UNIT 4 : AJAX (ASYNCHRONUS JAVASCRIPT AND XML)

4.1 Fundamentals of AJAX technology:

4.1.1 Difference between Synchronous and Asynchronous web application

4.1.2 XMLHttpRequest technology

4.2 XMLHttpRequest

4.2.1 Properties :(onReadyStateChange, readyState, responseText,
responseXML)

4.2.2 XMLHttpRequest Methods : (Open(), send(), setRequestHeader())

4.3 Working of AJAX and its architecture

Introduction

- □ AJAX is not a new technology, in fact, Ajax is not even really a technology at all.
- □ It is getting tremendous industry momentum and several tool kit and frameworks are emerging.
- □ AJAX is just a term to describe the process of exchanging data from a web server asynchronously through JavaScript, without refreshing the page. But at the same time, AJAX has browser incompatibility and it is supported by JavaScript, which is hard to maintain and debug

- AJAX is an acronym for Asynchronous JavaScript and XML. It is a group of interrelated technologies like JavaScript, DOM, XML, HTML/XHTML, CSS, XMLHttpRequest etc.
- Ajax is just a means of loading data from the server and selectively updating parts of a web page without reloading the whole page. So it is fast.
- Basically, what Ajax does is make use of the browser's built-in XMLHttpRequest(XHR) object to send and receive information to and from a web server asynchronously, in the background, without blocking the page or interfering with the user's experience.

- AJAX is not a programming language.

- AJAX just 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)

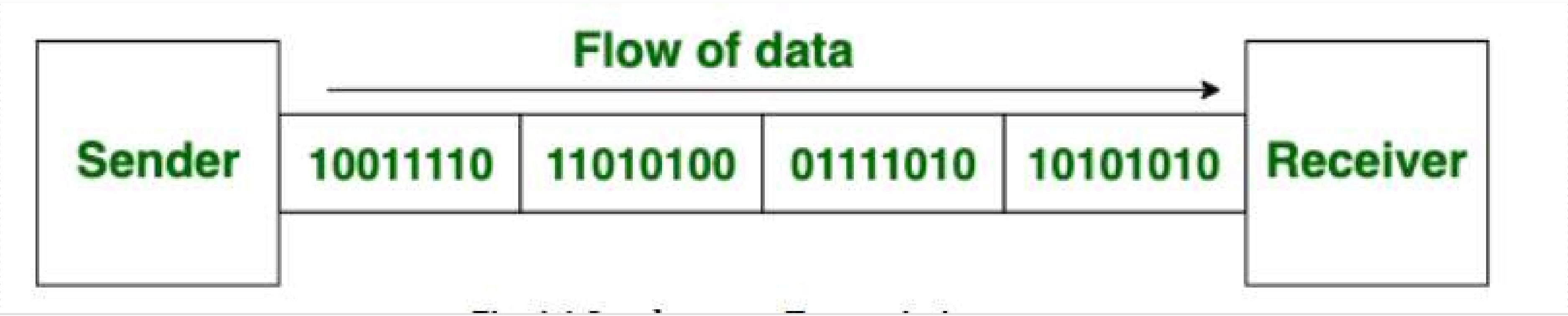
4.1 Fundamentals of AJAX technology:

