Trabajo Práctico: reduce en JavaScript

**Paradigmas de programación**Curso: \_\_\_C1\_\_\_\_\_\_\_\_\_ Estudiante: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Objetivos de aprendizaje

* - Comprender el funcionamiento del método reduce en JavaScript.
* - Resolver problemas de suma, producto, conteo, agrupamiento y transformaciones con una sola pasada.
* - Practicar el diseño de acumuladores (número, string, objeto, array, Map).

# Instrucciones

1. 1) Leé atentamente cada enunciado.
2. 2) Implementá la solución usando EXCLUSIVAMENTE reduce en la parte central (podés usar helpers si lo necesitás).
3. 3) Probá con los ejemplos de entrada/salida y agregá casos borde.
4. 4) Escribí el código con nombres claros y comentarios breves.

# Recordatorio de reduce

Recordatorio rápido de reduce:

array.reduce((acumulador, elementoActual) => nuevaVersionDelAcumulador, valorInicial)

• Reduce transforma “muchos” en “uno”.

• El acumulador puede ser de cualquier tipo: número, string, objeto, array, Map, etc.

• Siempre definí un valor inicial para evitar errores con arrays vacíos.

## Nivel Básico

### 1) Suma total

Dado un array de números, obtené la suma de todos sus elementos.

Entrada de ejemplo: [2, 3, 5]

Salida esperada: 10

Pista: Acumulá un número que empieza en 0.

### 2) Producto total

Calculá el producto de todos los números de un array.

Entrada de ejemplo: [2, 3, 4]

Salida esperada: 24

Pista: Empezá en 1 y andá multiplicando.

### 3) Concatenación con separador "-"

Uní todas las palabras de un array en un solo string separados por “-”.

Entrada de ejemplo: ["hola","mundo","js"]

Salida esperada: "hola-mundo-js"

Pista: Cuidá no agregar “-” adelante del primero.

### 4) Cantidad de ocurrencias

Contá cuántas veces aparece cada fruta y devolvé un objeto { fruta: cantidad }.

Entrada de ejemplo: ["pera","manzana","pera","uva"]

Salida esperada: { pera: 2, manzana: 1, uva: 1 }

Pista: Usá un objeto y sumá 1 por clave.

### 5) Eliminar duplicados

A partir de un array con repetidos, devolvé un array sin duplicados (mismo orden de primera aparición).

Entrada de ejemplo: [3,3,1,2,1]

Salida esperada: [3,1,2]

Pista: Si ya lo viste, no lo agregues.

## Nivel Intermedio

### 6) Aplanar un nivel

Aplaná un array de arrays (una sola profundidad) en un único array.

Entrada de ejemplo: [[1,2],[3],[4,5]]

Salida esperada: [1,2,3,4,5]

Pista: Concatená cada sub-array a un acumulador [].

### 7) Índice por id

Convertí un array de objetos en un índice por id: { [id]: objetoCompleto }.

Entrada de ejemplo: [{id:1,n:"A"},{id:2,n:"B"}]

Salida esperada: { 1:{id:1,n:"A"}, 2:{id:2,n:"B"} }

Pista: Acumulador {}, asigná acc[obj.id] = obj.

### 8) Máximo por propiedad

Dada una lista de empleados, obtené el empleado con mayor salario.

Entrada de ejemplo: [{n:"Ana",s:50},{n:"Luis",s:80},{n:"Sol",s:70}]

Salida esperada: {n:"Luis",s:80}

Pista: Guardá el “mejor hasta ahora”.

### 9) Particionar pares e impares

Separá números en un objeto con dos arrays: { pares:[], impares:[] }.

Entrada de ejemplo: [1,2,3,4,5]

Salida esperada: { pares:[2,4], impares:[1,3,5] }

Pista: Elegí el array correcto y push.

### 10) Histograma de letras

Dado un string, devolvé un objeto con la cantidad de veces que aparece cada letra (ignorando espacios y en minúsculas).

Entrada de ejemplo: "Hola Hola"

Salida esperada: { h:2, o:2, l:2, a:2 }

Pista: Pasá a minúsculas, quitá espacios y contá.

## Nivel Avanzado

### 11) Totales por categoría

Dado un carrito con {categoria, precio}, calculá el total gastado por categoría.

