Introducción a Promesas en JavaScript

🎯 Objetivo de la clase

- Entender qué es una Promesa.
- Comprender `resolve` y `reject`.
- Usar correctamente `.then()` y `.catch()`.
- Encadenar promesas simples.
- Identificar errores comunes al no retornar una promesa.

🟢 Ejercicio 1 – Mi Primera Promesa

🎯 Objetivo

Comprender la estructura básica de una promesa.

📘 Enunciado

Crear una función:

```
function saludar(nombre)
```

La función debe:

- Esperar 2 segundos usando `setTimeout`
- Si el nombre es `"(Tu nombre)"` → resolver con:

  ```
  "Hola (Tu nombre), bienvenido"
  ```

- Si no → rechazar con:

```
"No te conozco"
```

📌 Requisitos

- Usar `new Promise`
- Usar `setTimeout`
- Consumir la promesa con `.then()` y `.catch()`

💻 Ejemplo de uso esperado

```
saludar("Alejo")
  .then((mensaje) => {
    console.log(mensaje);
  })
  .catch((error) => {
    console.error(error);
  });
```

🟡 Ejercicio 2 – Encadenamiento Simple

🎯 Objetivo

Entender que el valor retornado en un `.then()` pasa al siguiente `.then()`.

📘 Enunciado

Crear dos funciones:

1️⃣ `obtenerNumero()`

- Espera 2 segundos.
- Resuelve con el número `10`.

2️⃣ `multiplicarPorDos(numero)`

- Espera 2 segundos.
- Resuelve con el número multiplicado por 2.

💻 Flujo esperado

```
obtenerNumero()
  .then((num) => {
    return multiplicarPorDos(num);
  })
  .then((resultado) => {
    console.log(resultado);
  })
  .catch((error) => {
    console.error(error);
  });
```

✅ Resultado esperado

```
20
```

⚠️ Ejercicio de análisis

Analiza este código:

```
obtenerNumero()
  .then((num) => {
    multiplicarPorDos(num);
  })
  .then((resultado) => {
    console.log(resultado);
  });
```

Responder:

- ¿Qué imprime?
- ¿Por qué?
- ¿Qué faltó?
- ¿Qué retorna realmente `.then()`?

🟠 Ejercicio 3 – Encadenamiento con Validación

🎯 Objetivo

Aprender cómo detener el flujo cuando ocurre un error.

📘 Enunciado

Crear dos funciones:

1️⃣ `obtenerEdad()`

- Espera 2 segundos.
- Resuelve con el número `17`.

2️⃣ `verificarMayorDeEdad(edad)`

- Si edad ≥ 18 → resolver con:

```
"Eres mayor de edad"
```

- Si edad < 18 → rechazar con:

```
"Eres menor de edad"
```

💻 Flujo esperado

```
obtenerEdad()
  .then((edad) => {
    return verificarMayorDeEdad(edad);
  })
  .then((mensaje) => {
    console.log(mensaje);
  })
  .catch((error) => {
    console.error(error);
  });
```

🎯 Resultado esperado

Debe ejecutarse el `.catch()` porque la edad es 17.

🧩 Conceptos Clave para Anotar en el Tablero

```
Promesa → resolve(valor) → .then(valor)
Promesa → reject(error) → .catch(error)
```

```
.then() SIEMPRE retorna una nueva promesa
```

📊 Criterios de Evaluación

Criterio
Usa correctamente `new Promise`
Retorna correctamente las promesas

Criterio
Usa `.then()` correctamente
Maneja errores con `.catch()`
Entiende el flujo asincrónico
Sustenta correctamente