En JavaScript, los arrays son muy flexibles y pueden contener diferentes tipos de datos, incluso mezclados.

ARRAYS: Ejemplos

1. Array con números

```
let numeros = [10, 20, 30, 40, 50];
console.log(numeros);
```

Aquí almacenamos varios valores numéricos dentro de un solo array.

2. Array con cadenas de texto (strings)

```
let nombres = ["Ana", "Luis", "Carlos", "María"];
console.log(nombres);
```

Cada elemento del array es una cadena de texto.

3. Array con objetos

```
let person = [{nombre: "Ana", edad: 25}, {nombre: "Luis", edad: 30}, {nombre: "Carlos", edad: 28}];
console.log(person);
```

Cada elemento es un objeto con propiedades.

4. Array con funciones

```
let operaciones = [
  function(a, b) { return a + b; },
  function(a, b) { return a - b; }
];
console.log(operaciones[0](5, 3)); // 8
console.log(operaciones[1](5, 3)); // 2
```

Aquí almacenamos funciones dentro del array y las ejecutamos usando su índice.

📚 5. Array con otros arrays (arrays multidimensionales)

```
let matriz = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ];
console.log(matriz[1][2]); // 6
```

Este tipo de estructura se usa mucho para representar tablas o matrices.

6. Array con diferentes tipos de datos

JavaScript permite mezclar tipos en un mismo array:

```
let mixto = [
  10,
  "Hola",
  { nombre: "Ana" },
  function() { return "Función ejecutada"; },
  [1, 2, 3]
];
console.log(mixto);
```

Aunque es posible mezclar tipos, se recomienda mantener consistencia para un código más claro y organizado.

Cómo recorrer un Array en JavaScript
Recorrer un array significa **iterar sobre cada uno de sus elementos** para leerlos o realizar alguna acción con ellos.

Usando for (bucle tradicional)
Es la forma más clásica y te da control total sobre el índice.

```
let frutas = ["Manzana", "Banana", "Naranja", "Uva"];

for (let i = 0; i < frutas.length; i++) {
  console.log("Fruta:", frutas[i]);
}
```

¿Cómo funciona?

- let i = 0 → comienza en la posición 0
- i < frutas.length → se repite mientras el índice sea menor al tamaño del array
- i++ → aumenta el índice en 1 en cada vuelta

frutas[i] permite acceder al elemento según su posición.

Usando forEach()
Es un método propio de los arrays y es más moderno y limpio.

```
let frutas = ["Manzana", "Banana", "Naranja", "Uva"];

frutas.forEach(function(fruta) {
  console.log("Fruta:", fruta);
});
```

¿Cómo funciona?

- forEach() ejecuta una función por cada elemento del array.
- fruta representa cada elemento del array en cada iteración.

Versión más moderna (Arrow Function)

```
frutas.forEach((fruta, indice) => {
  console.log(indice + ":", fruta);
});
```

Aquí usamos:

- fruta → valor actual
- indice → posición del elemento

Comparación rápida

Característica	for	forEach()
Usa índice manual	✅	Opcional
Más flexible	✅	❌
Más fácil de leer	❌	✅
Permite usar break o continue	✅	❌

Ejemplo práctico con números

```
let numeros = [10, 20, 30, 40];

numeros.forEach(numero => {
  console.log(numero * 2);
});
```

Salida: 20 40 60 80

Conclusión

- Usa for cuando necesites control total del índice o quieras detener el ciclo.
- Usa forEach() cuando solo necesites recorrer y ejecutar algo en cada elemento.

Ejercicios Prácticos – Recorrer Arrays en JavaScript

Aquí tienes ejercicios para practicar for y forEach().
Incluyen enunciado y ejemplo de cómo debería verse el resultado.

