AJAX

Asynchronous JavaScript And XML

What is AJAX?

Asynchronous JavaScript and XML

- A way of developing faster, more interactive and responsive web applications.
- Term coined by Jesse James in 2005
- A type of programming made popular in 2005 by Google.
- Used as far back as 1998 in Internet Explorer (IE5) (Outlook Web Access).
- Not a "new" technology - A new way to use existing technologies: JavaScript, DOM, XML, HTML, CSS.
- Not a new programming language.

Why AJAX?

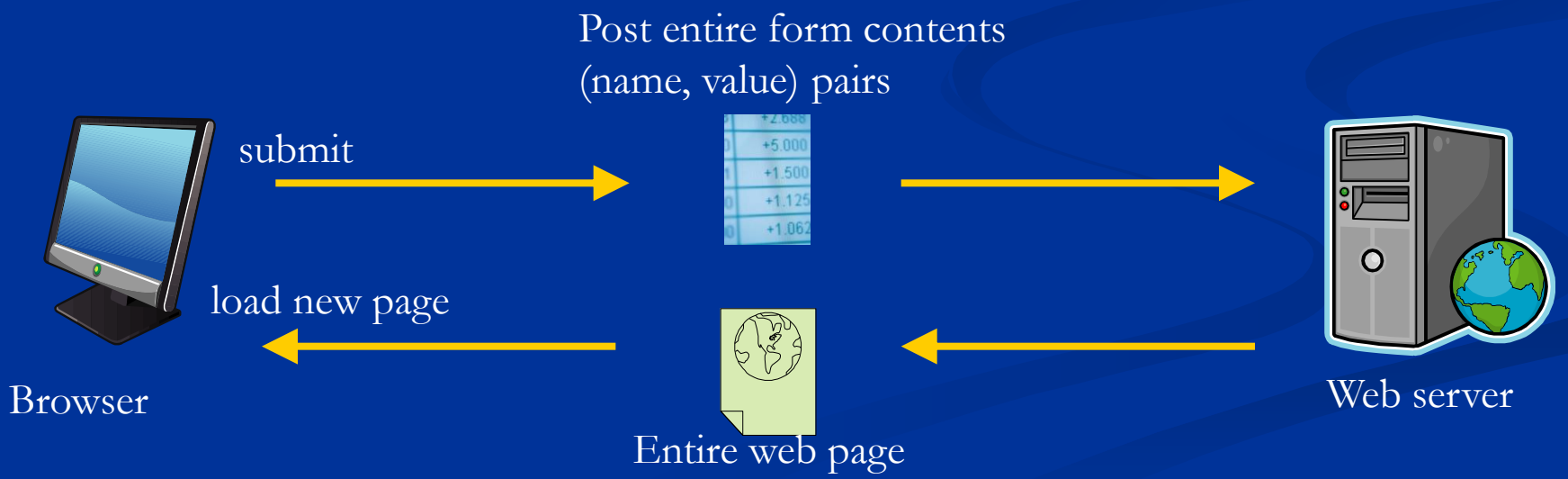
- Web based applications have many advantages, but …
- The user interface of web based applications have traditionally not been as rich and user friendly as desktop applications.
- AJAX allows the creation of web based user interfaces that are better, faster and more interactive/responsive .
  - Enables communication with the server and page updates without reloading the entire page – more user friendly.
  - Faster because we can download code and data on demand.
- **Asynchronous** is the key.