Entrada de ejemplo: [{c:"libros",p:100},{c:"tech",p:200},{c:"libros",p:50}]

Salida esperada: { libros:150, tech:200 }

Pista: Sumá precio por categoria.

### 12) Promedios por grupo (una pasada)

Dada una lista de notas {curso, nota}, devolvé {curso: promedio} sin recorrer dos veces (acumulá sumas y conteos en el mismo reduce).

Entrada de ejemplo: [{c:"A",n:8},{c:"B",n:6},{c:"A",n:10}]

Salida esperada: { A:9, B:6 }

Pista: Acumulá {suma, cuenta} por curso y al final calculás suma/cuenta.

### 13) Pipeline de funciones

Dada una lista de funciones unarias y un valor inicial, aplicá todas en cadena con reduce (composición izquierda→derecha).

Entrada de ejemplo: fns=[x=>x+1, x=>x\*2], valor=3

Salida esperada: 8

Pista: El acumulador es el valor actual transformado.

### 14) Verificador de paréntesis

Con reduce, verificá si una cadena de paréntesis está balanceada. Si en algún momento el conteo baja de 0 o termina distinto de 0, es inválida.

Entrada de ejemplo: "(()())" y "())("

Salida esperada: true y false, respectivamente

Pista: "(" suma 1, ")" resta 1.

### 15) Suma acumulada (prefix sum)

Generá un nuevo array con la suma acumulada de los elementos.

Entrada de ejemplo: [3,1,4,2]

Salida esperada: [3,4,8,10]

Pista: El siguiente valor es previo + actual.

# PARTE B – Soluciones (solo uso docente)

## 1) Suma total

const suma = nums => nums.reduce((acc, n) => acc + n, 0);

## 2) Producto total

const producto = nums => nums.reduce((acc, n) => acc \* n, 1);

## 3) Join con "-"

const joinGuion = palabras => palabras.reduce((acc, w, i) => (i ? acc + "-" + w : w), "");

## 4) Conteo de ocurrencias

const contar = arr => arr.reduce((acc, x) => { acc[x] = (acc[x] || 0) + 1; return acc; }, {});

## 5) Eliminar duplicados

const unicos = arr => arr.reduce((acc, x) => (acc.includes(x) ? acc : [...acc, x]), []);

## 6) Aplanar un nivel

const aplanar1 = arrs => arrs.reduce((acc, sub) => acc.concat(sub), []);

## 7) Índice por id

const indexarPorId = arr => arr.reduce((acc, obj) => (acc[obj.id] = obj, acc), {});

## 8) Máximo por propiedad

const mayorSalario = empleados => empleados.slice(1).reduce((m, e) => e.s > m.s ? e : m, empleados[0]);  
const mayorSalarioSafe = empleados => empleados.length ? mayorSalario(empleados) : null;

## 9) Particionar pares/impares

const particionar = nums => nums.reduce((acc, n) => { (n % 2 === 0 ? acc.pares : acc.impares).push(n); return acc; }, { pares: [], impares: [] });

## 10) Histograma de letras

const histo = str => str.toLowerCase().replace(/\s+/g, "").split("").reduce((acc, ch) => (acc[ch] = (acc[ch] || 0) + 1, acc), {});

## 11) Totales por categoría

const totalPorCategoria = items => items.reduce((acc, it) => { acc[it.c] = (acc[it.c] || 0) + it.p; return acc; }, {});

## 12) Promedios por grupo

const promedios = notas => { const tmp = notas.reduce((acc, {c, n}) => { (acc[c] ??= { suma: 0, cuenta: 0 }); acc[c].suma += n; acc[c].cuenta += 1; return acc; }, {}); return Object.fromEntries(Object.entries(tmp).map(([c, {suma, cuenta}]) => [c, suma / cuenta])); };

## 13) Pipeline de funciones

const aplicarPipeline = (fns, valor) => fns.reduce((acc, f) => f(acc), valor);

## 14) Paréntesis balanceados

const balanceados = s => { const fin = [...s].reduce((acc, ch) => { if (acc < 0) return acc; if (ch === "(") return acc + 1; if (ch === ")") return acc - 1; return acc; }, 0); return fin === 0; };

## 15) Suma acumulada

const prefixSum = arr => arr.reduce((acc, n, i) => { acc.push(i === 0 ? n : acc[i - 1] + n); return acc; }, []);