ARRAYS

CONCEPTO

CREACIÓN

LECTURA Y ESCRITURA

RECORRER LOS ARRAYS

MÉTODOS DE ARRAY

CONCEPTO

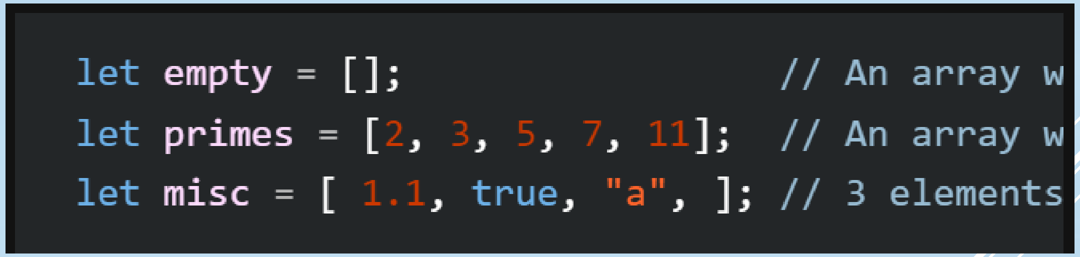
COLECCIÓN ORDENADA DE VALORES

- Cada valor es un **elemento**
- Cada valor tiene una posición conocida como **índice**
- Son **untyped**, cada valor puede ser diferente (string, otro array, números)
- **Zero-based** (el primer índice es el 0)
- **Dinámicos**: no necesitan definir un tamaño de memoria con antelación
- Heredan propiedades del **Array.prototype**

CREACIÓN

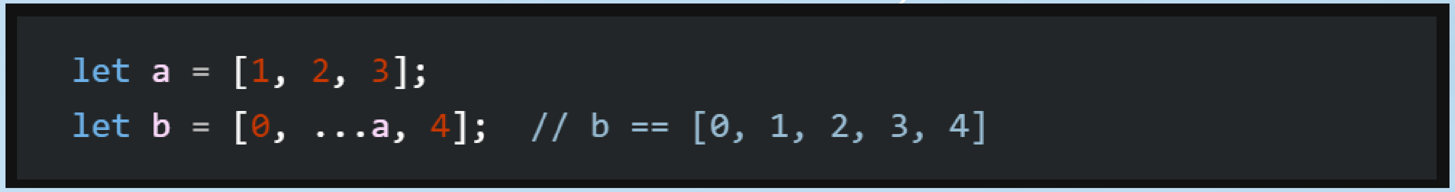
DISTINTAS MANERAS DE CREARSE

- Literales



```
let empty = [];                 // An array w
let primes = [2, 3, 5, 7, 11];  // An array w
let misc = [ 1.1, true, "a", ]; // 3 elements
```

- Operador Spread

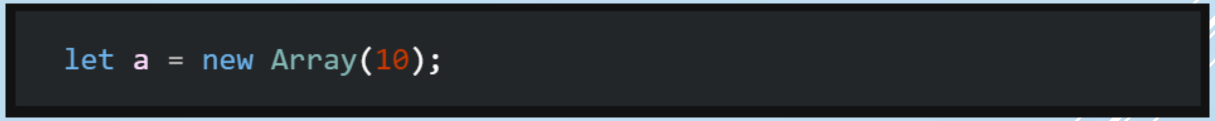


```
let a = [1, 2, 3];
let b = [0, ...a, 4];  // b == [0, 1, 2, 3, 4]
```

CREACIÓN

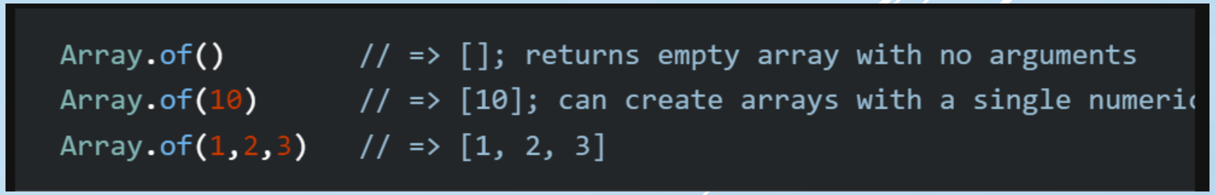
DISTINTAS MANERAS DE CREARSE

- Constructor (1 número, tamaño, más de 1 número, elementos del array)



