Asincronía en JavaScript

JavaScript es un lenguaje de programación basado en eventos y asíncrono, lo que significa que puede realizar tareas sin bloquear la ejecución del código. Aquí hay una explicación básica con ejemplos:

1. Callbacks:

```
/ Ejemplo de función con un callback
function hacerAlgo(callback) {
  setTimeout(function() {
    console.log("Haciendo algo...");
    callback();
  }, 1000);
}

// Uso de la función con un callback
hacerAlgo(function() {
  console.log("Terminado");
});
```

2. Promesas:

```
// Ejemplo de promesa
let miPromesa = new Promise(function(resolve, reject) {
  setTimeout(function() {
```

```
    resolve("Éxito");
    // O puedes usar reject("Error") para manejar errores
  }, 1000);
});

// Uso de la promesa
miPromesa.then(function(resultado) {
  console.log("La promesa se resolvió con: " + resultado);
}).catch(function(error) {
  console.log("La promesa se rechazó con: " + error);
});
```

3. Async/Await:

```
// Función asíncrona
async function ejecutarTareas() {
  try {
    let resultado1 = await tarea1();
    let resultado2 = await tarea2();
    console.log("Todas las tareas completadas");
  } catch (error) {
    console.error("Error: " + error);
  }
}

// Uso de la función asíncrona
```

```
ejecutarTareas();
```

4. Eventos asíncronos:

```
// Ejemplo de evento asíncrono
document.getElementById("miBoton").addEventListener("click", function() {
  setTimeout(function() {
    console.log("Botón clickeado después de 1 segundo");
  }, 1000);
});

// Esto se ejecutará antes de que se complete la tarea del botón
console.log("Continuando con otras tareas...");
```

Estos son solo algunos ejemplos básicos de cómo trabajar con asincronía en JavaScript. La asincronía es esencial para manejar operaciones que pueden llevar tiempo, como solicitudes a servidores, lectura de archivos o interacciones de usuario.