**Blinda tu Código JavaScript con try…catch**

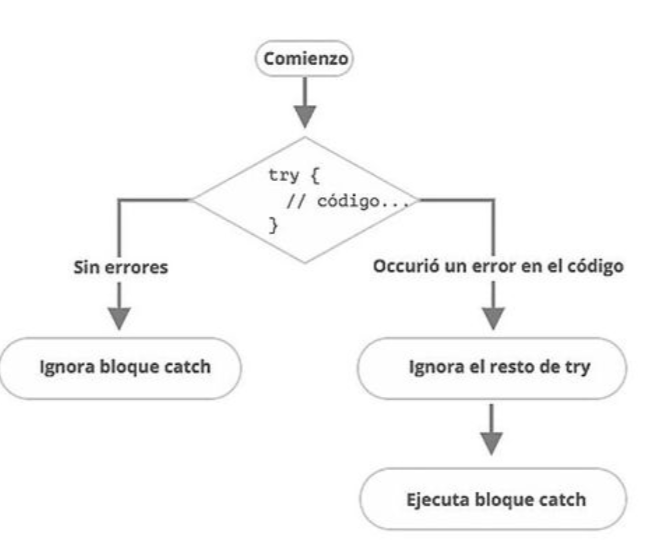
Generalmente, un programa JavaScript se detiene (die) inmediatamente si encuentra un error, dejando un mensaje en la consola.

El problema es que solo otros desarrolladores/as conocen la existencia de la consola, por lo que es posible que el usuario/a final de tu Web se quede mirando una página en blanco o un eterno mensaje de "cargando...".

Usando **try (intentar)...catch(capturar**) puedes capturar esos errores sin detener la ejecución, para procesarlos como más te convenga.

**Como funciona try..catch**

Primero, **se ejecuta el código dentro de try**. Si en su interior se presenta algún error, **se omite el resto del código try y se salta al bloque catch.**



**Veamos un ejemplo:**

const funcion= () =>alerT("Hola Mundo")

try{

    funcion()

}catch(error){

    console.error("Error capturado", error)

}

console.log("Continuamos con la ejecucion")

//Error capturado ReferenceError:

// alerT is not defined

//Continuamos con la ejecucion 🡪 llega hasta acá gracias a try..catch

**Procesando errores**Si se da un error y entramos en catch, **tenemos acceso a un objeto** que contiene información del **error**: una **propiedad name** **con el tipo de error** y otra **message** **con el error en sí**, entre muchas más.

const nombre = "Juan"

try{

    nombre()

}catch(error){

    console.error(

        `Tipo: ${error.name},

        Error: ${error.message}`

    )

}

//Tipo: TypeError,

//Error: nombre is not a function

**Errores personalizados**

Pero, que ocurre si no hay ningún error sintáctico o programático, ¿pero aun así queremos detectar cuando no se cumple alguna de nuestras condiciones?

¿La solución?, crear nuestro propio tipo de errores (se los conoce como **excepciones**) con el **operador** **throw**(lanzar).

Throw **genera un error**, para ello acepta **un objeto**(con name y message) o las clases de error nativo javascript(como TypeError, ReferenceError, SyntaError… además del genérico Error).

 throw new SyntaxError("Mensaje del error");

Al lanzar nuestro error personalizado, saltamos directamente al bloque catch, **impidiendo continuar con el resto del código. (guardarRegistro(frameworks)).**

const frameworks = ["React", "Angular", "Nuxt"]

try{

    if(!frameworks.includes("Vue"))

    throw new Error("Debes usar Vue")

    guardarRegistro(frameworks)

}catch(error){

    console.error(error.message)

}

//Debes usar Vue

**Bloque finally**

Try..catch tiene una cláusula final (**opcional**) llamada *finally,* que se **ejecuta siempre**  independientemente de si se dan errores o no en todo el bloque.

Se usa generalmente para **ejecutar alguna tarea** (liberar recurso, cerrar conexión, etc.) **cuando el proceso ha terminado.**

const conexion // API, webservice, etc

try{

    const conexion = conexion.abrir()

    const data = obtenerDatos(conexion)

}catch(error){

    console.error(error)

} finally{

    conexion.cerrar()

}