# Exception Handling

#### Error life cycle:

E rror Lifecycle

   Browser Error


   window.onerror


      try-catch


        Error
 

* Identify where the ***Errors*** madeoccurs

# Types of Errors

* Coercion
* Data Type Errors
* Communication Errors
* Invalid url/Post
* Server response status
* No network connection
* Server response content

## No network connection:

Internet explorer throws an error when calling open() button then goes to normal life cycle.

Firefox fails silently but throws an error if you try to access any property (status , status text ,response text)

# Server response content

* A status of 200/304 is not enough
* A server errors often return HTML
* If possible set status to 500

# Throw or Try-catch

Errors should be thrown in the low-level parts of the application

Utilities core libraries , etc

Use try catch blokes in higher level parts

Application specific

Client side business logic.

# Non-fatal

Won’t interfere with users main tasks

Effects only the portion of the page

--Easily disabled/ ignored

Recovery is possible

A repeat of the action may result in the appropriate result

Don’t tell the user it isn’t working unless absolutely necessary

# Fatal errors

The application absolutely cannot continue

Significantly interferes with user’s ability to be productive

Other errors will occur if the application continues

Message the user immediately

Reload

# Fatal or Non-fatal

Don’t allow your code to determine what is and is not fatal

Watch out for loops

The user experience comes first.

# Debug Mode

Assign a variable that is globally available

Try-catch should re throw the error

Window.onerror should return false

Allow the browser to handle the error.

# Rules

1. Assume your code will fail
2. Log errors to the server
3. You, not the browser handle errors
4. Identify where errors might occur
5. Throw your own errors
6. Fatal vs Non-fatal
7. Provide a debug mode