There are three types of errors in programming: (a) Syntax Errors, (b) Runtime Errors, and (c) Logical Errors.

Syntax Errors

Syntax errors, also called **parsing errors,** occur at compile time in traditional programming languages and at interpret time in JavaScript.

For example, the following line causes a syntax error because it is missing a closing parenthesis.

<script type = "text/javascript">

<!--

window.print(;

//-->

</script>

When a syntax error occurs in JavaScript, only the code contained within the same thread as the syntax error is affected and the rest of the code in other threads gets executed assuming nothing in them depends on the code containing the error.

Runtime Errors

Runtime errors, also called **exceptions,** occur during execution (after compilation/interpretation).

For example, the following line causes a runtime error because here the syntax is correct, but at runtime, it is trying to call a method that does not exist.

<script type = "text/javascript">

<!--

window.printme();

//-->

</script>

Exceptions also affect the thread in which they occur, allowing other JavaScript threads to continue normal execution.

Logical Errors

Logic errors can be the most difficult type of errors to track down. These errors are not the result of a syntax or runtime error. Instead, they occur when you make a mistake in the logic that drives your script and you do not get the result you expected.

You cannot catch those errors, because it depends on your business requirement what type of logic you want to put in your program.

The try statement lets you test a block of code for errors.

The catch statement lets you handle the error.

The throw statement lets you create custom errors.

The finally statement lets you execute code, after try and catch, regardless of the result.

Errors Will Happen!

When executing JavaScript code, different errors can occur.

Errors can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things.

The try statement allows you to define a block of code to be tested for errors while it is being executed.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

The JavaScript statements try and catch come in pairs:

try {  
  *Block of code to try*}  
catch(*err*) {  
  *Block of code to handle errors*}

**Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<p>Please input a number between 5 and 10:</p>  
  
<input id="demo" type="text">  
<button type="button" onclick="myFunction()">Test Input</button>  
<p id="p01"></p>  
  
<script>  
function myFunction() {  
  var message, x;  
  message = document.getElementById("p01");  
  message.innerHTML = "";  
  x = document.getElementById("demo").value;  
  try {  
    if(x == "") throw "empty";  
    if(isNaN(x)) throw "not a number";  
    x = Number(x);  
    if(x < 5) throw "too low";  
    if(x > 10) throw "too high";  
  }  
  catch(err) {  
    message.innerHTML = "Input is " + err;  
  }  
}  
</script>  
  
</body>  
</html>

**Reference Error**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Errors</h2>

<p>You cannot use the value of a non-existing variable:</p>

<p id="demo"></p>

<script>

var x;

try {

x = y + 1;

}

catch(err) {

document.getElementById("demo").innerHTML = err.name;

}

</script>

</body>

</html>

**Type Error**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Errors</h2>

<p>You cannot convert a number to upper case:</p>

<p id="demo"></p>

<script>

var num = 1;

try {

num.toUpperCase();

}

catch(err) {

document.getElementById("demo").innerHTML = err.name;

}

</script>

</body>

</html>

**Divide by zero error**

<html>

<head>

<script type = "text/javascript">

function myFunc() {

var a = 100;

var b = 0;

try {

if ( b == 0 ) {

throw( "Divide by zero error." );

} else {

var c = a / b;

}

}

catch ( e ) {

alert("Error: " + e );

}

}

</script>

</head>

<body>

<p>Click the following to see the result:</p>

<form>

<input type = "button" value = "Click Me" onclick = "myFunc();" />

</form>

</body>

</html>

## The onerror() Method

The **onerror** event handler was the first feature to facilitate error handling in JavaScript. The **error** event is fired on the window object whenever an exception occurs on the page.

<html>

<head>

<script type = "text/javascript">

<!--

window.onerror = function () {

alert("An error occurred.");

}

//-->

</script>

</head>

<body>

<p>Click the following to see the result:</p>

<form>

<input type = "button" value = "Click Me" onclick = "myFunc();" />

</form>

</body>

</html>

The **onerror** event handler provides three pieces of information to identify the exact nature of the error −

* **Error message** − The same message that the browser would display for the given error
* **URL** − The file in which the error occurred
* **Line number**− The line number in the given URL that caused the error

Here is the example to show how to extract this information.

[Live Demo](http://tpcg.io/axqshk)

<html>

<head>

<script type = "text/javascript">

<!--

window.onerror = function (msg, url, line) {

alert("Message : " + msg );

alert("url : " + url );

alert("Line number : " + line );

}

//-->

</script>

</head>

<body>

<p>Click the following to see the result:</p>

<form>

<input type = "button" value = "Click Me" onclick = "myFunc();" />

</form>

</body>

</html>

Object creation

<!DOCTYPE html>

<html>

<body>

<script>

// Create an object:

function person(x)

{

this.age=x;

}

var x=window.prompt("Enter the age","");

var p=new person(x);

window.alert("Age is " + p.age);

p.isAge=function age() {

window.alert("From the function "+this.age);

}

p.isAge();

</script>

</body>

</html>