VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

BCSE404L – Internet and Web Programming

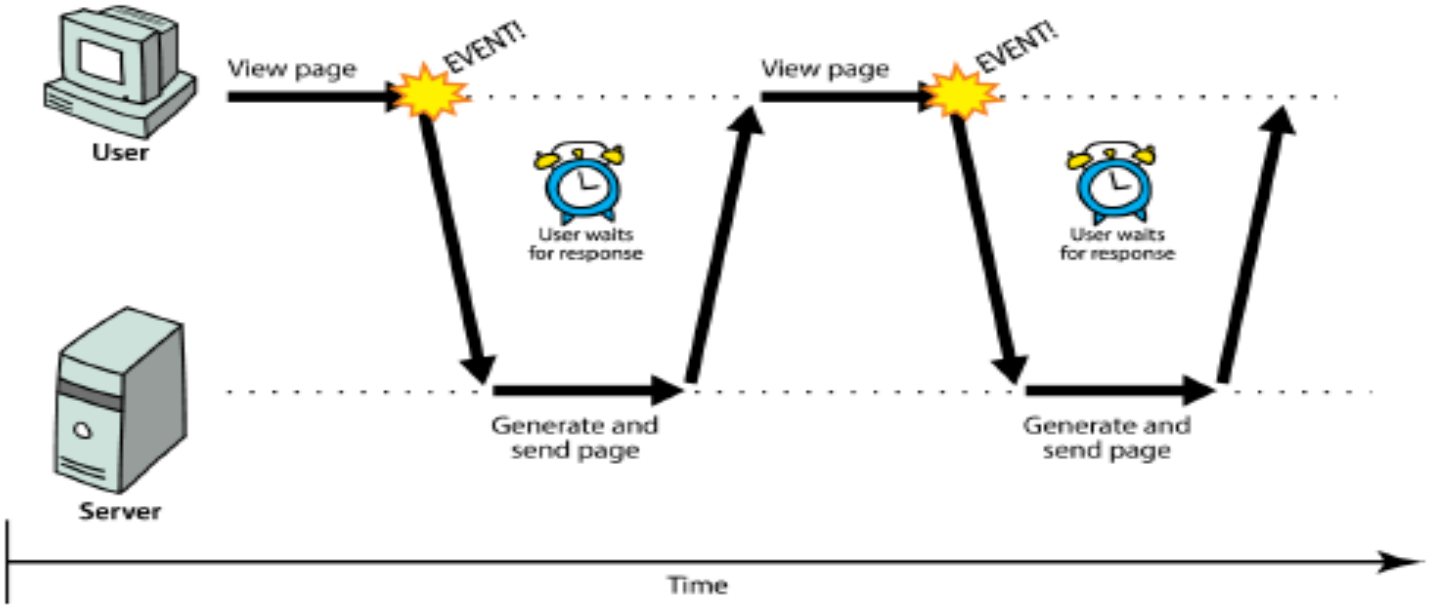
M6L3- AJAX

Dr. P. Saravanan

Associate Professor - SCOPE

VITCC

Synchronous web communication



- synchronous: user must wait while new pages load
  - the typical communication pattern used in web pages (click, wait, refresh)

Web applications and Ajax

3

- **web application**: a dynamic web site that mimics the feel of a desktop app
  - presents a continuous user experience rather than disjoint pages
  - examples: Gmail, Google Maps, Google Docs and Spreadsheets, Flickr, A9

