DECT-KS© TOOLS

Web Engineering Front-end Pt. 3

11. JavaScript: Troubleshooting

erb 僱員再培訓局 Employees Retraining Board
SCOPE School of Continuing and Professional Education 專業進修學院 香港城市大學 City University of Hong Kong
KRYSTAL INSTITUTE DIGITAL ECONOMY CORE TECHNOLOGY

Contents

11.1 Errors

What are errors?

In programming, errors are problems within the code that make the program run in unexpected ways, or not run at all.

Types of errors

Errors are usually divided into 3 categories, syntax errors, runtime errors, and logic errors.

Syntax errors

A syntax error happens when there is typo within the source code of the program.

Can you spot the syntax errors in the following code?

Syntax error examples

```
console.log(“Hello);

math.pow(2, 3);

var dog = {
  name: "Marley";
  age: 2;
  breed: "Labrador retriever";
}
```

Syntax error examples

```
console.log(“Hello); // Missing closing quotation mark

math.pow(2, 3); // “math” should have a capitalized M

var dog = {
   name: "Marley";
   age: 2;
   breed: "Labrador retriever";
}
// Object properties are separated with commas (,)
```

Syntax errors

Most IDEs will show error messages when there is a syntax error f

```
<body>
    <script>
        var dog = {
            name: "Marley";
            age: 2;
            breed: "Labrador retriever";
        }
    </script>
</body>
```

```
',' expected. javascript
View Problem (Alt+F8)    No quick fixes available
```

Runtime errors

A runtime error is an error that occurs when a program is executed.

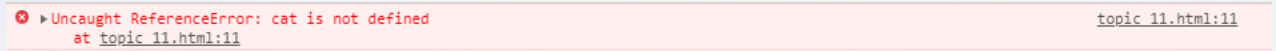
Runtime error example

```
var dog = {
   name: "Marley“,
   age: 2,
   breed: "Labrador retriever“,
}

console.log(cat.name);
```

Error messages

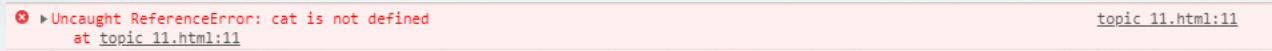
If you try to execute a program with syntax or runtime errors, an error message will appear in the console.

Error messages



Error messages allow you to locate the faults in your code quickly, as they give you a brief explanation on what the error is, as well as where it was found.

Error messages



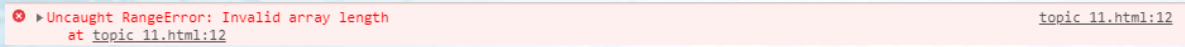
In this case, the error detected was a ReferenceError. It was caused by trying to access an object that does not exist named “cat”. And it was found in the file “topic 11.html” at line 11.

Common error message types

Type	Description
Reference Error	Represents an error when a non-existent variable is referenced
SyntaxError	Represents an error when trying to execute code with incorrect syntax
RangeError	Represents an error when a value is outside the range of accepted values
TypeError	Represents an error when an operation is attempted on an unexpected data type

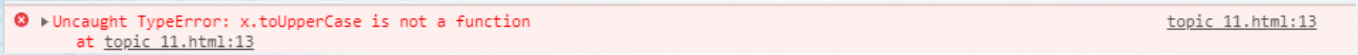
RangeError example

```
var array = [];

array.length = Number.MAX_VALUE;
```



TypeError example

```
var x = 123;

x.toUpperCase();
```



Logic errors

Unlike the other two kinds of errors, programs with logic errors will still execute and won't terminate abruptly. Rather, they don't work as the developer intended them to.

Logic error example

```
// Goal is to create a rectangle shape made of asterisks (*)
var output = “”;
for (var x = 1; x <= 5; x++) {
  for (var y = 1; y <= 5; y++) {
    output += “*” + “ “;
  }
  console.log(output);
}
```

Logic error example

Logic errors

In the previous example, the shape didn’t come out as expected because we forgot to reset “output” after each row.

Logic error example

```
// Goal is to create a rectangle shape made of asterisks (*)
var output = “”;
for (var x = 1; x <= 5; x++) {
  for (var y = 1; y <= 5; y++) {
    output += “*” + “ “;
  }
  console.log(output);
  output = “”;
}
```

Logic errors

However, even though the result is incorrect, there is no error message. Because to JavaScript, there is nothing wrong with the code in the “incorrect” example.

Logic errors

Because they don’t produce error messages, logic errors are the hardest type of error to detect. It is up to the developer to go back to the code to find the problem.

Activity: Errors you've encountered

Can you recall some of the errors that you've made in the past few lessons? What kind of errors were they?

