



Somos un **ecosistema** de desarrolladores de software

Java Script



BEGIN NAVIGATION

```
">Home</a></li>
        .html">Home Events</a></li>
    menu.html">Multiple Column Menus
    <a href="#" class="current">
        button-header.html">Tall Buttons
        logo.html">Image Logo</a></li>
        href="#">Tall Logo</a></li>
        f="#">Carousels</a>
        th-slider.html">Variable Width Sliders
        testimonial.html">Testimonials
```

Array

Propiedades

Un **array** es una colección o agrupación de elementos en una misma variable, cada uno de ellos ubicado por la posición que ocupa en el array. En algunas ocasiones también se les suelen llamar **arreglos** o **vectores**.

Constructor	Descripción
<code>new Array(NUMBER size)</code>	Crea un array vacío de tamaño size. Sus valores no están definidos, pero son UNDEFINED.
<code>new Array(e1, e2...)</code>	Crea un array con los elementos indicados.
<code>[e1, e2...]</code>	Simplemente, los elementos dentro de corchetes: <code>[]</code> . Notación preferida .

Array

Sintaxis y propiedades

- □ ×

```
// Forma tradicional (no se suele usar en Javascript)
const letters = new Array("a", "b", "c");    // Array con 3 elementos
const letters = new Array(3);                  // Array vacío de tamaño 3

// Mediante literales (notación preferida)
const letters = ["a", "b", "c"];   // Array con 3 elementos
const letters = [];              // Array vacío (0 elementos)
const letters = ["a", 5, true];   // Array mixto (String, Number, Boolean)
```

Array

Acceso a elementos del array

Forma	Descripción
NUMBER .length	Propiedad que devuelve el número de elementos del array.
OBJECT[pos]	Operador que devuelve (o modifica) el elemento número pos del array.
OBJECT.at(pos)	Método que devuelve el elemento en la posición pos. Números negativos en orden inverso.

Array

El operador []

- □ ×

```
const letters = ["a", "b", "c"];  
  
letters.length;    // 3  
letters[0];        // 'a'  
letters[2];        // 'c'  
letters[5];        // undefined
```

Array

El operador []

- □ ×

```
const letters = ["a", "b", "c"];  
  
letters[1] = "Z"; // Devuelve "Z" y modifica letters a ["a", "Z", "c"]  
letters[3] = "D"; // Devuelve "D" y modifica letters a ["a", "Z", "c", "D"]  
letters[5] = "A"; // Devuelve "A" y modifica letters a ["a", "Z", "c", "D", undefined, "A"]
```

Array

Metodo .at()

- □ ×

```
const letters = ["a", "b", "c"];  
  
letters.at(0);    // "a"  
letters.at(1);    // "b"  
letters.at(3);    // undefined  
letters.at(-1);   // "c"  
letters.at(-2);   // "b"
```

Array

Añadir o eliminar elementos

Método	Descripción
NUMBER.push(e1, e2, e3...) ⚡	Añade uno o varios elementos al final del array. Devuelve el tamaño del array.
OBJECT.pop() ⚡	Elimina el último elemento del array. Devuelve dicho elemento.
NUMBER.unshift(e1, e2, e3...) ⚡	Añade uno o varios elementos al inicio del array. Devuelve el tamaño del array.
OBJECT.shift() ⚡	Elimina el primer elemento del array. Devuelve dicho elemento.

- □ ×

Recuerda que estos métodos sirven para modificar (**mutar**) el array original.

Array

Añadir o eliminar elementos

- □ ×

- *Los métodos `.push()` y `.pop()` insertan al final del array.
- *Los métodos `.unshift()` y `.shift()` insertan al inicio del array.

- □ ×

```
const elements = ["a", "b", "c"]; // Array inicial

elements.push("d");           // Devuelve 4. Ahora elements = ['a', 'b', 'c', 'd']
elements.pop();               // Devuelve 'd'. Ahora elements = ['a', 'b', 'c']

elements.unshift("z");        // Devuelve 4. Ahora elements = ['z', 'a', 'b', 'c']
elements.shift();             // Devuelve 'z'. Ahora elements = ['a', 'b', 'c']
```

Array

Alternativas para crear arrays

Método	Descripción
Array.from(obj)	Intenta convertir el obj en un array.
Array.from(obj, fmap)	Idem, pero ejecuta la función fmap por cada elemento. Equivalente a .map()
Array.from({ length:size})	Crea un array a partir de un OBJECT de tamaño size, relleno de UNDEFINED
.concat(e1, e2, e3...)	Devuelve los elementos pasados por parámetro concatenados al final del array.
STRING.join(sep)	Une los elementos del array mediante separadores sep en un .