Web applications and Ajax

4

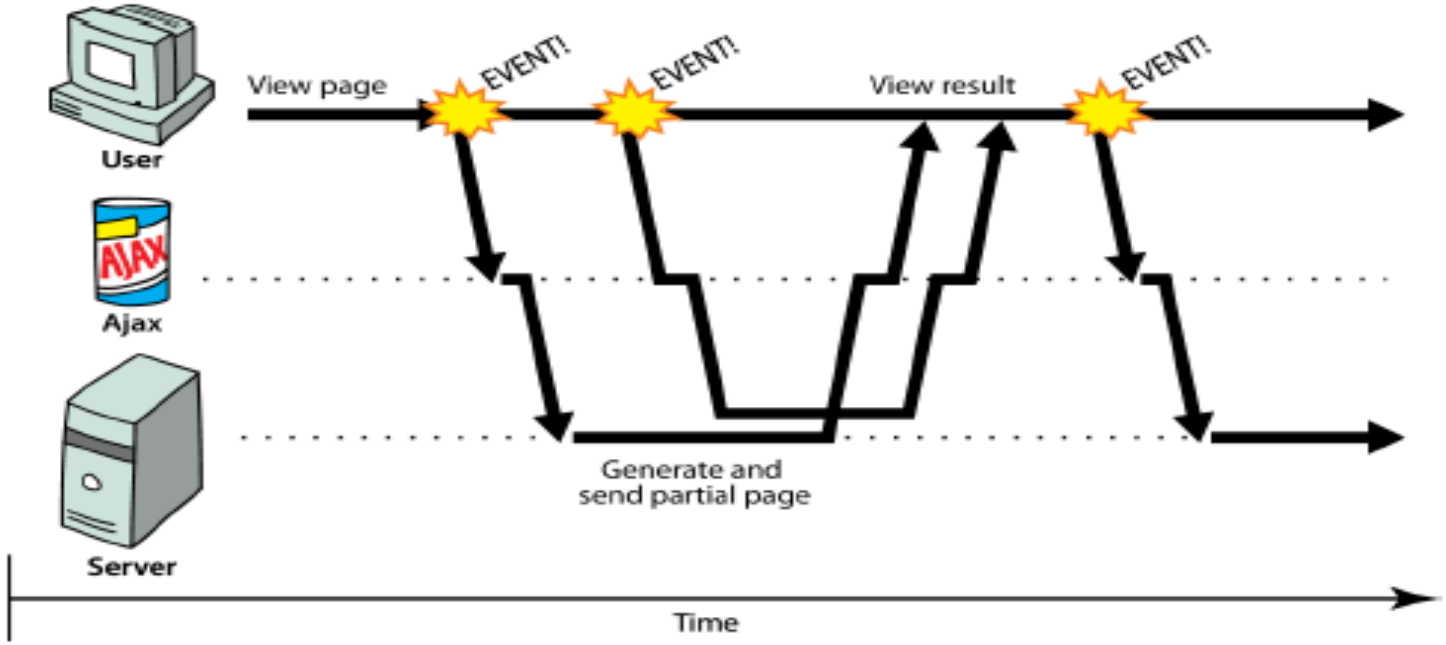
Ajax: Asynchronous JavaScript and XML

- Not a programming language; a particular way of using JavaScript
- A group of related existing technologies compiled together or technique to make web pages feel more responsive
- Makes interactive web pages by providing a way for the web page to interact with the server
- AJAX is a framework
- Downloads data from a server in the background
- Allows dynamically updating a page without making the user wait
- Avoids the "click-wait-refresh" pattern

