Métodos Intermedios y Avanzados de Arrays en JavaScript

🧠 1️⃣ Introducción

Los métodos intermedios de arrays permiten **recorrer, transformar, filtrar y analizar** colecciones de datos.
En lugar de usar bucles for, podemos hacerlo con funciones más expresivas como map(), filter(), find() o reduce().

👉 En esta unidad aprenderás:

- Cómo funcionan los métodos más usados.
- Cómo aplicarlos a arrays de objetos.
- Cómo combinarlos en pequeñas prácticas.

🧩 2️⃣ Array base (de productos)

Usaremos este mismo array durante toda la práctica:

```
const productos = [
  { id: 1, nombre: "Teclado", precio: 25, categoria: "Periféricos" },
  { id: 2, nombre: "Ratón", precio: 15, categoria: "Periféricos" },
  { id: 3, nombre: "Monitor", precio: 180, categoria: "Pantallas" },
  { id: 4, nombre: "Auriculares", precio: 45, categoria: "Audio" },
  { id: 5, nombre: "Laptop", precio: 950, categoria: "Informática" }
];
```

🔷 3️⃣ Método forEach() – recorrer el array

Sintaxis:

array.forEach(elemento => { ... });

👉 Recorre cada elemento del array, pero **no devuelve nada** (solo ejecuta una acción).

Ejemplo práctico:

```
productos.forEach(p => {
```

```
 console.log(`Producto: ${p.nombre} - Precio: ${p.precio}€`);
});
```

💡 Ideal para mostrar información o aplicar acciones a cada elemento.

🔷 4️⃣ Método map() – transformar un array

Sintaxis:

array.map(elemento => nuevaForma);

👉 Devuelve un **nuevo array** del mismo tamaño, con los valores transformados.

Ejemplo práctico:

```
const nombres = productos.map(p => p.nombre);
console.log(nombres);
// ["Teclado", "Ratón", "Monitor", "Auriculares", "Laptop"]
```

📘 También se puede usar para crear una **lista HTML**:

```
const lista = productos.map(p => `<li>${p.nombre} -
${p.precio}€</li>`).join('');
document.body.innerHTML = `<ul>${lista}</ul>`;
```

🔷 5️⃣ Método filter() – filtrar elementos

Sintaxis:

array.filter(elemento => condición);

👉 Devuelve un **nuevo array** con solo los elementos que cumplan la condición.

Ejemplo práctico:

```
const caros = productos.filter(p => p.precio > 100);
console.table(caros);
// 🧱 Puedes combinarlo con otra propiedad:
const perifericos = productos.filter(p => p.categoria === "Periféricos");
```

```
console.log(perifericos);
```

🔷 6 Método find() – buscar un elemento

Sintaxis:

array.find(elemento => condición);

👉 Devuelve el **primer elemento** que cumpla la condición (no un array).

Ejemplo práctico:

```
const encontrado = productos.find(p => p.id === 3);
console.log(encontrado);
// { id: 3, nombre: "Monitor", precio: 180, categoria: "Pantallas" }
```

💡 Si no encuentra nada, devuelve undefined.

🔷 7 Método some() y every() – validaciones

- some() → devuelve true si **al menos un elemento** cumple la condición.
- every() → devuelve true si **todos** la cumplen.

Ejemplo práctico:

```
console.log(productos.some(p => p.precio > 500)); // true
console.log(productos.every(p => p.precio > 10)); // true
```

🔷 8 Método reduce() – reducir a un solo valor

Sintaxis:

array.reduce((acumulador, elemento) => nuevaSuma, valorInicial);

👉 Acumula todos los valores en uno solo (por ejemplo, suma total).

Ejemplo práctico:

const total = productos.reduce((acum, p) => acum + p.precio, 0);

console.log(`Precio total del catálogo: ${total}€`);

📘 También puedes usarlo para encontrar el **producto más caro**:

```
const masCaro = productos.reduce((max, p) => p.precio > max.precio ? p :
max);
console.log(masCaro);
```

🔷 9️⃣ Método sort() – ordenar arrays

Sintaxis:

array.sort((a, b) => comparación);

👉 Ordena el array **en el lugar (modifica el original)**.

Ejemplo práctico:

```
const total = productos.reduce((acum, p) => acum + p.precio, 0);
console.log(`Precio total del catálogo: ${total}€`);
//🔍 También puedes usarlo para encontrar el producto más caro:
const masCaro = productos.reduce((max, p) => p.precio > max.precio ? p :
max);
console.log(masCaro);
```

🔷 🔟 Método slice() y splice()

Método	Qué hace	Modifica original
slice(inicio, fin)	Crea una copia parcial	❌ No
splice(inicio, cantidad, reemplazo?)	Elimina o reemplaza	✅ Sí

Ejemplo práctico:

```
const copia = productos.slice(0, 3);
console.log(copia); // copia los primeros tres

productos.splice(1, 1, { id: 6, nombre: "Tablet", precio: 300 });
console.table(productos);
```

🔁 11 Bucles modernos: for...of y for...in

for...of → recorre arrays

```
for (const p of productos) {
  console.log(p.nombre);
}
```

for...in → recorre propiedades de objetos

```
const producto = { nombre: "Teclado", precio: 25 };
for (const prop in producto) {
  console.log(prop, producto[prop]);
}
```