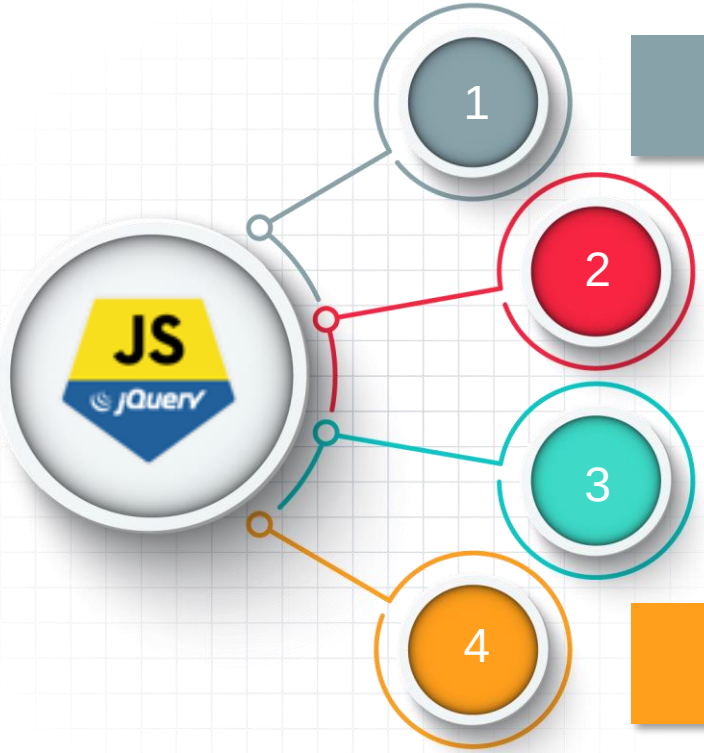
edureka!

JavaScript & JQuery

Course Outline



1. Introduction to JavaScript
2. Functions and Internal Memory Management
3. Event Handling and Objects
4. **Libraries and Frameworks**

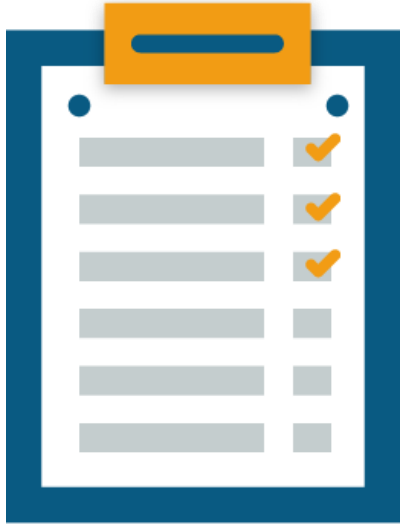
Module 4 – Libraries and Frameworks

edureka!

Objectives

After completing this module, you should be able to:

- Identify and work with Errors/Exceptions
- Use JSON and AJAX
- Categorize libraries, frameworks and tools
- Differentiate between JQuery and JavaScript



edureka!

JavaScript – Errors

- When executing JavaScript code, different errors can occur
- Errors can be:
  - coding errors made by the programmer
  - errors due to wrong input
  - other unforeseeable things

edureka!

Error Handling Statements

- While coding different errors might occur
- For normal execution of the program, handling errors is important
- The following are the error handling statements :

try	• lets you test a block of code for errors
catch	• lets you handle the error
throw	• lets you create custom errors
finally	• lets you execute code, after try and catch, regardless of the result

edureka!

Try and Catch Statements

- The **try** statement allows you to define a block of code to be tested for errors, while it is being executed
- The **catch** statement allows you to define a block of code to be executed, if an error occurs in the try block
- The JavaScript statements **try** and **catch** come in pairs
- Syntax:

```
try {
   statement n;                // Block of code to be tested
}
catch(err) {
      statement n;             //   Block of code to handle errors
}
```

edureka!

