Day 14 - Error Handling

JavaScript is loosely-typed and sometimes you'll get a runtime error when you try to access an undefined variable or called an undefined function.

JavaScript is similar to python or Java in that it provides an error-handling mechanism to catch runtime errors using try-catch-finally block.

```
try {
  // code that may throw an error
} catch (err) {
  // code to be executed if an arror occurs
} finally {
  // code to be executed regardless of if an error occurs or not
}
```

try

Wrap suspicious code that may throw an error in a try block. The try statement lets us define a block of code to be tested for errors while it is being executed.

catch

Write code to do something in catch block when an error occurs. Catch block can have parameters that will give you error information. Catch block is used to log an error or display specific messages to the user.

finally

Finally block will always be executed regardless of the occurrence of an error. Can be used to complete the remaining task or reset variables that might have changed before error occurred in try block.

```
try{
    let one = 1;
    console.log(one, two)
} catch (err){
    console.log('two is not a defined variable')
}

try {
  let lastName = 'Guerrero'
  let fullName = fistName + ' ' + lastName
} catch (err) {
  console.error(err) // we can use console.log() or console.error()
} finally {
  console.log('In any case I will be executed')
}
```

The catch block takes a parameter, that parameter being e, err, or error. The parameter is an object and has name and message keys.

```
try {
  let lastName = 'Guerrero'
  let fullName = fistName + ' ' + lastName
} catch (err) {
  console.log('Name of the error', err.name)
  console.log('Error message', err.message)
} finally {
  console.log('In any case I will be executed')
}

// log:
Name of the error ReferenceError
Error message fistName is not defined
In any case I will be executed
```

throw

The throw statement lets us create a custom error. We can throw a string, number, boolean or an object. We use the throw statement to throw an exception. When you throw an exception, the expression specifies that value of the exception.

```
throw 'Error2' // generates an exception with a string value
throw 42 // generates exception with value 42
throw true // generates exception with value true
throw new Error('Required') // Generates error object with the message of Required
```

```
const throwErrorExampleFun = () => {
  let message
  let x = prompt('Enter a number: ')
  try {
    if (x == '') throw 'empty'
    if (isNaN(x)) throw 'not a number'
    x = Number(x)
    if (x < 5) throw 'too low'
    if (x > 10) throw 'too high'
  } catch (err) {
    console.log(err)
  }
}
throwErrorExampleFun()
```

Error Types

ReferenceError

An illegal reference has occurred. ReferenceError is thrown if we use a variable that has not been declared.

```
let firstName = 'Chris'
let fullName = firstName + ' ' + lastName
console.log(fullName)

//log:
Uncaught ReferenceError: lastName is not defined
    at <anonymous>:2:35
```

SyntaxError

A syntax error has occurred

```
 let square = 2 x 2
console.log(square)
console.log('Hello, world")

//log:
Uncaught SyntaxError: Unexpected identifier
```

TypeError

A type error has occurred.

```
let num = 10
console.log(num.toLowerCase())

//log:
```

```
Uncaught TypeError: num.toLowerCase is not a function
    at <anonymous>:2:17
```