

<script>



# *Client Side Technologies*

## **Error Handling**

# Error Object



❑ Whenever an error occurs, an instance of the Error object is created to describe the error.

❑ Error objects can be created in 2 ways:

- Explicitly:

```
var newErrorObj = new Error();
```

- Implicitly:

➔ thrown using the throw statement.

# Error Object(Cont.)



## ❑ Error Object Properties:

| Property           | Description  |
|--------------------|--|
| <b>description</b> | Plain-language description of error                      |
| <b>fileName</b>    | URI of the file containing the script throwing the error |
| <b>lineNumber</b>  | Source code line number of error                         |
| <b>message</b>     | Plain-language description of error (ECMA)               |
| <b>name</b>        | Error type (ECMA)  |
| <b>number</b>      | Microsoft proprietary error number                       |

# Error Object(Cont.)



## ❑ Error constructor:

– `var e = new Error();`

## ❑ Six additional Error constructor ones exist and they all inherit Error:

|                       |   |
|-----------------------|---|
| <b>EvalError</b>      | Raised by eval when used incorrectly  |
| <b>RangeError</b>     | Numeric value exceeds its range   |
| <b>ReferenceError</b> | Invalid reference is used   |
| <b>SyntaxError</b>    | Used with invalid syntax  |
| <b>TypeError</b>      | Raised when variable is not the type expected   |
| <b>URIError</b>       | Raised when <code>encodeURIComponent( )</code> or <code>decodeURIComponent( )</code> are used incorrectly |

Using ***instanceOf*** when catching the error lets you know if the error is one of these built-in types.

# Error Object(Cont.)



## ❑ Error Object standard Properties:

- **Name:** The name of the error constructor used to create the object
- **Example:**

```
var e = new Error('Oops');
```

- **Message:** Additional error information
- **Example:**

```
var e = new EvalError('jaavcsritp is _not_ how you spell it');  
document.write(e.name); //EvalError  
document.write(e.message); // jaavcsritp is _not_ how you
```

spell it

# try...catch



- ❑ The try...catch statement allows you to test a block of code for errors.
- ❑ The try block contains the code to be run.
- ❑ The catch block contains the code to be executed if an error occurs.
- ❑ **Syntax:**

```
try
{
    //Run some code here
}
catch(err) → Implicitly Created  
Error object “err”
{
    //Handle errors here
}
```

# try...catch (Cont.)



- ❑ If you have any functionality that needs to be processed regardless of success or failure, you can include this in the finally block (will be executed even if there is a break, throw, or return statement in the catch block).

- ❑ **Syntax:**

```
try
{
    //Run some code here
}
catch(err)
{
    //Handle errors here
}
finally
{
    //Here code that will be executed on both cases
}
```

# try...catch (Cont.)



```
try{  
    if(x<100)  
        throw "less100"  
    else if(x>200)  
        throw "more200"  
}  
catch(er){  
    if(er=="less100")  
        alert("Error! The value is too low")  
    if(er == "more200")  
        alert("Error! The value is too high")  
}
```



# onerror



- ❑ The old standard solution to catch errors in a web page.
- ❑ The onerror event is fired whenever there is a script error in the page.
- ❑ onerror event can be used to:
  - Suppress error.
  - Retrieve additional information about the error.

# Suppressing errors



```
function supError()
{
    alert("Error occured")
}
window.onerror=supError;
```

OR

```
function supError()
{
    return true;
}
window.onerror=supError;
```

- ❑ return true is added to avoid the yellow alert that appears at the window's lower left corner when an error occurs.

# Retrieve additional information about the error



```
onerror=handleErr
function handleErr(msg,url,l)
{
    //Handle the error here
    return true;
}
```

## □ where

- msg → Contains the message explaining why the error occurred.
- url → Contains the url of the page with the error script
- l → Contains the line number where the error occurred

# throw statement



- ❑ The throw statement allows you to create an exception.
- ❑ Using throw statement with the try...catch, you can control program flow and generate accurate error messages.
- ❑ **Syntax:**

```
throw(exception)
```

- ❑ The exception can be a string, integer, Boolean or an object.

# Throw Error (Cont.)

A circular icon with a white background and a blue border, containing the letters 'JS' in a stylized purple font.

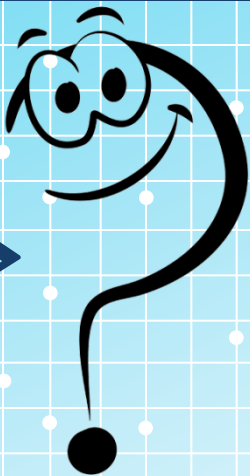
```
try{  
    if(x<100)  
        throw "less100"  
    else if(x>200)  
        throw "more200"  
}  
catch(er){  
    if(er=="less100")  
        alert("Error! The value is too low")  
    if(er == "more200")  
        alert("Error! The value is too high")  
}
```

`<script>`



JavaScript

`</script>`

`<SCRIPT >`  `</SCRIPT>`

```
<script>document.writeln("Thank  
You!")</script>
```