8. Arreglos [Arrays]

Los **arreglos** (o arrays) son una de las estructuras de datos más importantes en JavaScript. Permiten almacenar colecciones ordenadas de valores, como números, strings, objetos u otros arrays. En este capítulo, exploraremos cómo crear, manipular y trabajar con arreglos, además de aprender sobre sus métodos integrados para realizar operaciones comunes.

¿Qué Son los Arreglos?

Un arreglo es una lista ordenada de elementos, donde cada elemento tiene un índice numérico que indica su posición en el arreglo. Los índices comienzan en `0`, lo que significa que el primer elemento está en la posición `0`, el segundo en la posición `1`, y así sucesivamente.

Ejemplo:

```
let frutas = ["manzana", "plátano", "uva"];
console.log(frutas[0]); // manzana
console.log(frutas[1]); // plátano
```

Creación de Arreglos

Existen varias formas de crear arreglos en JavaScript:

1. Notación Literal

Es la forma más común y recomendada de crear arreglos.

```
let numeros = [1, 2, 3, 4];
let colores = ["rojo", "verde", "azul"];
```

2. Constructor `Array`

Puedes usar el constructor `Array` para crear un arreglo. Sin embargo, esta forma es menos común y puede ser confusa en algunos casos.

```
let numeros = new Array(1, 2, 3, 4);
let colores = new Array.of("rojo", "verde", "azul");
```

> Nota: Si pasas un solo número al constructor Array, crea un arreglo vacío con esa longitud.

```
let arregloVacio = new Array(3); // [ <3 empty items> ]
```

Luego lo puedes rellenar con el método fill

```
const cienGaviotas = Array(100).fill("Gaviota")
```

Propiedades y Métodos Básicos

Propiedad `length`

La propiedad `length` devuelve el número de elementos en un arreglo.

```
let frutas = ["manzana", "plátano", "uva"];
console.log(frutas.length); // 3
```

Acceder a Elementos

Puedes acceder a un elemento específico usando su índice.

```
let frutas = ["manzana", "plátano", "uva"];
console.log(frutas[1]); // plátano
```

Modificar Elementos

Puedes modificar un elemento asignando un nuevo valor a un índice específico.

```
let frutas = ["manzana", "plátano", "uva"];
frutas[1] = "naranja";
console.log(frutas); // ["manzana", "naranja", "uva"]
```

Métodos Comunes de los Arreglos

JavaScript proporciona una amplia gama de métodos integrados para manipular arreglos. A continuación, exploraremos los más utilizados.

1. Agregar y Eliminar Elementos

`push(elemento)`

Agrega uno o más elementos al final del arreglo.

```
let frutas = ["manzana", "plátano"];
frutas.push("uva");
console.log(frutas); // ["manzana", "plátano", "uva"]
```

`pop()`

Elimina el último elemento del arreglo y lo devuelve.

```
let frutas = ["manzana", "plátano", "uva"];
let ultimaFruta = frutas.pop();
```

```
console.log(frutas); // ["manzana", "plátano"]
console.log(ultimaFruta); // uva
```

unshift(elemento)

Agrega uno o más elementos al inicio del arreglo.

```
let frutas = ["manzana", "plátano"];
frutas.unshift("uva");
console.log(frutas); // ["uva", "manzana", "plátano"]
```

shift()

Elimina el primer elemento del arreglo y lo devuelve.

```
let frutas = ["manzana", "plátano", "uva"];
let primeraFruta = frutas.shift();
console.log(frutas); // ["plátano", "uva"]
console.log(primeraFruta); // manzana
```

---

2. Iteración sobre Arreglos

forEach(callback)

Ejecuta una función para cada elemento del arreglo.

```
let numeros = [1, 2, 3];
numeros.forEach(function(numero, indice) {
      console.log(indice + " X " + numero + " = " + indice * numero);
});
// Salida:
// 0 X 1 = 0
// 1 X 2 = 2
// 2 X 3 = 6
```

map(callback)

Crea un nuevo arreglo con los resultados de aplicar una función a cada elemento.

```
let numeros = [1, 2, 3];
let duplicados = numeros.map(function(numero) {
   return numero * 2;
});
console.log(duplicados); // [2, 4, 6]
```

---

3. Búsqueda y Filtrado

indexOf(valor)

Devuelve el índice de la primera aparición de un valor en el arreglo. Si no se encuentra, devuelve `-1`.

```
let frutas = ["manzana", "plátano", "uva"];
console.log(frutas.indexOf("plátano")); // 1
console.log(frutas.indexOf("cereza")); // -1
```

includes(valor)

Verifica si un valor existe en el arreglo.

```
let frutas = ["manzana", "plátano", "uva"];
console.log(frutas.includes("uva")); // true
console.log(frutas.includes("cereza")); // false
```

filter(callback)

Crea un nuevo arreglo con todos los elementos que cumplan una condición.

```
let numeros = [1, 2, 3, 4, 5];
let pares = numeros.filter(function(numero) {
  return numero % 2 === 0;
});
console.log(pares); // [2, 4]
```

---

4. Ordenamiento y Reversión

sort()

Ordena los elementos de un arreglo alfabéticamente o numéricamente (requiere una función de comparación para números).

```
let letras = ["b", "c", "a"];
letras.sort();
console.log(letras); // ["a", "b", "c"]

let numeros = [3, 1, 2];
numeros.sort((a, b) => a - b);
console.log(numeros); // [1, 2, 3]
```

reverse()

Invierte el orden de los elementos en el arreglo.

```
let letras = ["a", "b", "c"];
letras.reverse();
console.log(letras); // ["c", "b", "a"]
```

---

5. Reducción

`reduce(callback, valorInicial)`

Reduce el arreglo a un único valor ejecutando una función acumuladora.

```
let numeros = [1, 2, 3, 4];
let suma = numeros.reduce(function(acumulador, numero) {
   return acumulador + numero;
}, 0);
console.log(suma); // 10
```

Arreglos Multidimensionales

Un arreglo puede contener otros arreglos, lo que permite crear estructuras multidimensionales.

```
let matriz = [
   [1, 2, 3],
   [4, 5, 6],
   [7, 8, 9]
];

console.log(matriz[1][2]); // 6
```

Desestructuración de Arreglos

La desestructuración permite extraer valores de un arreglo y asignarlos a variables individuales.

```
let frutas = ["manzana", "plátano", "uva"];
let [primera, segunda] = frutas;

console.log(primera); // manzana
console.log(segunda); // plátano
```