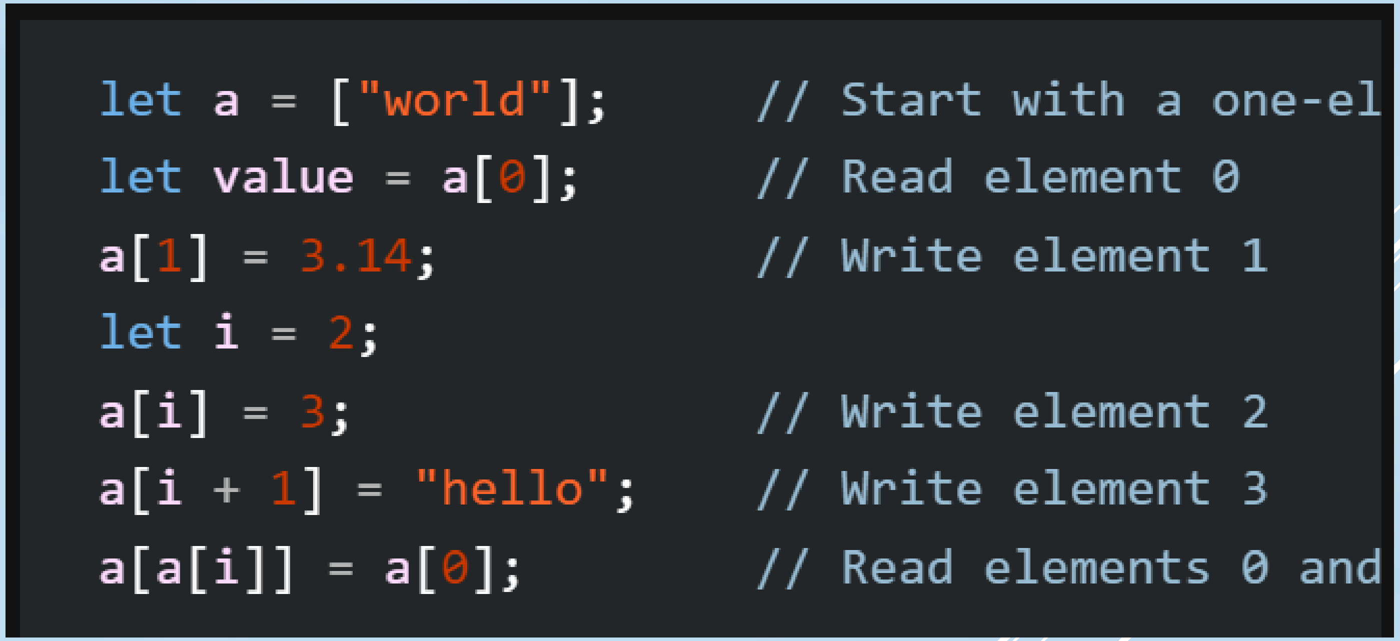
```
let a = new Array(10);
```

- Array.of

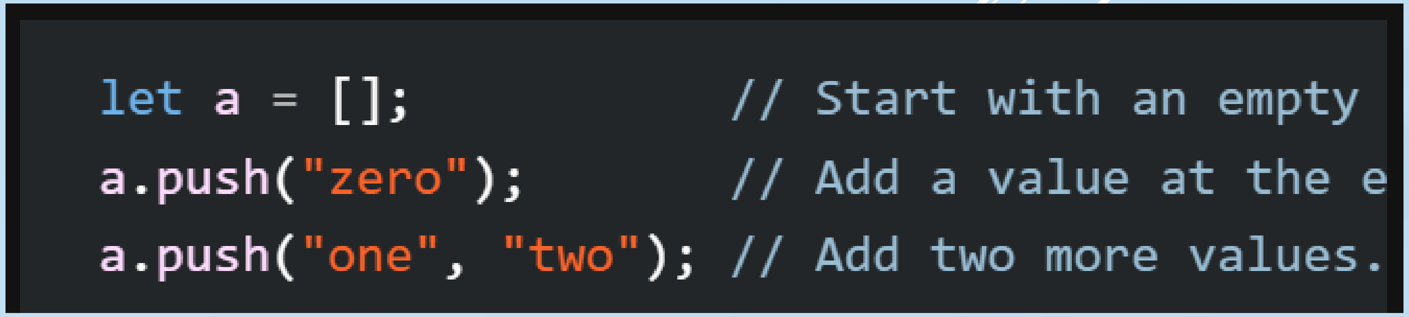


```
Array.of()        // => []; returns empty array with no arguments
Array.of(10)      // => [10]; can create arrays with a single numeri
Array.of(1,2,3)   // => [1, 2, 3]
```

LECTURA Y ESCRITURA



```
let a = ["world"];      // Start with a one-el
let value = a[0];       // Read element 0
a[1] = 3.14;            // Write element 1
let i = 2;
a[i] = 3;               // Write element 2
a[i + 1] = "hello";     // Write element 3
a[a[i]] = a[0];         // Read elements 0 and
```



```
let a = [];                // Start with an empty
a.push("zero");            // Add a value at the e
a.push("one", "two"); // Add two more values.
```

LECTURA Y ESCRITURA

Notas

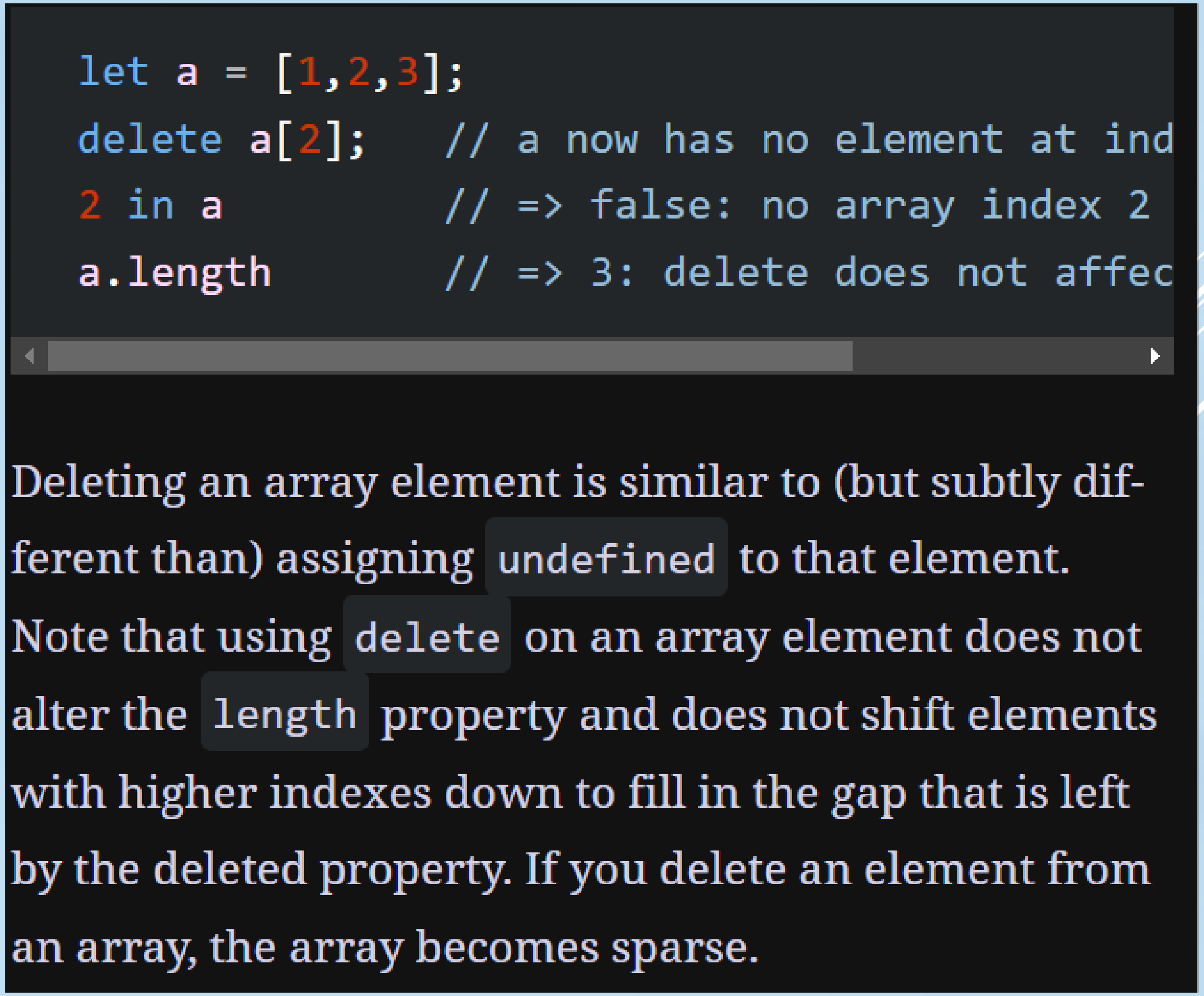
- Si ponemos la propiedad .length a cero, es como borrar el array.

