REPORTE DE CASO DE ESTUDIO ASYNC/AWAIT.

Programación Web

15 DE ENERO DE 2026

UNIVERSIDAD DEL CARIBE
Axel Jesus Chimal Chan

1.- Introducción

El propósito fue entender cómo se pueden manejar procesos que tardan cierto tiempo en ejecutarse, como esperas o consultas a internet, sin que el código se vuelva complicado. Para realizar la práctica desarrollé una página web sencilla con dos ejercicios. Cada uno tiene su propio botón y muestra los resultados tanto en la consola como directamente en la página.

2.- Ejercicio 1: Simulación de validación

El primer ejercicio consiste en simular la validación de un usuario. Se utilizó una función que devuelve una Promesa, la cual se resuelve después de unos segundos usando setTimeout.

Gracias a la palabra clave await, el programa espera a que termine el proceso antes de continuar con la ejecución del código.

Código del Ejercicio 1:

En este caso, al presionar el botón, se muestra un mensaje de validación, se espera dos segundos y posteriormente se presenta el resultado final tanto en la página como en la consola.

```
5   function simularValidacion(){
6       return new Promise((resolve) => {
7           setTimeout(() => {
8               resolve("Usuario validado correctamente.");
9           }, 2000);
10      });
11  }
12
13  async function verificarUsuario(){
14
15      const area = document.getElementById("salida1");
16
17      area.innerHTML = "Validando usuario...";
18      console.log("Proceso de validación iniciado");
19
20      const resultado = await simularValidacion();
21
22      area.innerHTML = resultado;
23  }
```

Ejercicio 1 - Simular validación usuario

3.- Ejercicio 2: Consulta de Productos desde una API

En el segundo ejercicio se realizó una petición a una API pública para obtener información de productos en tiempo real. Para ello se utilizó la función fetch() junto con async/await.

También se implementó un bloque try/catch para manejar posibles errores, como problemas de conexión.

Código del Ejercicio 2:

Este ejercicio permite descargar datos reales desde internet y mostrarlos directamente en la página web, no solamente en la consola.

```
async function consultarProductos(){

    const area = document.getElementById("salida2");

    try{

        area.innerHTML = "Consultando productos...";
        console.log("Solicitando datos a la API");

        const respuesta = await fetch("https://fakestoreapi.com/products");

        const datos = await respuesta.json();

        const primerosProductos = datos.slice(0, 3);

        let html = "";

        primerosProductos.forEach(prod => {
            html += `
                <div class="producto">
                    <strong>${prod.title}</strong><br>
                    Precio: $${prod.price}
                </div>
            `;
        });

        area.innerHTML = html;
        console.log("Productos recibidos:", datos);

    }catch(error){

        area.innerHTML = "Error al obtener los productos.";
        console.error("Error:", error);

    }

}
```

Ejercicio 2 – Consultar Productos

4.- Conclusión

El uso de async/await facilita mucho la lectura y comprensión del código asíncrono. A diferencia del uso tradicional de callbacks, esta estructura permite escribir instrucciones de forma más ordenada, como si fueran secuenciales.

Además, se comprobó que es posible interactuar con APIs externas y mostrar la información en la página de manera dinámica, lo cual es fundamental en el desarrollo web moderno.