Ejercicio 1: Mostrar todos los elementos

Enunciado: Crea un array con 5 colores y recórrelo mostrando cada color en consola.

let colores = ["Rojo", "Azul", "Verde", "Amarillo", "Negro"];
Usa:

- for
- forEach()

Ejercicio 2: Sumar todos los números

Enunciado: Dado el siguiente array, muestra la suma total de sus elementos.

let numeros = [5, 10, 15, 20];
Resultado esperado:
La suma es: 50

Ejercicio 3: Mostrar solo números mayores a 10

Enunciado: Recorre el array y muestra únicamente los números mayores a 10.

let numeros = [4, 12, 7, 25, 3, 18];
Resultado esperado:
12
25
18

Ejercicio 4: Mostrar índice y valor

Enunciado: Recorre el array mostrando el índice y el valor.

let frutas = ["Manzana", "Banana", "Naranja"];
Resultado esperado:
0 - Manzana
1 - Banana
2 - Naranja

Ejercicio 5: Multiplicar cada número por 3

Enunciado: Recorre el array y muestra cada número multiplicado por 3.

let numeros = [2, 4, 6, 8];
Resultado esperado:
6
12
18
24

Ejercicios Nivel Intermedio

Ejercicio 6: Contar números pares
Enunciado: Cuenta cuántos números pares hay en el array.

let numeros = [1, 2, 3, 4, 5, 6, 8, 9];
Resultado esperado:
Cantidad de números pares: 4

Ejercicio 7: Buscar un elemento
Enunciado: Recorre el array y verifica si existe el nombre "Carlos".

let nombres = ["Ana", "Luis", "Pedro", "Carlos"];
Resultado esperado:
Carlos fue encontrado

Desafío Final
Crea un programa que:

1. Recorra un array de edades.
2. Muestre cuántas personas son mayores de edad (18 o más).

let edades = [12, 18, 25, 16, 30, 14, 21];

Ejercicios Nivel Alto con Arrays en JavaScript

Estos ejercicios están pensados para que practiques lógica avanzada, combinación de métodos (for, forEach, map, filter, reduce) y manejo de objetos.

Eliminar elementos duplicados (sin usar Set)
Enunciado: Dado el siguiente array, crea uno nuevo sin elementos repetidos.

let numeros = [1, 2, 3, 2, 4, 1, 5, 3, 6];

Resultado esperado:
[1, 2, 3, 4, 5, 6]
Restricción: No usar Set.

Contar cuántas veces se repite cada elemento
Enunciado: Cuenta cuántas veces aparece cada palabra.
let palabras = ["hola", "adios", "hola", "hola", "adios", "bien"];

Resultado esperado:
```
{
  hola: 3,
  adios: 2,
  bien: 1
}
```
Pista: Usa un objeto como contador.

Agrupar objetos por propiedad
Enunciado: Agrupa los productos por categoría.

```
let productos = [
  { nombre: "Laptop", categoria: "Tecnologia" },
  { nombre: "Camisa", categoria: "Ropa" },
  { nombre: "Celular", categoria: "Tecnologia" },
  { nombre: "Pantalon", categoria: "Ropa" }
];
```

Resultado esperado:

```
{
  Tecnologia: [
    { nombre: "Laptop", categoria: "Tecnologia" },
    { nombre: "Celular", categoria: "Tecnologia" }
  ],
  Ropa: [
    { nombre: "Camisa", categoria: "Ropa" },
    { nombre: "Pantalon", categoria: "Ropa" }
  ]
}
```

Encontrar el segundo número mayor
Enunciado: Encuentra el segundo número más grande del array.

```
let numeros = [10, 50, 30, 20, 90, 80];
```

Resultado esperado:
80
No usar sort().

Crear una matriz transpuesta
Enunciado:
Dada una matriz 2D, genera su matriz transpuesta.

```
let matriz = [
  [1, 2, 3],
  [4, 5, 6]
];
```

Resultado esperado:

```
[
  [1, 4],
  [2, 5],
  [3, 6]
]
```

Simular carrito de compras (Total con impuestos)
Enunciado: Calcula el total a pagar incluyendo 19% de impuesto.

```
let carrito = [
  { producto: "Laptop", precio: 1000 },
  { producto: "Mouse", precio: 50 },
  { producto: "Teclado", precio: 80 }
];
```

Resultado esperado:
Total con impuesto: 1344.9

Aplanar un array multidimensional (sin flat())
Enunciado: Convierte el siguiente array en uno plano.

let datos = [1, [2, 3], [4, [5, 6]], 7];
Resultado esperado:
[1, 2, 3, 4, 5, 6, 7]
No usar .flat().