- Length siempre será igual o mayor al número de elementos del array
- Podemos poner un element en un índice alejado del final

```
let a = [];
a[23] = 12
```

LECTURA Y ESCRITURA

DELETE

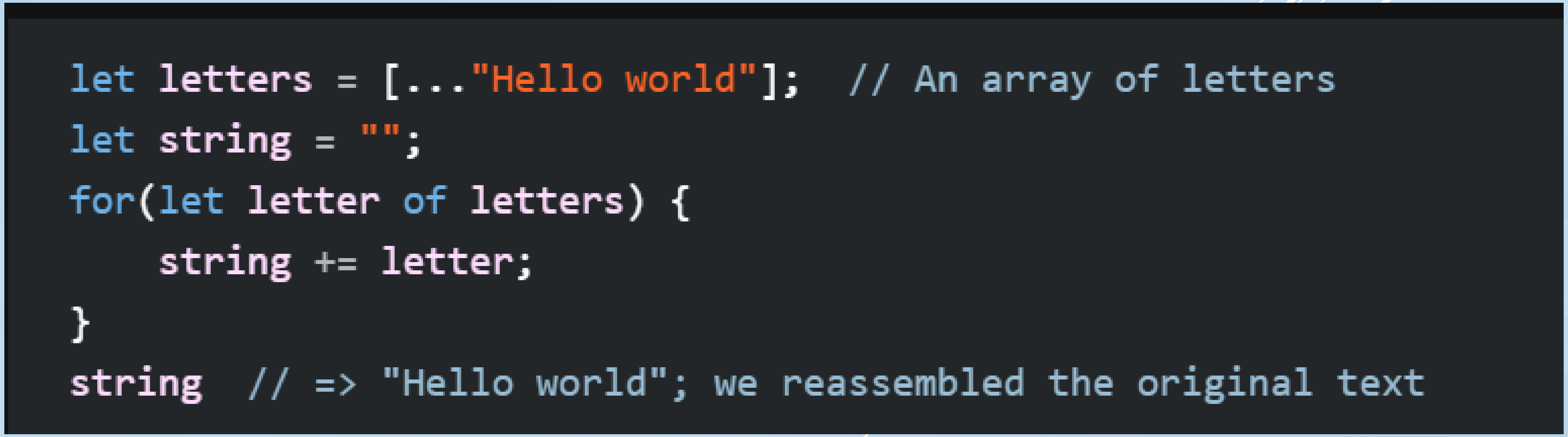


```
let a = [1,2,3];
delete a[2];   // a now has no element at ind
2 in a         // => false: no array index 2
a.length       // => 3: delete does not affec
```

Deleting an array element is similar to (but subtly different than) assigning `undefined` to that element. Note that using `delete` on an array element does not alter the `length` property and does not shift elements with higher indexes down to fill in the gap that is left by the deleted property. If you delete an element from an array, the array becomes sparse.