4.1.1 Difference between Synchronous and Asynchronous web application

4.1.2 XMLHttpRequest technology

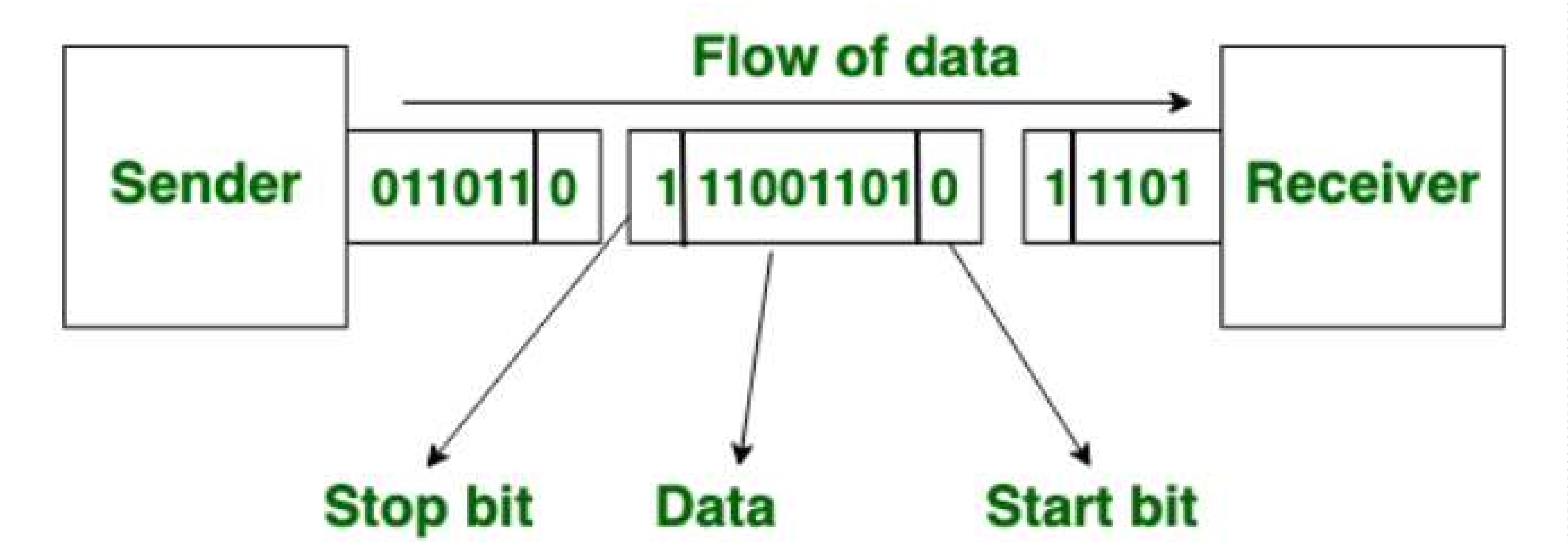
Synchronous web application

- In **Synchronous Transmission**, data is sent in form of blocks or frames. This transmission is the full duplex type. Between sender and receiver, the synchronization is compulsory. In Synchronous transmission, there is no gap present between data. It is more efficient and more reliable than asynchronous transmission to transfer the large amount of data