Support for AJAX

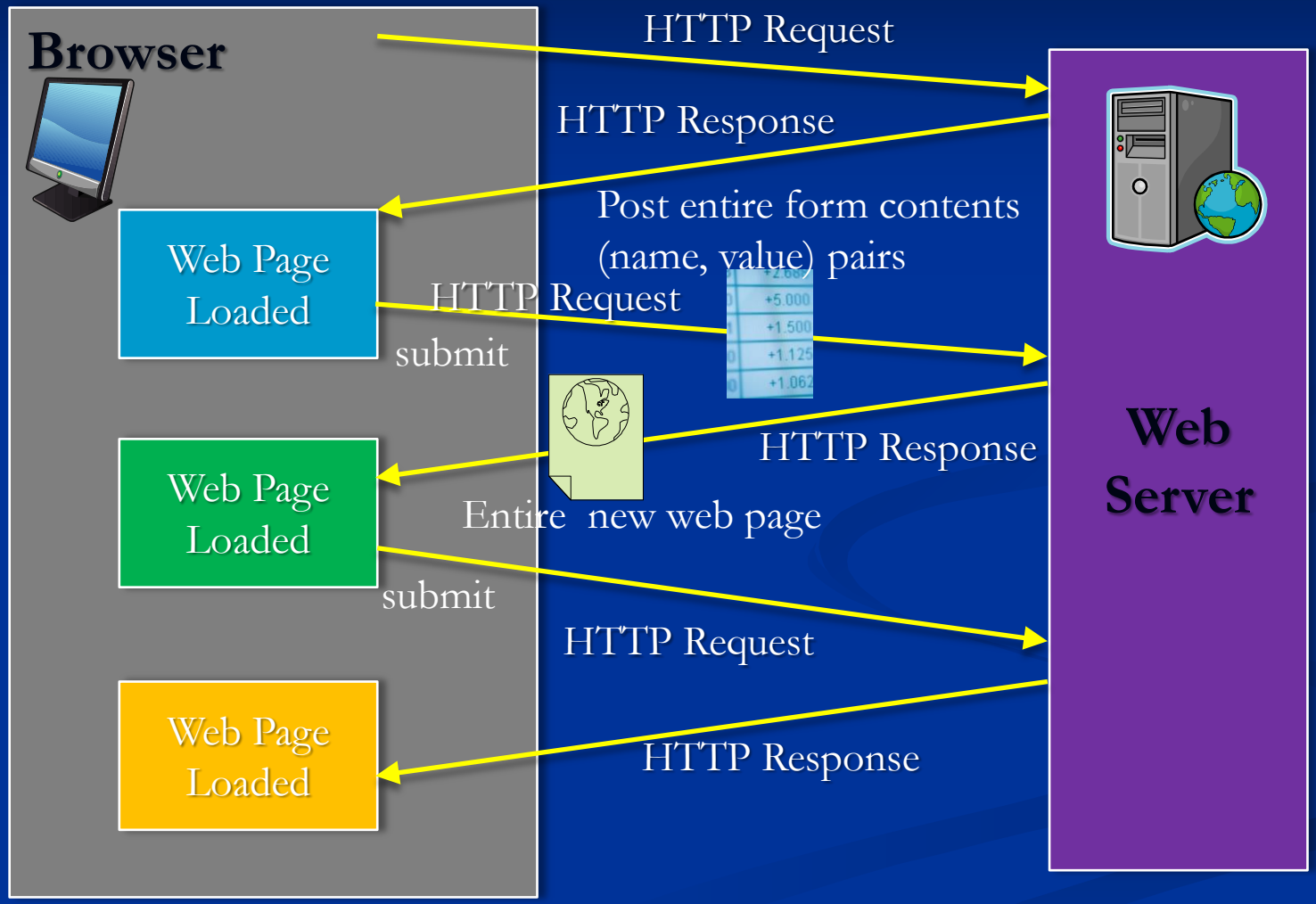
- The XMLHttpRequest object is supported in:
  - Internet Explorer 7.0+
  - Safari 1.2
  - Mozilla Firefox 1.0
  - Opera 8+
  - Netscape 7
  - Chrome
- Works with any server side technology.

Normal Web Page Lifecycle

- Client side submit action causes HTTP request message to be sent to the server with list of (name, value) pairs for all controls on the web form.
- Web server responds with an entirely new web page which must be loaded by the browser.