Ordenar objetos por propiedad
Enunciado: Ordena los estudiantes por nota de mayor a menor.

```
let estudiantes = [
  { nombre: "Ana", nota: 85 },
  { nombre: "Luis", nota: 95 },
  { nombre: "Carlos", nota: 78 }
];
```
Resultado esperado:
```
[
  { nombre: "Luis", nota: 95 },
  { nombre: "Ana", nota: 85 },
  { nombre: "Carlos", nota: 78 }
]
```
Desafío PRO
Crea una función que:
1. Reciba un array de números.
2. Elimine duplicados.
3. Ordene de menor a mayor.
4. Devuelva solo los números pares.
5. Multiplique cada número por 2.

Todo en una sola línea usando métodos encadenados.

Ejercicios de Objetos y Funciones Predeterminadas en JavaScript
(Nivel Inicial, Intermedio y Avanzado — 25 ejercicios)

NIVEL INICIAL (1–8)

1) Crear un objeto simple: Crea un objeto persona con propiedades: nombre, edad y ciudad. Muestra cada propiedad en consola.

2) Modificar propiedades: Agrega una nueva propiedad profesion al objeto anterior y cambia la edad.

3) Eliminar propiedades: Elimina la propiedad ciudad del objeto.

4) Recorrer un objeto: Usa for...in para mostrar todas las propiedades y valores.

5) Usar Object.keys()
Dado un objeto: Muestra todas las claves usando Object.keys().
let auto = { marca: "Toyota", modelo: "Corolla", año: 2022 };

6) Usar Object.values(): Muestra todos los valores del objeto auto.

7) Usar Object.entries(): Recorre el objeto mostrando clave y valor.

8) Verificar propiedad: Usa hasOwnProperty() para verificar si existe la propiedad "modelo".

NIVEL INTERMEDIO (9–17)

9) Contar propiedades: Crea una función que reciba un objeto y devuelva cuántas propiedades tiene.
10) Convertir objeto a array: Convierte un objeto en un array de pares [clave, valor].

11) Fusionar objetos: Une dos objetos usando Object.assign().

12) Clonar un objeto: Crea una copia exacta de un objeto sin modificar el original.

13) Congelar un objeto: Usa Object.freeze() y verifica que no se puedan modificar sus propiedades.

14) Sellar un objeto: Usa Object.seal() y prueba qué modificaciones permite.

15) Obtener claves y valores dinámicamente: Crea una función que reciba un objeto y muestre: La clave es: X y el valor es: Y

16) Comparar objetos: Crea una función que compare si dos objetos tienen las mismas propiedades y valores.

17) Ordenar propiedades alfabéticamente: Dado un objeto, devuelve uno nuevo con las claves ordenadas.

NIVEL AVANZADO (18–25)

18) Agrupar elementos por propiedad: Dado un array de objetos, agrúpalos por una propiedad específica.

19) Crear método dentro de objeto: Crea un objeto calculadora que tenga métodos:

- sumar
- restar
- multiplicar
- dividir

20) Encadenamiento de métodos: Crea un objeto que permita:
obj.sumar(5).restar(2).multiplicar(3)

21) Transformar objeto: Dado un objeto, crea otro donde los valores sean transformados (por ejemplo multiplicados por 2).

22) Invertir claves y valores
Convierte: { a: 1, b: 2 }
En: { 1: "a", 2: "b" }

23) Objeto dinámico: Crea una función que reciba un array y lo convierta en objeto:
["nombre", "edad", "ciudad"]
Resultado:
```
{
  nombre: null,
  edad: null,
  ciudad: null
}
```

24) Validar estructura de objeto: Crea una función que valide si un objeto tiene ciertas propiedades obligatorias.