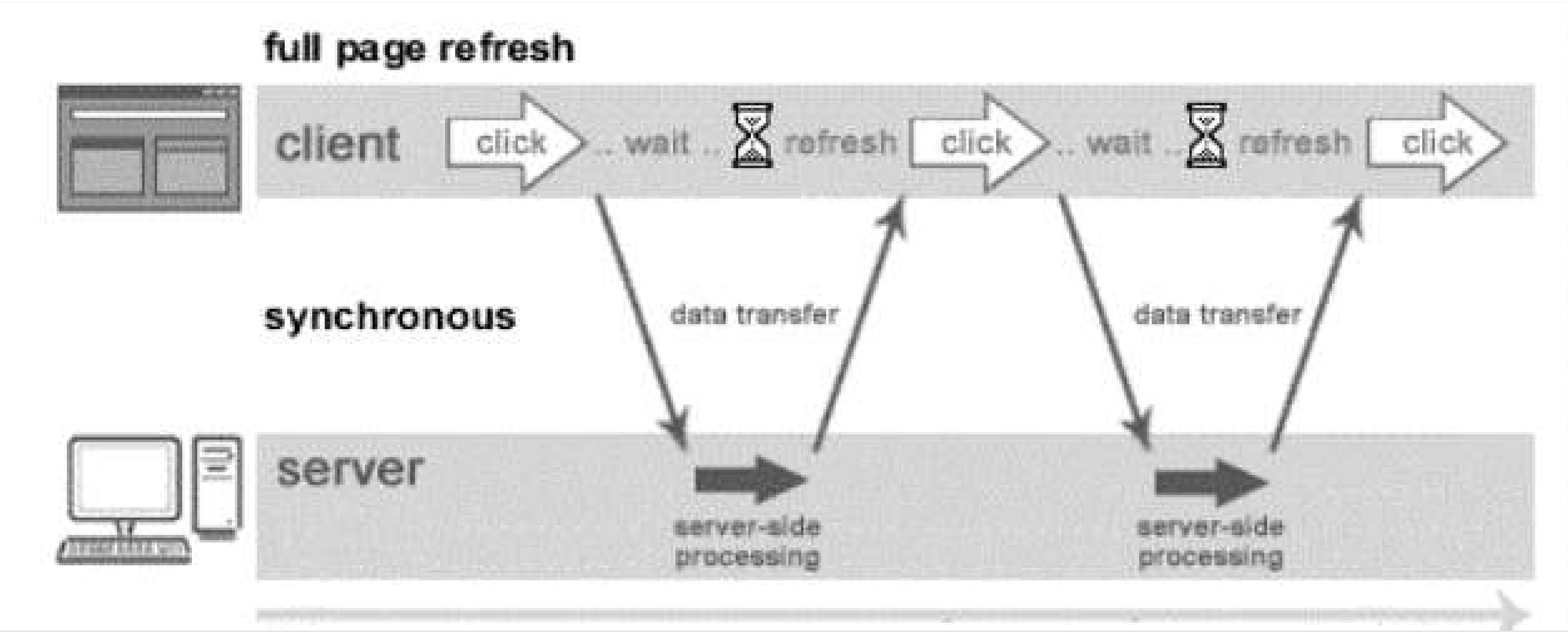

Flow of data
Sender
10011110
11010100
01111010
10101010
Receiver

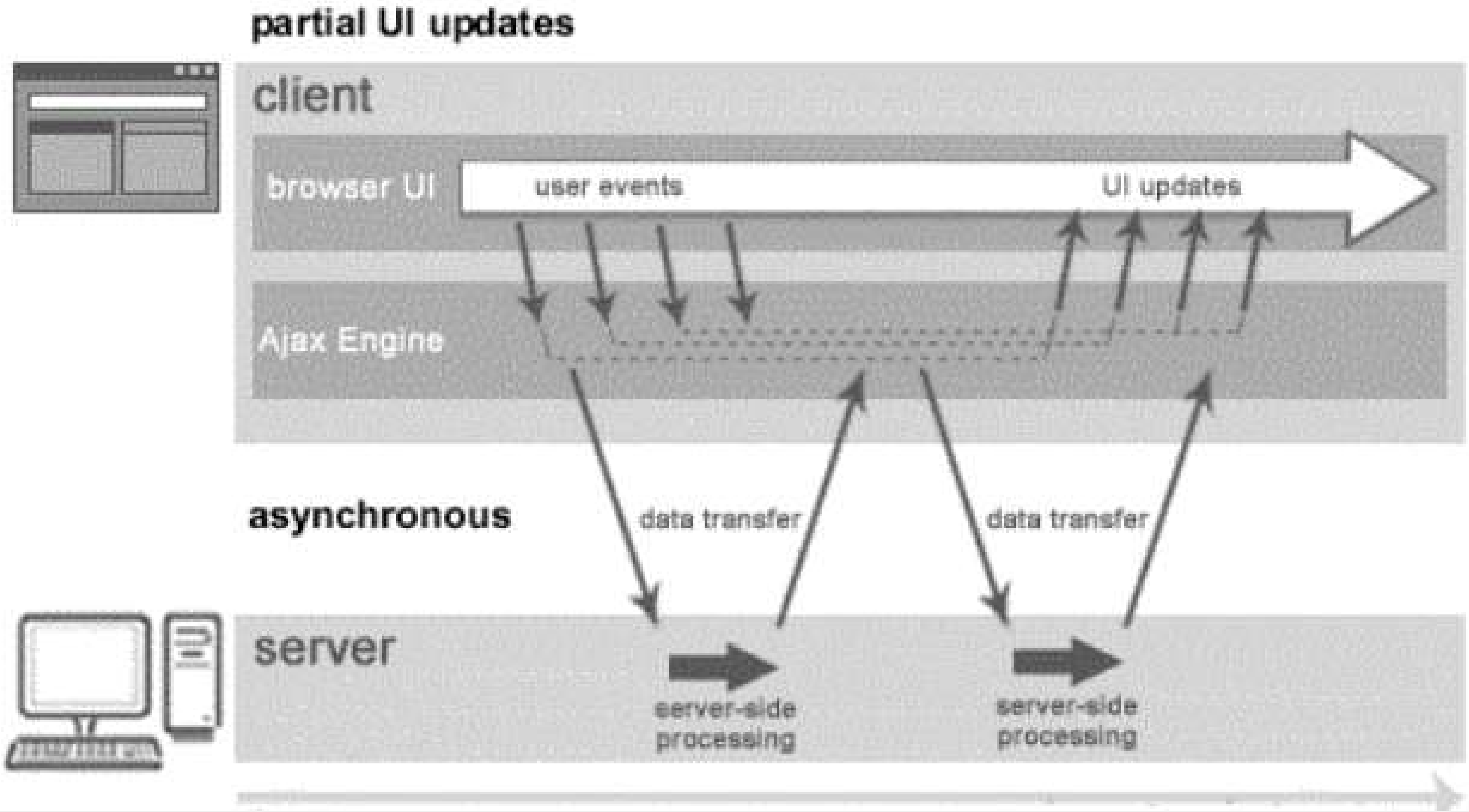
Asynchronous web application

- In **Asynchronous Transmission**, data is sent in form of byte or character. This transmission is the half duplex type transmission.
- In this transmission start bits and stop bits are added with data. It does not require synchronization.


Flow of data
Sender
011011
0
1
11001101
0
1
1101
Receiver
Stop bit
Data
Start bit

4.1.1 Difference between Synchronous and Asynchronous web application




partial UI updates
client
browser UI
user events
UI updates
Ajax Engine
asynchronous
data transfer
data transfer
server
server-side processing
server-side processing

4.1.2 XMLHttpRequest technology

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

- XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.
- The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.

4.2 XMLHttpRequest

4.2.1 Properties :(onReadyStateChange, readyState, responseText,responseXML)

4.2.2 XMLHttpRequest Methods : (Open(), send(), setRequestHeader())

4.3 Working of AJAX and its architecture

- *Create Object:*
- *variable* = new XMLHttpRequest();

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method,url,async,user,psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

- 4.2.1 Properties
  - onReadyStateChange
  - readyState
  - responseText
  - responseXML

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

AJAX code to read TEXT file

demo.txt

Hello

Very Good Morning

SYBCA THE GREAT CLASS

readdata.htm

```
<html>
<head>
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML =this.responseText;
  }
  xhttp.open("GET", "demo.txt");
  xhttp.send();
}
</script>

</head>
<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>
<h2>SYBCA</h2>

</body>
```

AJAX code to read XML file

demo.xml

```
<student>
        <stud>
        <fname>Rohit</fname>
        <lname>Sharma</lname>
