John" };
var reservas = [reserva1, reserva2, reserva3, reserva4];

//Ejemplo de bucle:

var total = 0;
for (var i in reservas) {
  total += reservas[i].price;
}

// total: 172.95
```

Sentencia for...of

Este tipo de bucle recorre un array accediendo a los valores de cada posición (no a sus índices como el caso anterior). En cada iteración tendremos disponible el elemento de la posición correspondiente.

```
var ciudades = ["Bali", "Roma", "Tokio", "Montevideo"];
for (var ciudad of ciudades) {
  console.log("Estamos visitando " + ciudad);
}

/*
"Estamos visitando Bali"
"Estamos visitando Roma"
"Estamos visitando Tokio"
"Estamos visitando Montevideo"
*/
```

Ejemplo: vamos a calcular el total de todas las reservas de un hotel a partir de un listado de reservas:

```
var reserva1 = { id: 1, price: 24.5, client: "Caroline" };
var reserva2 = { id: 2, price: 61.5, client: "Liam" };
var reserva3 = { id: 3, price: 49.85, client: "Andrew" };
var reserva4 = { id: 4, price: 37.1, client: "John" };
var reservas = [reserva1, reserva2, reserva3, reserva4];

//Ejemplo de bucle:

var total = 0;
for (var reserva of reservas) {
  total += reserva.price;
}

// total: 172.95
```

Sentencia while

Un bucle while se ejecuta mientras la condición sea verdadera, no tiene inicialización de la condición, ni post incremento como en un bucle for. Por tanto, las variables que forman parte de la condición deben ser modificadas dentro del bloque de código que ejecuta el bucle para que, en algún momento, finalice. **La condición se evalúa antes de ejecutar el bloque de código del bucle**.

```
var ciudades = ["Bali", "Roma", "Tokio", "Montevideo"];
var i = 0;
while (i < ciudades.length) {
  console.log("Estamos visitando " + ciudades[i]);
  i++;
}
```

Ejemplo: vamos a calcular el total de todas las reservas de un hotel a partir de un listado de reservas:

```
var reserva1 = { id: 1, price: 24.5, client: "Caroline" };
var reserva2 = { id: 2, price: 61.5, client: "Liam" };
var reserva3 = { id: 3, price: 49.85, client: "Andrew" };
var reserva4 = { id: 4, price: 37.1, client: "John" };
var reservas = [reserva1, reserva2, reserva3, reserva4];

//Ejemplo de bucle:

var total = 0;
var i = 0;
while (i < reservas.length) {
  total += reservas[i].price;
  i++;
}

// total: 172.95
```

Sentencia do... while

Un bucle do...while se repite mientras la condición especificada sea verdadera. **A diferencia del bucle while, esta sentencia ejecuta el bloque de código antes de evaluar la condición.** Por tanto, la primera iteración siempre se ejecutará.

```
var ciudades = ["Bali", "Roma", "Tokio", "Montevideo"];
var i = 0;

do {
  console.log("Estamos visitando " + ciudades[i]);
  i++;
} while (i < ciudades.length);
```

Ejemplo: vamos a calcular el total de todas las reservas de un hotel a partir de un listado de reservas:

```
var reserva1 = { id: 1, price: 24.5, client: "Caroline" };
var reserva2 = { id: 2, price: 61.5, client: "Liam" };
var reserva3 = { id: 3, price: 49.85, client: "Andrew" };
var reserva4 = { id: 4, price: 37.1, client: "John" };
var reservas = [reserva1, reserva2, reserva3, reserva4];

//Ejemplo de bucle:

var total = 0;
var i = 0;
do {
  total += reservas[i].price;
  i++;
} while (i < reservas.length);

// total: 172.95
```