11.2 Debugging

What is debugging?

Debugging refers to the process of removing errors, or “bugs”, from a program.

try and catch statements

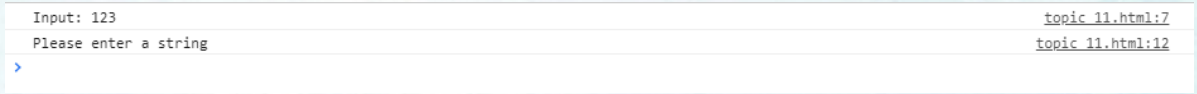
The try and catch statements are used to test a block of code for errors and to handle the caught errors, respectively.

try and catch statements syntax

```
try {
  // Code to be tested
} catch (error) {
  // Code to handle errors
}
```

try and catch statements example

```
try {
  var x = 123;
  console.log(“Input: ”+x);

  x.toUpperCase();

  console.log(x);
} catch {
  console.log(“Please enter a string”);
}
```



try and catch statements

The program will jump to the catch statement when an error is detected. The code before the point where the error happens still executes.

try and catch statements

“error” is an optional parameter that represents the error message object.

Use .name to access the error type, and .message to access the error description.

try and catch statements example

```
try {
   var x = 123;
   console.log(“Input: ”+x);

   x.toUpperCase();

   console.log(x);
} catch (e) {
   console.log(“Please enter a string”);
   console.log(“Error type: “+e.name);
   console.log(“Error description: “+e.message);
}
```

try and catch statements

Catching errors allows the program to continue running despite of the error, though subsequent code might be affected.

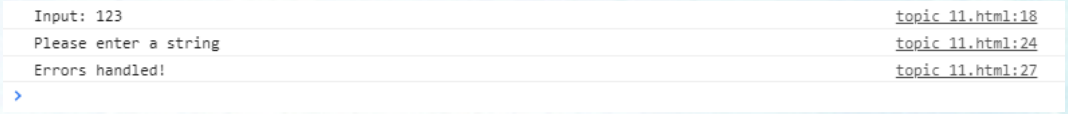
Uncaught error example

```
var x = 123;
console.log(“Input: ”+x);

x.toUpperCase();

console.log(x);
console.log(“Errors handled!”);
```

```
Input: 123                                                    topic 11.html:17
▶Uncaught TypeError: x.toUpperCase is not a function          topic 11.html:19
    at topic 11.html:19
>
```

Caught errors example

```
try {
  var x = 123;
  console.log(“Input: ”+x);

  x.toUpperCase();

  console.log(x);
} catch {
  console.log(“Please enter a string”);
}

console.log(“Errors handled!”);
```



finally statement

The finally statement can be used to execute code after error handling, regardless of the error results.

finally statement syntax

```
try {
  // Code to be tested
} catch (error) {
  // Code to handle errors
} finally {
  // Code to be executed after error handling
}
```

finally statement syntax

Alternatively, you can use only the try and finally statements.

```
try {
  // Code to be tested
} finally {
  // Code to be executed after error handling
}
```

finally statement example

```
try {
   var x = 123;
   console.log(“Input: ”+x);

   x.toUpperCase();
} catch {
   console.log(“Please enter a string”)
} finally {
   console.log(x);
}

console.log(“Errors handled!”);
```

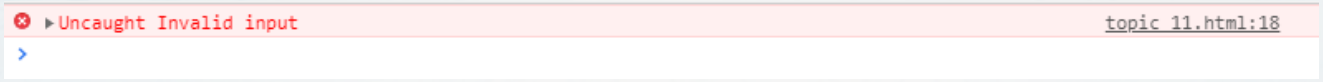
throw statement

The throw statement allows us to create a custom error on JavaScript

Technically we can throw an exception (throw an error) for our own testing

throw statement example

throw (“Invalid input”);

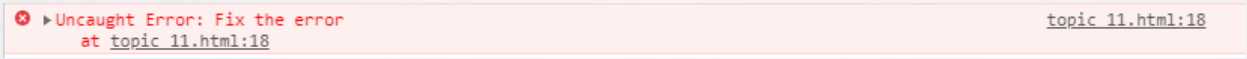


Error object

To throw a more detailed error message that looks more like the default ones we can create an Error object.

Error object example

throw new Error(“Fix the error”);



Practice: Debugging

Go back to the code you have written in the past lessons and try to add some error handling statements. Also try creating your own specific error messages.

DECT-KS©
TOOLS

The End

References 1: Bits and Pieces 7 Types of Native Errors in JavaScript You Should Know

https://blog.bitsrc.io/types-of-native-errors-in-javascript-you-must-know-b8238d40e492

Reference 2: MDN JavaScript reference

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference