Reporte de Evaluación - Fork de GitHub

Información General

Estudiante: paula andrea gil vargas
Repositorio: GGP113/act_web1_s7
Fecha de evaluación: 11/9/2025, 14:12:34
Evaluado por: Sistema de Evaluación Masiva

Resumen de Calificaciones

Calificación general: 4.2/5.0
Actividades completadas: 10/10
Porcentaje de completitud: 100.0%

Detalle de Actividades

#	Descripción	Archivo	Encontrado	Calificación
1	Gestión de Inventario Básico - Crea un a...	ejercicios/ejercicio_01.js	Sí	4.0
2	Filtrado de Productos por Categoría - Us...	ejercicios/ejercicio_02.js	Sí	5.0
3	Transformación de Datos con map() - Crea...	ejercicios/ejercicio_03.js	Sí	4.0
4	Análisis de Ventas con reduce() - Dado u...	ejercicios/ejercicio_04.js	Sí	3.0
5	Búsqueda y Verificación - Crea un array ...	ejercicios/ejercicio_05.js	Sí	5.0
6	Manipulación de Arrays - Crea un array i...	ejercicios/ejercicio_06.js	Sí	5.0
7	Ordenamiento y Reversión - Crea arrays d...	ejercicios/ejercicio_07.js	Sí	4.0
8	Desestructuración de Arrays - Dado el ar...	ejercicios/ejercicio_08.js	Sí	3.0
9	Desestructuración de Objetos - Crea un o...	ejercicios/ejercicio_09.js	Sí	4.0
10	Métodos de Objeto - Crea un objeto y dem...	ejercicios/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: ejercicios/ejercicio_01.js

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Podría mejorar la presentación del inventario mostrando más detalles de cada producto y calcular el valor total considerando el stock.

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: ejercicios/ejercicio_02.js

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. El código es claro, conciso y cumple con todos los requisitos de la actividad utilizando correctamente el método filter y mostrando la información de manera legible.

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 >= 70, 'Reprobado' si < 70).

Archivo esperado: ejercicios/ejercicio_03.js

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Se puede mejorar la legibilidad del código al evitar el uso de `estudiantes.indexOf(est)` en el cálculo del promedio, lo cual puede ser ineficiente. Considera calcular el promedio dentro del primer `map` para evitar múltiples iteraciones sobre el array.

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: ejercicios/ejercicio_04.js

Estado: Archivo encontrado

Calificación: 3.0/5.0

Retroalimentación:

La lógica para `totalIngresos` es correcta. Sin embargo, el cálculo del `promedioVentas` solo suma las cantidades, no los ingresos por venta. La función reduce para encontrar el producto más vendido parece funcionar pero la inicialización del acumulador no es la correcta, podría dar resultados inesperados si la lista está vacía. Además, hay errores de tipeo menores ('vantal').

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: ejercicios/ejercicio_05.js

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y completa, implementando todas las funcionalidades requeridas. El código es limpio, legible y utiliza las funciones find(), findIndex(), some() y every() de manera eficiente.

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: ejercicios/ejercicio_06.js

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

La solución es correcta y demuestra el uso de los métodos de array solicitados. El código es claro y conciso, facilitando su comprensión y mantenimiento.

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: ejercicios/ejercicio_07.js

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La solución es correcta y funcional. Sería bueno incluir la función reverse() después de cada ordenamiento para ver su efecto individualmente y agregar comentarios para mayor claridad.

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: ejercicios/ejercicio_08.js

Estado: Archivo encontrado

Calificación: 3.0/5.0

Retroalimentación:

La extracción de los primeros 3 y el primero/último se realiza de forma poco óptima (slice/indexación directa en lugar de desestructuración). El resto de la desestructuración es correcta, pero la salida por consola podría ser más clara.

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: ejercicios/ejercicio_09.js

Estado: Archivo encontrado

Calificación: 4.0/5.0

Retroalimentación:

La desestructuración es correcta y cubre los puntos solicitados. Sin embargo, las variables 'nombre1' y 'edad1' no están definidas correctamente, deberían usar el nombre original para la desestructuración básica. Además, es importante eliminar las variables sin utilizar.

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: ejercicios/ejercicio_10.js

Estado: Archivo encontrado

Calificación: 5.0/5.0

Retroalimentación:

Excelente solución. Demuestra el uso correcto de Object.keys(), Object.values(), Object.entries() y la iteración con forEach(). El código es claro y funcional.

Resumen General

Excelente trabajo. Completó 10/10 actividades (100%) con una calificación promedio de 4.2/5. Demuestra buen dominio de los conceptos.

Recomendaciones

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