RECORRIENDO ARRAYS

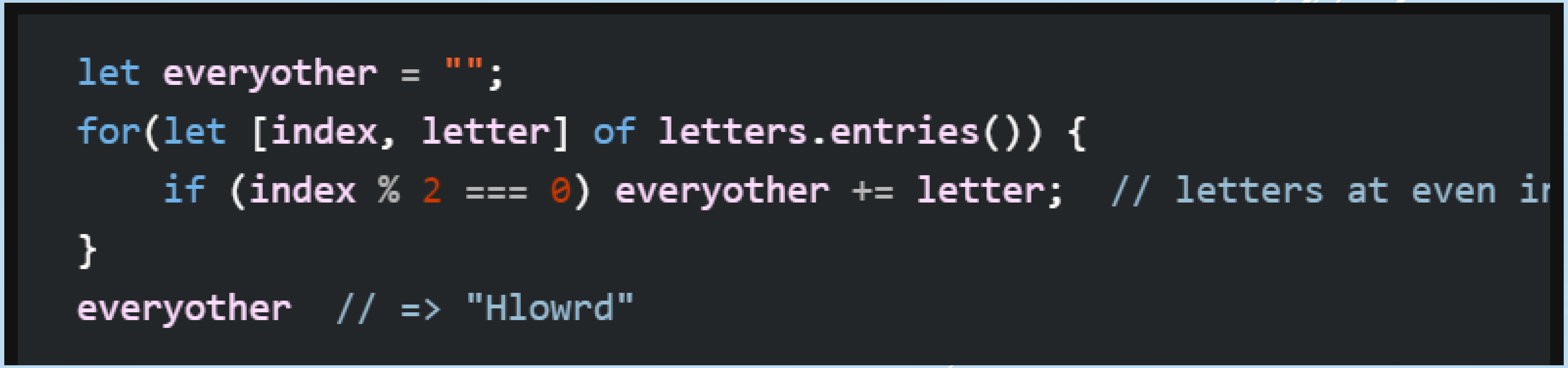
LET..OF



```
let letters = [..."Hello world"];  // An array of letters
let string = "";
for(let letter of letters) {
    string += letter;
}
string  // => "Hello world"; we reassembled the original text
```

RECORRIENDO ARRAYS

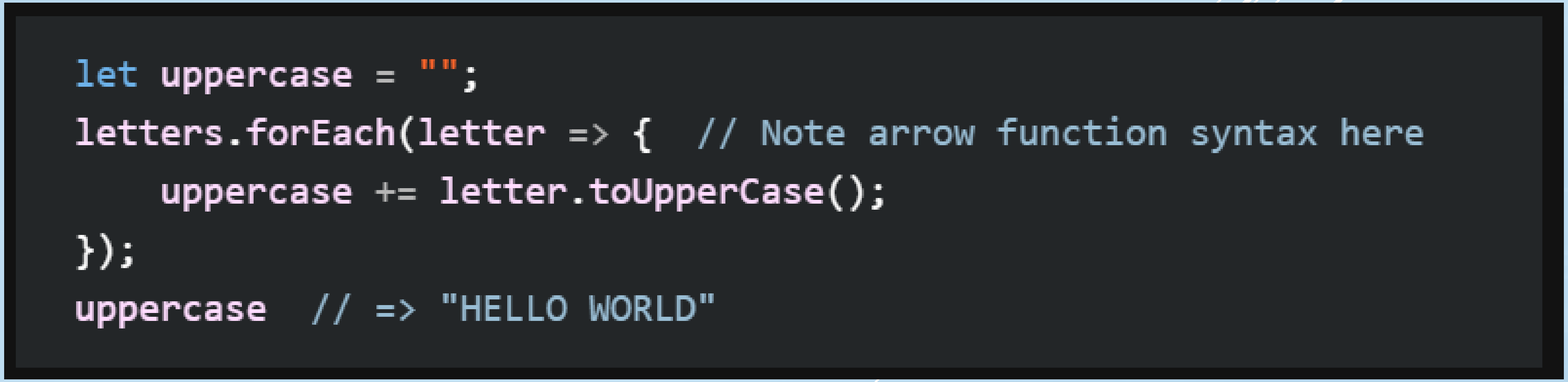
LET..OF WITH INDEX



```
let everyother = "";
for(let [index, letter] of letters.entries()) {
    if (index % 2 === 0) everyother += letter;  // letters at even in
}
everyother  // => "Hlowrd"
```

RECORRIENDO ARRAYS

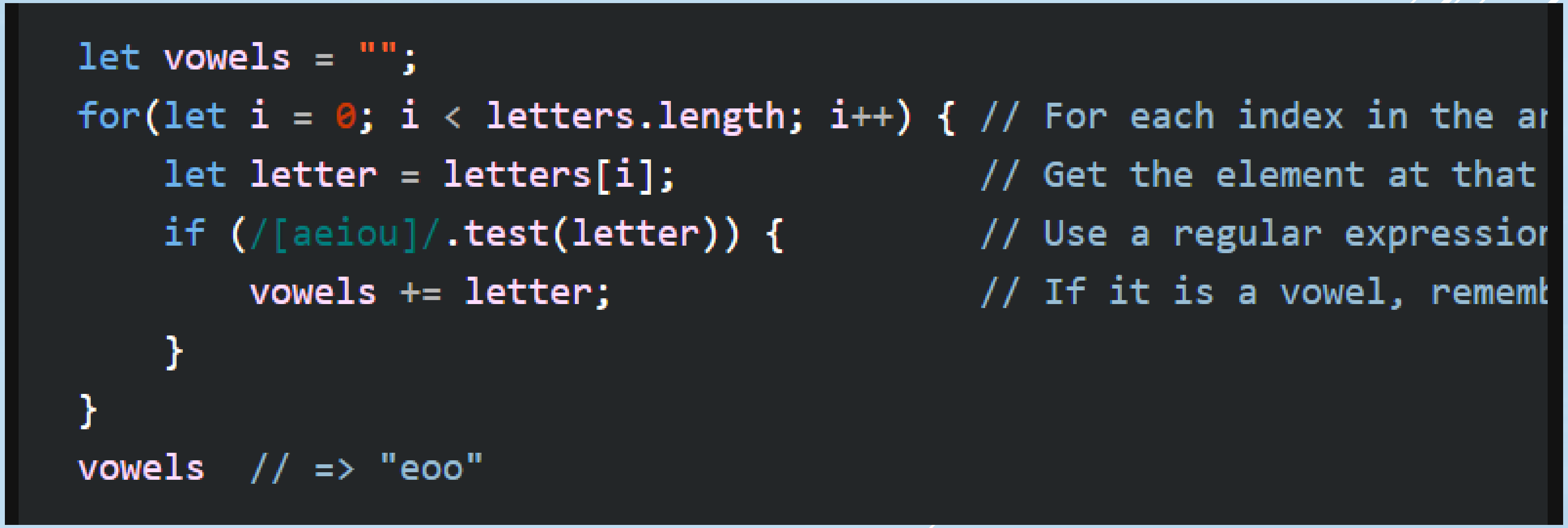
FOREACH



```
let uppercase = "";
letters.forEach(letter => {  // Note arrow function syntax here
    uppercase += letter.toUpperCase();
});
uppercase  // => "HELLO WORLD"
```

