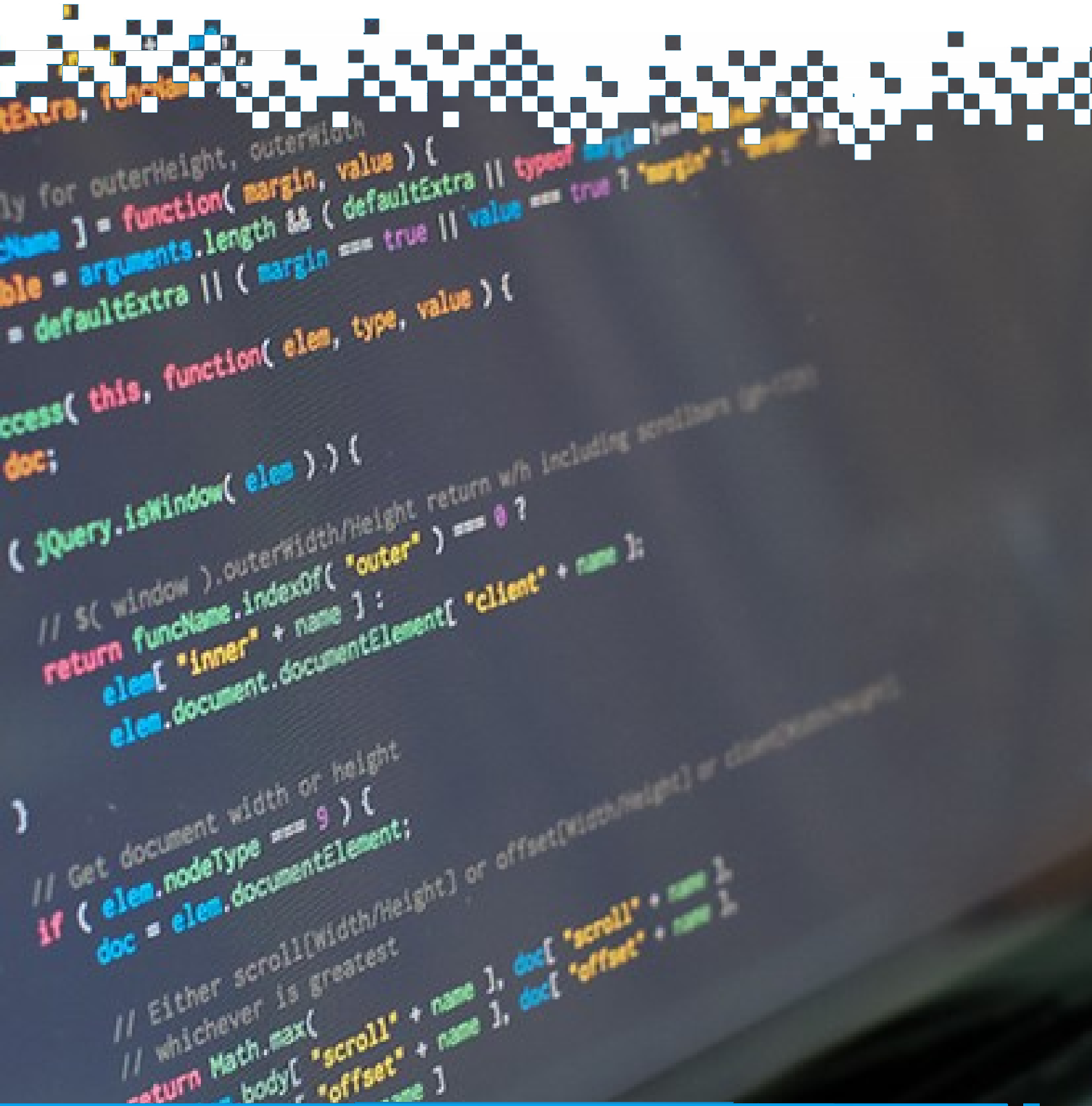
NASSCOM® FOUNDATION

Directorate General of Training

Powered by Microsoft



Session 1: Use of Error Handling Techniques in JavaScript

Session 1: Use of Error Handling Techniques in JavaScript

What is exception?

Exception

- An exception signifies the presence of an abnormal condition that requires special operable techniques.
- In programming terms, an exception is an anomalous code that breaks the normal flow of the code.
- Such exceptions require specialised programming constructs for its execution.

Exception Handling

- In programming, it is a process or method for handling abnormal statements in the code and executing them.
- It also enables handling the flow control of the code or program.
- For handling the code, various handlers are used that process the exception and execute the code.
- Division of a non-zero value with zero will result in infinity always.
- This is an exception. Thus, with the help of exception handling, it can be executed and handled.

Types of Errors

There are many types of errors in a programming language that can interfere with the proper execution of a program. Usually, there can be three types of errors in a code. They are:

Syntax Error

- Due to a mistake in the pre-defined syntax of a programming language, a syntax error may appear.
- Syntax errors are also called parsing errors that occur at compile time in traditional programming languages and interpret time in JavaScript.
- For example, the following line causes a syntax error because it is missing a closing parenthesis.
- When a syntax error occurs in JavaScript, only the code contained within the same thread as the syntax error is affected and

Runtime Error

- When an error occurs during the execution, such an error is known as a runtime error.
- The codes that create runtime errors are known as exceptions. Thus, exception handlers are used for handling runtime errors.

