JS Error Handling

Lesson Time: 60 Minutes

Errors happen. We must handle them to prevent problems.

Errors will happen. Error Handling involves adding additional code to your scripts to look for errors and resolve them to prevent unintended and unexpected behavior.

JS performs error handling similar to most other languages by using a combination of keywords **throw**,**try**, **catch**, and **finally**

throw

throw allows us to generate an error. This is called “throwing an error”. Why would we want to do this? The most common reasons are:

1. To detect when a problem is about to happen and prevent it
2. To give the user a friendly message to let them know they’ve made an error & need to try again.
3. To let the user something unexpected happened and give them information to troubleshoot with.

For example, say a user is making a payment online and they enter -50.00 as the payment amount. If we don’t plan for this, the system might try to process negative payment, the software might even credit the user the money if it’s not programmed well!

We can error check for this to prevent the error from occurring.

```
1   var amount = -50.00
2
3   if (amount <= 0.00){
4
5       throw "amount can not be a negative number"
6   }
```

```
    ^
amount can not be a negative number
Waiting for the debugger to disconnect...
Error: "amount can not be a negative number"
```

Here we are performing error checking and creating our own error to prevent the user from doing something we don't want them to do. When this error happens, the JS code will stop running and no additional code with execute.

We don't want our JS script to stop completely because of the error, so our next step is to **handle** the error. We will handle it with a **try..catch** statement.

Try...Catch

```
function amountErrorCheck(amount){
    if (amount <= 0.00){
        throw "amount can not be a negative number"
    }
}

var amount = -50.00

try{
    amountErrorCheck(amount)
}
catch{
    amount = 0.00
    console.log("Handled error -- amount can not be a nagative number")
}
```

```
ROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
C:\Program Files\nodejs\node.exe --inspect-brk=44403 errors.js
Debugger listening on ws://127.0.0.1:44403/cf641754-d70c-4e53-85ef-fb6f60102b61
For help, see: https://nodejs.org/en/docs/inspector
Handled error -- amount can not be a nagative number
```

Here we've turned our amount check into a function we can reuse called amountErrorCheck().

Now we'll test to see if an error will occur by executing the function in a **try** block. The error will occur, so now the **catch** block will execute. We set the amount to zero and print a message that the error has been handled. Our script does not stop and everything continues to function.

The **finally** block contains statements to execute after the try and catch blocks execute but before the rest of our script resumes. The finally block executes whether or not an exception is thrown. If an exception is thrown, the statements in the finally block execute even if no catch block handles the exception.

In summary

- try — code to try that may produce errors
- catch — code that will handle any errors
- finally — code that will run after a try/catch block regardless of the outcome
- throw — a keyword you can use to throw your own custom errors

Key Terms	Try, Catch, Finally, Throw
Lesson Files	
Additional Resources	
Further Learning	