Estamos desarrollando una página web que obtiene datos de una **API pública** y los muestra en la interfaz.

Las peticiones pueden tardar **diferentes tiempos en responder**, por lo que usaremos varias estrategias con Promise.all(), Promise.allSettled(), Promise.race() y Promise.any() para manejar estas llamadas asíncronas de forma eficiente.

**Código HTML + JavaScript**

Este código tiene una interfaz básica con botones para ejecutar cada tipo de promesa y mostrar los resultados en pantalla.

**index.html**

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Ejercicio Promesas</title>

<style>

body { font-family: Arial, sans-serif; text-align: center; margin: 30px; }

button { margin: 10px; padding: 10px; cursor: pointer; }

pre { text-align: left; background: #f4f4f4; padding: 15px; border-radius: 5px; }

</style>

</head>

<body>

<h1>Ejercicio con Promesas</h1>

<button onclick="fetchAll()">Promise.all()</button>

<button onclick="fetchAllSettled()">Promise.allSettled()</button>

<button onclick="fetchRace()">Promise.race()</button>

<button onclick="fetchAny()">Promise.any()</button>

<h2>Resultados:</h2>

<pre id="output"></pre>

<script src="script.js"></script>

</body>

</html>

**script.js (Lógica con Promesas)**

// Elemento donde mostraremos los resultados

const output = document.getElementById("output");

// Función para obtener datos simulando tiempos de respuesta variables

function fetchData(url) {

return new Promise((resolve, reject) => {

const time = Math.random() \* 3000; // Simula tiempos entre 0 y 3 segundos

setTimeout(() => {

fetch(url)

.then(response => {

if (!response.ok) throw new Error(`Error en ${url}`);

return response.json();

})

.then(data => resolve({ url, data, time }))

.catch(error => reject(error));

}, time);

});

}

// URLs de prueba (API pública)

const userAPI = "https://jsonplaceholder.typicode.com/users";

const postsAPI = "https://jsonplaceholder.typicode.com/posts";

const commentsAPI = "https://jsonplaceholder.typicode.com/comments";

// Promise.all() - Espera a todas las respuestas o falla si una falla

function fetchAll() {

output.textContent = "Cargando datos con Promise.all()...";

Promise.all([fetchData(userAPI), fetchData(postsAPI)])

.then(results => {

output.textContent = "Promise.all() - Datos obtenidos:\n" + JSON.stringify(results, null, 2);

})

.catch(error => {

output.textContent = "Promise.all() - Error:\n" + error.message;

});

}

// Promise.allSettled() - Espera a todas sin importar si fallan

function fetchAllSettled() {

output.textContent = "Cargando datos con Promise.allSettled()...";

Promise.allSettled([fetchData(userAPI), fetchData("https://wrong.url"), fetchData(postsAPI)])

.then(results => {

output.textContent = "Promise.allSettled() - Resultados:\n" + JSON.stringify(results, null, 2);

});

}

// Promise.race() - Toma el primer resultado exitoso o fallido

function fetchRace() {

output.textContent = "Cargando datos con Promise.race()...";

Promise.race([fetchData(userAPI), fetchData(postsAPI)])

.then(result => {

output.textContent = "Promise.race() - Primer en responder:\n" + JSON.stringify(result, null, 2);

})

.catch(error => {

output.textContent = "Promise.race() - Error:\n" + error.message;

});

}

// Promise.any() - Toma el primer éxito y descarta los errores

function fetchAny() {

output.textContent = "Cargando datos con Promise.any()...";

Promise.any([fetchData("https://wrong.url"), fetchData(userAPI), fetchData(postsAPI)])

.then(result => {

output.textContent = "Promise.any() - Primer éxito:\n" + JSON.stringify(result, null, 2);

})

.catch(error => {

output.textContent = "Promise.any() - Todas fallaron:\n" + error.message;

});

}

**Explicación del código**

1. **HTML (index.html)**
   * Tiene **cuatro botones**, cada uno ejecuta un tipo de promesa.
   * Muestra los resultados en un <pre> para mejorar la legibilidad.
2. **JavaScript (script.js)**
   * La función fetchData(url) usa fetch() para obtener datos de la API y simula un **retraso aleatorio**.
   * Luego, se implementan las **cuatro variantes de promesas**:

**Promise.all() (Todas o ninguna)**

Promise.all([fetchData(userAPI), fetchData(postsAPI)])

Espera **todas las respuestas** y las muestra.  
Si **una sola** falla, toda la operación falla.

**Promise.allSettled() (Todas sin importar el estado)**

Promise.allSettled([fetchData(userAPI), fetchData("https://wrong.url"), fetchData(postsAPI)])

Devuelve **todas las respuestas**, exitosas o fallidas.  
No falla si alguna API responde mal, solo muestra los resultados con status: "fulfilled" o status: "rejected".

**Promise.race() (El primero en responder)**

Promise.race([fetchData(userAPI), fetchData(postsAPI)])

Devuelve **el primer resultado**, ya sea éxito o error.  
No espera a las demás respuestas.

**Promise.any() (El primer éxito, ignora errores)**

Promise.any([fetchData("https://wrong.url"), fetchData(userAPI), fetchData(postsAPI)])

Devuelve **el primer éxito** e **ignora** los errores.  
Si **todas fallan**, devuelve un AggregateError.