JavaScript

6 – Métodos de la clase Array

Insertar y borrar

- **.unshift(...items)** → Inserta al principio del array
- **.push(...items)** → Inserta al final del array
- **.shift()** → Elimina y devuelve la primera posición
- **.pop()** → Elimina y devuelve la última posición

Transformar a cadena

- Para unir los elementos de un array en una cadena separados por coma llamamos a **.toString()**
  - Convertirá a string cada valor si no lo es
- Podemos utilizar el método **.join(separador)** si queremos establecer un separador diferente de la coma.
  - Por defecto separa por coma

```
let a = [3, 21, 15, 61, 9];
console.log(a.toString()); // 3,21,15,61,9
console.log(a.join()); // 3,21,15,61,9
console.log(a.join("-#-")); // 3-#-21-#-15-#-61-#-9
```

Concatenar arrays

- El método **a1.concat(a2)** devuelve un nuevo array con los elementos de **a1** seguidos de los de **a2**
  - No modifica ningún array existente

```
let a = ["a", "b", "c"];
let b = ["d", "e", "f"];
let c = a.concat(b);
console.log(c); // ["a", "b", "c", "d", "e", "f"]
console.log(a); // ["a", "b", "c"] -> No modificado
```

.slice(inicio, fin)

- Devuelve un nuevo array con las posiciones desde **inicio** (incluida), hasta **fin** (excluida).
  - No modifica el array original

```
let a = ["a", "b", "c", "d", "e", "f"];
let b = a.slice(1, 3); // Posiciones 1 y 2
console.log(b); // ["b", "c"]
console.log(a); // ["a", "b", "c", "d", "e", "f"]
// Si se omite el segundo parámetro -> hasta el final
console.log(a.slice(3)); // ["d", "e", "f"]
```

.splice(inicio, cantidad, ...items)

- Desde la posición **inicio**, borra **cantidad** elementos. También puede insertar nuevos elementos ahí.
  - Modifica el array original
  - Devuelve los elementos eliminados

Ordenar elementos

- **.reverse()** → Orden inverso.

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

- **.sort()** → Ordena alfabéticamente los elementos del array.

```
let a = ["Pedro", "Ana", "Tomás", "Juan", "Marta"];
a.sort();
console.log(a); // ["Ana", "Juan", "Marta", "Pedro", "Tomás"]
```

Buscar elementos

- **.indexOf(valor)** → Devuelve el índice donde se encuentra el valor o -1 si no lo encuentra
- **.indexOf(valor, inicio)** → Empieza a buscar desde la posición **inicio** (por defecto 0)
- **.lastIndexOf** funciona igual pero empieza desde el final

```
let a = [3, 21, 15, 61, 9, 15];
console.log(a.indexOf(15)); // 2
console.log(a.indexOf(15, 3)); // 5
console.log(a.indexOf(56)); // -1
console.log(a.lastIndexOf(15)); // 5
```

.every y .some

- **.every** → Devuelve true si todos los elementos del array cumplen una condición
  - Recibe una función anónima que comprueba cada elemento

```
let a = [3, 21, 15, 61, 9, 54];
console.log(a.every(num =>  num < 100)); // true
console.log(a.every(num => num % 2 == 0)); // false
```

- **.some** → Devuelve true si al menos un elemento del array la cumple

```
let a = [3, 21, 15, 61, 9, 54];
console.log(a.some(num => num % 2 == 0)); // true
```

.map

- Devuelve un nuevo array pasando una función de transformación a cada elemento
  - La función anónima debe devolver el nuevo elemento de cada posición

```
let a = [4, 7, 15, 25];
let b = a.map(num => num * 2);
console.log(b); // [8, 14, 30, 50]

let words = ["rana", "sombrero", "tela", "mesa"];
let letters = words.map(word => {
    return word.charAt(0).toLocaleUpperCase();
});
console.log(letters); // ["R", "S", "T", "M"]
```

.filter

- Devuelve un nuevo array con los elementos del actual que cumplan una condición
  - La función anónima debe devolver true (se queda) o false (va fuera)

```
let nums = [4, 7, 12, 25, 8];
let pares = nums.filter(num => num % 2 === 0);
console.log(pares); // [4 ,12 ,8]

let nombres = ["Pedro", "Marta", "Jose", "Antonio", "Ana"];
let nombresConA = nombres.filter(nombre => {
    return nombre.match(/.*a.*/i);
});
console.log(nombresConA); // ["Marta", "Antonio", "Ana"]
```

.reduce

- A partir de un array devuelve un sólo valor que calcula recorriendo sus elementos de principio a fin
  - Segundo parámetro (opcional) → valor inicial
  - La función anónima recibe el valor acumulado y el elemento actual
  - Para la operación inversa (final - principio) → reduceRight

```
let nums = [3, 5, 8, 11, 12];
let suma = nums.reduce((total, num) => total + num, 0);
console.log(suma); // 39 (0 + 3 + 5 + 8 + 11 + 12)
let resta = nums.reduceRight((total, num) => total - num);
console.log(resta); // -15
```

.fill

- Rellena (inicializa) todas las posiciones del array con un valor

```
let a = new Array(6);
a.fill(0);
console.log(a); // [0, 0, 0, 0, 0, 0]

let nums = [4, 7, 12, 25, 8];
nums.fill(0, 1, 4);
console.log(nums); // [4, 0, 0, 0, 8]
```

.find y findIndex

- Busca la el primer elemento que cumple una condición
  - La condición se comprueba en una función anónima que devuelve un booleano

```
let nums = [3, 5, 8, 11, 12];
let par = nums.find(num => num % 2 === 0);
let indicePar = nums.findIndex(num => num % 2 === 0);

console.log(`En la posición ${indicePar} está el número ${par}`);
// En la posición 2 está el número 8
```