Client-side Technologies

Eng. Niveen Nasr El-Den
SD & Gaming CoE
iTi

Day 8

“ These are the Golden Days of JavaScript

JavaScript Built-in Objects

- String
- Number
- Array
- Date
- Math
- Boolean
- RegExp
- Error
- Function
- Object

Error
built-in Objects

Error Object Creation

- Whenever an error occurs, an instance of error object is created to describe the error.
- Error objects are created either by the environment (the browser) or by your code.
- Developer can create Error objects by 2 ways:
  - Explicitly:
    - var newErrorObj = new Error();
    - thrown using the throw statement
  - Implicitly

Error Object Construction

- Error constructor
  - var e = new Error();
- More than Six additional Error constructor ones exist and they all inherit Error:

EvalError	Raised by eval when used incorrectly
RangeError	Numeric value exceeds its range
ReferenceError	Invalid reference is used
SyntaxError	Used with invalid syntax
TypeError	Raised when variable is not the type expected
URIError	Raised when encodeURI( ) or decodeURI( ) are used incorrectly

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

Error Object Properties

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

Error Object Standard Properties

- name →The name of the error constructor used to create the object
  - Example:
    - var e = new EvalError('Oops');
    - e.name;

      → "EvalError“

- Message → Additional error information:
  - Example:
    - var e = new Error('jaavcsritp is _not_ how you spell it');
    - e.message

      →" jaavcsritp is _not_ how you spell it"

Example!

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 Example

```
if(x<100)
      throw “less100"
else if(x>200)
      throw “more200“
```

```
var e= new Error(“more200“)
if(x<100)
      throw new Error(“less100" )
else if(x>200)
      throw e
```

Example!

Error Handling

JavaScript Error Handling

- There are two ways of catching errors in a Web page:
    1. ***try…catch*** statement.
    2. ***onerror*** event.

try...catch Statement

- 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){
    //Handle errors here
}
```

$\xrightarrow{}$ **Implicitly an Error object “err” is created**

try…catch Statement (no error)

```
try {
        ✓ no error.
        ✓ no error.
        ✓ no error.
    }
    catch( exception )
     {
        ✓ ~~error handling code will not run.~~
     }
    ✓ execution will be continued.
```

try...catch Statement (error in try)

try {

- ✓ no error.
- ✓ no error.

an error! *control is passed to the catch block here.*

this will never execute.

}

catch(exception)

{

- ✓ error handling code is run here

}

✓ execution continues from here.

Example!

try...catch Statement (error in catch)

```
try {
        ✓ no error.
        ✓ no error.
    an error! control is passed to the catch block here.
              this will never execute.
}
catch( exception )
 {
        ✓ error handling code is run here
      an error!
      ~~error handling code is run here will never execute.~~
 }
~~execution wont be continued.~~
```

Example!

try...catch & throw Example

```
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")
     }
```

Example!

Adding the *finally* statement

- If you have any functionality that needs to be processed regardless of success or failure, you can include this in the *finally* block.

try…catch…finally Statement (no error)

try {

- ✓ no error.
- ✓ no error.
- ✓ no error.

}

catch(exception)

{

- ✓ ~~error handling code will not run~~.

}

finally {

- ✓ This code will run even there is no failure occurrence.

}

✓ execution will be continued.

try…catch…finally Statement (error in try)

try {

✓ no error.

✓ no error.

an error! *control is passed to the catch block here.*

this will never execute.

}

catch(exception)

{

✓ error handling code is run here

✓ error handling code is run here

✓ error handling code is run here

}

finally {

✓ This code will run even there is failure occurrence.

}

✓ execution will be continued.

Example!

try…catch…finally Statement (error in catch)

try {

✓ no error.

✓ no error.

an error! *control is passed to the catch block here.*

this will never execute.

}

catch(exception)

{

✓ error handling code is run here

an error!

~~error handling code is run here will never execute.~~

}

finally {

✓ This code will run even there is failure occurrence.

}

~~execution wont be continued.~~

Example!

onerror Event

- 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.

Suppress error

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

OR

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

The value returned determines whether the browser displays a standard error message.

true the browser does not display the standard error message.

false the browser displays the standard error message in the JavaScript console

Retrieve additional information about the error

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

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
col	→ Column number for the line where the error occurred
err	→ Contains the error object

Assignment