**Hablemos un poco sobre el tema de las promesas en JavaScript.**

**Concepto básico**

Las promesas en JavaScript son como las promesas en la vida real. Imagina que un jefe te dice:

*"Necesito este reporte para mañana."*

Y tú respondes:

*"Te prometo que mañana lo tendrás en el escritorio."*

Luego, trabajas en el reporte, y tu jefe puede estar haciendo cualquier otra cosa, como ver Netflix. Una vez que terminas el reporte, le notificas que está listo.

Esto representa una **promesa exitosa**.

Si, por otro lado, no puedes completar el reporte, también debes notificar al jefe que **la promesa falló**.

El jefe no necesita saber los detalles del proceso; lo único que le importa es si la promesa se cumple o no.

Esto mismo ocurre en JavaScript con las promesas.

**Crear una promesa en JavaScript**

Aquí tienes el código básico para crear una promesa:

javascript

Copiar código

const promesa = new Promise((resolve, reject) => {

// Cuerpo de la promesa

});

Las promesas son **asíncronas**, lo que significa que primero se ejecuta todo el código **síncrono** y, una vez completado, se procesan las tareas pendientes de la promesa.

**Ejemplo con setTimeout**

javascript

Copiar código

const promesa = new Promise((resolve, reject) => {

setTimeout(() => {

console.log("Dos segundos después...");

resolve(); // Notifica que la promesa se cumplió

}, 2000);

});

promesa

.then(() => {

console.log("Then de la promesa");

})

.catch((error) => {

console.error("Algo salió mal:", error);

});

**Métodos principales de una promesa**

1. **then()**: Se ejecuta cuando la promesa es exitosa (llamamos resolve).
2. **catch()**: Se ejecuta cuando la promesa falla (llamamos reject).
3. **finally()**: Se ejecuta después de then() o catch(), sin importar el resultado.

**Cómo pasar valores al método then**

javascript

Copiar código

const promesa = new Promise((resolve, reject) => {

setTimeout(() => {

const resultado = "Información procesada";

resolve(resultado);

}, 2000);

});

promesa

.then((resultado) => {

console.log("Resultado de la promesa:", resultado);

})

.catch((error) => {

console.error("Error:", error);

});

**Ejercicio práctico**

Queremos simular la obtención de datos de una API utilizando una función llamada getHeroeById. Este es el proceso:

1. **Exportar la función** desde un archivo externo.
2. **Importarla** en nuestro archivo principal.
3. Usar un setTimeout para simular la obtención de datos.

**Código de ejemplo: getHeroeById.js**

javascript

Copiar código

const heroes = [

{ id: 1, name: "Batman" },

{ id: 2, name: "Superman" },

{ id: 3, name: "Wonder Woman" },

];

export const getHeroeById = (id) => {

return heroes.find((heroe) => heroe.id === id);

};

**Código de ejemplo: app.js**

javascript

Copiar código

import { getHeroeById } from "./getHeroeById";

const obtenerHeroe = new Promise((resolve, reject) => {

setTimeout(() => {

const heroe = getHeroeById(2);

if (heroe) {

resolve(heroe);

} else {

reject("Héroe no encontrado");

}

}, 2000);

});

obtenerHeroe

.then((heroe) => {

console.log("Héroe encontrado:", heroe);

})

.catch((error) => {

console.error("Error:", error);

});

**Resultado esperado:**

css

Copiar código

Dos segundos después...

Héroe encontrado: { id: 2, name: "Superman" }