

## ORDENACIÓN CON ARREGLOS

### Ejemplo:

```
let estudiantes = [  
  { nombre: "Ana", edad: 22, promedio: 8.5 },  
  { nombre: "Luis", edad: 20, promedio: 9.2 },  
  { nombre: "María", edad: 23, promedio: 7.8 },  
  { nombre: "Mario", edad: 21, promedio: 8.9 }  
];  
  
// De menor a mayor  
estudiantes.sort((a, b) => a.edad - b.edad);  
console.log("Ordenados por edad:");  
console.log(estudiantes);  
  
// De mayor a menor  
estudiantes.sort((a, b) => b.promedio - a.promedio);  
console.log("Ordenados por promedio descendente:");  
console.log(estudiantes);  
  
// Orden alfabético  
estudiantes.sort((a, b) => a.nombre.localeCompare(b.nombre));  
console.log("Ordenados por nombre:");  
console.log(estudiantes);  
estudiantes.sort((a, b) => {  
  if (b.promedio !== a.promedio) {  
    return b.promedio - a.promedio; // Descendente  
  }  
  return a.nombre.localeCompare(b.nombre); // Ascendente  
});  
console.log("Ordenados por promedio y luego nombre:");  
console.log(estudiantes);
```



**INSTRUCCIÓN: ESTUDIANTES.SORT((A, B) => A.EDAD - B.EDAD);**

## 1. Contexto

- estudiantes es un **arreglo de objetos** (cada objeto tiene propiedades como nombre, edad, promedio).
- Array.prototype.sort() es el método de JavaScript que ordena **in place** (modifica directamente el arreglo).
- El método .sort() recibe una **función de comparación** que indica cómo comparar dos elementos del arreglo.


## 2. La función (a, b) => a.edad - b.edad

Aquí, a y b representan **dos objetos** del arreglo que el motor de JavaScript compara entre sí.

La expresión a.edad - b.edad devuelve un número que decide el orden:

- **Si el resultado es negativo (< 0):**  
a se coloca antes que b en el arreglo.
- **Si el resultado es positivo (> 0):**  
b se coloca antes que a.
- **Si es 0:**  
se consideran equivalentes y se mantiene su orden relativo (en navegadores modernos, porque el sort ya es **estable**).

estudiantes.sort((a, b) => a.edad - b.edad) ordena el arreglo **directamente en memoria**, colocando primero a los estudiantes con menor edad y al final a los de mayor edad.



## INSTRUCCIÓN: ESTUDIANTES.SORT((A, B) => B.PROMEDIO - A.PROMEDIO);

### 1. Contexto

- estudiantes sigue siendo un arreglo de objetos.
- Cada objeto tiene la propiedad promedio (un número con la nota/promedio del alumno).
- .sort() recibe una función de comparación que decide cómo reordenar los elementos.

### 2. Diferencia clave respecto al caso anterior

En el ejemplo de edad, se usa:

a.edad - b.edad

que ordenaba de menor a mayor (ascendente).

Aquí se invierte:

b.promedio - a.promedio

lo cual hace que el ordenamiento sea de mayor a menor (descendente).

### 3. Explicación de la función (a, b) => b.promedio - a.promedio

- Si b.promedio - a.promedio devuelve un número negativo ( $< 0$ ), significa que b.promedio es menor  $\rightarrow$  a se coloca antes que b.
- Si devuelve un número positivo ( $> 0$ ), significa que b.promedio es mayor  $\rightarrow$  b se coloca antes que a.
- Si es 0, ambos tienen el mismo promedio y se mantienen en el mismo orden relativo (en motores modernos, porque el sort es estable).

*Resumen: Ordena el arreglo en orden descendente, es decir, coloca primero a los estudiantes con mayor promedio y al final a los de menor promedio.*

**INSTRUCCIÓN: ESTUDIANTES.SORT((A, B) =>  
A.NOMBRE.LOCALECOMPARE(B.NOMBRE));**

### 1. ¿Qué hace sort aquí?

- El método .sort() ordena los elementos de un arreglo **en su lugar** (modifica el arreglo original).
- Para decidir el orden, recibe una **función comparadora** que compara de a dos elementos del arreglo (a y b).

### 2. ¿Qué es localeCompare?

localeCompare es un método de **strings** que devuelve un número según el orden alfabético:

"cadena1".localeCompare("cadena2")

- **< 0 (negativo)** → "cadena1" va antes que "cadena2".
- **> 0 (positivo)** → "cadena1" va después que "cadena2".
- **0** → se consideran iguales para efectos de ordenamiento.

Ventaja: localeCompare **respeta acentos, caracteres especiales y reglas de idioma** (no se limita al simple código ASCII).

### 3. Aplicación al arreglo de objetos

La función comparadora:

(a, b) => a.nombre.localeCompare(b.nombre)

- Compara el campo nombre de cada objeto.
- Ordena los objetos **alfabéticamente (A → Z)** según esa propiedad.

### Resumen

estudiantes.sort((a, b) => a.nombre.localeCompare(b.nombre));

- ✓ Ordena el arreglo **alfabéticamente de A a Z** según el campo nombre.
- ✓ Considera **acentos, tildes y caracteres especiales** gracias a localeCompare.
- ✓ Modifica directamente el arreglo original.