Programación Asíncrona: Promesas y Callbacks

Julian F. Latorre

17 de octubre de 2024

Agenda

- Introducción a la Programación Asíncrona
- Callbacks
- Promesas
- async/await
- Comparación de enfoques

¿Qué es la Programación Asíncrona?

- Paradigma que permite que múltiples operaciones ocurran simultáneamente
- Esencial en el desarrollo web moderno
- Mejora la experiencia del usuario y el rendimiento de la aplicación

Callbacks

- Función pasada como argumento a otra función
- Se ejecuta después de que la función principal termina

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

function despedirse() {
    console.log('Adiós!');
}

saludar('Juan', despedirse);
```

Callback Hell

```
operacion1(function(resultado1) {
    operacion2(resultado1, function(resultado2) {
        operacion3(resultado2, function(resultado3) {
            operacion4(resultado3, function(resultado4
                ) {
                // ... y así sucesivamente
            });
        });
    });
});
```

Promesas

- Objeto que representa la eventual finalización de una operación asíncrona
- Estados: Pendiente, Cumplida, Rechazada
- Métodos principales: .then() y .catch()

Creación y Uso de Promesas

```
const miPromesa = new Promise((resolve, reject) => {
    // Código asíncrono
    if (exito) {
        resolve("Operación exitosa");
    } else {
        reject("La operación falló");
    }
});

miPromesa
    .then((mensaje) => {
        console.log(mensaje);
    })
    .catch((error) => {
        console.error(error);
    });
```

async/await

- Sintaxis más reciente para trabajar con Promesas
- Hace que el código asíncrono se parezca al síncrono

```
async function obtenerDatos() {
    try {
        const response = await fetch('https://api.
            ejemplo.com/datos');
        const data = await response.json();
        console.log(data);
    } catch (error) {
        console.error('Hubo un error:', error);
    }
}
```

Comparación: Callbacks vs Promesas vs async/await

- Callbacks: Primera solución, pero puede llevar a código complicado
- Promesas: Mejor manejo de errores y encadenamiento de operaciones
- async/await: Sintaxis más limpia y fácil de leer

Conclusión

- La programación asíncrona es fundamental en el desarrollo web moderno
- Cada enfoque (Callbacks, Promesas, async/await) tiene su lugar
- Es importante entender cuándo y cómo usar cada uno
- Práctica y experiencia son claves para dominar estos conceptos