RECORRIENDO ARRAYS

INDEXING

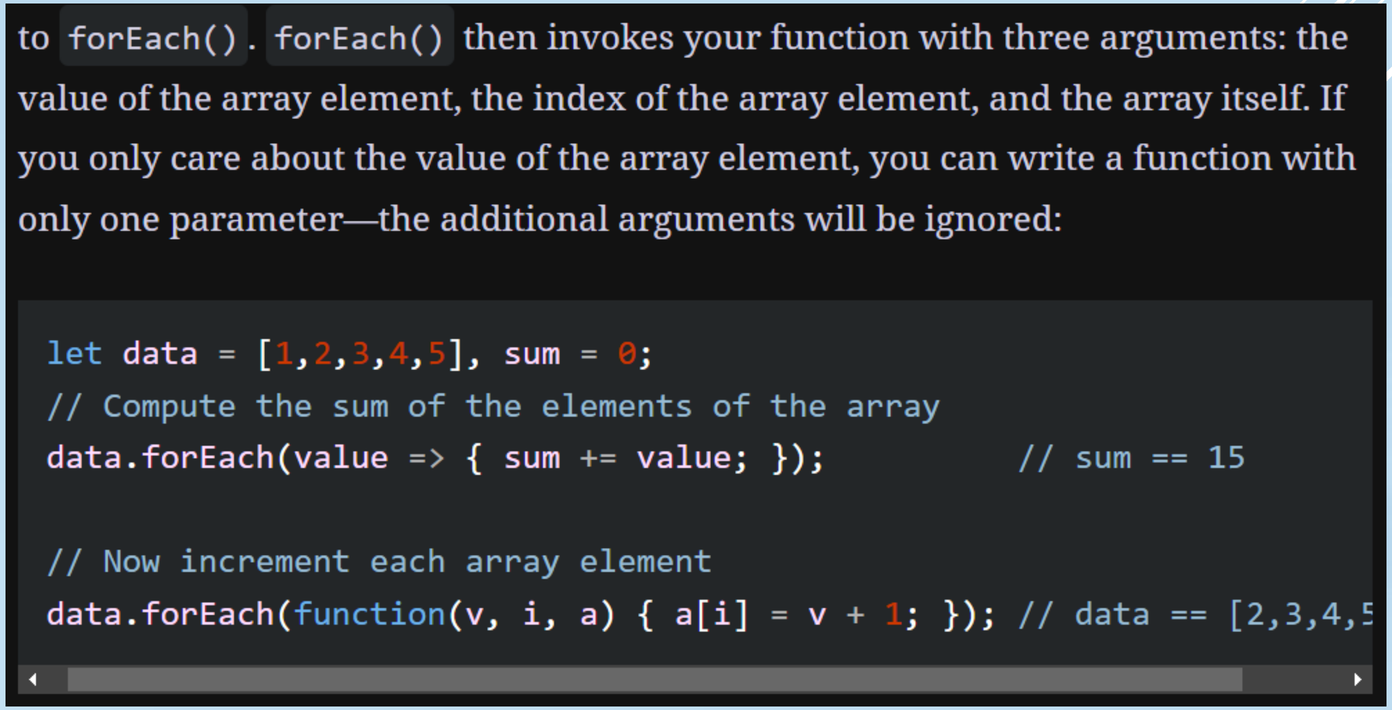


```
let vowels = "";
for(let i = 0; i < letters.length; i++) { // For each index in the ar
    let letter = letters[i];              // Get the element at that
    if (/[aeiou]/.test(letter)) {         // Use a regular expressior
        vowels += letter;                 // If it is a vowel, rememt
    }
}
vowels  // => "eoo"
```

MÉTODOS DE ARRAY

FOREACH (ya visto)

- No devuelve array, modifica el existente



to `forEach()`. `forEach()` then invokes your function with three arguments: the value of the array element, the index of the array element, and the array itself. If you only care about the value of the array element, you can write a function with only one parameter—the additional arguments will be ignored:

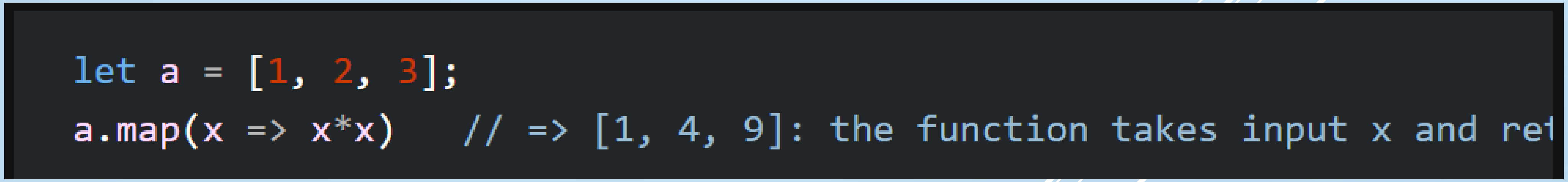
```
let data = [1,2,3,4,5], sum = 0;
// Compute the sum of the elements of the array
data.forEach(value => { sum += value; });          // sum == 15

// Now increment each array element
data.forEach(function(v, i, a) { a[i] = v + 1; }); // data == [2,3,4,5
```

