INSTITUTO KHIPU

Semestre IV

Sesión 26

PROGRAMA DE ESTUDIOS

DESARROLLOS DE SISTEMAS DE INFORMACIÓN

DESARROLLO FRONTEND DE SISTEMAS DE WEB

Tema

INTRODUCCIÓN A LA PROGRAMACIÓN ASÍNCRONA: PROMESAS Y CALLBACKS

Introducción a la Programación Asíncrona: Promesas y Callbacks

Objetivo:

Comprender el concepto de programación asíncrona en JavaScript y cómo se manejan tareas que requieren espera (como operaciones de red) usando **callbacks** y **promesas**.

PARTE CONCEPTUAL

¿Cómo podemos manejar código que depende de procesos tardados como consultas a una API?

¿Qué es la programación asíncrona?

Es un modelo de ejecución que **permite que el código no se bloquee** mientras espera una operación lenta (como leer archivos, consultar una API o usar temporizadores). JavaScript es de un solo hilo (single-threaded), por lo tanto, necesita este tipo de manejo para no detenerse.

Programación síncrona vs asíncrona

Síncrona:	Asíncrona:

`console.log("Inicio");`
`console.log("Fin");`
	`console.log("Inicio");`
	`setTimeout(() => console.log("Tarea demorada"), 2000);`
	`console.log("Fin");`

¿Qué es un callback?

Una función que se pasa como argumento a otra y se ejecuta después de que algo haya ocurrido.

```
function saludar(nombre, callback) {
  console.log("Hola " + nombre);
  callback();
}
```

```
saludar("Ana", () => {
  console.log("Este es un callback");
});
```

El problema del "Callback Hell"

```
javascript
CopiarEditar
setTimeout(() => {
  console.log("Paso 1");
  setTimeout(() => {
    console.log("Paso 2");
    setTimeout(() => {
      console.log("Paso 3");
    }, 1000);
  }, 1000);
}, 1000);
```

Esto vuelve el código difícil de leer y mantener.

💡 ¿Qué es una Promesa?

Una promesa representa un **valor que puede estar disponible ahora, en el futuro o nunca**. Su uso mejora la legibilidad del código asíncrono.

```
let promesa = new Promise((resolve, reject) => {
  // operación asíncrona simulada
  setTimeout(() => resolve("¡Éxito!"), 2000);
});

promesa.then(mensaje => console.log(mensaje));
```

Estados de una promesa:

- `pending` (pendiente)
- `fulfilled` (resuelta)
- `rejected` (rechazada)

Métodos:

- `.then()` para manejar éxito
- `.catch()` para manejar errores

PARTE DEMOSTRATIVA

1 Callback con setTimeout

```
function esperar(mensaje, callback) {
  setTimeout(() => {
    console.log(mensaje);
    callback();
  }, 2000);
}

esperar("Esperando...", () => {
  console.log("Listo");
});
```

2 Misma lógica con Promesas

```
function esperarPromesa(mensaje) {
  return new Promise(resolve => {
    setTimeout(() => resolve(mensaje), 2000);
  });
}
```

```
esperarPromesa("Esperando...").then(msg => {
  console.log(msg);
  console.log("Listo");
});
```

3 Simulación de una API con `fetch()`

```
fetch("https://jsonplaceholder.typicode.com/users/1")
  .then(response => response.json())
  .then(data => {
    console.log("Usuario:", data.name);
  })
  .catch(error => console.error("Error:", error));
```

✍ ACTIVIDAD PRÁCTICA INDIVIDUAL (20 minutos)

🎯 Instrucciones:

Implementa los siguientes 3 ejercicios en Visual Studio Code.

- **Callback simple:**
  Crea una función que reciba un mensaje y lo muestre después de 2 segundos usando `setTimeout()` y un **callback**.
- **Promesa equivalente:**
  Convierte esa función en una **promesa** que se resuelva después del mismo tiempo.
- **Petición a una API pública:**
  Usa `fetch()` para obtener un usuario de la API `https://jsonplaceholder.typicode.com/users/1` y muestra el nombre en consola.

Reflexión final:

¿Cuáles son las ventajas de usar promesas en comparación con callbacks?

Ideas clave:

- Código más legible
- Facilita manejo de errores
- Mejora el mantenimiento de lógica asíncrona

Revisión de ejercicios:

- ¿Funcionan los temporizadores?
- ¿La promesa resuelve correctamente?
- ¿La petición a la API se ejecuta bien y devuelve datos?

Tarea:

Crear una función que simule una solicitud a una API con `setTimeout()` y devuelva un valor usando una promesa. Sugerencia: Muestra en el navegador un mensaje de “Cargando…” mientras espera la promesa, y luego reemplázalo por el resultado.

INSTITUTO

KHIPU