Ejercicios Propuestos 3

Ejercicios Practica 04-07 JS Métodos Arrays para Programación funcional con funciones callback. includes,isArray ,every ,some ,find ,findLast ,findIndex ,findLastIndex ,lastIndexOf

Estos ejercicios muestran cómo puedes utilizar funciones como Array.includes(), Array.isArray(), Array.every(), Array.some(), Array.find(), entre otras, para resolver problemas comunes, como la verificación de datos, búsqueda y validación de condiciones dentro de arrays. Cada ejercicio está contextualizado en situaciones reales que pueden ser útiles en tu trabajo diario con arrays en JavaScript. ¡Espero que te sean útiles!

Ejercicio 1: Verificación de productos en un inventario

Contexto: Tienes un inventario de productos y quieres verificar si un producto específico está en stock.

1. **Descripción:** Tienes un array con los nombres de los productos [“manzana”, “naranja”, “plátano”, “pera”]. Luego:
    - Usa Array.includes() para verificar si la "naranja" está en el inventario.
    - Usa Array.includes() nuevamente para comprobar si "mango" está en el inventario, pero empezando la búsqueda desde el índice 2.
2. **Resultado esperado:**

```
true  // "naranja" está en el inventario
false // "mango" no está en el inventario desde el índice 2
```

Ejercicio 2: Verificar si una variable es un array

Contexto: Estás procesando datos y necesitas asegurarte de que ciertas variables sean arrays.

1. **Descripción:** Tienes una variable data que podría ser un array o no. Luego:
    - Usa Array.isArray() para verificar si la variable data es realmente un array o no.
    - Si es un array, debes recorrerlo y mostrar los elementos. Si no es un array, muestra un mensaje de error.
2. **Resultado esperado:**

```
const data = [1, 2, 3];
Array.isArray(data); // true
```

Ejercicio 3: Validación de formularios

Contexto: Estás validando un formulario en el que se espera que todos los campos estén completados.

1. **Descripción:** Tienes un array que representa el estado de varios campos en un formulario, donde cada valor es un booleano indicando si el campo está lleno (true) o vacío (false): [true, true, false, true]. Luego:
    - Usa Array.every() para verificar si todos los campos están completos.
    - Si algún campo está vacío, muestra un mensaje de advertencia.
2. **Resultado esperado:**

```
const campos = [true, true, false, true];
campos.every(campo => campo); // false, aún hay campos vacíos
```

Ejercicio 4: Verificación de permisos de usuarios

Contexto: Estás verificando si al menos uno de los usuarios tiene permiso para realizar una acción.

1. **Descripción:** Tienes un array que indica si ciertos usuarios tienen permisos [false, false, true, false]. Luego:
    - Usa Array.some() para verificar si al menos un usuario tiene permisos para realizar una acción específica.
2. **Resultado esperado:**

```
const permisos = [false, false, true, false];
permisos.some(permiso => permiso); // true, al menos un usuario tiene permisos
```

Ejercicio 5: Encontrar el primer producto con stock bajo

Contexto: Tienes un inventario de productos con sus cantidades y necesitas encontrar el primer producto con un stock menor a 10 unidades.

1. **Descripción:** Tienes un array con las cantidades de stock [15, 7, 25, 5]. Luego:
    - Usa Array.find() para encontrar el primer producto con stock menor a 10.
2. **Resultado esperado:**

```
const stock = [15, 7, 25, 5];
stock.find(cantidad => cantidad < 10); // 7, es el primer producto con stock bajo
```

Ejercicio 6: Encontrar el último pedido urgente

Contexto: Estás gestionando pedidos y necesitas encontrar el último pedido con prioridad alta.

1. **Descripción:** Tienes un array con las prioridades de los pedidos [“baja”, “media”, “alta”, “baja”]. Luego:
    - Usa Array.findLast() para encontrar el último pedido con prioridad "alta".
2. **Resultado esperado:**

```
const prioridades = ["baja", "media", "alta", "baja"];
prioridades.findLast(prioridad => prioridad === "alta"); // "alta", es el último pedido urgente
```

Ejercicio 7: Encontrar el índice del primer producto en oferta

Contexto: En una tienda en línea, necesitas encontrar el índice del primer producto que está en oferta.

1. **Descripción:** Tienes un array que indica el precio de los productos [100, 50, 30, 150]. Luego:
    - Usa Array.findIndex() para encontrar el índice del primer producto cuyo precio es menor a 50 (producto en oferta).
2. **Resultado esperado:**

```
const precios = [100, 50, 30, 150];
precios.findIndex(precio => precio < 50); // 2, es el índice del primer producto en oferta
```

Ejercicio 8: Encontrar el índice del último cliente VIP

Contexto: Tienes una lista de clientes y necesitas encontrar el índice del último cliente VIP.

1. **Descripción:** Tienes un array con el estatus de los clientes [“regular”, “VIP”, “regular”, “VIP”]. Luego:
    - Usa Array.findLastIndex() para encontrar el índice del último cliente con estatus "VIP".
2. **Resultado esperado:**

```
const clientes = ["regular", "VIP", "regular", "VIP"];
clientes.findLastIndex(cliente => cliente === "VIP"); // 3, es el índice del último cliente VIP
```

Ejercicio 9: Búsqueda del índice de un producto específico

Contexto: Tienes una lista de productos y necesitas encontrar la posición de uno específico.

1. **Descripción:** Tienes un array con los nombres de los productos [“manzana”, “naranja”, “plátano”]. Luego:
    - Usa Array.indexOf() para encontrar el índice del producto "plátano".
    - Usa Array.lastIndexOf() para buscar el último índice del producto "manzana", en caso de que esté duplicado.
2. **Resultado esperado:**

```
const productos = ["manzana", "naranja", "plátano"];
productos.indexOf("plátano"); // 2
productos.lastIndexOf("manzana"); // 0, si es el único
```

Ejercicio 10: Validación de usuarios con un tipo específico

Contexto: Estás verificando si en una lista de usuarios hay un tipo de usuario específico (por ejemplo, "administrador").

1. **Descripción:** Tienes un array con los roles de los usuarios [“usuario”, “usuario”, “admin”, “moderador”]. Luego:
    - Usa Array.includes() para verificar si hay un administrador.
    - Usa Array.some() para verificar si al menos uno de los usuarios es un administrador.
2. **Resultado esperado:**

```
const roles = ["usuario", "usuario", "admin", "moderador"];
roles.includes("admin"); // true
roles.some(rol => rol === "admin"); // true
```