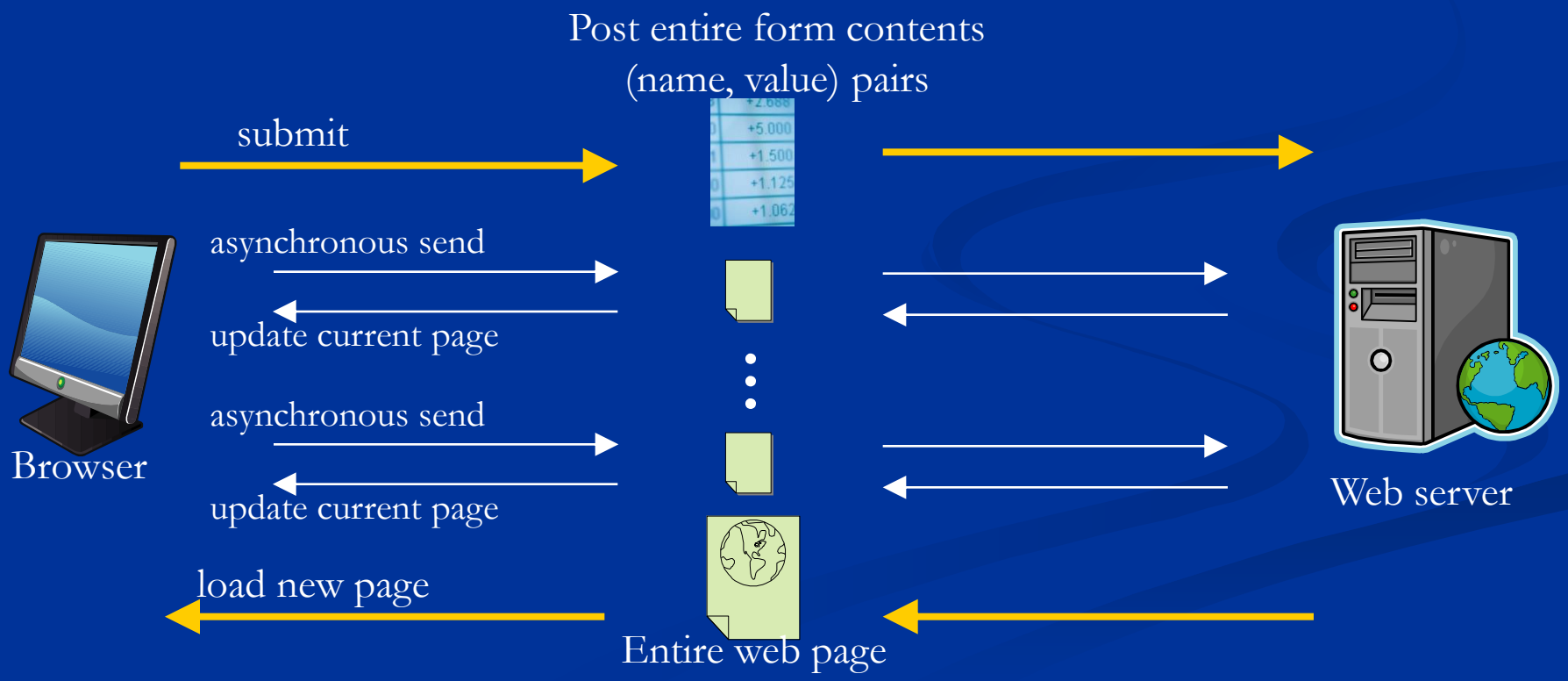


Normal Web Page Interaction

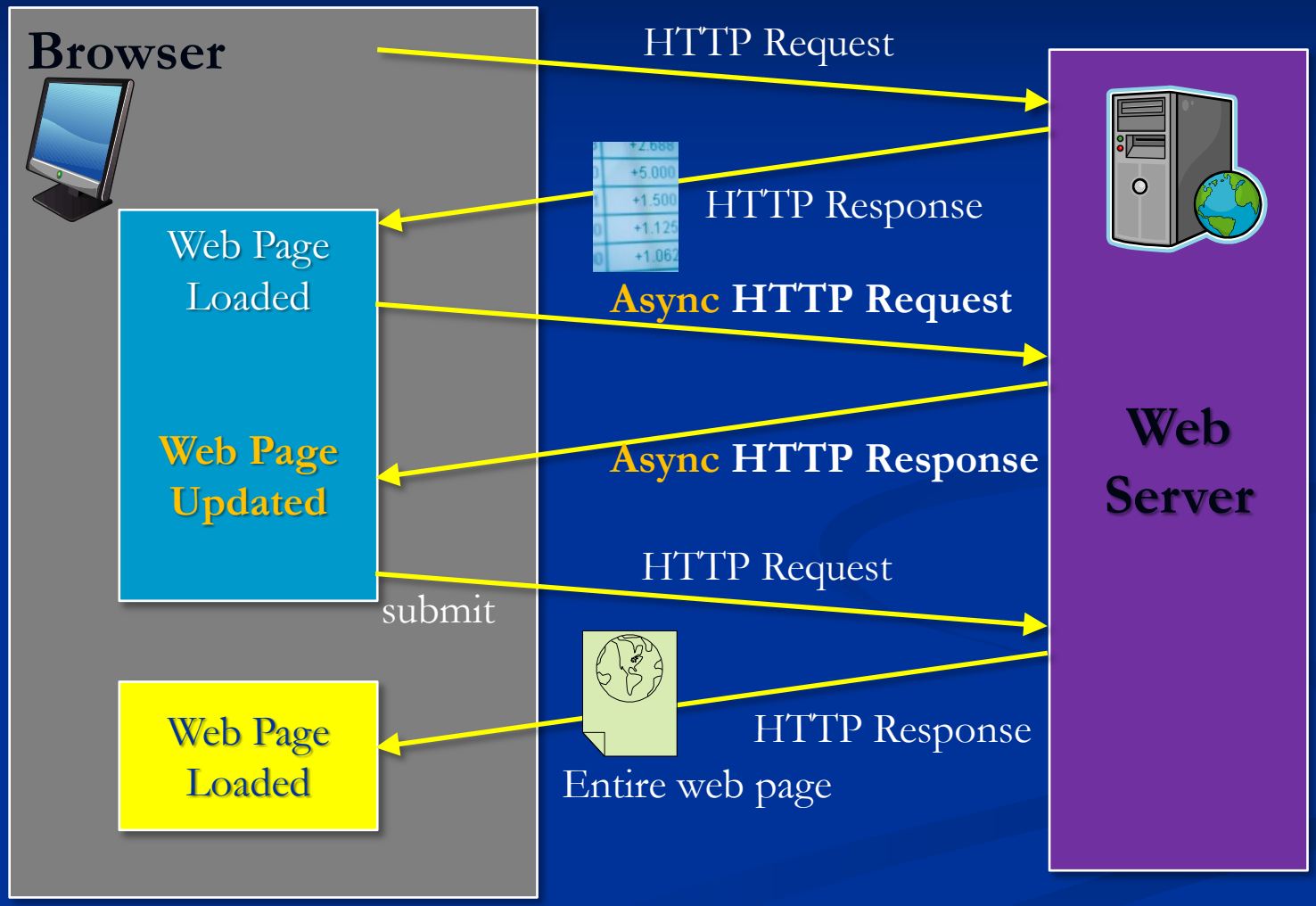


AJAX Web Page Lifecycle

- In between submit actions, browser can send requests to the server (asynchronously during normal user interaction with the web page).
- When the browser receives the response from the web server for this request it updates the current page (by using JavaScript and DOM) rather than loading a new page.



