Web Design & Development (JavaScript-3)

Class-16 (30/1/21)

Jibitesh Mishra

try … catch

- try
- catch
- throw
- finally
- try..catch comes in pair
- throw statement allows one to create an error
- finally statement lets one to execute code after try and catch

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>
```

Cont..

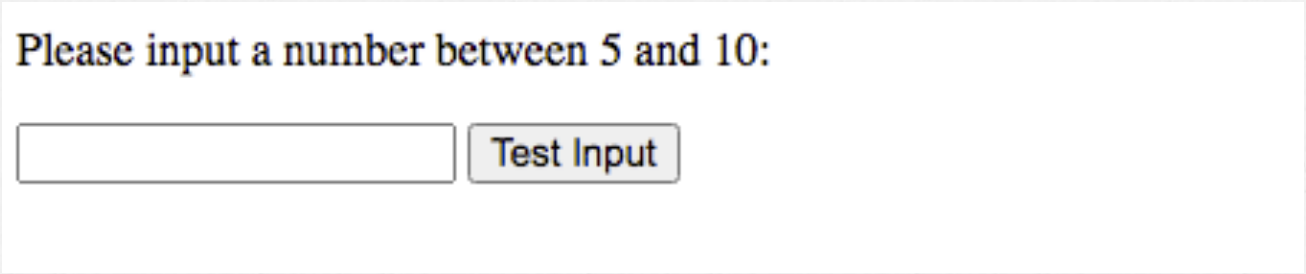
```
<function myFunction() {
  var message, x;
  message = document.getElementById("p01");
  message.innerHTML = "";
  x = document.getElementById("demo").value;
  try {
    if(x == "")  throw "empty";
    if(isNaN(x)) throw "not a number";
    x = Number(x);
```

Cont..

```
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>
```

output



Error object properties

- name : sets or returns an error name e.g.
  - EvalError
  - rangeError
  - referenceError
  - syntaxError
  - typeError
- message : sets or returns an error message (a string)

example

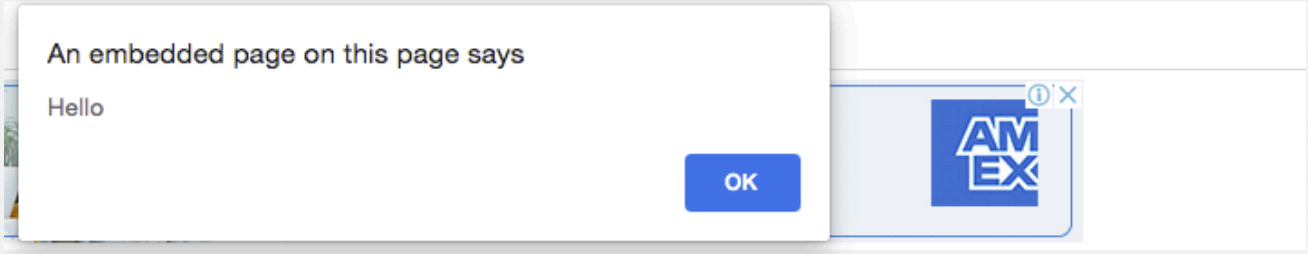
```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Errors</h2>
<p>You cannot evaluate code that contains a syntax error:</p>
<p id="demo"></p>
<script>
try {
  eval("alert('Hello')");
}
```

example

```
catch(err) {

document.getElementById("demo").innerHTML
= err.name;
}
</script>
</body>
</html>
```

output



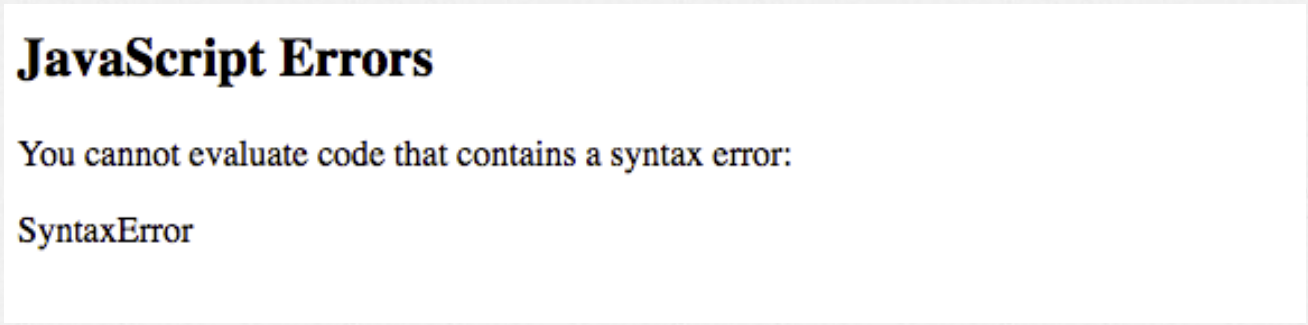
JavaScript Errors

You cannot evaluate code that contains a syntax error:

Syntax error

```
try {
  eval("alert('Hello)");
}
```

output


JavaScript Errors
You cannot evaluate code that contains a syntax error:
SyntaxError

Another example

```
<!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 = 2;
```

Cont..

