

## [5] ERROR HANDLING

- Errors, also known as exceptions, occur when something unexpected happens during the execution of your code.
- JavaScript provides mechanisms for detecting, handling, and responding to errors, ensuring that your code can gracefully recover from these situations without crashing.

### Try-Catch Blocks

The most common way to handle errors in JavaScript is by using try and catch blocks. Wrap the potentially error-prone code within the try block, and if an error occurs, it's caught and handled in the catch block.

```
try {  
  // Code that might throw an error  
  const result = someFunction();  
} catch (error) {  
  // Handle the error  
  console.error("An error occurred:", error.message);  
}
```

### Throwing Errors

You can manually throw an error using the throw statement. This allows you to create custom error messages and trigger error handling.

```
function divide(a, b) {  
  if (b === 0) {  
    throw new Error("Division by zero is not allowed.");  
  }  
  return a / b;  
}
```

### Error Objects

JavaScript provides various built-in error types, such as Error, SyntaxError, TypeError, etc. You can use these error objects to provide more context about the error.

```
const data = JSON.parse(invalidJSON); // Throws a SyntaxError
} catch (error) {
  if (error instanceof SyntaxError) {
    console.error("Invalid JSON format:", error.message);
  } else {
    console.error("An error occurred:", error.message);
  }
}
```

## Finally Block

You can use a finally block in conjunction with try-catch to specify code that runs regardless of whether an error occurred or not.

```
try {
  // Code that might throw an error
} catch (error) {
  // Handle the error
} finally {
  // Code that always runs
}
```

## Promise Error Handling

When working with Promises, you can use `.catch()` to handle errors in asynchronous code.

```
someAsyncFunction()
  .then(result => {
    // Handle the result
  })
  .catch(error => {
    // Handle the error
  })
```

## Global Error Handling:

You can set up a global error handler using `window.onerror` in the browser or `process.on('uncaughtException')` in Node.js to catch unhandled errors that occur at the top level of your application.

```
window.onerror = function(message, source, lineno, colno, error) {
```

```
};
```