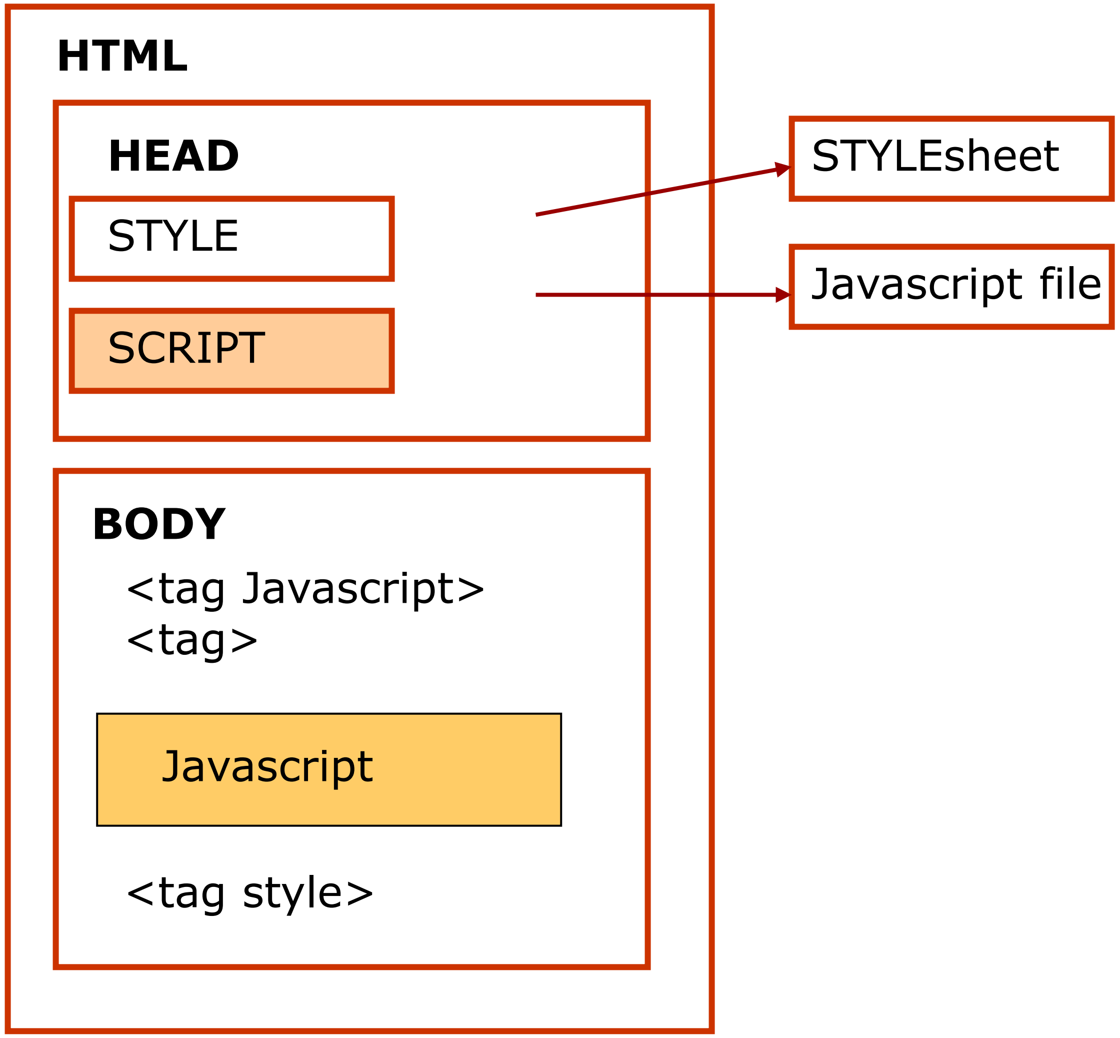
Client side web programming

Dynamic Applications

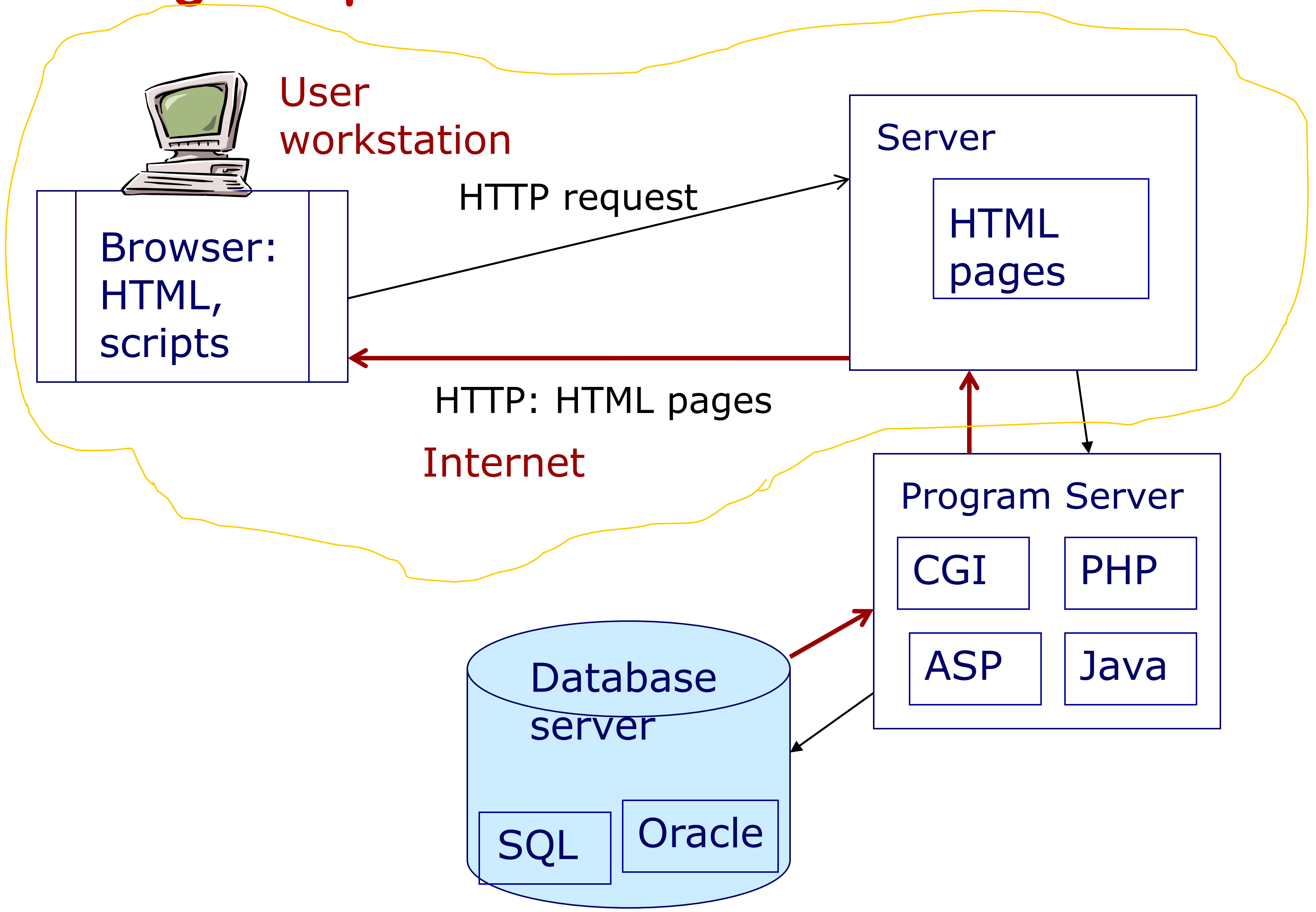
Jaana Holvikivi, DSc.
School of ICT

Helsinki Metropolia University of Applied Sciences

Scripts and styles on an HTML page



Page requests on the Web



HTTP requests

GET /index.html HTTP/1.1
Host: www.evtek.fi
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.3) Gecko/20070309 Firefox/2.0.0.3
a blank line *

- The client lists the Multipurpose Internet Mail Extension (MIME) types it will accept in return.
- Finally, the client sends a blank line indicating it has completed its request.

