Guía de Actividades
Promesas en JS

Guía de Actividades de Promesas en JavaScript

Actividad 1: Crear una promesa básica

- **Objetivo**: Familiarizarse con la creación y resolución de una promesa básica.
- **Instrucciones**:
    1. Crea una promesa que se resuelva después de 2 segundos.
    2. Cuando la promesa se resuelva, muestra el mensaje `"Promesa resuelta"` en la consola.
- **Tip**: Usa `setTimeout` para simular un retraso en la promesa.

Actividad 2: Promesa con datos de entrada y salida

- **Objetivo**: Aprender a pasar y devolver datos con promesas.
- **Instrucciones**:
    1. Crea una promesa que reciba un número como entrada.
    2. Si el número es mayor a 10, la promesa debe resolverse con el mensaje `"Número válido"`.
    3. Si el número es menor o igual a 10, debe rechazarse con el mensaje `"Número inválido"`.
    4. Usa `.then()` para manejar la resolución y `.catch()` para el rechazo.

Actividad 3: Simular una consulta a API con `setTimeout`

- **Objetivo**: Practicar el manejo de promesas simulando una respuesta de una API.
- **Instrucciones**:
    1. Crea una función que devuelva una promesa que se resuelve después de 3 segundos con un objeto que represente un usuario (nombre, edad, email).
    2. Muestra el objeto en la consola cuando la promesa se resuelva.
    3. Agrega un mensaje de `"Error de conexión"` en caso de rechazo de la promesa.

Actividad 4: Encadenamiento de promesas

- **Objetivo**: Aprender a encadenar promesas.
- **Instrucciones**:
    1. Crea una función `doblarNumero` que devuelva una promesa que tome un número, lo multiplique por 2 y lo devuelva después de 1 segundo.
    2. Crea otra función `sumarDiez` que tome un número, le sume 10 y lo devuelva como una promesa.
    3. Usa encadenamiento para tomar un número inicial, doblarlo, y luego sumar diez, mostrando el resultado final en la consola.

Actividad 5: Manejar errores en encadenamiento

- **Objetivo**: Profundizar en el manejo de errores en promesas encadenadas.
- **Instrucciones**:
    1. Extiende el ejercicio anterior: modifica la función `doblarNumero` para rechazar la promesa si el número es negativo.
    2. Usa `.catch()` en el encadenamiento para capturar el error en caso de que se pase un número negativo y muestra el mensaje `"Número no válido"`.

Actividad 6: Usar `Promise.all` para ejecutar promesas en paralelo

- **Objetivo**: Practicar la ejecución en paralelo de múltiples promesas.
- **Instrucciones**:
    1. Crea tres promesas que simulen llamadas a una API para obtener diferentes tipos de datos: usuario, posts y comentarios.
    2. Usa `Promise.all` para ejecutar todas las promesas en paralelo y espera a que todas se completen.
    3. Muestra en la consola un mensaje que diga `"Datos recibidos"` junto con los resultados cuando todas las promesas se hayan resuelto.

Actividad 7: Crear una función `delay` que use promesas

- **Objetivo**: Crear una función reusable de temporizador usando promesas.
- **Instrucciones**:
    1. Crea una función `delay` que reciba un tiempo en milisegundos y devuelva una promesa que se resuelva después de ese tiempo.

2. Usa esta función para crear un temporizador que muestre el mensaje `"Tiempo completado"` después de 5 segundos.
3. Usa `delay` en un encadenamiento de promesas para mostrar diferentes mensajes en distintos tiempos (2s, 3s, 5s).