

Gestión de errores en JavaScript

Fundamentals

Tipos de Error

Syntax Error (Errores de Sintaxis)

Es un tipo comun de error que suele ocurrir cuando hay errores en la sintaxis del codigo escrito.

Runtime Error (Error en tiempo de ejecucion)

Comunmente conocido como excepcion, son errores que se presentan durante la ejecucion de un programa y detienen el flujo del mismo.

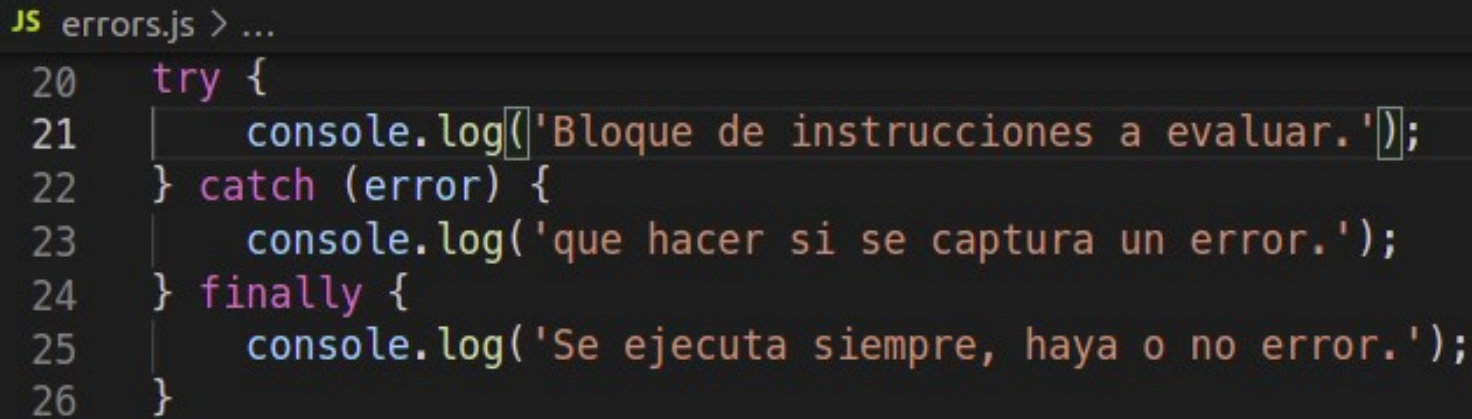
Logical Error (Errores Logicos)

Son errores que se presentan en la logica de un programa y son los mas dificiles de detectar, ya que estos permiten la ejecucion normal del programa, pero arrojan resultados diferentes a los esperados.

try – catch – finally

La instrucción try – catch – finally, nos permite hacer una correcta gestion de los errores en JavaScript, su sintaxis es la siguiente:

```
JS errors.js > ...
20  try {
21      console.log('Bloque de instrucciones a evaluar.');
```



```
22  } catch (error) {
23      console.log('que hacer si se captura un error.');
```

```
24  } finally {
25      console.log('Se ejecuta siempre, haya o no error.');
```

```
26  }
```

try – catch – finally

try – Evalua un conjunto de instrucciones, y lanza un error en caso de encontrarse.

catch – captura el error lanzado por la instrucción try, y ejecuta una serie de acciones definidas por el usuario.

finally – este bloque se ejecuta siempre, no importa si se lanza o no un error.

Podemos encontrar tres variantes de estas instrucciones:

1. try – catch
2. try – finally
3. try – catch – finally

try – catch – finally

```
JS errors.js > ...
1  'use strict'
2
3  function sumaEnteros(a, b) {
4      return a + b;
5  }
6
7  try {
8      console.log(add(5, 3));
9  } catch (error) {
10     console.error(error);
11 }
12 finally {
13     console.log('Programa terminado.');
```

Podemos imprimir el tipo de error y su mensaje asociado utilizando las propiedades **name** y **message**, del objeto **error**.

error.name: nombre del error
error.message: descripcion

Reto 1

Defina el tipo de error que representa el siguiente fragmento de código.

```
JS errors.js >  productoEntero
```

```
30 function productoEntero(multiploA, multiploB) {  
31     return multiploA + multiploB;  
32 }  
33  
34  
35 console.log(`El producto de ${5} x ${3} es: ${productoEntero(5, 3)}`);
```

Reto 2

Defina el tipo de error que representa el siguiente fragmento de código.

```
JS errors.js >  productoEntero
```

```
30 function productoEntero(multiploA, multiploB) {  
31     return multiploA + multiploB;  
32  
33  
34
```

```
35 console.log(`El producto de ${5} x ${3} es: ${productoEntero(5, 3)}`);
```

Reto 3

Desarrolle el siguiente programa utilizando lo aprendido hasta ahora.

```
JS errors.js > ...
```

```
20  /*  
21  * Desarrolle la funcion esPositivo, que reciba un numero entero leído por teclado  
22  * Si el numero es mayor que cero imprima en consola "YES"  
23  * Si el numero es menor que cero imprima en consola "Negative Error"  
24  * Si el numero es cero imprima en consola "Zero Error"  
25  * Utilize el metodo error del objeto console.  
26  */
```


throw

throw – Esta instrucción nos permite lanzar un error, su sintaxis es la siguiente:

throw new Error(mensaje);

Errores del navegador.

Tipo de error	Significado	Ejemplo
EvalError	Representa un error relacionado con la ejecución de la función eval()	throw new EvalError();
RangeError	Representa que un valor está fuera del rango permitido para una variable o parámetro.	throw new RangeError();
ReferenceError	Representa que se ha intentado invocar una función u objeto y no existe en ese ámbito	throw new ReferenceError();
SyntaxError	Representa un error en la sintaxis del código que se pretende ejecutar con eval()	throw new SyntaxError();
TypeError	Representa un error debido a que una variable o parámetro no tienen un tipo válido.	throw new TypeError();
URIError	Representa un error cuando se pasan parámetros no válidos a las funciones encodeURIComponent() ó decodeURI()	Throw new URIError();

Reto 4

Modifique el programa del reto anterior, para crear errores utilizando la instrucción **throw**.

```
JS errors.js > ...
```

```
20  /*
21  * Desarrolle la funcion esPositivo, que reciba un numero entero leído por teclado
22  * Si el numero es mayor que cero imprima en consola "YES"
23  * Si el numero es menor que cero imprima en consola "Negative Error"
24  * Si el numero es cero imprima en consola "Zero Error"
25  * Utilize el metodo error del objeto console.
26  */
```

Reto 5 – Super–Reto

Usted ha sido seleccionado para desarrollar una aplicación de gestión de un pequeño teatro. Nuestro cliente requiere gestionar las sillas de un pequeño teatro que cuenta con 5 filas, cada una con 5 sillas.

Se requiere crear la siguiente funcionalidad:

Cada silla representa un objeto que tiene propiedades, nombre y estado.

Los estados de la silla son: Libre (L), Reservado (R), Comprado (C).

Cree la funcionalidad necesaria para, reservar y/o comprar la silla en el teatro.

El programa desarrollado debe contar con un menú, para las funcionalidades mencionadas.

Si el usuario selecciona una silla que tiene estado reservado y/o comprado, debe mostrar un error en consola y permitirle seleccionar otra silla.

Por cada cambio se debe mostrar el estado de las sillas en la sala.

El usuario debe poder contar con la opción de salir del programa.