Ejercicios async/await

Ejercicio 1: Cargar Datos de Múltiples APIs

Enunciado:
Crea una función que obtenga datos de dos APIs diferentes: https://jsonplaceholder.typicode.com/users y https://jsonplaceholder.typicode.com/posts. Primero, carga la lista de usuarios y luego carga las publicaciones. Muestra en la consola el nombre del primer usuario junto con los títulos de sus publicaciones.

Requisitos:

1. Usa async/await para realizar las solicitudes.
2. Usa Promise.all para cargar los datos en paralelo si es necesario.
3. Maneja errores con try/catch.

Ejercicio 2: Reintento Automático de Solicitudes Fallidas

Enunciado:
Crea una función fetchConReintento que intente obtener datos desde https://jsonplaceholder.typicode.com/posts/1. Si la solicitud falla, intenta nuevamente hasta 3 veces antes de mostrar un error.

Requisitos:

1. Implementa la lógica de reintento usando async/await.
2. Después de 3 intentos fallidos, muestra un mensaje de error.

Ejercicio 3: Carga Condicional de Datos

Enunciado:
Crea una función que primero obtenga un usuario desde https://jsonplaceholder.typicode.com/users/1. Si el usuario tiene un username que comienza con "B", carga sus publicaciones desde https://jsonplaceholder.typicode.com/posts?userId=1. Si no, muestra un mensaje indicando que no se cargarán publicaciones.

Requisitos:

1. Usa async/await y estructuras condicionales.
2. Maneja errores con try/catch.

Ejercicio 4: Temporizador de Carga con async/await

Enunciado:
Crea una función que simule un retraso de 3 segundos antes de cargar los datos de https://jsonplaceholder.typicode.com/photos. Muestra los primeros 5 títulos de las fotos en la consola después del retraso.

Requisitos:

1. Utiliza setTimeout combinado con async/await para simular el retraso.
2. Maneja errores con try/catch.

Ejercicio 5: Transformar Datos Obtenidos de una API

Enunciado:
Obtén una lista de tareas desde https://jsonplaceholder.typicode.com/todos. Filtra las tareas que estén completadas y transforma los datos para que solo se muestre el título en mayúsculas.

Requisitos:

1. Usa async/await para obtener los datos.
2. Filtra y transforma los datos antes de mostrarlos en la consola.
3. Maneja errores con try/catch.

Ejercicio 6: Actualización Parcial con PATCH

Enunciado:
Crea una función que actualice el título de una publicación con ID 1 en https://jsonplaceholder.typicode.com/posts/1 usando el método PATCH. Cambia el título a "Título Actualizado Parcialmente".

Requisitos:

1. Usa async/await y el método PATCH.
2. Muestra la respuesta del servidor en la consola.
3. Maneja errores con try/catch.

Ejercicio 7: Paginar Resultados con async/await

Enunciado:
Crea una función que obtenga publicaciones desde https://jsonplaceholder.typicode.com/posts. Muestra los resultados de 10 en 10, permitiendo que el usuario avance a la siguiente página presionando un botón.

Requisitos:

1. Usa async/await para obtener los datos.
2. Implementa la lógica de paginación en el DOM.
3. Maneja errores con try/catch.

Ejercicio 8: Simulación de Error Aleatorio

Enunciado:
Crea una función que realice una solicitud a https://jsonplaceholder.typicode.com/posts/1. Simula un error con una probabilidad del 50%. Si la solicitud tiene éxito, muestra los datos; si falla, muestra un mensaje de error.

Requisitos:

1. Usa async/await y Math.random() para simular el error.
2. Maneja errores con try/catch.

Ejercicio 9: Cargar Datos y Guardar Localmente

Enunciado:
Crea una función que obtenga datos de https://jsonplaceholder.typicode.com/users y los guarde en localStorage. Luego, crea otra función que lea estos datos desde localStorage y los muestre en la consola.

Requisitos:

1. Usa async/await para obtener los datos.
2. Usa localStorage para guardar y recuperar la información.
3. Maneja errores con try/catch.

Ejercicio 10: Validación de Respuesta del Servidor

Enunciado:
Realiza una solicitud a https://jsonplaceholder.typicode.com/posts/1. Antes de procesar la respuesta, verifica que el código de estado (status) sea 200. Si no lo es, lanza un error personalizado.

Requisitos:

1. Usa async/await para realizar la solicitud.
2. Valida el código de estado y maneja errores con try/catch.