Reporte de Evaluación - Fork de GitHub

Información General

Estudiante: David Quiroz Gonzalez
Repositorio: Strikys12/act_web1_s7
Fecha de evaluación: 11/9/2025, 14:13:42
Evaluado por: Sistema de Evaluación Masiva

Resumen de Calificaciones

Calificación general: 3.4/5.0
Actividades completadas: 9/10
Porcentaje de completitud: 90.0%

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	4.0
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	3.0
6	Manipulación de Arrays - Crea un array i...	ejercicios/ejercicio_06.js	Sí	4.0
7	Ordenamiento y Reversión - Crea arrays d...	ejercicios/ejercicio_07.js	Sí	3.0
8	Desestructuración de Arrays - Dado el ar...	ejercicios/ejercicio_08.js	Sí	5.0
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	No	0.0

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, pero se puede mejorar la eficiencia en el cálculo del total del inventario. Utilizar `reduce` directamente sobre el `inventario` para calcular el valor total y el total de stock sería más eficiente y legible.

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: 4.0/5.0

Retroalimentación:

La lógica de filtrado es correcta y el código funciona. Se debe prestar atención a los nombres de las variables (catLacteos, menor, mayor) para que sean más descriptivos. Además, la descripción de la actividad solicitaba filtrar por 'Electrónicos' en lugar de 'lacteos'.

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. Sin embargo, podrías agregar la propiedad 'estado' directamente dentro del segundo `map` para evitar el bucle `for`, mejorando la eficiencia y legibilidad del código.

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:

Calcula correctamente el total de ingresos usando `reduce()`, pero falta implementar la lógica para el producto más vendido y el promedio de venta por transacción. El código es legible pero incompleto.

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: 3.0/5.0

Retroalimentación:

La solución implementa las funciones find, findIndex, some y every correctamente, pero el email de Carlos no es válido, afectando la prueba de 'every'. Además, la lógica de some debería buscar usuarios INACTIVOS, no activos.

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: 4.0/5.0

Retroalimentación:

El código funciona correctamente y demuestra las operaciones solicitadas. Podrías añadir comentarios explicando cada operación para mejorar la claridad y mostrar el estado del array en cada paso con `console.log`.

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: 3.0/5.0

Retroalimentación:

La solución es parcialmente correcta. El ordenamiento de números y strings funciona, pero el ordenamiento de objetos por edad no es el correcto, ya que solo se ordena un array aparte con las edades y no el array original de objetos. Además, falta aplicar `reverse()` a los arrays de números y nombres.

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: 5.0/5.0

Retroalimentación:

La solución es correcta y utiliza la desestructuración de arrays de forma efectiva para resolver todos los requerimientos del ejercicio. El código es limpio y fácil de entender.

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 anidada y el rest operator están bien aplicados. Falta un ejemplo de valores por defecto y algunas variables no se usan después de desestructurar, lo cual resta claridad.

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 no encontrado

Calificación: 0.0/5.0

Resumen General

Buen trabajo general. Completó 9/10 actividades (90%) con una calificación promedio de 3.4/5. Hay oportunidades de mejora en algunos aspectos.

Recomendaciones

- Completar los archivos faltantes: ejercicios/ejercicio_10.js