AJAX Interaction



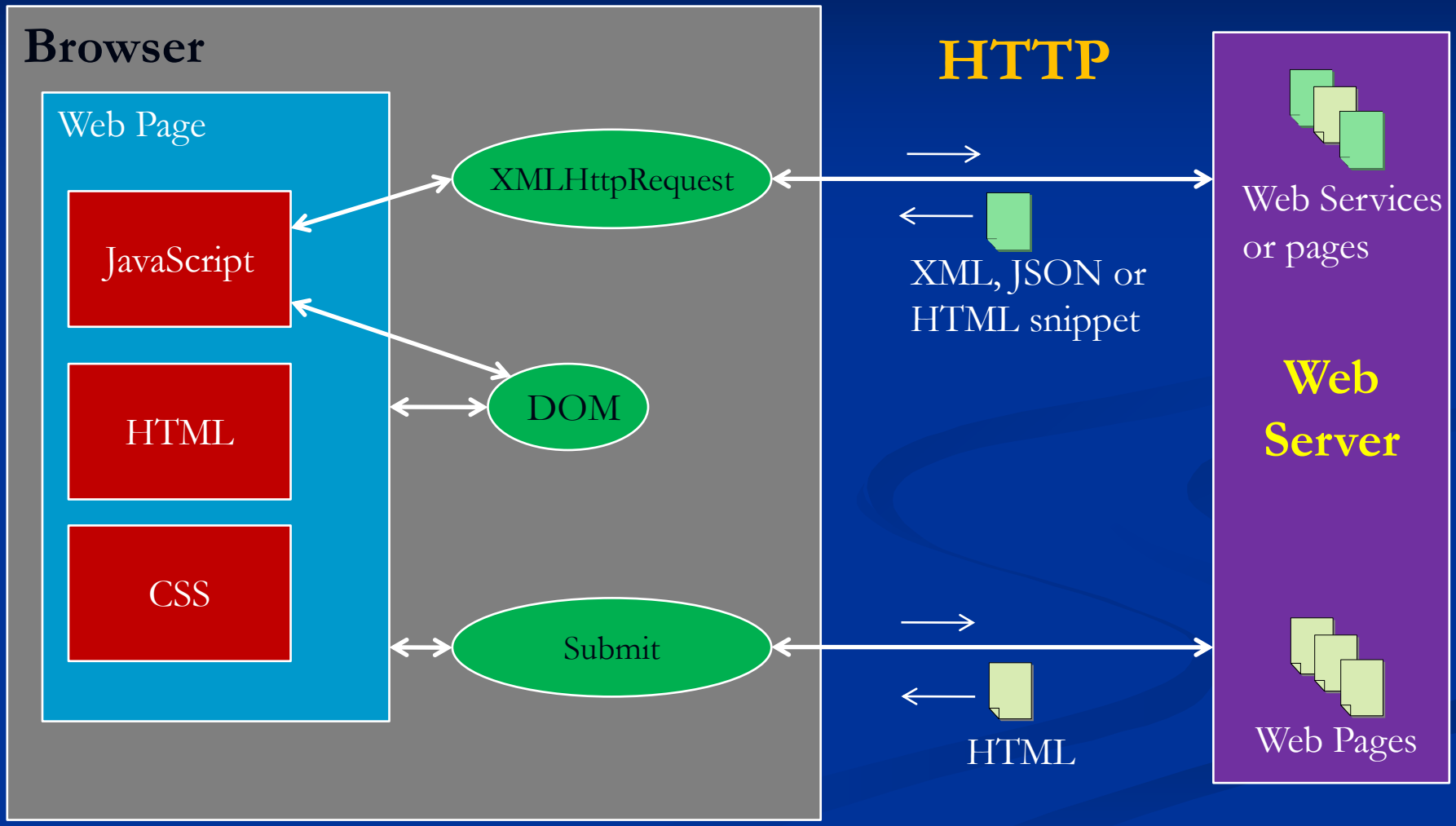
Asynchronous

- At some point the client-side JavaScript decides to send an (async) message to the server.
  - Normally in response to some client-side event
  - The message is sent, but the JavaScript doesn't halt and wait for a reply.
  - The user can continue to interact with the web page.
- At some point in the future we hopefully receive a reply from the server.
  - The client-side JavaScript processes the response via a callback function registered when the message was originally sent. The client-side JavaScript can use the contents of the response message to update parts of the current page.
- The client does not need to wait for a response to previous requests before issuing new async requests
  - So, multiple async requests can be in progress at any point in time – each with their own callback function.

