Arrays

Web Development

JavaScript

Los arrays son estructuras de datos que permiten almacenar varios valores en una sola variable.

Cada valor tiene una posición o índice, que empieza en 0.

```
const frutas = ["Manzana", "Banana", "Naranja"];

console.log(frutas[0]); // "Manzana"

console.log(frutas[2]); // "Naranja"
```

JavaScript

JavaScript ofrece muchas funciones para trabajar fácilmente con arreglos.

length: Devuelve la cantidad de elementos en el arreglo.

```
const numeros = [10, 20, 30];

console.log(numeros.length); // 3
```

push() y pop()

- push() agrega un elemento al final del arreglo.
- pop() elimina el último elemento del arreglo.

```
const colores = ["Rojo", "Azul"];

colores.push("Verde");   // ["Rojo", "Azul", "Verde"]

colores.pop();           // ["Rojo", "Azul"]
```

unshift() y shift()

- unshift() agrega un elemento al inicio del arreglo.
- shift() elimina el primer elemento.

```
const letras = ["B", "C"];

letras.unshift("A");  // ["A", "B", "C"]

letras.shift();       // ["B", "C"]
```

indexOf() y includes()

Sirven para buscar elementos dentro del arreglo.

```
const numeros = [5, 10, 15];

console.log(numeros.indexOf(10)); // 1

console.log(numeros.includes(20)); // false
```

forEach()

Ejecuta una función para cada elemento del arreglo.

```
const animales = ["Perro", "Gato", "Loro"];

animales.forEach(animal => console.log(animal));
```

map()

Crea un nuevo arreglo con los resultados de aplicar una función a cada elemento.

```
const nums = [1, 2, 3];

const doble = nums.map(n => n * 2);

console.log(doble); // [2, 4, 6]
```

filter() y find()

- filter() devuelve un nuevo arreglo con los elementos que cumplan una condición.
- find() devuelve el primer elemento que cumple la condición.

```
const edades = [12, 18, 25, 30];

const adultos = edades.filter(e => e >= 18);

const primerAdulto = edades.find(e => e >= 18);


console.log(adultos); // [18, 25, 30]

console.log(primerAdulto); // 18
```

join()

Une todos los elementos del arreglo en una sola cadena.

```
const palabras = ["Hola", "mundo"];

console.log(palabras.join(" ")); // "Hola mundo"
```

concat() y slice()

- concat() une dos o más arreglos.
- slice() extrae una parte del arreglo sin modificar el original.

```
const a = [1, 2];

const b = [3, 4];

const c = a.concat(b); // [1, 2, 3, 4]


const parte = c.slice(1, 3); // [2, 3]
```

sort() y reverse()

- sort() ordena los elementos (alfabéticamente o numéricamente).
- reverse() invierte el orden.

```
const nombres = ["Ana", "Luis", "Pedro"];

nombres.sort();     // ["Ana", "Luis", "Pedro"]

nombres.reverse();  // ["Pedro", "Luis", "Ana"]
```

Iterar sobre arreglos en JavaScript

for clásico (tradicional):

Es la forma más antigua y detallada de recorrer un arreglo.

Usa un contador para acceder a cada elemento por su índice.

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

for (let i = 0; i < numeros.length; i++) {
  console.log("Índice:", i, "→ Valor:", numeros[i]);
}
```

Iterar sobre arreglos en JavaScript

Ventajas:

- Puedes usar el índice (i) directamente.
- Permite modificar los elementos del arreglo durante la iteración.
- Total control sobre el recorrido (inicio, condición y paso).

Desventajas:

- Más código.
- Más propenso a errores si se confunde el límite o el índice.
- Menos legible cuando solo se necesita el valor.

Iterar sobre arreglos en JavaScript

for...of (moderno o mejorado)

- Es la forma moderna y sencilla de recorrer arreglos o cualquier estructura iterable (como strings o sets).
- Recorre directamente los valores sin necesidad de índices.

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


for (let n of numeros) {
  console.log("Valor:", n);
}
```

Iterar sobre arreglos en JavaScript

Ventajas:

- Más simple y legible.
- Ideal cuando solo necesitas los valores del arreglo.
- Menos errores de sintaxis.

Desventajas:

- No accede directamente al índice (aunque puedes obtenerlo con un contador externo o entries()).

ACTIVIDAD

Resuelva los siguientes ejercicios en JavaScript sin usar inteligencia artificial o google o cualquier otro recurso para encontrar la solución

1. Crea un programa que recorra un arreglo de números y calcule la suma total de todos sus elementos.
2. Crea un programa que recorra un arreglo de números y separe los pares e impares en dos nuevos arreglos.
3. Crea un arreglo con varias palabras y muestra en consola la palabra más larga.
4. Crea un programa que muestre el arreglo invertido, sin usar el método reverse().
5. Crea un programa que reciba un arreglo de números y un número límite.

   El programa debe mostrar solo los números que sean mayores al límite.