Técnicas de Programación

Carrera Programador full-stack

TRY - CATCH

Gestión de errores

No importa que tan buenos seamos programando a menudo nos encontraremos con errores durante la ejecución de un código, pueden ocurrir debido a descuidos, una entrada inesperada del usuario, una respuesta errónea del servidor o por otras razones, Sin importar la naturaleza del error esto ocasiona que nuestro programa se detenga inmediatamente evitando que el usuario pueda seguir con su tarea.

Gestión de errores

Para estos casos hay una construcción sintáctica que nos permite “atrapar” errores para antes de que nuestro programa termine su ejecución debido a un error.

Esta es las palabras reservadas

TRY y CATCH….

Gestión de errores

La construcción `try...catch` tiene dos bloques principales: `try`, y luego `catch`:

```
1  try {
2
3    // código...
4
5  } catch (err) {
6
7    // manipulación de error
8
9  }
```

Gestión de errores

Como Funciona:

1. Primero, se ejecuta el código en `try {...}`.

```
try {
        //Código que vamos a ejecutar
        // Si se produce un error se lanza una excepción y se salta al catch
}
```

Gestión de errores

Como Funciona:

2. Si no hubo errores, se ignora `catch (err)`: la ejecución llega al final de `try` y continúa, omitiendo `catch`.

Al igual que pasaría con el IF ELSE si la condición resulta ser verdadera, ignora la secuencia de código que se encuentra dentro del ELSE

Gestión de errores

Como Funciona:

3. Si se produce un error, la ejecución de `try` se detiene y el control fluye al comienzo de `catch (err)`. La variable `err` (podemos usar cualquier nombre para ella) contendrá un objeto de error con detalles sobre lo que sucedió.

```
catch (e) {
  // e representa el error lanzado
  // mensajes de alerta, acciones a ejecutar, etc.
}
```

Gestión de errores

Entonces, un error dentro del bloque `try{...}` detendrá la ejecución del programa por lo tanto y tendríamos la oportunidad de manejarlo en el bloque `catch`.

Gestión de errores

¡Probamos que imprime..!

```
try {

  alert('Inicio de intentos de prueba');  // (1) <--

  // ...no hay errores aquí

  alert('Fin de las ejecuciones de try');   // (2) <--

} catch (err) {

  alert('Se ignora catch porque no hay errores'); // (3)

}
```

Gestión de errores

¡Probamos que imprime..!

```
try {

  alert('Inicio de ejecuciones try');  // (1) <--

  lalala; // error, variable no está definida!

  alert('Fin de try (nunca alcanzado)');  // (2)

} catch (err) {

  alert(`¡Un error ha ocurrido!`); // (3) <--

}
```

Gestión de errores

¡Probamos que imprime..!

```
try {
  {{{{{{{{{{{
} catch(err) {
  alert("El motor no puede entender este código, no es válido.");
}
```

Gestión de errores

¡Probamos que imprime..!

```
try {
  {{{{{{{{{{{
} catch(err) {
  alert("El motor no puede entender este código, no es válido.");
}
```

Para que `try..catch` funcione, el código debe ser ejecutable. En otras palabras, debería ser JavaScript válido.No funcionará si el código es sintácticamente incorrecto!

Ejercicios

Carrera Programador full-stack

Ejercicio

Realizar un programa que lea un archivo de texto y lo muestre en un html. En caso de no existir el archivo el programa debe de indicar el error.

Ejercicio

Realizar cambios en el programa de modo de provocar distintos posibles errores, capturarlos y mostrarlos en el html.
Usar el estado de una promesa para mostrar su código en el html. Ej: status: 200 ok