Throw Statement

- When an error occurs, JavaScript will generate an error message (throw an exception)
- The throw statement allows you to create a custom error
- Technically, you can throw an exception (throw an error)
- The exception can be a JavaScript String, Number, Boolean or Object
- Syntax:

```
try {
  statement n;                    //Block of code to try
}
catch(err) {
   statement n;                   //Block of code to handle errors
}
finally {
  statement n;                    // Block of code to be executed regardless of the try / catch result
}
```

Finally Statement

- After try and catch, whatever be the result, the **finally** statement lets you execute the code

- Syntax:

```
try {
   Block of code to try
}
catch(err) {
   Block of code to handle errors
}
finally {
   Block of code to be executed regardless of the try / catch result
}
```

Error Object

- JavaScript has a built in Error Object
- Error object contains information about errors
- It is used as a parameter in the catch block, where the type of error is returned by the try block
- It has two main properties:
  - Error Name : Returns an Error Name
  - Error Message : Returns an Error Message

Error Property – Name : Message

EvalError : Returns the error message of the type of error in eval()

- An error has occurred in the eval() function

RangeError : x should be in between 1 to 50

- A number "out of range" has occurred

ReferenceError : x not defined

- An illegal reference has occurred

SyntaxError : Invalid or unexpected token

- A syntax error has occurred

TypeError : func() Is not a function

- A type error has occurred

URIError : URI malformed

- An error in encodeURI() has occurred

Demo – Validate input for any Errors

edureka!

Validate Input Number which is in between 5 and 10

Step1: Type your HTML code. Take a textbox and a button. Add onclick Event Listener to the button, which will call myFunction()

Step2: Create a paragraph element with "message" as its id. Error message will be displayed inside this paragraph

```
<html>

<body>
<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="message"></p>
```

Validate Input Number which is in between 5 and 10

Step3: First clear the inner text of the element, which will display the error message thrown

Step4: Store value of the input, which is to be checked, in variable "x"

```
<script>
function myFunction() {
    var message, x;
    message = document.getElementById("message");
    message.innerHTML = "";
    x = document.getElementById("demo").value;
    try {
        if(x == "")  throw "is empty";
        if(isNaN(x)) throw "is not a number";
        x = Number(x);
        if(x > 10)   throw "is too high";
        if(x < 5)    throw "is too low";
    }
    catch(err) {
        message.innerHTML = "Input " + err;
    }
    finally {
        document.getElementById("demo").value = "";
    }
}
</script>

</body>
```

Step5: The try block consists the code that is to be checked for errors

Step6: The throw statement throws a custom error, if the conditions do not match our requirements

edureka!

Validate Input Number which is in between 5 and 10

Step7: When the error is thrown, the catch block starts executing and displays the error that has been thrown by the try block

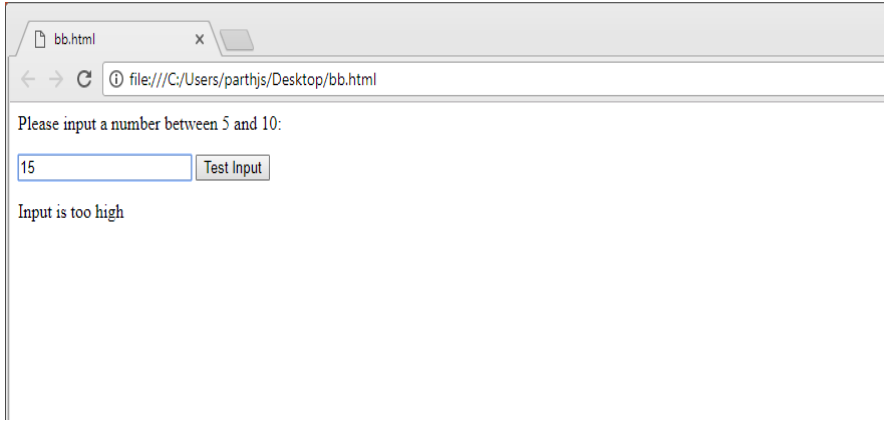
```
<script>
function myFunction() {
    var message, x;
    message = document.getElementById("message");
    message.innerHTML = "";
    x = document.getElementById("demo").value;
    try {
        if(x == "")  throw "is empty";
        if(isNaN(x)) throw "is not a number";
        x = Number(x);
        if(x > 10)   throw "is too high";
        if(x < 5)    throw "is too low";
    }
    catch(err) {
        message.innerHTML = "Input " + err;
    }
    finally {
        document.getElementById("demo").value = "";
    }
}
</script>

</body>
```