25) Crear un sistema de inventario:
Crea un objeto que:

- Agregue productos
- Elimine productos
- Actualice stock
- Calcule el valor total del inventario

Ejercicios Nivel PRO con Arrays y Objetos en JavaScript

1) Normalizar datos anidados
Dado:
```
let usuarios = [
  {
    id: 1,
    nombre: "Ana",
    pedidos: [
      { id: 101, total: 50 },
      { id: 102, total: 75 }
    ]
  },
  {
    id: 2,
    nombre: "Luis",
    pedidos: [
      { id: 103, total: 100 }
    ]
  }
];
```
📌 Genera un array plano con esta estructura:
```
[
  { usuarioId: 1, pedidoId: 101, total: 50 },
  { usuarioId: 1, pedidoId: 102, total: 75 },
  { usuarioId: 2, pedidoId: 103, total: 100 }
]
```

2) **Encontrar duplicados con conteo:** Devuelve un objeto con los elementos repetidos y cuántas veces aparecen.

3) **Implementar tu propio map():** Crea una función miMap(array, callback) que funcione como .map() sin usarlo internamente.

4) **Implementar tu propio reduce():** Crea una versión personalizada de reduce().

5) Deep Flatten infinito: Aplana un array con profundidad infinita sin usar .flat().

6) **Detectar ciclos en objeto:** Crea una función que detecte si un objeto tiene referencias circulares.

7) **Comparación profunda (Deep Equal):** Crea una función que compare dos objetos profundamente.

8) **Agrupación múltiple:**
Agrupa por más de una propiedad:

```
[
  { pais: "MX", ciudad: "CDMX" },
  { pais: "MX", ciudad: "Guadalajara" },
  { pais: "US", ciudad: "NY" }
]
```

Agrupar primero por país y luego por ciudad.

9) **Histograma dinámico:** Convierte un array numérico en rangos dinámicos
[5, 12, 18, 25, 33, 47]
Resultado:
0-10: 1
11-20: 2
21-30: 1
31-40: 1
41-50: 1

10) **Ordenamiento complejo**
Ordena por:

1. Edad ascendente
2. Si edad es igual → nombre alfabético

11) **Memorización:** Crea una función que memorice resultados para evitar recalcular.

12) **Índice inverso**
Convierte:

```
[
  { id: 1, nombre: "Ana" },
  { id: 2, nombre: "Luis" }
]
```

En:

```
{
  1: { id: 1, nombre: "Ana" },
  2: { id: 2, nombre: "Luis" }
}
```

13) Resolver dependencia jerárquica
Dado:
```
[
  { id: 1, parent: null },
  { id: 2, parent: 1 },
  { id: 3, parent: 2 }
]
```
Construye un árbol jerárquico.

14) Simular base de datos en memoria
Crea funciones:
- insert
- update
- delete
- find
- findById

Usando arrays de objetos.

15) Pipeline funcional:
Crea una función pipe() que permita:
```
pipe(
  eliminarDuplicados,
  ordenar,
  filtrarPares,
  multiplicarPorDos
)(array)
```

16) Cache con expiración: Implementa un sistema de cache usando objetos donde cada entrada tenga tiempo de expiración.

17) Diferencia entre arrays de objetos: Devuelve los objetos que están en A pero no en B.

18) Merge profundo: Fusiona dos objetos profundamente sin sobrescribir objetos anidados.

19) Resolver estructura tipo Graph
Dado un array de conexiones:
```
[
  ["A", "B"],
  ["A", "C"],
  ["B", "D"]
]
```
Construye un grafo como objeto.

20) Motor de reglas dinámicas
Dado:
```
let reglas = [
  { campo: "edad", operador: ">", valor: 18 },
  { campo: "pais", operador: "==", valor: "MX" }
];
```
Crea una función que filtre un array de usuarios aplicando reglas dinámicas.

21) Proxy avanzado:
Usa Proxy para:

- Evitar que se asignen valores negativos
- Registrar cambios en consola

22) Deep Clone: Crea una función que haga una copia profunda (deep copy) sin usar structuredClone().

23) Crear un mini ORM: Convierte objetos en formato JSON en instancias de una clase automáticamente.

24) ULTRA DESAFÍO
Crea un mini framework que:

- Valide esquemas de objetos
- Permita transformaciones
- Soporte middlewares
- Tenga sistema de eventos
- Permita hooks antes y después de modificar datos

Todo usando arrays y objetos.