```
try {
  num.toUpperCase();
  document.getElementById("demo").innerHTML =
num.toUpperCase();
}
catch(err) {
  document.getElementById("demo").innerHTML =
err.name;
}
</script>
</body>
</html>
```

output

JavaScript Errors

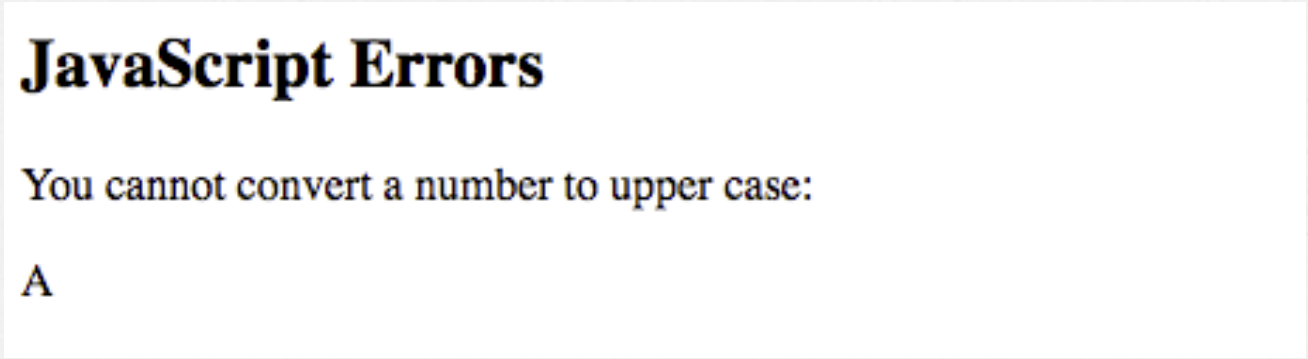
You cannot convert a number to upper case:

TypeError

If num="a"

```
<!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 = "a";
</html>
```

output


JavaScript Errors
You cannot convert a number to upper case:
A

JavaScript function scope

- There are two types of scope: local & global
- Each Function creates a new scope
- Variables declared within a javascript function is LOCAL
- Variables declared outside a javascript function is GLOBAL
- GLOBAL means all scripts and functions in a web pages can access that

example

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Scope</h2>
<p>Outside myFunction() carName is undefined.</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
```

Cont..

```
myFunction();
function myFunction() {
  var carName = "Volvo";
  document.getElementById("demo1").innerHTML =
typeof carName + " " + carName;
}
document.getElementById("demo2").innerHTML =
typeof carName;
</script>
</body>
</html>
```

output

JavaScript Scope

Outside myFunction() carName is undefined.

string Volvo

undefined

Scope cont..

- If a value is assigned to a variable not defined before, it will become automatically GLOBAL
- All global variables belong to the Windows object
- Once one closes the browser window or tab, the global variable is deleted i.e. its life gets over
- Once a function is completed, local variable is deleted i.e. its life gets over

strict

- In order to create a secure javascript, one should use Strict
- All new browsers understands Strict
- Duplictaing a parameter will not be allowed
- Deleting a function will not be allowed e.g.

```
<script>
"use strict";
function x(p1, p2) {};
delete x;       // This will cause an error
</script>
```

this keyword

- this key word refers to the object it refers to
- In an object method, this refers to the owner of the method
- In the next example, the person object is the owner of the fullName method
- When used alone, the owner is the global object
- In a browser, windows is the global object

Javascript object

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p>An object method is a function definition, stored as a property value.</p>
<p id="demo"></p>
<script>
// Create an object:
var person = {
  firstName: "John",
  lastName : "Doe",
  id     : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
// Display data from the object:
document.getElementById("demo").innerHTML = person.fullName();
</script>
</body>
</html>
```

output

JavaScript Objects

An object method is a function definition, stored as a property value.

John Doe

this is the event handler

```
<!DOCTYPE html>
<html>
<body>
<h2>The JavaScript <i>this</i> Keyword</h2>
<button onclick="this.style.display='none'">Click to Remove Me!</button>
</body>
</html>
```

Output: click on button to remove it