# Gestión de Errores en JavaScript

La gestión de errores en JavaScript se realiza mediante el uso de bloques try, catch, finally y lanzando objetos Error. Esto permite manejar situaciones inesperadas y realizar acciones específicas en caso de que se produzca un error.

**1. Bloques Try-Catch:**

```
try {
  // Código que podría generar un error
  throw new Error('Este es un error personalizado');
} catch (error) {
  // Captura el error y maneja la situación
  console.error(`Error: ${error.message}`);
} finally {
  // Se ejecuta siempre, independientemente de si hay un error o no
  console.log('Este bloque siempre se ejecuta');
}
```

**2. Lanzar Errores Personalizados:**

```
function dividir(a, b) {
  if (b === 0) {
    throw new Error('No se puede dividir por cero');
  }
```

```
  return a / b;
}

try {
  const resultado = dividir(10, 0);
  console.log(resultado);
} catch (error) {
  console.error(`Error: ${error.message}`);
}
```

**3. Jerarquía de Objetos Error:**

JavaScript tiene una jerarquía de objetos Error. Puedes crear errores personalizados que hereden de Error.

```
class MiError extends Error {
  constructor(message) {
    super(message);
    this.name = 'MiError';
  }
}

try {
  throw new MiError('Este es un error personalizado');
} catch (error) {
  if (error instanceof MiError) {
    console.error(`Error personalizado: ${error.message}`);
```

```
  } else {
    console.error(`Error: ${error.message}`);
  }
}
```

**4. Asincronía y Promesas:**

Manejar errores en operaciones asíncronas se hace a menudo a través del método catch en promesas.

```
function operacionAsincrona() {
  return new Promise((resolve, reject) => {
    // Simulación de operación asíncrona
    setTimeout(() => {
      const exito = true; // Cambiar a false para simular un error
      if (exito) {
        resolve('Operación exitosa');
      } else {
        reject(new Error('Error en la operación'));
      }
    }, 1000);
  });
}

operacionAsincrona()
  .then((resultado) => {
    console.log(resultado);
```

```
  })
  .catch((error) => {
    console.error(`Error asincrónico: ${error.message}`);
  });
```

La gestión de errores es esencial para construir aplicaciones robustas, y estas técnicas te permiten anticipar y manejar problemas que puedan surgir durante la ejecución del código.