Node.js Try Catch

Node.js Try Catch

Node.js Try Catch is an Error Handling mechanism. When a piece of code is expected to throw an error and is surrounded with try, any exceptions thrown in the piece of code could be addressed in catch block. If the error is not handled in any way, the program terminates abruptly, which is not a good thing to happen.

Note : It is a good practice to use **Node.js Try Catch** only for synchronous operations. We shall also learn in this tutorial as to why **Try Catch** should not be used for asynchronous operations.

Example of Node.js Try Catch

In this example, we shall use a **Try Catch** around the piece of code that tries to read a file synchronously.

nodejs-try-catch-example.js

```
# example for Node.js
Try Catch
```

```
# example for Node.js Try Catch
var fs = require('fs');

try{
    // file not presenet
    var data = fs.readFileSync('sample.html');
} catch (err){
    console.log(err);
}

console.log("Continuing with other statements..");
```

When the above program is run..

Terminal Output

```
arjun@arjun-
VBCEH26EN:~/nodejs$
```

```
arjun@arjun-VPCEH26EN:~/nodejs$ node nodejs-try-catch-example.js
{ Error: ENOENT: no such file or directory, open 'sample.html'
    at Object.fs.openSync (fs.js:652:18)
    at Object.fs.readFileSync (fs.js:553:33)
    at Object.<anonymous> (/nodejs/nodejs-try-catch-example.js:5:16)
    at Module._compile (module.js:573:30)
    at Object.Module._extensions..js (module.js:584:10)
    at Module.load (module.js:507:32)
    at tryModuleLoad (module.js:470:12)
    at Function.Module._load (module.js:462:3)
    at Function.Module.runMain (module.js:609:10)
    at startup (bootstrap_node.js:158:16)
  errno: -2,
  code: 'ENOENT',
  syscall: 'open',
  path: 'sample.html' }
Continuing with other statements..
```

Observe that the program did not terminate abruptly, but continued with the execution of subsequent statements.

Now we shall see what happens if we do not use try catch for the same operation above.

nodejs-try-catch-example-1.js

```
# error without Node.js
Try Catch
```

```
# error without Node.js Try Catch
var fs = require('fs');

// try to read a file synchronously, file not presenet
var data = fs.readFileSync('sample.html');

console.log("Continuing with other statements..");
```

There is no error handling mechanism incorporated in the code. And the program terminates abruptly and the subsequent statements are not executed.

When the above program is run..

Terminal Output

```
arjun@arjun-
VPCEH26EN:~/nodejs$
```

```
arjun@arjun-VPCEH26EN:~/nodejs$ node nodejs-try-catch-example-1.js
/home/arjun/nodejs/nodejs-try-catch-example-1.js:1
(function (exports, require, module, __filename, __dirname) { # example for Node.js Try Catch
                                                              ^

SyntaxError: Invalid or unexpected token
    at createScript (vm.js:74:10)
    at Object.runInThisContext (vm.js:116:10)
    at Module._compile (module.js:537:28)
    at Object.Module._extensions..js (module.js:584:10)
    at Module.load (module.js:507:32)
    at tryModuleLoad (module.js:470:12)
    at Function.Module._load (module.js:462:3)
    at Function.Module.runMain (module.js:609:10)
    at startup (bootstrap_node.js:158:16)
    at bootstrap_node.js:598:3
```

Why Node.js Try Catch should not be used for catching errors in Asynchronous Operations

Consider the following example where we try to read a file asynchronously with a callback function and throw an error if something goes wrong. And we surround the task with a Try Catch Block, hoping to catch the error thrown.

nodejs-try-catch-example-2.js

```
# Node.js Try Catch
with Asynchronous
```

```
# Node.js Try Catch with Asynchronous Callback Function
var fs = require('fs');

try{
    fs.readFile('sample.txta',
        // callback function
        function(err, data) {
            if (err) throw err;
    });
} catch(err){
    console.log("In Catch Block")
    console.log(err);
}
console.log("Next Statements")
```

Terminal Output

```
arjun@arjun-
```

```
arjun@arjun-VPCEH26EN:~/nodejs/try-catch$ node nodejs-try-catch-example-2.js
Next Statements
/home/arjun/nodejs/try-catch/nodejs-try-catch-example-2.js:8
        if (err) throw err;
                 ^

Error: ENOENT: no such file or directory, open 'sample.txta'
```

If you observe the output,console.log("Next Statements") has been executed prior to if(err) throw err which is because, reading file is being done asynchronously and the control does not wait the file operation to complete, instead it proceeds with the next statement. Which means, the control is out of try catch block. If an error occurs during the asynchronous operation, there is no try catch block the control could know of. Hence our Try Catch block cannot catch errors that could occur during asynchronous operations and developers should avoid catching errors thrown by asynchronous tasks with **Node.js Try Catch** blocks.

What happens to exceptions in asynchronous code ?

If you are puzzled as to what happens to exceptions in asynchronous code, here is the answer. If you have gone through Callback Function, and observed an error object as an argument, this is where any errors or exceptions are reported back to Node.js.

Conclusion :

In this Node.js Tutorial – **Node.js Try Catch**, we have learnt the scenarios where to use Try Catch, and where not to use it.