MÉTODOS DE ARRAY

MAP

- Sí devuelve array

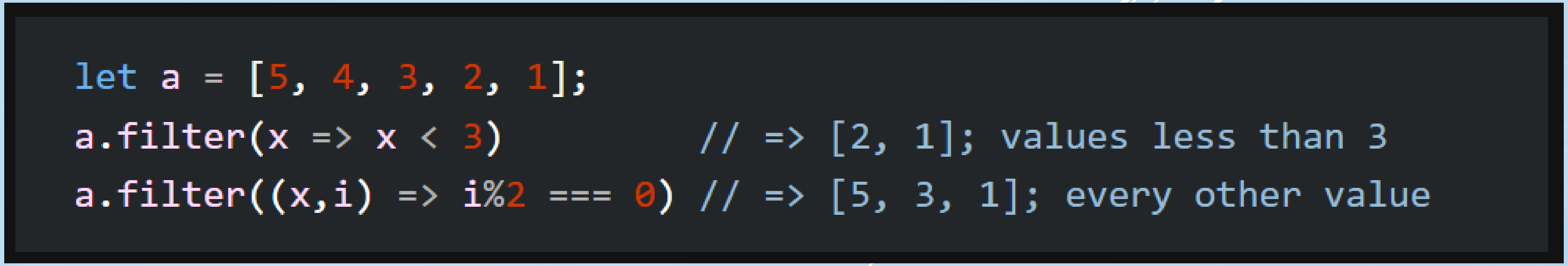


```
let a = [1, 2, 3];
a.map(x => x*x)   // => [1, 4, 9]: the function takes input x and ret
```

MÉTODOS DE ARRAY

FILTER

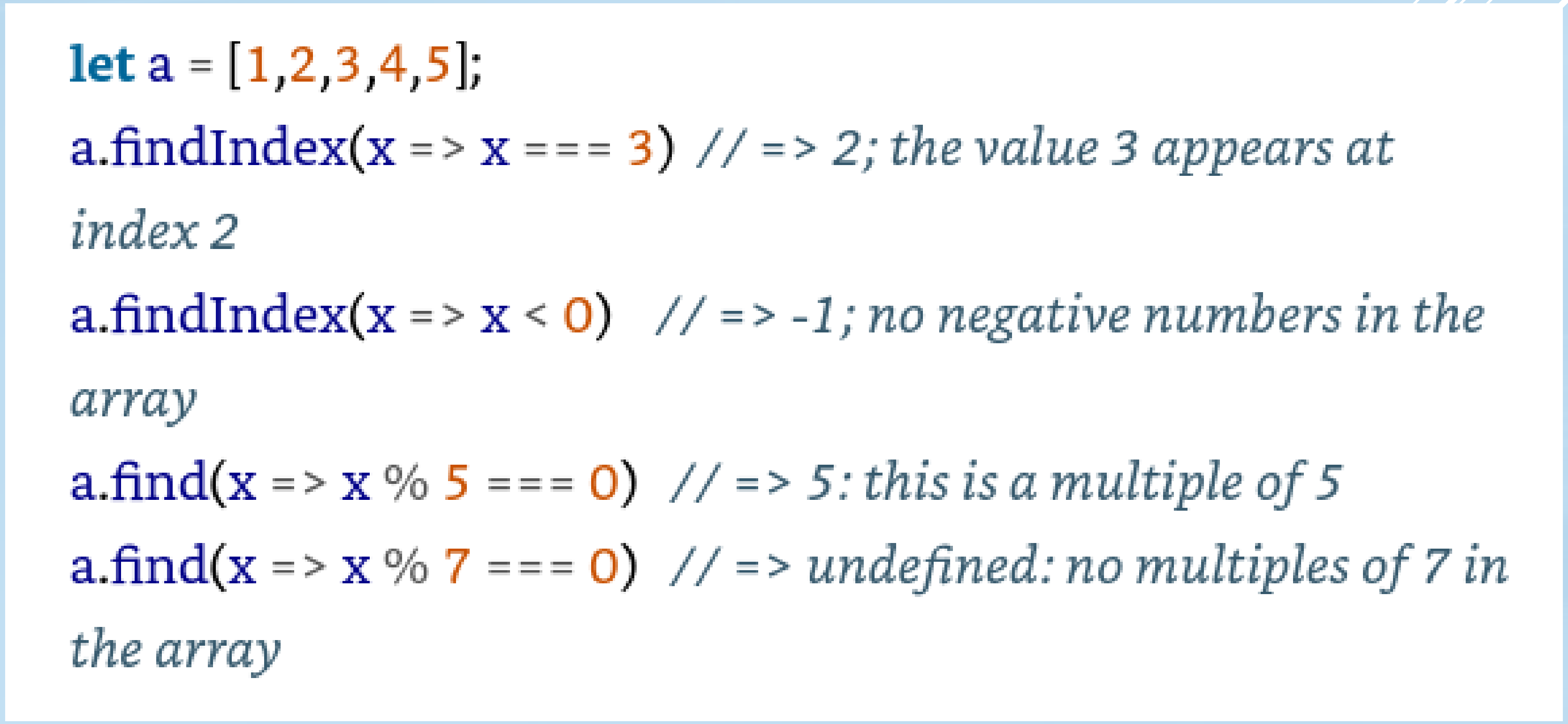
- Devuelve un array conteniendo un subset de elementos del array principal



```
let a = [5, 4, 3, 2, 1];
a.filter(x => x < 3)         // => [2, 1]; values less than 3
a.filter((x,i) => i%2 === 0) // => [5, 3, 1]; every other value
```