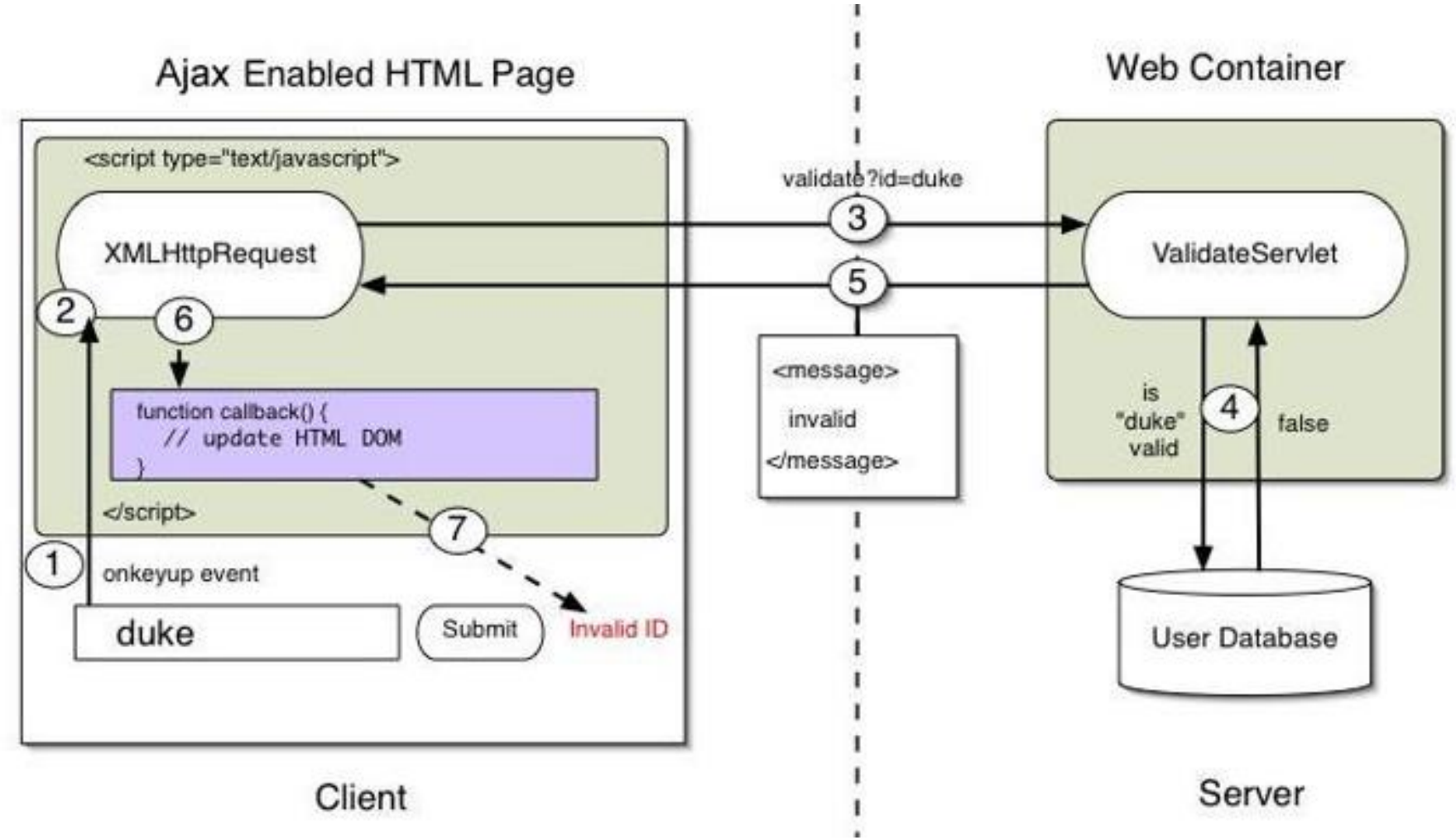
Based on Internet Standards

- XHTML/HTML and CSS
  - To display the data
- JavaScript (XMLHttpRequest calls)
  - To exchange data asynchronously with the server
- XML
  - To tranfer the data
- DOM (document object model)
  - To navigate the hierarchy of X/HTML elements

Asynchronous web communication



- **asynchronous**: user can keep interacting with page while data loads
  - communication pattern made possible by Ajax


Ajax Enabled HTML Page
Web Container
<script type="text/javascript">
XMLHttpRequest
function callback() {
// update HTML DOM
}
</script>
onkeyup event
duke
Submit
Invalid ID
validate?id=duke
<message>
invalid
</message>
ValidateServlet
is
"duke"
valid
false
User Database
1
2
3
4
5
6
7
Client
Server

A typical Ajax request

1. user clicks, invoking an event handler
2. handler's code creates an `XMLHttpRequest` object
3. `XMLHttpRequest` object requests page from server
4. server retrieves appropriate data, sends it back
5. `XMLHttpRequest` fires an event when data arrives
   - this is often called a callback
   - you can attach a handler function to this event
6. your callback event handler processes the data and displays it

JavaScript/XMLHttpRequest

- Provides connection between the X/HTML element(s) and how they behave
- Sends XMLHttpRequests on demand when the user creates events
- Handles events both from the user and the replies from the server
- Can parse XML data and map it to data objects needed in the JavaScript
- Updates the X/HTML elements as needed

The XMLHttpRequest object

- The `XMLHttpRequest` object is the backbone of every Ajax method.
- This depends upon the web browser that the client is using because of the different ways in which the object has been implemented in the browsers.
- Firefox, Safari, Opera, and some other browsers can create one of these objects simply using the "new" keyword.

```
<script type="text/javascript">
                ajaxRequest = new XMLHttpRequest();

</script>
```

- Microsoft Internet Explorer implements this object using its proprietary ActiveX technology.

```
request = new ActiveXObject("MSXML2.XMLHttp.6.0");
```

The XMLHttpRequest object (cont.)


```
function getXMLHttpRequest()
/*   This function attempts to get an Ajax request object by trying
     a few different methods for different browsers.  */
{
   var request, err;
   try {
          request = new XMLHttpRequest();    // Firefox, Safari, Opera, etc.
       }
   catch(err) {
       try {               //  first attempt for Internet Explorer
          request = new ActiveXObject("MSXML2.XMLHttp.6.0");
           }
       catch (err) {
                      try {     //  second attempt for Internet Explorer
                      request = new ActiveXObject("MSXML2.XMLHttp.3.0");
                          }
                      catch (err) {
                           request = false;  // oops, can't create one!
                                  }
                  }
              }
   return request;
}
```

XMLHttpRequest object properties

Property	Description
readyState	An integer from 0. . .4. (0 means the call is uninitialized, 4 means that the call is complete.)
onreadystatechange	Determines the function called when the objects readyState changes.
responseText	Data returned from the server as a text string (read-only).
responseXML	Data returned from the server as an XML document object (read-only).
status	HTTP status code returned by the server
statusText	HTTP status phrase returned by the server

We use the `readyState` to determine when the request has been completed, and then check the `status` to see if it executed without an error.

