Error handling in JavaScript

Errors may be occurred some times in our code. They may occur because of our mistakes, an unexpected user input, an erroneous server response, and for a thousand other reasons.

When it occurs, a script "immediately stops" in case of an error, printing it to console.

By handling errors ,we can make our program to execute without stopping by that error.

We can use **try** and **catch** to handle errors.

```
try {

  // code...

} catch (err) {

  // error handling

}
```

It works like this:

1. First, the code in `try {...}` is executed.
2. If there were no errors, then `catch(err)` is ignored: the execution reaches the end of `try` and goes on, skipping `catch`.
3. If an error occurs, then the `try` execution is stopped, and control flows to the beginning of `catch(err)`. The `err` variable (we can use any name for it) will contain an error object with details about what happened.

So, an error inside the try {...} block does not kill the script – we have a chance to handle it in catch.

Ex:

```
let x=20;

try{
console.log(a)//error
}
catch(err){
    console.error("error name is  ",err.name)
    console.error("error message is ",err.message)
}

console.log(x)
console.log("good bye...")
```

Error object have key as "name" and value as "message".

Important notes about try..catch:

1. **try..catch only works for runtime errors**
2. **try..catch works synchronously**

finally block

"finally" block can be addaed after try or catch block which will be executed whether error is occurred or not. It can be used to perform "must doing" operation before program is exited like closing files, closing DB connection etc....

```
try {
  ... try to execute the code ...
} catch(e) {
  ... handle errors ...
} finally {
  ... execute always ...
}
```

Throwing an error

When an error occurs, JavaScript will normally stop and generate an error message.

The technical term for this is: JavaScript will **throw an exception (throw an error)**.

The `throw` statement allows you to create a custom error.

Technically you can **throw an exception (throw an error)**.

If you use `throw` together with `try` and `catch`, you can control program flow and generate custom error messages

Ex:

```
try {
  throw new Error(“error-message”);
} catch(e) {
  … handle errors …
} finally {
  … execute always …
}
```