Objeto

Convertir a Array.from

- □ ×

```
const text = "12345";
text.constructor.name;                                // "String"

const letters = Array.from(text);                     // ["1", "2", "3", "4", "5"]
const letters = [...text];                           // ["1", "2", "3", "4", "5"]

const divs = document.querySelectorAll("div");
divs.constructor.name;                               // "NodeList"

const elements = Array.from(divs);                  // [div, div, div]
const elements = [...divs];                         // [div, div, div]
const text = "12345";

const numbers = Array.from(text, (number) => Number(number)); // [1, 2, 3, 4, 5]
const numbers = Array.from(text, Number);            // Equivalente al
anterior

// Equivalente a los dos anteriores
const numbers = [...text].map(Number);
```

Array

Concatenar arrays

- □ ×

```
const elements = [1, 2, 3];  
  
elements.push(4, 5, 6);      // Devuelve 6. Ahora elements = [1, 2, 3, 4, 5, 6]  
elements.push([7, 8, 9]);    // Devuelve 7. Ahora elements = [1, 2, 3, 4, 5, 6, [7,  
8, 9]]
```

Array

Concatenar arrays

- □ ×

```
const firstPart = [1, 2, 3];
const secondPart = [4, 5, 6];

firstPart.concat(firstPart);           // Devuelve [1, 2, 3, 1, 2, 3]
firstPart.concat(secondPart);          // Devuelve [1, 2, 3, 4, 5, 6]

// Se pueden pasar elementos sueltos
firstPart.concat(4, 5, 6);            // Devuelve [1, 2, 3, 4, 5, 6]

// Se pueden concatenar múltiples arrays e incluso mezclarlos con elementos
// sueltos
firstPart.concat(firstPart, secondPart, 7); // Devuelve [1, 2, 3, 1, 2, 3, 4, 5,
                                             6, 7]
```

Array

Separar y unir strings

- □ ×

```
const letters = ["a", "b", "c"];
```

// Une elementos del array por el separador indicado

```
letters.join("→");           // Devuelve 'a→b→c'
```

```
letters.join(".");          // Devuelve 'a.b.c'
```

// Separa elementos del string por el separador indicado

```
"a.b.c".split(".");        // Devuelve ['a', 'b', 'c']
```

```
"5-4-3-2-1".split("-");   // Devuelve ['5', '4', '3', '2', '1']
```

Array

Buscar elementos en un array

Método	Descripción
BOOLEAN.includes(element)	Comprueba si element está incluido en el array.
BOOLEAN.includes(element, from)	Idem, pero partiendo desde la posición from del array.
NUMBER .indexOf(element)	Devuelve la posición de la primera aparición de element. Devuelve -1 si no existe.
NUMBER.indexOf(element, from)	Idem, pero partiendo desde la posición from del array.
NUMBER.lastIndexOf(element)	Devuelve la posición de la última aparición de element. Devuelve -1 si no existe.
NUMBER.lastIndexOf(element, from)	Idem, pero partiendo desde la posición from del array.

Array

¿El elemento existe? (includes)

- □ ×

```
const numbers = [5, 10, 15, 20, 25];
```

```
numbers.includes(3);    // false
```

```
numbers.includes(15);   // true
```

```
numbers.includes(15, 1); // true
```

```
numbers.includes(15, 4); // false
```

Array

Buscar la posición (indexOf)

- □ ×

```
const numbers = [5, 10, 15, 20, 25,  
20, 15, 10, 5];  
  
numbers.indexOf(5);    // 0  
numbers.indexOf(15);  // 2  
numbers.indexOf(25);  // 4  
numbers.indexOf(99);  // -1  
  
numbers.indexOf(15, 1); // 2  
numbers.indexOf(15, 4); // 6  
numbers.indexOf(15, 7); // -1
```

Array

Buscar la posición (lastIndexOf)

```
const numbers = [5, 10, 15, 20, 25,  
20, 15, 10, 5];
```

```
numbers.lastIndexOf(5);    // 8  
numbers.lastIndexOf(15);   // 6  
numbers.lastIndexOf(25);   // 4  
numbers.lastIndexOf(99);   // -1
```

```
numbers.lastIndexOf(15, 8); // 6  
numbers.lastIndexOf(15, 5); // 2  
numbers.lastIndexOf(15, 1); // -1
```

Array

Buscar un elemento en array

- □ ×

```
const names = [
    { name: "María", age: 20 },
    { name: "Bernardo", age: 28 },
    { name: "Pancracio", age: 22 },
    { name: "Andrea", age: 19 },
    { name: "Sara", age: 29 },
    { name: "Jorge", age: 32 },
    { name: "Yurena", age: 38 },
    { name: "Ayoze", age: 18 }
];

// Busca el primer elemento con la edad indicada, sino devuelve -1
const findElement = (array, searchedAge) => {
    for (let i = 0; i < array.length; i++) {
        const element = array[i];
        if (element.age === searchedAge) {
            return element;
        }
    }
    return -1;
}
findElement(names, 19);      // { name: "Andrea", age: 19 }
findElement(names, 32);      // { name: "Jorge", age: 32 }
findElement(names, 33);      // -1
```

</Be a
coder>