        </stud>
        <stud>
        <fname>Virat</fname>
        <lname>Kohli</lname>
        </stud>
</student>
```

readdata.html

```
<html>
<head>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
    myFunction(this);
    }
  };
  xhttp.open("GET", "data.xml", true);
  xhttp.send();
}
function myFunction(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var table="<table border='2'><tr><th>Title</th><th>Artist</th></tr>";
  var x = xmlDoc.getElementsByTagName("stud");
  for (i = 0; i <x.length; i++) {
    table += "<tr><td>" +
    x[i].getElementsByTagName("fname")[0].childNodes[0].nodeValue +
    "</td><td>" +
    x[i].getElementsByTagName("lname")[0].childNodes[0].nodeValue +
    "</td></tr>";
  }
  document.getElementById("demo").innerHTML = table;
}
</script>
</head>
<body>
<div id="demo"></div>
<button onclick="loadDoc()">Click</button>
</body>
</html>
```

AJAX code to read JSON file

demo.json

```
{
        "fname":"Virat",
        "lname":"Kohli"
}
```

readdata.html

```
<html>
<head>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
                                        const obj=JSON.parse(this.responseText);
                                        document.getElementById("demo").innerHTML = obj.fname;
    }
  };
  xhttp.open("GET", "data.json", true);
  xhttp.send();
}

</script>
</head>
<body>
<div id="demo"></div>
<button onclick="loadDoc()">Click</button>
</body>
</html>
```

demo.json

```
["Ford", "BMW", "Fiat"]
```

readjson.htm

```
<html>
<head>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
                                        const obj=JSON.parse(this.responseText);
                                        for(var i=0;i<obj.length;i++)
                                        document.getElementById("demo").innerHTML += obj[i];
    }
  };
  xhttp.open("GET", "demo.json", true);
  xhttp.send();
}

</script>
</head>
<body>
<div id="demo"></div>
<button onclick="loadDoc()">Click</button>
</body>
</html>
```

