Ejercicios promesas

Ejercicio 1: Simulación de búsqueda de usuarios

Enunciado
Crea una aplicación web que permita simular una búsqueda de usuarios por nombre utilizando una promesa.

1. Cuando el usuario ingrese un nombre en un campo de texto y haga clic en un botón de "Buscar", muestra el mensaje "Buscando..." mientras esperas la respuesta.
2. Después de 2 segundos, resuelve la promesa mostrando si el usuario existe o no en una lista de usuarios simulada.
3. Si ocurre un error en el proceso de búsqueda (por ejemplo, el nombre está vacío), rechaza la promesa con un mensaje de error.

Requisitos:

- Usa un array de nombres para simular los datos de los usuarios.
- Muestra mensajes de error en rojo.

Ejercicio 2: Simulación de temporizador con promesas

Enunciado
Crea una aplicación web que simule un temporizador.

1. Pide al usuario ingresar un número de segundos en un campo de texto y haz clic en "Iniciar temporizador".
2. Utiliza una promesa para esperar los segundos indicados.
3. Al finalizar, muestra un mensaje diciendo: "El tiempo ha terminado".
4. Si el usuario no ingresa un número válido, rechaza la promesa con un mensaje de error.

Requisitos

- El temporizador debe aceptar solo valores numéricos positivos.
- Muestra errores si el usuario deja el campo vacío o ingresa valores incorrectos.

Ejercicio 3: Simulación de carga paralela con Promise.race

Enunciado:
Crea una aplicación web que cargue tres promesas que simulen tareas con tiempos distintos (por ejemplo, "Imagen A", "Imagen B" y "Imagen C").

1. Usa Promise.race para mostrar cuál de las tareas terminó primero.
2. Muestra el mensaje "La tarea más rápida fue: [nombre de la tarea]".
3. Si ocurre un error en cualquiera de las tareas, rechaza la promesa y muestra "Error al cargar las tareas".

Requisitos:

- Simula los tiempos de carga con setTimeout.
- Implementa un diseño simple con botones para iniciar la carga.

Ejercicio 4: then y encadenamiento de promesas

Enunciado:
Crea una aplicación web que simule un flujo de procesamiento encadenado:

1. Simula una promesa que representa la descarga de un archivo en 3 segundos.
2. Una vez descargado, otra promesa debe procesar los datos del archivo (2 segundos).
3. Finalmente, una tercera promesa debe guardar los datos procesados en un sistema (1 segundo).
4. Muestra mensajes en el DOM indicando en qué paso del proceso se encuentra la aplicación.

Requisitos:

- Usa encadenamiento con .then para gestionar las promesas.
- Si alguna promesa falla, muestra el error en rojo y detén el proceso.

Ejercicio 5: Creación de una promesa personalizada

Enunciado:
Crea una promesa personalizada que simule el lanzamiento de un dado.

1. Cuando el usuario haga clic en un botón, lanza el dado (genera un número aleatorio entre 1 y 6).
2. Si el resultado es mayor o igual a 4, resuelve la promesa mostrando "¡Éxito! Obtuviste un [número]".
3. Si el resultado es menor a 4, rechaza la promesa mostrando "Fallaste. Obtuviste un [número]".

Requisitos:

- Implementa un diseño simple con un botón y un área para mostrar los resultados.
- Utiliza estilos para distinguir entre éxito y fallo.

Ejercicio 6: Uso de Promise.allSettled

Enunciado:
Crea una aplicación web que simule la carga de cinco tareas con resultados variados (éxito o error).

1. Usa Promise.allSettled para gestionar todas las promesas.
2. Muestra una lista con el estado de cada tarea ("cumplida" o "rechazada") y el resultado correspondiente.
3. El proceso de cada tarea debe tomar un tiempo aleatorio entre 1 y 3 segundos.

Requisitos:

- Muestra los estados de las tareas en una lista.
- Diferencia visualmente entre tareas cumplidas y rechazadas.

Ejercicio 7: Simulación de pagos con una promesa

Enunciado:
Crea una aplicación web que simule el procesamiento de pagos con tarjeta.

1. El usuario debe ingresar un número de tarjeta, el CVV y la cantidad a pagar, y hacer clic en "Pagar".
2. Valida los datos ingresados (por ejemplo, que no estén vacíos y tengan el formato correcto).
3. Usa una promesa para simular el tiempo de procesamiento (2 segundos).
4. Si la promesa se resuelve, muestra "Pago realizado con éxito". Si se rechaza, muestra "Error al procesar el pago".

Requisitos:

- Los errores deben mostrarse en rojo.
- Valida los campos antes de enviar los datos.

Ejercicio 8: Conversión de callbacks a promesas

Enunciado:
Convierte una función que usa callbacks en una función que retorna una promesa.

1. Implementa una función llamada simularDescarga que acepte un callback y simule la descarga de un archivo (usa setTimeout para simular el tiempo de espera).
2. Reescribe la función para que en lugar de callbacks utilice promesas.
3. Usa la nueva función basada en promesas para descargar tres archivos y mostrar un mensaje al finalizar.

Requisitos:

- Asegúrate de usar la nueva versión con promesas en lugar de la anterior con callbacks.
- Muestra mensajes en el DOM indicando cuándo se descarga cada archivo.