XMLHttpRequest object methods

Method	Description
open('method', 'URL', asyn)	Specifies the HTTP method to be used (GET or POST as a string, the target URL, and whether or not the request should be handled asynchronously (asyn should be true or false, if omitted, true is assumed).
send(content)	Sends the data for a POST request and starts the request, if GET is used you should call send(null).
setRequestHeader('x','y')	Sets a parameter and value pair x=y and assigns it to the header to be sent with the request.
getAllResponseHeaders()	Returns all headers as a string.
getResponseHeader(x)	Returns header x as a string.
abort()	Stops the current operation.

The `open` object method is used to set up the request, and the `send` method starts the request by sending it to the server (with data for the server if the POST method is used).

A general skeleton for an Ajax application


```
<script type="text/javascript">
      // *****  include the getXMLHttpRequest function defined before
var ajaxRequest = getXMLHttpRequest();

if (ajaxRequest) {   //  if the object was created successfully

      ajaxRequest.onreadystatechange = ajaxResponse();
      ajaxRequest.open("GET", "search.php?query=Bob");
      ajaxRequest.send(null);
   }


function ajaxResponse()  //This gets called when the readyState changes.
{
   if (ajaxRequest.readyState != 4)  //  check to see if we're done
      {  return;  }
   else {
     if (ajaxRequest.status == 200) //  check to see if successful
          {   //  process server data here. . . }
     else {
       alert("Request failed: " + ajaxRequest.statusText);
          }
     }
}
</script>
```

A first example

- Here's an example to illustrate the ideas we've mentioned (inspired by an example in the book Ajax in 10 Minutes by Phil Ballard).
- The main idea is that we're going to get the time on the server and display it to the screen (and provide a button for a user to update this time). The point I want to demonstrate here is how to use Ajax to do this update without updating/refreshing the entire webpage.

- We use a (very) small PHP script to get the date from the server, and return it as a string as a response to the request. Here is the script:

```
<?php
    echo date('H:i:s');
    ?>
```

- I saved this as the file "telltime.php".
- The HTML file and JavaScript code follows.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
<title>Ajax Demonstration</title>
<style>
body {
   background-color: #CCCCCC;
   text-align: center;
     }
.displaybox {
   margin: auto;
   width: 150px;
   background-color: #FFFFFF;
   border: 2px solid #000000;
   padding: 10px;
   font: 1.5em normal verdana, helvetica, arial, sans-serif;
            }
</style>

<script type="text/javascript">
var ajaxRequest;

function getXMLHttpRequest()
/*   This function attempts to get an Ajax request object by trying
     a few different methods for different browsers.  */
{
      //  same code as before. . .
}
```

```
function ajaxResponse()  //This gets called when the readyState changes.
{
   if (ajaxRequest.readyState != 4)  //  check to see if we're done
      {  return;  }
   else {
     if (ajaxRequest.status == 200) //  check to see if successful
          {
                  document.getElementById("showtime").innerHTML =
                                  ajaxRequest.responseText; }
     else {
       alert("Request failed: " + ajaxRequest.statusText);
          }
     }
}

function getServerTime()   //  The main JavaScript for calling the update.
{
   ajaxRequest = getXMLHttpRequest();
   if (!ajaxRequest)  {
         document.getElementById("showtime").innerHTML = "Request error!";
         return;      }
   var myURL = "telltime.php";
   var myRand = parseInt(Math.random()*999999999999999);
   myURL = myURL + "?rand=" + myRand;
   ajaxRequest.onreadystatechange = ajaxResponse();
   ajaxRequest.open("GET", myURL);
   ajaxRequest.send(null);
}
</script>
</head>
```

```
<body>
<h1>Ajax Demonstration</h1>

<h2>Getting the server time without refreshing the page</h2>

<form>
    <input type="button" value="Get Server Time"
   onclick="getServerTime();" />
</form>
<div id="showtime" class="displaybox"></div>

</body>
</html>
```

The main functionality is handled by the `getServerTime()` function in setting up and sending the `XMLHttpRequest` object, and the `ajaxResponse()` function to display the time.