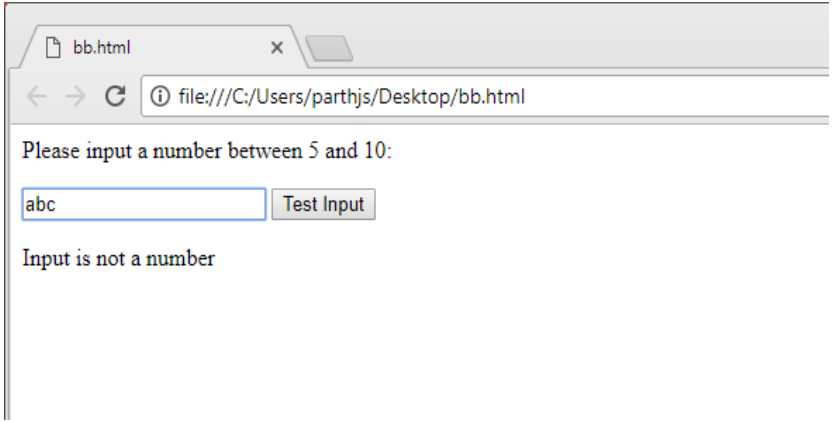
Step8: In the finally block we will clear the value of textbox, so that the user can enter another input

Validate Input Number which is in between 5 and 10

Step9: When input is above 10, it returns "Input is high"



Step10: When input is Not a Number, it returns "Input is not a number"



JavaScript Object Notation (JSON)

- A syntax for storing and exchanging data
- JSON data is written as name/value pairs, just like JavaScript object properties
- A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value

- Conversion of JavaScript objects to string format

```
alert(JSON.stringify(person));
```

- Conversion of JSON string to a JavaScript object

```
var person = { ”name”:"saya", ”age”:46 };
var jsonStr = JSON.stringify(person);
alert("Person Object in JSON format is: " + jsonStr);
var newPerson = JSON.parse(jsonStr);
alert("Your NEW name is: " + newPerson.name + "\nYour NEW age is: "
+ newPerson.age);
```

edureka!

AJAX

AJAX

Asynchronous **Ja**vaScript + **X**ML

AJAX uses a combination of

A browser built -in XMLHttpRequest object (to request data from a web server)

JavaScript and HTML DOM (to display or use the data)