HTTP server response

```
HTTP/1.1 200 OK
Date: Mon, 09 Apr 2007 12:39:22 GMT
   Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Set-Cookie: fe_typo_user=4f74f6c85b; path=/;
   domain=www.evtek.fi
   Last-Modified: Wed, 08 Jan 2007 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

GET and POST methods

The difference between these two methods is in the way of sending data to the page:

- GET method sends data using URL (size limit),

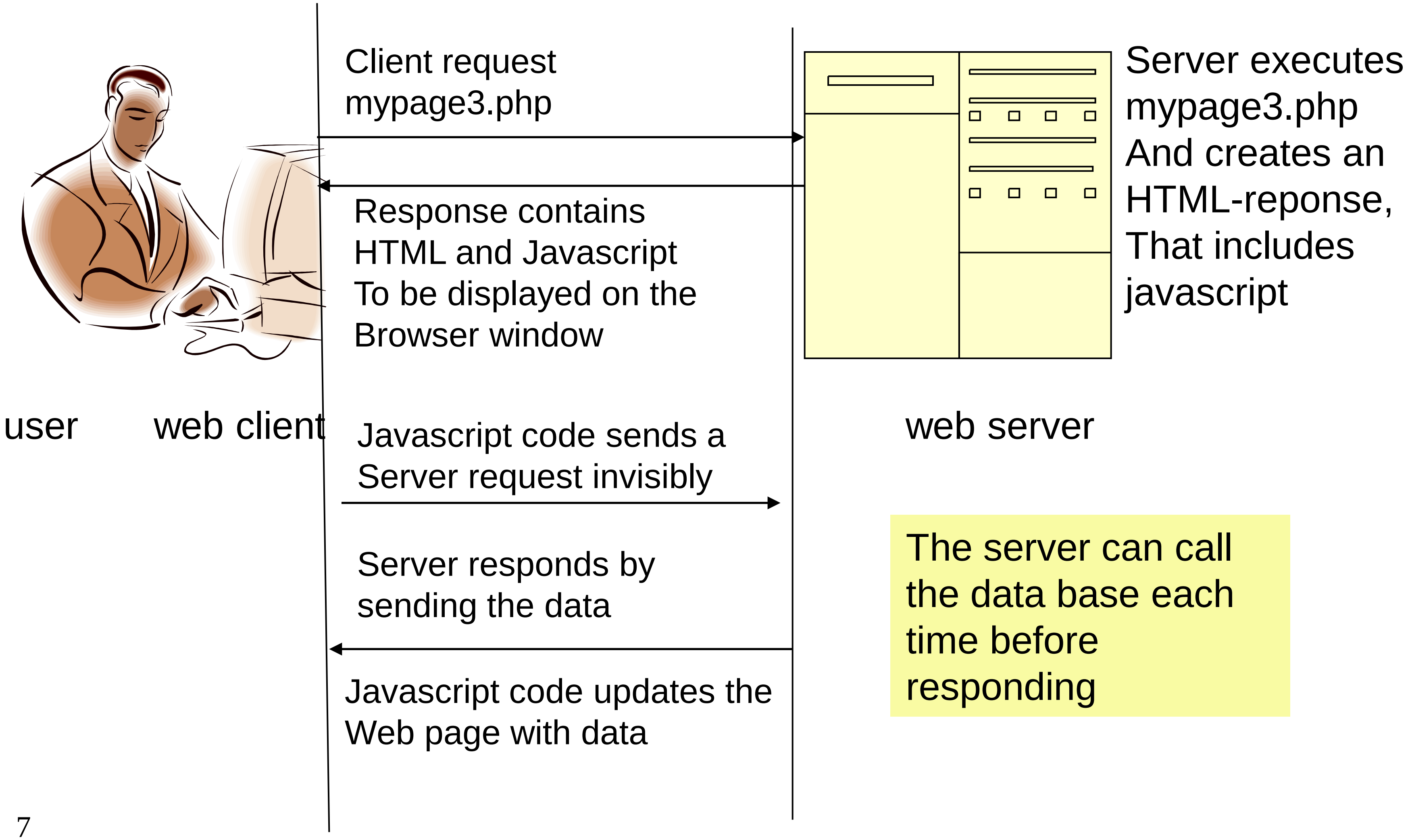
  <form method="GET" action="prog2.html">

http://www.google.com/search?sourceid=gmail&q=get%20method

- POST method sends data through a standard entrance

<form method="post" action="http://www.school.fi/cgi-bin/post-query">

A Client request with AJAX



