FCyT
Sede Concepción del Uruguay

Licenciatura en Sistemas de Información

Programación Avanzada - 2025

Ejercicios JS

03 Práctica de JavaScript

---

1) Suma de números únicos

Tema: Arrays + Set + Funciones
Consigna: Implementá sumUnique(nums) que reciba un array y devuelva la suma de los números únicos (ignora duplicados, ignora no-numéricos).
Ejemplos:

- sumUnique([1,2,2,3]) → 1+2+3 = 6
- sumUnique([1,'2',2,3,'a']) → 1+2+3 = 6
  Criterios: Usar Set o filtrar por primera aparición; validar Number.isFinite.
  Starter:

```
function sumUnique(nums) {
  // tu código
}
```

2) Seleccionar propiedades

Tema: Objetos + Funciones puras
Consigna: Escribí `pick(obj, keys)` que devuelva un **nuevo objeto** con solo las claves indicadas (ignorar las que no existan).
Ejemplos:
`pick({a:1,b:2,c:3}, ['a','c','z']) → {a:1, c:3}`
Criterios: No mutar `obj`.

function pick(obj, keys) {

// tu código

}

3) Agrupar por clave o función

Tema: Arrays + Objetos + HOF
Consigna: `groupBy(list, keyOrFn)` agrupa una lista por:

- **string**: nombre de propiedad
- **función**: `(item) => clave`
  Ejemplos:
  `groupBy([{t:'a'},{t:'b'},{t:'a'}], 't')` → `{ a:[...], b:[...] }`
  `groupBy([6,7,8,9], n => n%2?'impar':'par')` → `{ par:[6,8], impar:[7,9] }`
  Criterios: Retornar objeto plano; no mutar `list`.

function groupBy(list, keyOrFn) {

// tu código

}

4) Ordenar por múltiples campos

Tema: Arrays + sort + comparadores
Consigna: `sortByMany(list, specs)` donde `specs` es un array de reglas `{ key, dir }` con `dir='asc'|'desc'`.
Ejemplo:
Ordenar usuarios por `lastName asc` y `age desc`.
Criterios: No mutar el array original (clonar antes).

function sortByMany(list, specs) {

// tu código

}

5) deepEqual (objetos/arrays simples)

Tema: Objetos + Recursividad
Consigna: `deepEqual(a,b)` compara **primitivos**, **arrays** y **objetos planos** (sin funciones ni fechas).
Ejemplos:
`deepEqual({x:[1,2]}, {x:[1,2]}) → true`
`deepEqual({x:1}, {x:'1'}) → false`
Criterios: Mismo conjunto de claves; manejar `null`.

```
function deepEqual(a, b) {

  // tu código

}
```

6) Validador de paréntesis

Tema: Estructuras de datos (Stack)
Consigna: `isBalanced(s)` retorna `true` si `()[]{}` están balanceados y bien anidados.
Ejemplos:
`"([]{})" → true`, `"(]" → false`, `"([)]" → false`
Criterios: Usar un stack (array como pila).

```
function isBalanced(s) {

  // tu código

}
```

7) Frecuencia de palabras

Tema: Map/Set + Strings
Consigna: `wordFreq(text)` que devuelva un `Map` con la frecuencia de cada palabra **case-insensitive** y sin puntuación básica.
Ejemplo:
`"Hola, hola! chau." → { hola:2, chau:1 }`
Criterios: Normalizar a minúsculas; remover `[,.:;!?]`.

```
function wordFreq(text) {

  // tu código

}
```

9) Debounce

Tema: Funciones de orden superior + timers
Consigna: `debounce(fn, delay)` devuelve una función que **pospone** la ejecución de `fn` hasta que pasen `delay` ms sin ser llamada de nuevo.
Ejemplo: útil para `input` de búsqueda.
Criterios: Preservar `this` y argumentos; cancelar timer previo.

```
function debounce(fn, delay) {

  // tu código

}
```

10) Asincronismo: `withTimeout` + `allSettledLite`

Tema: Promesas + control de tiempo
Consigna A: `withTimeout(promise, ms)` rechaza con `Error('Timeout')` si `promise` no resuelve en `ms`.
Consigna B: `allSettledLite(promises)` devuelve un array de objetos `{ status:'fulfilled', value } | { status:'rejected', reason }` (sin usar `Promise.allSettled`).
Ejemplos:

- `withTimeout(fetchX(), 1000)`
- `allSettledLite([ok(), falla()]) → [{status:'fulfilled', value:...},{status:'rejected', reason:...}]`

```
function withTimeout(promise, ms) {
  // tu código
}
function allSettledLite(promises) {
  // tu código
}
```