Error Handling & Try Catch in Javascript | JavaScript Tutorial In Hindi #44

Error Handling & Try Catch in Javascript | JavaScript Tutorial In Hindi #44

In this tutorial, we will learn about the try, catch, throw, and, finally statements that handle JavaScript's exceptions. No matter how great we are at programming, sometimes our code has errors. They may occur because of unexpected user input and for a thousand other reasons. Usually, a code immediately stops executing in case of an error, printing it to console. But we can solve this problem by using the try..catch statement. It allows us to "catch" errors so the script can do something more reasonable instead of stopping. Before we learn about error handling statements, let us see about the types of errors in programming.

Types of Errors:-

There can two types of errors:

- **Syntax Error**: This is the error in the syntax. For example, if we write console.lo('JS');, the above program throws a syntax error. The spelling of the log is a mistake in the above code.
- **Runtime Error**: The runtime error occurs during the execution of the program. For example, calling an invalid function or a variable.

Now,let's see how we can handle these exceptions.

What is Try Catch in JavaScript?

Just like other programming languages, JavaScript also has exception handling capabilities. JavaScript implements the try-catch statements as well as the throw operator to handle exceptions. Here is the basic syntax for try...catch:

```
try{//some code that has an error
}
catch (e) {
//this will run if the code in the try block errors}
```

With these statements, in JavaScript, we can also add a throw or a finally clause. Let us see what role they play.

- **throw:** This is a block of code nested within the try statement and allows the programmer to write their own error that they want to handle
- **finally:** This block of code runs once all the other statements have run

try/throw/catch:-

The throw operator generates an error. We can define and throw their own custom errors. When the throw statement is executed, the statements present after it does not execute. The control will directly pass to the catch block. In the following example, we create our own error ("This is a new error") in the throw block. Then try the code which throws an error which should be caught by the catch block.

```
try{
  throw new Error ('This is a new error')
}
catch(error){
  console.log(error)
  console.log("End of try-catch block")
}
```

try/catch/throw/finally:-

Finally is an optional block of statements that is executed after the execution of try and catch statements. It does not matter that any exception is thrown or not, finally block code will definitely execute if it is present. In this example, we will see how to use the finally statement with the other three statements. In this example, we do not show the entire error. We just logged that the error has been handled in the catch block.

```
try{
  console.log("This statement works")
  throw new Error('This statement throws an error')
}
catch(error){
  console.log("Error has been handled")
}
finally{
  console.log("Everything has been handled")
}
```

Code file tut44.js as described in the video

```
console.log("This is tutorial 44");
```

```
// Pretend this is coming from a server as response
let a = "Harry bhai";
a = undefined;
if (a !=undefined){
    throw new Error('This is not undefined');
}
else{
    console.log('This is undefined');
}


try {
    null.console
    console.log("We are inside try block");

    functionHarry();

} catch(error) {
    console.log(error)
    console.log("Are you okay?");
    console.log(error.name);
    console.log(error.message);

} finally {
    console.log("Finally we will run this")
}
```

Previous Next

CodeWithHarry