Why do Developers prefer it

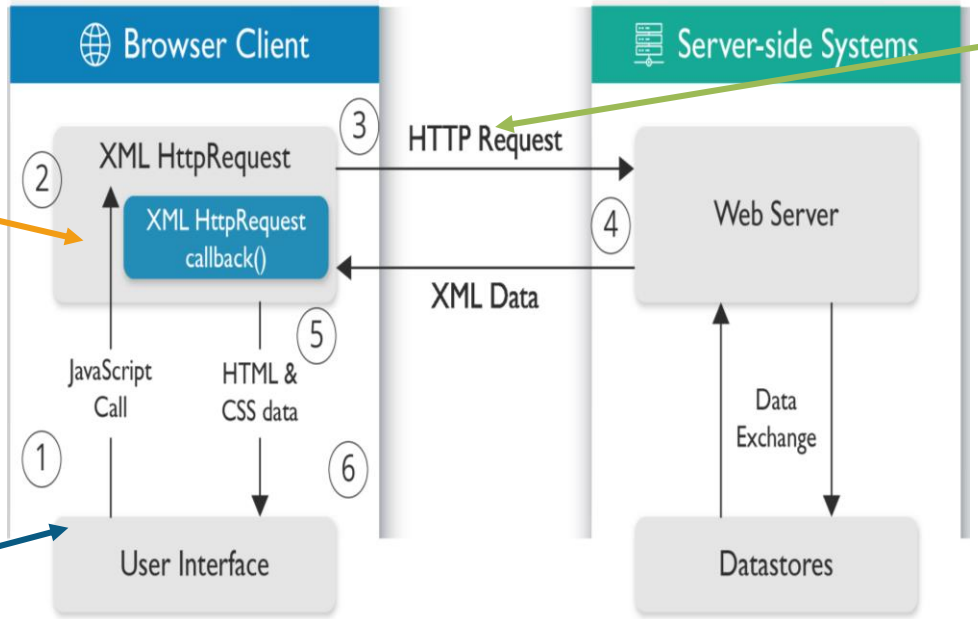
Read data from a web server - after the page has loaded

Update a web page without reloading the page

Send data to a web server - in the background

AJAX – Working

Step2: An XMLHttpRequest object is created by JavaScript

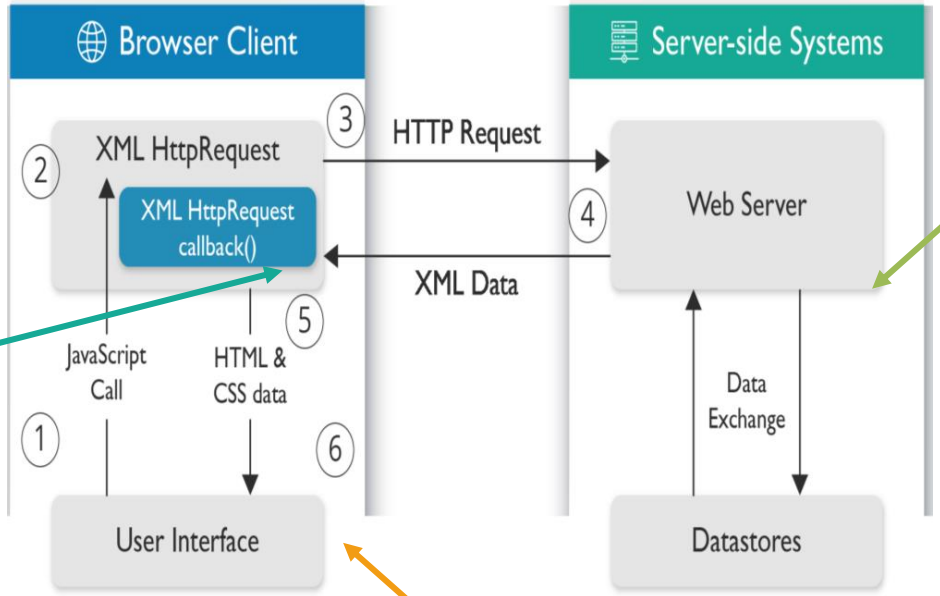


Step3: The XMLHttpRequest object sends an HTTP Request to a web server

Step1: An event occurs in a web page (the page is loaded, a button is clicked) calls a function

edureka!

AJAX – Working



Step4: After receiving the request, the server processes the request and replies with the appropriate response

Step5: Server sends XML data or JSON data to the XMLHttpRequest callback() function

Step6: Client receives data from server and displays using HTML, CSS data on the browser

AJAX- GET and POST

The two most common "methods" for sending a request to a server are GET and POST

GET : Should be used for operations where you are only "getting" data from the server, not changing data on the server

- A search query can be a GET request
- GET requests generally sends their data by appending into URL using query string

POST - The POST method should be used for operations where you are changing data on the server

- A user saving a blog post can be a POST request
- POST requests are generally not cached by the browser
- A query string can be part of the URL, but the data tends to be sent separately as post data

edureka!