Client request
mypage3.php

Response contains
HTML and Javascript
To be displayed on the
Browser window

Server executes
mypage3.php
And creates an
HTML-reponse,
That includes
javascript

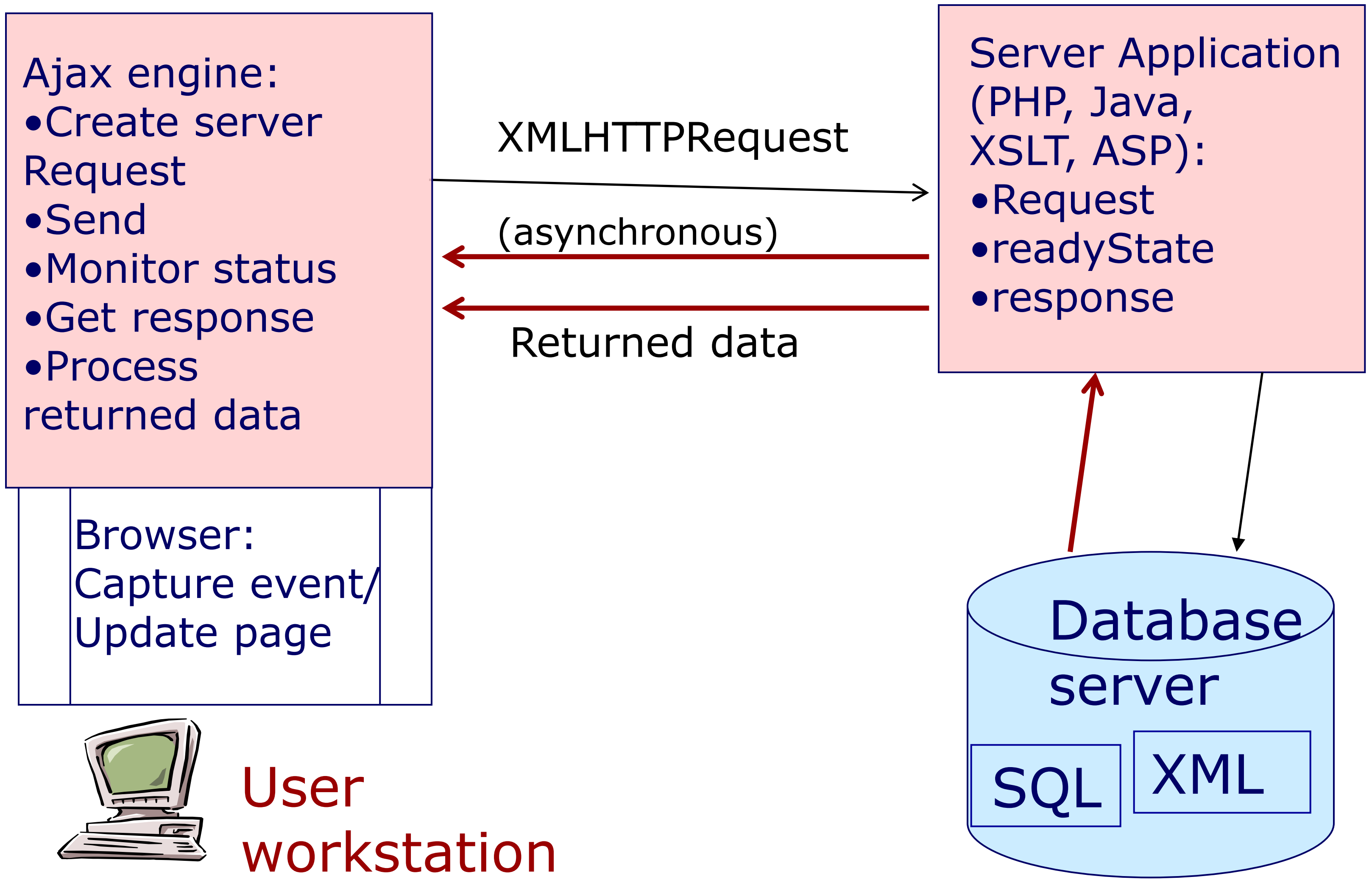
Javascript code sends a
Server request invisibly

Server responds by
sending the data

Javascript code updates the
Web page with data

The server can call
the data base each
time before
responding

Page requests: XMLHTTPRequest



Ajax components

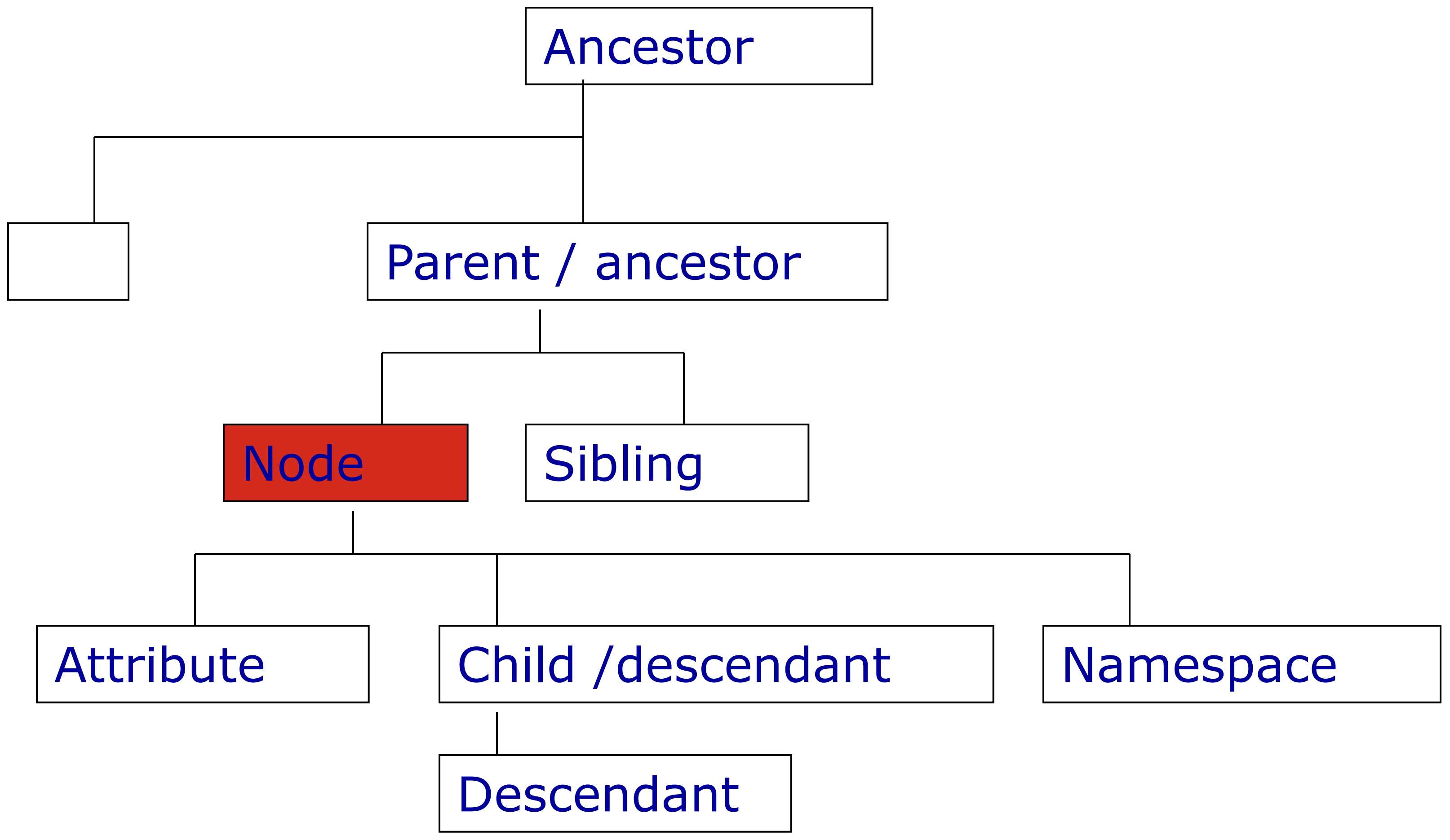
- Modern browsers
- Javascript & libraries
- XMLHttpRequest
- XHTML & CSS
- DOM
- XML
- Server side programs

Document Object Model (DOM)

- Used by many programming languages (php, Java, C#, C++, Javascript…)
- and understood by browsers (Firefox, IE, Chrome, Safari)

- XML -document is a collection of nodes that make a hierarchical tree structure
- The hierarchy is used in navigating the tree to locate information
- Inefficient use of memory: the tree structure is created in the memory
- DOM enables adding, moving, deleting and changing of nodes

Document tree



Processing of the tree

- Start with the root node (document-object)
- Element properties
  - firstChild
  - lastChild
  - nextSibling
  - parentNode
- Methods to navigate the tree
  - getElementByID(id)
  - getElementsByName(name)
  - getElementsByTagName(name)
  - getAttribute(name)

XML DOM

Note.xml

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this
weekend!</body>
</note>
```

