# Módulo Javascript

# Manejo de Errores

**Objetivos:** Conocer los tipos de errores en javascript y cómo manejarlos.

## Introducción

Manejar errores es una parte importante de cualquier programa en JavaScript. Los errores pueden ocurrir en cualquier parte del código y pueden tener consecuencias graves si no se manejan adecuadamente.

## Tipo de errores

Antes de profundizar en cómo manejar errores en JavaScript, es importante comprender los diferentes tipos de errores que pueden ocurrir. Hay dos tipos principales de errores en JavaScript: errores de sintaxis y errores en el tiempo de ejecución.

Los errores de sintaxis ocurren cuando se escribe código que no sigue la sintaxis correcta de JavaScript. Estos errores generalmente son detectados por el editor de código o el navegador y se muestran en la consola con un mensaje de error descriptivo.

Por otro lado, los errores en el tiempo de ejecución ocurren durante la ejecución del código. Estos errores pueden ser causados por una amplia variedad de factores, como entradas incorrectas del usuario, problemas de red, problemas de memoria y otros errores lógicos en el código.

## Manejo de errores

El manejo de errores en JavaScript se puede hacer de varias maneras. Una de las formas más comunes es utilizando el bloque try-catch.

El bloque try-catch permite al programador intentar ejecutar un bloque de código y capturar cualquier error que ocurra dentro de ese bloque. Aquí hay un ejemplo:

```javascript
try {
  // Código a intentar
} catch (error) {
  // Código para manejar el error
}
```

En este ejemplo, cualquier error que ocurra dentro del bloque **try** será capturado por el bloque **catch** y se almacenará en la variable **error**. Dentro del bloque **catch**, el programador puede escribir código para manejar el error de la manera que sea apropiada.

Además del bloque try-catch, también se puede utilizar la instrucción **throw** para lanzar un error explícitamente. La instrucción **throw** se utiliza para lanzar una excepción y detener la ejecución del programa en caso de que ocurra un error. Aquí hay un ejemplo:

```javascript
function dividir(a, b) {
  if (b === 0) {
    throw new Error("No se puede dividir entre cero");
  }
  return a / b;
}
```

En este ejemplo, la función **dividir** lanzará un error si el segundo argumento es cero. El mensaje de error se almacenará en una instancia de **Error**, que se puede personalizar para proporcionar información adicional sobre el error.

Además del bloque try-catch, también existe el bloque finally, que se ejecuta independientemente de si se produjo una excepción o no. El bloque **finally** es útil para realizar tareas de limpieza, como cerrar una conexión de red o liberar recursos.

JavaScript

```
try {
   // Código que puede lanzar una excepción
} catch (error) {
   // Código para manejar la excepción
} finally {
   // Código que se ejecuta independientemente de si se produjo una excepción o no
}
```

## Mostrar mensajes de error al usuario

En la mayoría de los casos, los mensajes de error se mostrarán en la consola del navegador o en el registro de errores del servidor. Sin embargo, en algunos casos, puede ser necesario mostrar mensajes de error al usuario.

Una forma común de hacer esto es utilizando la instrucción alert. La instrucción alert muestra un mensaje emergente en el navegador con el mensaje de error. Aquí hay un ejemplo:

JavaScript

```
try {
  // Código a intentar
} catch (error) {
  alert("Se produjo un error: " + error.message);
}
```

En este ejemplo, se muestra un mensaje emergente al usuario que contiene el mensaje de error.

Otra forma de mostrar mensajes de error al usuario es utilizando una biblioteca de notificaciones, como Toastr o SweetAlert. Estas bibliotecas proporcionan una interfaz más atractiva y fácil de