- 4.2.2 XMLHttpRequest Methods
  - open()
  - send()
  - setRequestHeader()

- The XMLHttpRequest object is used to exchange data with a server.

-
  ```
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
  ```

Method	Description
open(*method, url, async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

setRequestHeader()

- The XMLHttpRequest method setRequestHeader() sets the value of an HTTP request header. When using setRequestHeader(), you must call it after calling open(), but before calling send(). If this method is called several times with the same header, the values are merged into one single request header.

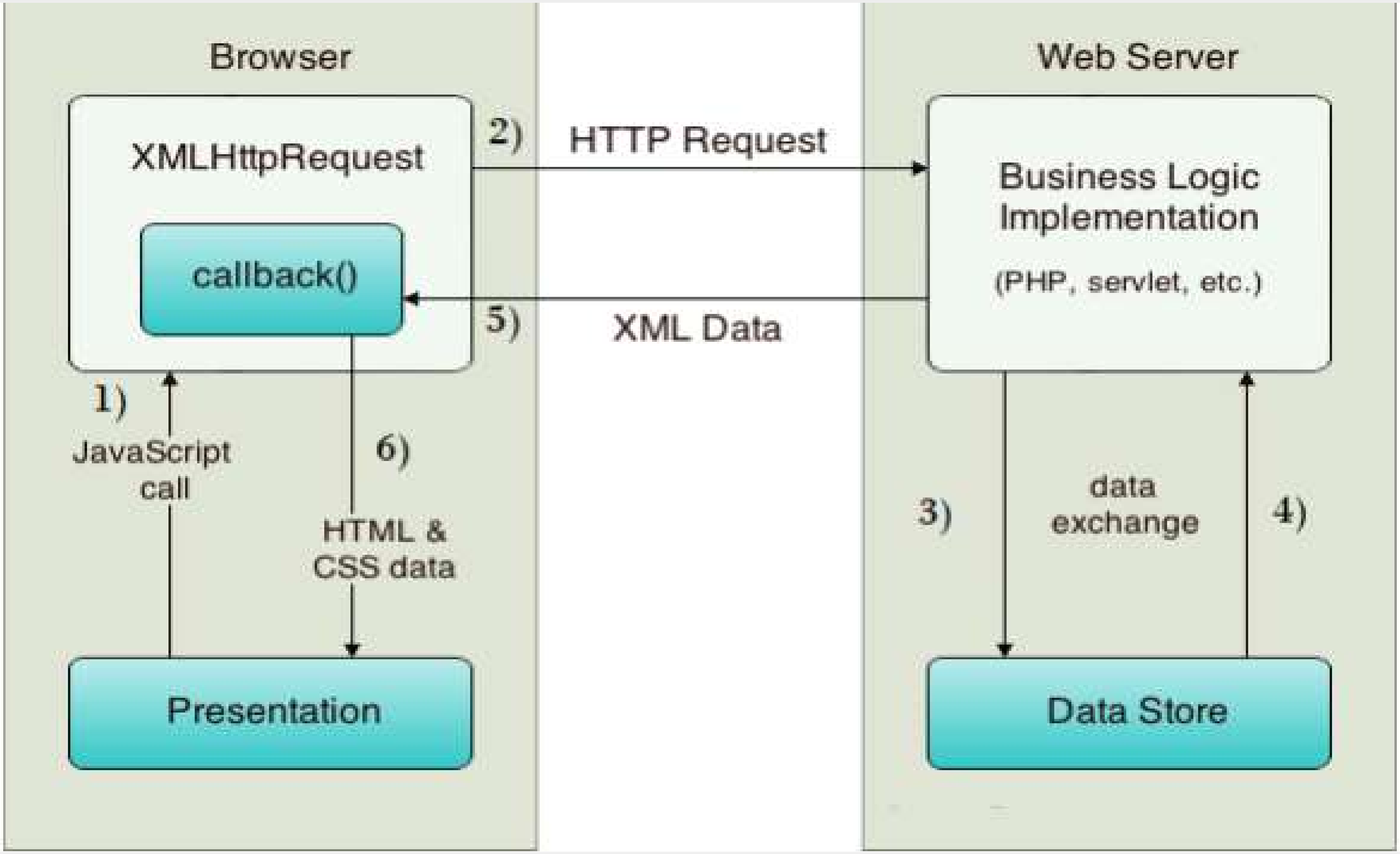
- XMLHttpRequest.setRequestHeader(header, value)
- header
  - The name of the header whose value is to be set.
- value
  - The value to set as the body of the header.