Logical Error

- It is an error that occurs due to any logical mistake in the program that may not produce the desired output and may terminate abnormally.
- Logical errors are difficult to track down.
- These errors are not the result of a syntax or runtime error. Instead of making mistakes logically, it drives your script but you do not get the result you expected.
- This cannot catch such errors because it depends on your

Standard Built-in Error Types

The following are standard built-in error types:

Eval Error:

It creates an instance for the error that occurred in the eval(), which is a global function used for evaluating the js string code.

Internal Error:

It creates an instance when the js engine throws an internal error.

Range Error:

It creates an instance for the error that occurs when a numeric variable or parameter is out of its valid range.

Reference Error:

It creates an instance for the error that occurs when an invalid reference is de-referenced.

Syntax Error:

An instance is created for the syntax error that may occur while parsing the eval().

Type Error:

When a variable is not a valid type, an instance is created for such an error.

URI Error:

An instance is created for the error that occurs when invalid parameters are passed

in encodeURI() or decodeURI().

Exception Handling Statements

Try and Catch Statement

- A try and catch is a commonly used statement in various programming languages.
- It is used to handle the error-prone part of a code.

Syntax:

```
try{
expression; } //code to be written
catch(error){
expression; } // code for handling th
e error
```

Program:

```
<html>
<head> Exception Handling</br></head>
<body>
<script>
try{
var x= ["11","12","13","14","15","16","17"]; //x is an array
document.write(x);  // displays elements of x
document.write(y); //y is undefined but still trying to fetch
its value. Thus catch block will be invoked
}catch(e){
alert("Error Occurs: "+e.message); //Handling error
}
</script>
</body>
</html>
```

Output

Error Occurs: y is not defined

Throw Statement

- Throw statements are used for throwing user-defined errors. Users can define and throw their custom errors. When a throw statement is executed, the statements present after it will not be executed. The control will directly pass to the catch block.
- Below program is using throw statement with its corresponding output.

Syntax:

```
throw exception;
```

```
try{
throw exception; // user can define thei
r own exception
}
catch(error){
expression; }
// code for handling exception.
```

Program:

```
<html>
<head>Exception Handling</head>
<body>
<script>
try {
  throw new Error('Throwing Exception'); //user-
defined throw statement.
}
catch (e) {

document.write(e.message); // This will generate an error m
essage
}
</script>
</body>
</html>
```

Output

```
Exception Handling
Throwing Exception
```

Try, Catch and Finally

- Finally is an optional block of statements which is executed after the execution of try and catch statements. Finally block does not hold for the exception to be thrown. Any exception is thrown or not, finally block definitely gets executed. It does not care for the output too.
- Below program is using Finally statement with its corresponding output.

Syntax:

```
try{
expression;
}
catch(error){
expression;
}
finally{
expression; } //Executable code
```

Program:

```
<html>
<head>Exception Handling</head>
<body>
<script>
try{
var x=10;
if(x==10)
document.write("Good");
}
catch(Error){
document.write("Error Occurs"+e.message); //
Catching and displaying exception
}
finally{
document.write("Value of x is 10 "); //This
statement will execute anyway
}
</script>
</body>
</html>
```

Output

```
Exception Handling
Good
Value of x is 10
```

JavaScript Onerror Event to Handle Exception

Onerror event to handle exception

Any error that is not handled by the try-catch statement causes the onerror event to fire on the Window object. The onerror event does not create an event object; it is called by the window that has errors.

It accepts three arguments. They are:

1. The error messages
2. The Uniform Resource Locator (URL) of the page on which error has occurred
3. The line number contains the error

When the onerror event triggers, it displays the error message, the URL of the web page and the line number at which the error occurs.

```
<html>
 <head>
  <title>
   Example of onerror Event
  </title>
  <script type="text/javascript">
  // using onerror event to handle error
  window.onerror = function(ermessage, url, line)
  {
   document.write(ermessage+"<br/>");
   document.write(url+"<br/>");
   document.write(line+"<br/>");
  }
  // the below line will cause error
  document.write(x);
  </script>
 </head>
 <body>
  <!-- HTML body -->
 </body>
</html>
```

Output:

Uncaught ReferenceError: x is not defined — Error Message

https://www.cnkonline.com/code/play/web/?id=SOci69 — URL

10 — Line No.

The output is displaying three things:

- Error Message,
- URL and
- Line Number,

Recap:

- In programming terms, an exception is an anomalous code that breaks the normal flow of the code
- When a user makes a mistake in the pre-defined syntax of a programming language, a syntax error may appear
- When an error occurs during the execution, it is known as a runtime error
- Syntax errors are also called parsing errors, which occur at compile time in traditional programming languages and interpret time in JavaScript
- Runtime errors, also called exceptions, occur during execution (after compilation/interpretation)
- The code which needs possible error testing is kept within the try block
- Throw statements are used for throwing user-defined errors
- Finally is an optional block of statements, which is executed after the execution of try and catch statements
- Any error that is not handled by the try-catch statement causes the onerror event to fire on the Window object