MÉTODOS DE ARRAY

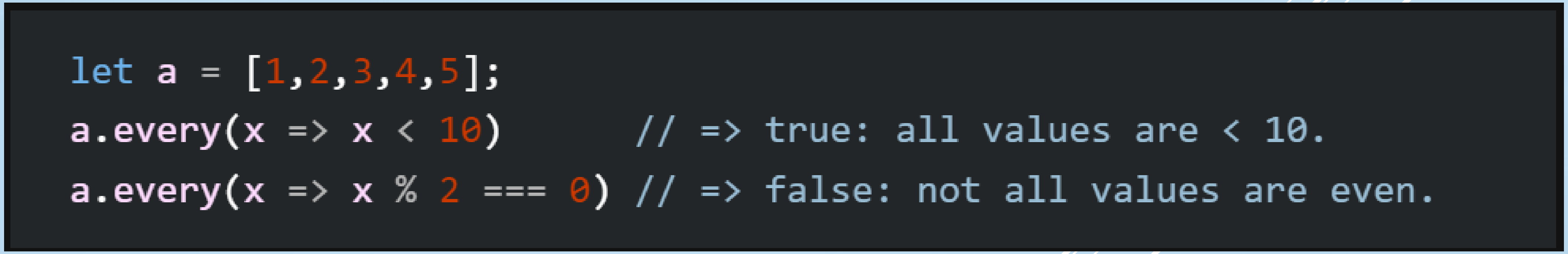
FIND AND FINDINDEX



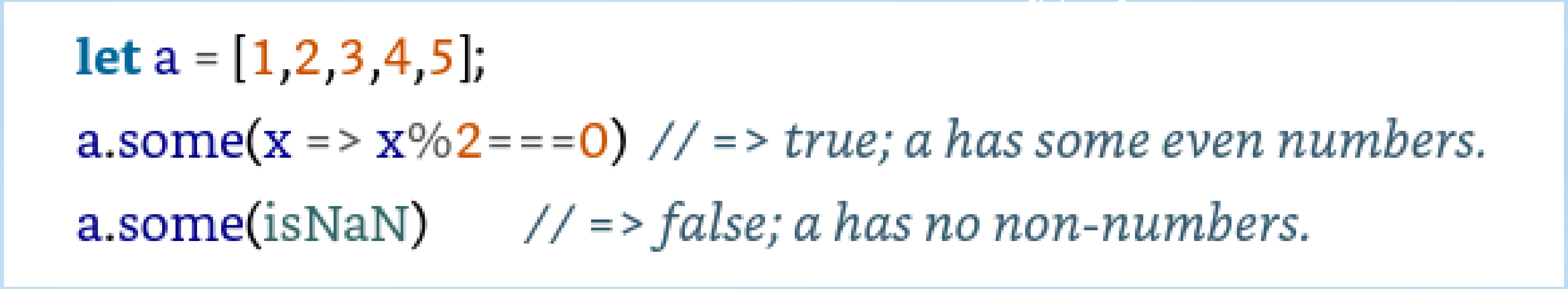
```
let a = [1,2,3,4,5];
a.findIndex(x => x === 3) // => 2; the value 3 appears at
index 2
a.findIndex(x => x < 0)   // => -1; no negative numbers in the
array
a.find(x => x % 5 === 0)  // => 5: this is a multiple of 5
a.find(x => x % 7 === 0)  // => undefined: no multiples of 7 in
the array
```

MÉTODOS DE ARRAY

EVERY AND SOME



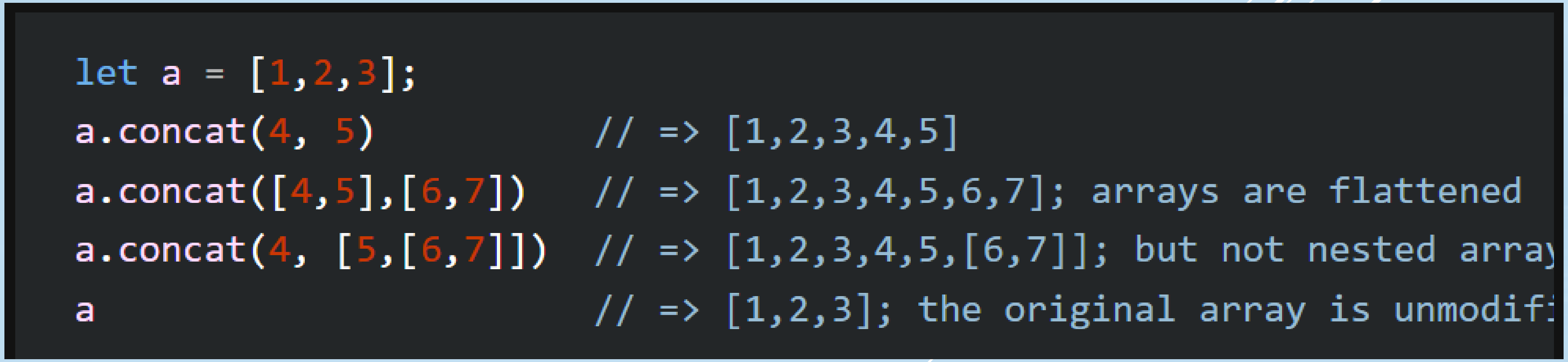
```
let a = [1,2,3,4,5];
a.every(x => x < 10)      // => true: all values are < 10.
a.every(x => x % 2 === 0) // => false: not all values are even.
```



```
let a = [1,2,3,4,5];
a.some(x => x%2===0) // => true; a has some even numbers.
a.some(isNaN)        // => false; a has no non-numbers.
```

