📝 Ejercicios Prácticos: Métodos Fundamentales de Arrays

A continuación, encontrarás un ejercicio para cada uno de los métodos de Array que hemos estudiado. Tu objetivo es escribir el código JavaScript necesario para resolver cada problema.

Parte 1: Métodos Modificadores (Mutan el Array Original)

Estos métodos cambian el contenido del array directamente.

1. `push()` - Añadir al Final

Objetivo: Agregar un nuevo elemento al final de una lista.

Ejercicio: Tienes una lista de tareas: `let tareas = ["Estudiar JS", "Hacer la compra"]`. Usa el método `push()` para añadir una nueva tarea al final: **"Pasear al perro"**. Luego, imprime la lista completa.

Pista: `push()` solo necesita el nuevo elemento como argumento.

2. `pop()` - Eliminar el Último

Objetivo: Eliminar el último elemento de un array.

Ejercicio: Usando la lista de tareas del ejercicio anterior, usa `pop()` para simular que terminaste la última tarea. Imprime **el elemento que se eliminó** y luego la lista final.

Pista: `pop()` no necesita argumentos.

3. `unshift()` - Añadir al Principio

Objetivo: Agregar un nuevo elemento al principio de un array.

Ejercicio: Tienes una lista de invitados: `let invitados = ["Ana", "Luis", "Carlos"]`. Usa `unshift()` para agregar a **"María"** como la primera invitada.

Pista: `unshift()` funciona igual que `push()`, pero añade al inicio.

4. `shift()` - Eliminar el Primero

Objetivo: Eliminar el primer elemento de un array.

Ejercicio: De la lista de invitados, usa `shift()` para eliminar y mostrar quién fue **el primer invitado** en irse.

Pista: `shift()` no necesita argumentos.

5. `splice()` - Reemplazar en Medio

Objetivo: Reemplazar un elemento en una posición específica.

Ejercicio: Tienes una lista de colores: `let colores = ["Rojo", "Verde", "Azul", "Amarillo"]`. El color "Verde" (índice 1) es incorrecto; debe ser **"Naranja"**. Usa `splice()` para reemplazar **solo** ese elemento.

Pista: Recuerda que para reemplazar es: `(índice de inicio, cuántos eliminar, nuevo elemento)`.

6. `sort()` - Ordenar

Objetivo: Ordenar números de forma ascendente.

Ejercicio: Tienes un array de puntuaciones: `let puntos = [40, 100, 1, 5, 25, 10]`. Usa `sort()` con una función de comparación para ordenarlos de forma **ascendente** (del menor al mayor).

Pista: La función de comparación para números debe ser `(a, b) => a - b`.

Parte 2: Métodos de Iteración y Transformación

Estos métodos son fundamentales para manipular datos y crear arrays nuevos.

7. `forEach()` - Iterar

Objetivo: Recorrer un array y ejecutar una acción por cada elemento.

Ejercicio: Tienes una lista de estudiantes: `let alumnos = ["Elena", "David", "Sofia"]`. Usa `forEach()` para imprimir un saludo personalizado para cada uno en la consola: **"¡Hola, [Nombre]! Bienvenido/a."**

Pista: La función *callback* de `forEach()` recibe el elemento como primer argumento.

8. `map()` - Transformar

Objetivo: Crear un nuevo array transformando cada elemento del original.

Ejercicio: Tienes un array de frutas en minúscula: `let frutas = ["manzana", "banana", "uva"]`. Usa `map()` para crear un **nuevo array** llamado `frutasMayusculas` donde todas las frutas estén en **MAYÚSCULAS**.

Pista: Debes usar el método de *string* `.toUpperCase()` dentro del `map()`.

9. `filter()` - Filtrar

Objetivo: Crear un nuevo array solo con los elementos que cumplan una condición.

Ejercicio: Tienes una lista de notas: `let notas = [4, 8, 3, 9, 6, 5]`. Usa `filter()` para crear un **nuevo array** llamado `aprobados` que contenga solo las notas que son **mayores o iguales a 7**.

Pista: La condición dentro del `filter()` debe devolver `true` o `false` (`nota >= 7`).

10. `reduce()` - Acumular

Objetivo: Sumar todos los valores de un array para obtener un único resultado.

Ejercicio: Tienes un array de precios: `let precios = [25.50, 10.00, 5.25, 50.00]`. Usa `reduce()` para calcular el **monto total** de la factura.

Pista: La función *callback* de `reduce()` siempre debe devolver el **acumulador** actualizado.

Parte 3: Métodos de Búsqueda y Validación

Estos métodos buscan o comprueban si los elementos cumplen ciertas condiciones.

11. `find()` - Encontrar el Primer Elemento

Objetivo: Encontrar el primer objeto que cumple una condición específica.

Ejercicio: Tienes una lista de productos: `let productos = [{id: 101, nombre: "Lápiz"}, {id: 102, nombre: "Cuaderno"}]`. Usa `find()` para obtener **el objeto completo** del producto cuyo `id` es **102**.

Pista: La condición debe verificar si el `producto.id` es exactamente igual a `102`.

12. `findIndex()` - Encontrar el Índice

Objetivo: Encontrar la posición (índice) de un elemento.

Ejercicio: Usando el array `productos` del ejercicio anterior, usa `findIndex()` para saber en qué **índice** se encuentra el producto llamado **"Lápiz"**.

Pista: La condición debe verificar si `producto.nombre` es igual a `"Lápiz"`.

13. `every()` - Verificar Todos

Objetivo: Comprobar si *todos* los elementos cumplen una regla.

Ejercicio: Tienes una lista de edades: `let edades = [22, 18, 30, 25]`. Usa `every()` para comprobar si **TODOS** los elementos son mayores o iguales a 18. Imprime el resultado (`true` o `false`).

Pista: `every()` solo retorna `true` si la condición es cierta para *todos* los elementos sin excepción.