AJAX - The XMLHttpRequest Object

- The XMLHttpRequest object can be used to exchange data with a web server behind the scenes
- This means that it is possible to update parts of a web page, without reloading the whole page
- Example:

```
<script>
 var httpReq;
 if (window.XMLHttpRequest)
 httpReq = new XMLHttpRequest(); // for IE7+,Firefox,Chrome,Opera,
Safari
    else
 httpReq =new ActiveXObject("Microsoft.XMLHTTP"); // code for IE6
    httpReq.onreadystatechange = function () {
        if (httpReq.readyState == 4)
            if(httpReq.status == 200)
                alert("The response is: " + httpReq.responseText);
}   httpReq.open("GET","http://127.0.0.1/index.php",true);
    httpReq.send();
</script>
```

edureka!

JavaScript Libraries

- A library is an organized collection/package of code that is called by our application to perform a particular task
- Libraries normally provide a higher level of abstraction, which smooths over implementation details and inconsistencies

A typical library could include functions to handle

- Strings
- Dates
- HTML DOM elements
- Events
- Cookies
- Animations
- Network requests

JavaScript Frameworks

- A framework is a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful
- Functionality such as events, storage, and data binding are normally provided for you

A framework can be everything you use in application development

- A library
- A collection of many libraries
- A collection of scripts
- Any piece of software you need to create your application

edureka!

JavaScript –Tools

- A **tool** aids development, but is not an integral part of your project
- Tools Include :

Tools include

- build systems
- compilers
- transpilers
- code minifiers
- image compressors
- deployment mechanisms

List of some Popular Libraries, Frameworks and Tools

Libraries

- Jquery
- React
- Lodash
- Underscore
- Polymer

Frameworks

- AngularJS 1.X
- AngularJS 2.X & 4.X
- Vue.js
- Backbone.js
- Ember.js
- Knockout.js

Tools

- Jasmine
- Mocha
- ESlint
- JS Hint
- Npm
- Grunt
- Webpack
- Browserify

JQuery – Introduction

- It is a free and open source JavaScript library
- It was released in 2006
- jQuery has changed the way that millions of people write JavaScript
- It is very fast and concise in:
  - The way it traverses through HTML Documents
  - Handling Events
  - Performing Animations

edureka!

Why JQuery?

jQuery is a JavaScript library designed to simplify the client-side scripting of HTML

It helps to create powerful dynamic web pages and web applications

Is intuitive and easy to learn

It integrates with Visual Studio IDE with ease

Helps in loading pages faster and is SEO friendly

Helps in creating animated pages like flash

Many of the biggest companies on the Web use jQuery, such as: Google, Microsoft, IBM and Netflix

JavaScript vs JQuery

JavaScript	JQuery
▪ It is a scripting language that works with all Browsers	▪ It is a JavaScript Library, made up from JavaScript
▪ Write your own script, which takes time	▪ Pre-written scripts in the libraries, reduces the time taken
Example : function changeBackground(color) { document.body.style.background = color; } onload= “changeBackground (‘red’);”	Example: $ (‘body’).css(‘background’, ‘#ccc’);

edureka!

JQuery – Getting Started

- There are 2 ways to Add JQuery on your webpage:
  - Download the library from www.jquery.com

    Syntax:

```
<head>
<script src ="jquery-3.2.1.min.js"></script>
</head>
```

  - Include JQuery from a CDN

    Syntax:

```
<head>
<script src ="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
</head>
```

OR

```
<head>
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.2.1.min.js"></script>
</head>
```

edureka!

Summary

In this module, you should have learnt :

- Handling of errors in JavaScript
- Working with AJAX and JSON
- The differences between Libraries, Frameworks and Tools
- The differences between JavaScript and JQuery

edureka!

Thank You

For more information please visit our website
www.edureka.co