From http://www.w3schools.com

```
<html><head>
<script type="text/javascript">
var xmlDoc;
function loadXML()
{
//load xml file
// code for IE6
if (window.ActiveXObject)
{
    xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
    xmlDoc.async=false;
    xmlDoc.load("note.xml");
    getmessage();
}
// code for Firefox, Opera, IE7+, Chrome, Safari
else if (document.implementation &&
    document.implementation.createDocument)
{
    xmlDoc=document.implementation.createDocument("","",null);
    xmlDoc.load("note.xml");
    xmlDoc.onload=getmessage;
}
else
{
    alert('Your browser cannot handle this script');
}
}
```

continues

```
function getmessage()
{
document.getElementById("to").innerHTML=xmlDoc.getElementsByTagNa
    me("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=xmlDoc.getElementsByTag
    Name("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=xmlDoc.getElementsB
    yTagName("body")[0].childNodes[0].nodeValue;
}
</script>
</head>

<body onload="loadXML()">
<h1>W3Schools Internal Note</h1>
<p><b>To:</b> <span id="to"></span><br />
<b>From:</b> <span id="from"></span><br />
<b>Message:</b> <span id="message"></span>
</p>
</body>
</html>
```

XMLHttpRequest

- XMLHttpRequest-object
  - Creation
  - Methods
  - Properties
- Server response handling
- Errors

Ajax request

- Client requests an event handler e.g. onclick=startaReq();
- XMLHttpRequest-object is created on client
- Callback handler is registered
- Start of request
  - httpReq.open(”GET”, stringA, true);
- Sending request
  - httpReq.send(null);
- Server executes the request and returns data to the client
- Client takes either text or xml response
  - httpReq.responseText
  - httpReq.responseXML

Helsinki Metropolia University of Applied Sciences

Creation of XMLHttpRequest

- Internet Explorer

```
if (window.ActiveXObject) {
        request = new ActiveXObject("Microsoft.XMLHTTP");
```

- Other browsers

```
if (window.XMLHttpRequest) {
        request = new XMLHttpRequest();
```

XMLHttpRequest properties

onreadystatechange	set up of callbackfunction
readystate	returns status of request:

- 0 = uninitialized
- 1 = loading
- 2 = loaded
- 3 = interactive
- 4 = complete

responseText	server response string
responseXML	server response XML document
Status	response status code
statusText	response status text

