

# Reporte de Evaluación - Fork de GitHub

## Información General

Estudiante: Mario Alonso Munera Velez  
Repositorio: MarioMunera1993/act\_web1\_s7  
Fecha de evaluación: 5/10/2025, 12:29:39  
Evaluado por: Sistema de Evaluación Masiva

## Resumen de Calificaciones

Calificación general: 4.5/5.0  
Actividades completadas: 10/10  
Porcentaje de completitud: 100.0%

## Detalle de Actividades

#	Descripción	Archivo	Encontrado	Calificación
1	Gestión de Inventario Básico - Crea un a...	src/ejercicio_01.js	Sí	4.0
2	Filtrado de Productos por Categoría - Us...	src/ejercicio_02.js	Sí	5.0
3	Transformación de Datos con map() - Crea...	src/ejercicio_03.js	Sí	4.0
4	Análisis de Ventas con reduce() - Dado u...	src/ejercicio_04.js	Sí	4.0
5	Búsqueda y Verificación - Crea un array ...	src/ejercicio_05.js	Sí	4.0
6	Manipulación de Arrays - Crea un array i...	src/ejercicio_06.js	Sí	5.0
7	Ordenamiento y Reversión - Crea arrays d...	src/ejercicio_07.js	Sí	4.0
8	Desestructuración de Arrays - Dado el ar...	src/ejercicio_08.js	Sí	5.0
9	Desestructuración de Objetos - Crea un o...	src/ejercicio_09.js	Sí	5.0
10	Métodos de Objeto - Crea un objeto y dem...	src/ejercicio_10.js	Sí	5.0

## Retroalimentación Detallada

**Actividad 1: Gestión de Inventario Básico - Crea un array de objetos que represente un inventario de productos. Cada producto debe tener: id, nombre, precio, categoria, stock. Declara al menos 5 productos y muestra todos los productos, total de productos en inventario y valor total del inventario.**

Archivo esperado: src/ejercicio\_01.js

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución resuelve correctamente el problema. Se recomienda usar nombres de propiedades en minúsculas (id, nombre, stock) para seguir convenciones de JavaScript y mejorar la legibilidad.

**Actividad 2: Filtrado de Productos por Categoría - Usando el inventario del ejercicio anterior, utiliza el método filter() para mostrar solo productos de la categoría 'Electrónicos', productos con stock menor a 10 y productos con precio mayor a \$500.**

Archivo esperado: src/ejercicio\_02.js

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y cumple con todos los requisitos. El código es legible y fácil de entender.

**Actividad 3: Transformación de Datos con map() - Crea un array de estudiantes con nombre, edad, notas (array de números). Usa map() para crear un nuevo array con solo los nombres, crear un array con el promedio de cada estudiante y agregar una propiedad estado ('Aprobado' si promedio  $\geq 70$ , 'Reprobado' si  $< 70$ ).**

Archivo esperado: src/ejercicio\_03.js

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y cumple con los requisitos. La lógica para calcular el promedio y el estado es funcional, pero la función `revision` podría simplificarse eliminando las variables redundantes. Además, el rango de aprobación en la función `revision` es incorrecto, debería ser  $\geq 70$  y  $< 70$ , no entre 3 y 5 (se asume que 70 se representa como 3.5 al dividir las notas entre 20).

**Actividad 4: Análisis de Ventas con reduce() - Dado un array de ventas con producto, cantidad, precio, fecha. Usa reduce() para calcular total de ingresos, producto más vendido (por cantidad) y promedio de venta por transacción.**

Archivo esperado: src/ejercicio\_04.js

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y utiliza `reduce()` adecuadamente. Podría mejorarse la legibilidad al encadenar operaciones o usar funciones auxiliares para el producto más vendido y evitar la mutación directa del acumulador en el primer `reduce`.

**Actividad 5: Búsqueda y Verificación - Crea un array de usuarios con id, nombre, email, activo. Implementa búsquedas usando find() para buscar usuario por email, findIndex() para obtener posición de usuario por id, some() para verificar si hay usuarios inactivos y every() para verificar si todos tienen email válido (contiene @).**

Archivo esperado: src/ejercicio\_05.js

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Se podrían añadir validaciones más robustas para el email y considerar el uso de funciones nombradas para mayor claridad.

**Actividad 6: Manipulación de Arrays - Crea un array inicial [1, 2, 3, 4, 5] y demuestra push() y pop() (agregar y quitar del final), shift() y unshift() (agregar y quitar del inicio), splice() (insertar elementos en posición específica) y slice() (extraer porción sin modificar original).**

Archivo esperado: src/ejercicio\_06.js

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La actividad se resolvió correctamente y el código es legible. Demuestra el uso correcto de los métodos de array solicitados y la salida es clara.

**Actividad 7: Ordenamiento y Reversión - Crea arrays de números desordenados (ordena ascendente y descendente), nombres de personas (ordena alfabéticamente), objetos con propiedad edad (ordena por edad) y usa reverse() para invertir el orden.**

Archivo esperado: src/ejercicio\_07.js

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La actividad se resuelve correctamente para números y nombres. Falta la ordenación de objetos por edad y el uso de `reverse()` para invertir el orden como se pide en la descripción.

**Actividad 8: Desestructuración de Arrays - Dado el array ['JavaScript', 'Python', 'Java', 'C++', 'Go']: extrae los primeros 3 lenguajes, extrae el primero y el último, usa rest operator para separar el primero del resto e intercambia dos variables usando desestructuración.**

Archivo esperado: src/ejercicio\_08.js

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y completa, cumpliendo con todos los requisitos de la actividad. El código es claro, conciso y utiliza las características de desestructuración de arrays de forma adecuada.

**Actividad 9: Desestructuración de Objetos - Crea un objeto persona con propiedades anidadas (dirección, contacto). Demuestra desestructuración básica, renombrado de variables, valores por defecto, desestructuración anidada y rest operator en objetos.**

Archivo esperado: src/ejercicio\_09.js

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. Demuestra un entendimiento completo de la desestructuración de objetos, incluyendo casos básicos, renombrado, valores por defecto, anidación y el uso del operador rest. El código es claro y bien estructurado.

**Actividad 10: Métodos de Objeto - Crea un objeto y demuestra Object.keys() (obtener claves), Object.values() (obtener valores), Object.entries() (obtener pares clave-valor) e iterar sobre el objeto con forEach().**

Archivo esperado: src/ejercicio\_10.js

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código es claro, conciso y demuestra correctamente el uso de Object.keys(), Object.values(), Object.entries() y la iteración con forEach().

## Resumen General

Excelente trabajo. Completó 10/10 actividades (100%) con una calificación promedio de 4.5/5. Demuestra buen dominio de los conceptos.

## Recomendaciones

- Continuar con el excelente trabajo y mantener la calidad del código