1. Array de Números

let numeros = [10, 20, 30, 40, 50];

console.log("Array de números:", numeros);

console.log("Primer elemento:", numeros[0]);

numeros[2] = 35; // Modificar elemento

console.log("Array modificado:", numeros);

Explicación:

- Se crea un **array de tipo numérico** con cinco elementos.
- Se muestra todo el array con console.log.
- Se accede al **primer elemento** usando numeros[0] (los arrays en JavaScript empiezan en índice 0).
- Se modifica el **tercer elemento** (numeros[2]) y se asigna un nuevo valor.
- **let** → es una palabra clave de JavaScript para **declarar una variable**.
    - Permite cambiar el valor de esa variable después de crearla.
    - A diferencia de const (constante) que no permite reasignar la variable completa.
- **numeros** → es el **nombre** que le estamos dando a esa variable.
    - Se recomienda que los nombres sean descriptivos, en este caso números indica que la variable almacenará **números**.
    - Es un **identificador**, no lleva comillas porque no es un texto, es el nombre de la variable.
- **= [10, 20, 30, 40, 50]** → el signo = asigna el valor a la variable, y el valor aquí es un **array** (arreglo) que contiene cinco números.
    - Los corchetes [] indican que es un array.
    - Los elementos están separados por comas.

En resumen:

- let numeros = crea una variable llamada **numeros**.
- [10, 20, 30, 40, 50] = es un **array de tipo numérico** que guardamos en esa variable.

2. Array de Cadenas de Texto

```
let frutas = ["Manzana", "Banana", "Cereza"];

frutas.push("Mango"); // Agregar al final

console.log("Frutas:", frutas);

frutas.pop(); // Eliminar último

console.log("Frutas después de pop():", frutas);

console.log("Número de frutas:", frutas.length);
```

Explicación:

- El array almacena **strings** (cadenas de texto).
- .push() agrega un elemento **al final** del array.
- .pop() elimina el **último elemento** del array.
- .length devuelve el **número de elementos** que tiene.

3. Recorrer un Array con for

```
let edades = [18, 21, 25, 30];

for (let i = 0; i < edades.length; i++) {
  console.log(` Edad ${i + 1}:` , edades[i]);
}
```

Explicación:

- for es un bucle tradicional que:
    1. Inicia i = 0 (primer elemento).
    2. Continua mientras i < edades.length (hasta el último elemento).
    3. Incrementa i en cada iteración.
- edades[i] accede a cada valor por posición.
- Se usa **template string** (` ${}`) para mostrar texto y variable juntas.

Estructura general del for

El for en JavaScript tiene esta estructura:

```
for (inicialización; condición; incremento) {
  // Código a ejecutar
}
```

En este caso:

- **Inicialización:** let i = 0
  - Se crea una variable i (contador) y se inicializa con el valor 0.
  - Este i será el índice para recorrer el array.
- **Condición:** i < edades.length
  - Se verifica si i es menor que la cantidad de elementos del array edades.
  - .length devuelve cuántos elementos tiene el array.
  - Mientras la condición sea verdadera, el bucle se seguirá ejecutando.
- **Incremento:** i++
  - Después de ejecutar el código dentro del for, i aumenta en 1 (i = i + 1).

Lo que pasa dentro del bucle

```
console.log(` Edad ${i + 1}:` , edades[i]);
```

- **edades[i]** → accede al elemento del array en la posición i.
  - Si i = 0, accede al primer elemento.
  - Si i = 1, accede al segundo elemento, etc.
- **Template string:** ` Edad ${i + 1}:`
  - Las comillas invertidas (`) permiten insertar variables dentro del texto usando ${}` .
  - i + 1 se usa para que al mostrar, empiece en **Edad 1** en lugar de **Edad 0** (porque los arrays empiezan en índice 0).
  - **console.log** muestra el mensaje en la consola con el texto y el valor correspondiente.

4. Recorrer un Array con forEach

```
let letras = ['A', 'B', 'C', 'D'];

letras.forEach(function(letra, indice) {
  console.log(` Índice ${indice}: ${letra}` );
});
```

Explicación:

- .forEach() recorre cada elemento **sin usar un índice manual**. Que realiza:
    - Internamente crea un contador.
    - Recorre todos los elementos.
    - Entrega directamente cada elemento y también el índice, sin que tener que manejarlo manualmente.
- Recibe una función que recibe dos parámetros:
    - letra → valor del elemento.
    - indice → posición en el array.
- Es más legible que un for tradicional cuando solo queremos leer datos.

5. Filtrar elementos de un Array

```
let numerosGrandes = [5, 12, 8, 130, 44];

let filtrados = numerosGrandes.filter(num => num > 10);

console.log("Números mayores a 10:", filtrados);
```

Explicación:

- .filter() crea un **nuevo array** con elementos que cumplan una condición.
- En este caso, la condición es num > 10.
- **No modifica** el array original, solo devuelve los que cumplen el filtro.

6. Ordenar un Array

let valores = [40, 100, 1, 5, 25, 10];

valores.sort((a, b) => a - b); // Ascendente

console.log("Orden ascendente:", valores);

valores.sort((a, b) => b - a); // Descendente

console.log("Orden descendente:", valores);

Explicación:

- .sort() ordena elementos, pero por defecto lo hace como **texto** (lexicográficamente).
- Para números, se usa una **función de comparación**:
    - a - b → ordena de menor a mayor.
    - b - a → ordena de mayor a menor.

7. Buscar un elemento

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

let existeVerde = colores.includes("Verde");

console.log("¿Existe el color Verde?:", existeVerde);

let indiceAzul = colores.indexOf("Azul");

console.log("Posición del color Azul:", indiceAzul);

Explicación:

- .includes(valor) → devuelve true si el valor existe en el array, false si no.
- .indexOf(valor) → devuelve la **posición** del elemento o -1 si no lo encuentra.