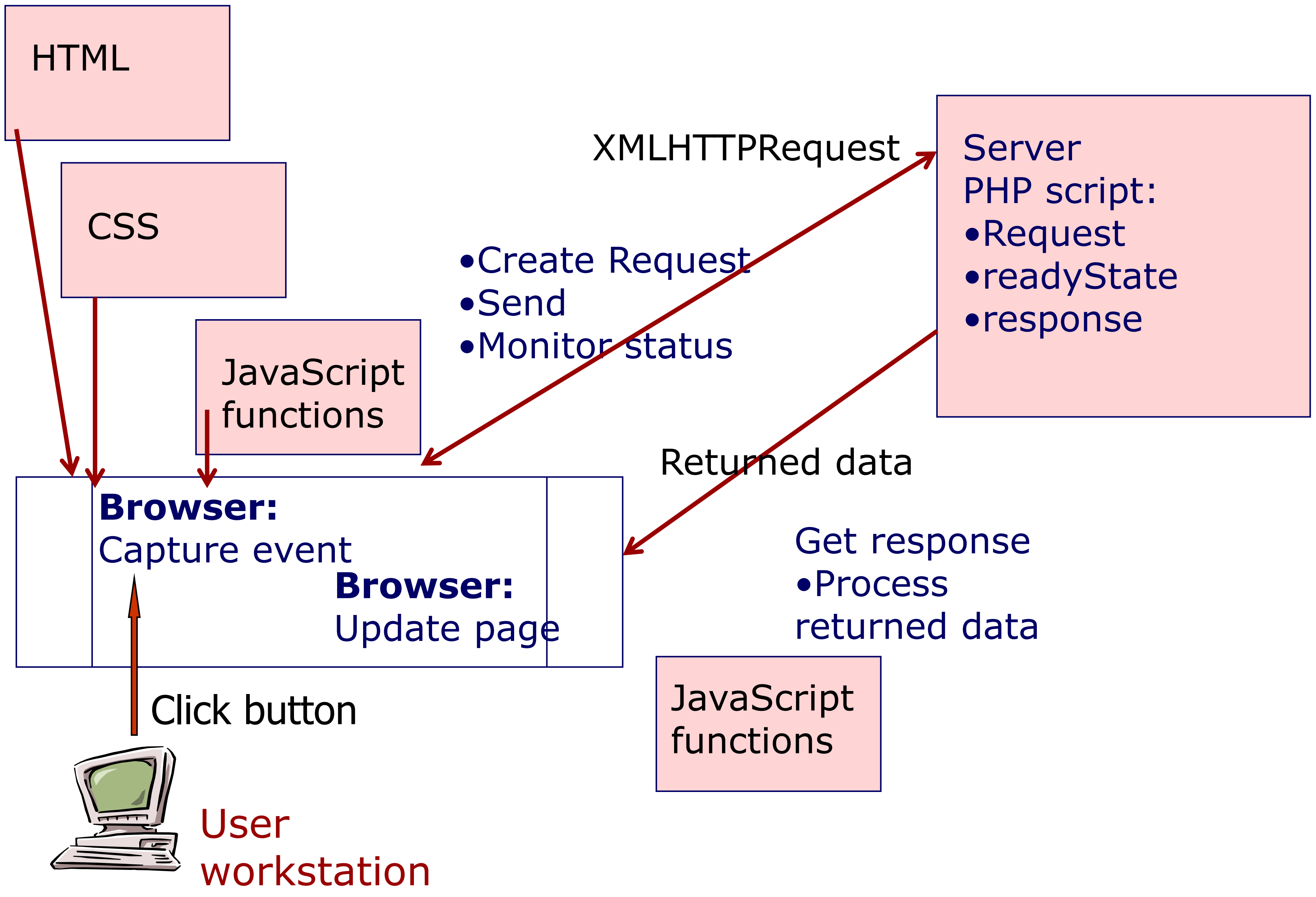
Document parsing error

```
function handleServerResponse()
{
  // read the message from the server
  var xmlResponse = xmlHttp.responseXML;
 // IE ja Opera
  if (!xmlResponse || !xmlResponse.documentElement)
           throw(”Invalid XML structure:\n”+xmlHttp.responseText);

// Firefox
  var rootNodeName = xmlResponse.documentElement.nodeName;
  if (rootNodeName == ”parsererror”)
           throw (”Invalid XML structure:\n”+xmlHttp.responseText);
  // obtain the XML's document element
  xmlRoot = xmlResponse.documentElement;
```

Helsinki Metropolia University of Applied Sciences

The browser in control



More Javascript

User defined functions

```
function Capitalize(mjono)
//return a string that has first capital letter
    {
    var firstletter, reststring, cap;
    firstletter = mjono.charAt(0);
    reststring = mjono.substring(1, mjono.length);
    cap= firstletter.toUpperCase() + reststring.toLowerCase()

    return cap;
}
```

Running this function gives the value of cap to mjono.

User defined functions

Printing output to an HTML-page:

```
<head>
<script>
function countdown()
{
    var count;
    count = document.getElementById("countBox").value;
    document.getElementById(“printing").value ="";
    while (count > 1){
    document.getElementById(“printing").value =
    document.getElementById(“printing").value + count + "\n";
    count = count -1;
}
    document.getElementById(“printing").value =
            document.getElementById(" printing ").value + "hep!";}
</script>
</head>
<body>
    <p>Give starting number for countdown:
    <input type ="text" id ="countBox" size = "3" value= "19"/>
    </p>
<p><input type ="button" value= ”Start countdown" onClick ="countdown();"/>
    </p>
<p>
    <textarea  id="printing" rows="20" cols="8">
    </textarea>
</p>
</body>
```

Loops: for

```
for (i = 0; i <= 10; i++)
{
    result += i;
    document.write(i + ": " + result + "<br/>");
}
```

Increment

    i=i+1 or   i++

Loops: while

```
var x = 1;
var result = 0;

while ( x <= 10 ) // repeated until x>10
{
result += x;
x++;
}

alert ("The result is " + result + " and x is " + x );
```

Nesting loops

```
var heads = 0, tails  = 0;
var i, j;
    for (j = 0; j <= 5;  j++)
    {
           for (i = 0; i < 100;  i++)
           {
           if ( Math.floor ((Math.random()*2)) == 1 )
             heads = heads + 1;
           else
             tails = tails + 1;
           }
           document.write ("Heads:  "+heads+ </br>
           "Tails  :  " +tails+ " "<p>");
           heads= 0; tails = 0;
    }
```