Example

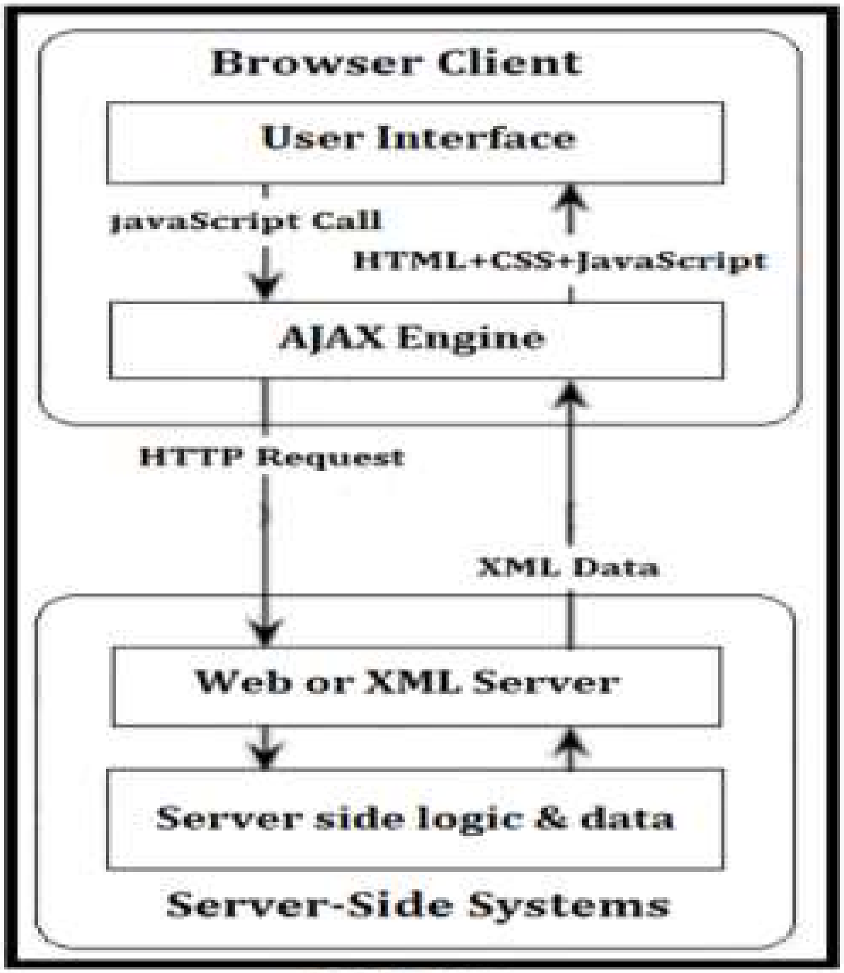
```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("POST", "demo_post2.php");
  xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xhttp.send("fname=Henry&lname=Ford");
}
</script>

</body>
</html>
```

- 4.3 Working of AJAX and its architecture

Working with Data



AJAX Application Model



AJAX Web
Application Model

Advantages:

- Speed is enhanced as there is no need to reload the page again.
- AJAX make asynchronous calls to a web server, this means client browsers avoid waiting for all the data to arrive before starting of rendering.
- Form validation can be done successfully through it.
- Bandwidth utilization – It saves memory when the data is fetched from the same page.
- More interactive.

Disadvantages

- Ajax is dependent on Javascript. If there is some Javascript problem with the browser or in the OS, Ajax will not support.
- Ajax can be problematic in Search engines as it uses Javascript for most of its parts.
- Source code written in AJAX is easily human readable. There will be some security issues in Ajax.
- Debugging is difficult.
- Problem with browser back button when using AJAX enabled pages