MÉTODOS DE ARRAY

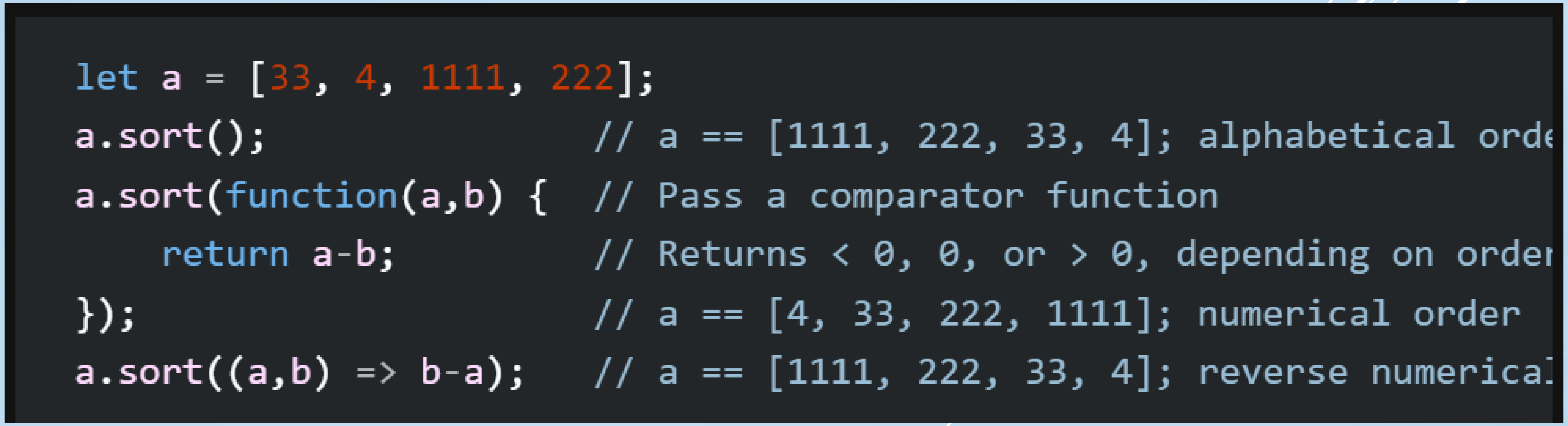
CONCAT



```
let a = [1,2,3];
a.concat(4, 5)          // => [1,2,3,4,5]
a.concat([4,5],[6,7])   // => [1,2,3,4,5,6,7]; arrays are flattened
a.concat(4, [5,[6,7]])  // => [1,2,3,4,5,[6,7]]; but not nested array
a                       // => [1,2,3]; the original array is unmodifi
```

MÉTODOS DE ARRAY

SORT

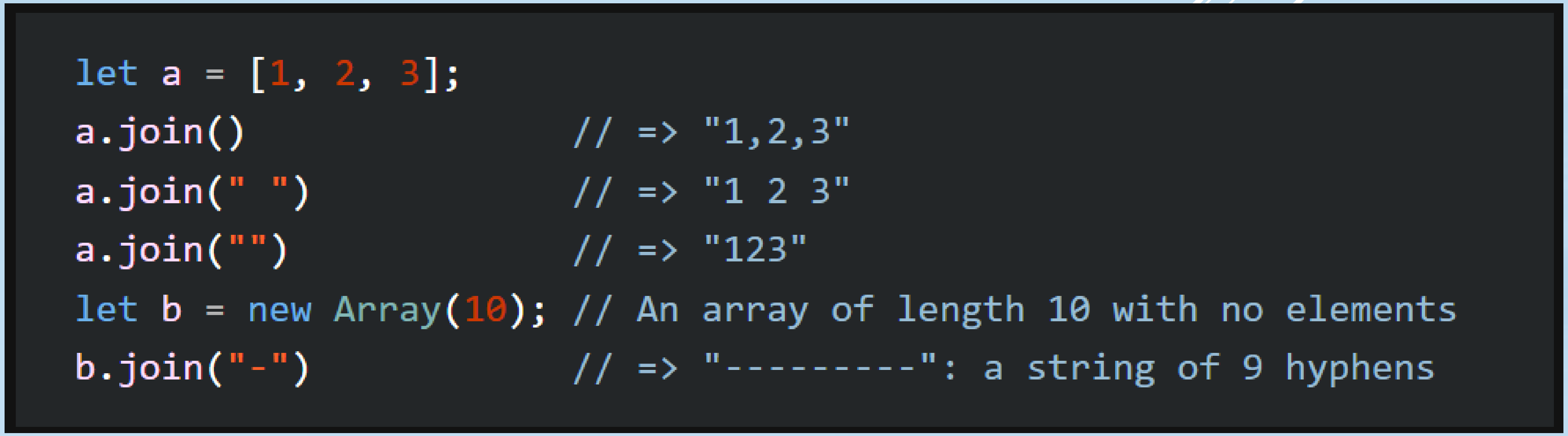


```
let a = [33, 4, 1111, 222];
a.sort();                   // a == [1111, 222, 33, 4]; alphabetical orde
a.sort(function(a,b) {  // Pass a comparator function
    return a-b;             // Returns < 0, 0, or > 0, depending on orde
});                         // a == [4, 33, 222, 1111]; numerical order
a.sort((a,b) => b-a);   // a == [1111, 222, 33, 4]; reverse numerical
```

MÉTODOS DE ARRAY

REVERSE AND JOIN

- Reverse ordena los elementos al reves
- Join convierte el array a string separándolo por el argumento
- Un string es un array de caracteres (UTF16)



```
let a = [1, 2, 3];
a.join()               // => "1,2,3"
a.join(" ")            // => "1 2 3"
a.join("")             // => "123"
let b = new Array(10); // An array of length 10 with no elements
b.join("-")            // => "---------": a string of 9 hyphens
```

BIBLIOGRAFÍA

1. JavaScript: The Definitive Guide, 7th Edition
2. Eloquent JavaScript, 3th edition, Marijn Haverbeke
3. Clean Code: A Handbook of Agile Software Craftsmanship (Robert C. Martin Series)
4. Modern c++ Programming with Test-Driven Development, Jeff Langr
5. Refactoring: Improve the design of existing Code, Martin Fowler
6. Game programming patterns, Robert Nystrom