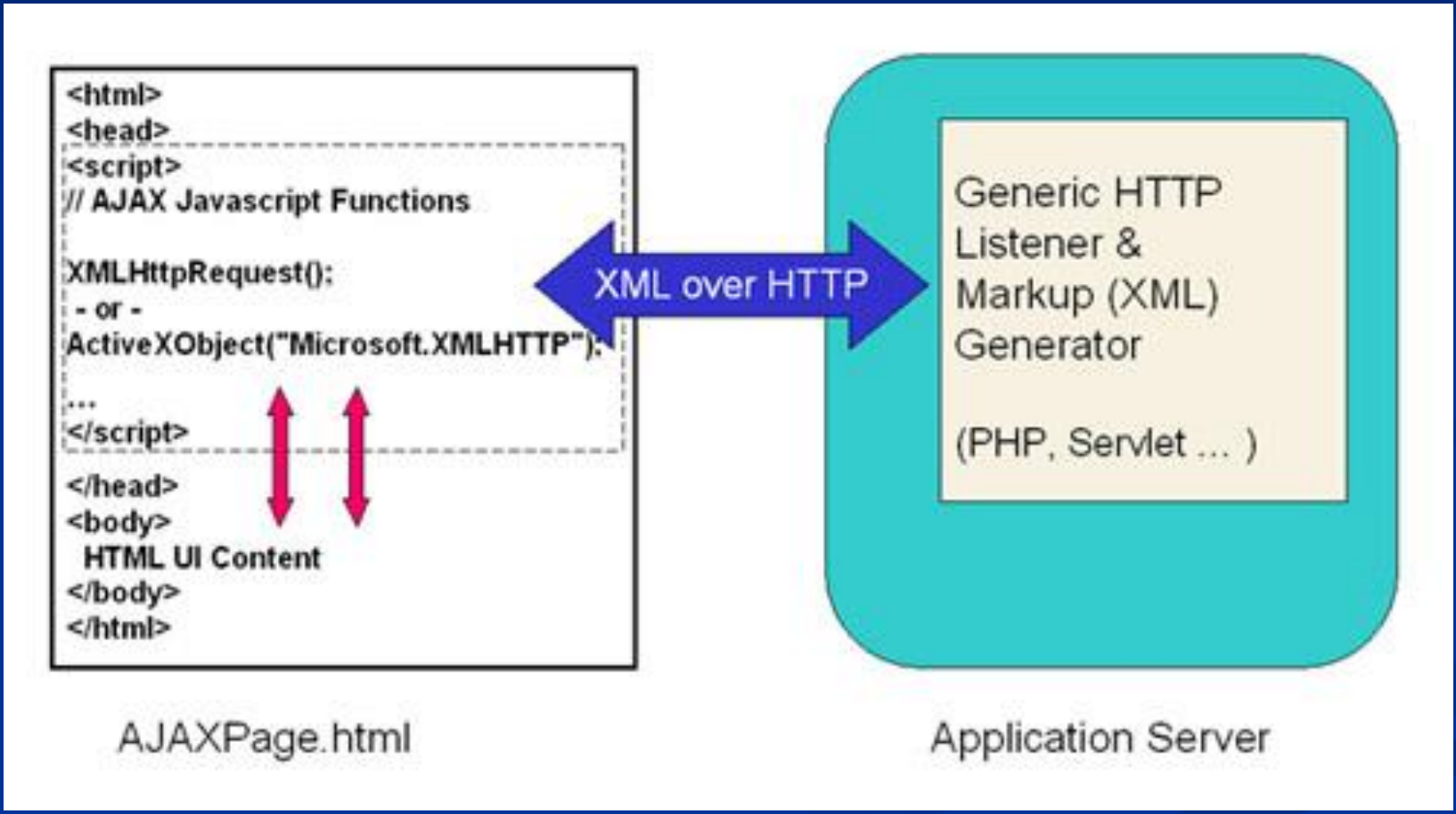
AJAX Technologies

- **XMLHttpRequest** Object
  - Requires browser capability/ support for asynchronous communication
- Client-side scripting **JavaScript**
  - To invoke asynchronous requests and process responses
- Client-side **DOM** (HTML Document Object Model)
  - To dynamically update web pages after loading by the web browser without reloading
- Client side HTML and CSS still used for rendering
- Server-side XML/JSON/HTML **Web Services**
  - Asynchronous request and response messages are often (but not always) encoded in XML.
  - To process and provide updates for the page
  - Any dynamic server side technology can be used for processing these new requests (JSP, ASP, CGI, PHP, etc).

AJAX Architecture



Core Ajax architecture



What Is Needed on the Client-Side ?

- A Javascript-enabled browser that supports either XMLHTTP or XMLHttpRequest objects
- An HTML page that contains:
  - UI elements that interact with Ajax Javascript functions
  - Javascript Functions that interact with Ajax server

What Is Needed on the Server-Side ?

- AJAX works with any server side technology.
- The server needs to be able to process http/https requests and respond with a stream of markup (XML/JSON) text.
- The XML/JSON is requested and returned to the client browser using standard http/https.

Server-Side Options

- Web Service or "Web Page"?
- Request may include:
  - HTTP parameters (list of name-value pairs) and/or
  - a message body containing XML or JSON
- Response message body contains:
  - POX (Plain Old XML), JSON or HTML (snippet)

- Usually not SOAP

Invoking Ajax Javascript Funcitons

- Responding to user interface interactions (like typing)
- Operate independently on their own timers.

Example Text filed

```
<input type="text" id="searchField"size="20" onkeyup="lookup('searchField');">
```

Creating XMLHttpRequest

- Firefox, Opera 8.0+, Safari, IE 7.0

```
new XMLHttpRequest();
```

- Internet Explorer 5, 6:

```
new ActiveXObject("Microsoft.XMLHTTP");
```

Microsoft.XMLHTTP maps to:

- Msxml2.XMLHTTP.2.6
- Msxml2.XMLHTTP.3.0
- Msxml2.XMLHTTP.4.0
- Msxml2.XMLHTTP.5.0
- Msxml2.XMLHTTP.6.0
- Msxml2.XMLHTTP.7.0

Creating XMLHttpRequest example

```
<html>
    <body>

    <script type="text/javascript">
    function ajaxFunction()
    {
    var xmlhttp;
    if (window.XMLHttpRequest)
      {
      // code for IE7+, Firefox, Chrome, Opera, Safari
      xmlhttp=new XMLHttpRequest();
      }
    else if (window.ActiveXObject)
      {
      // code for IE6, IE5
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
      }
    else
      {
      alert("Your browser does not support XMLHTTP!");
      }
    }
    </script>

    <form name="myForm">
    Name: <input type="text" name="username" />
    Time: <input type="text" name="time" />
    </form>

    </body>
</html>
```

AJAX Java Script

Browser Independence

HTML UI elements

Browser Independence

```
function  GetHttpRequest()
{
  if (window.XMLHttpRequest)
      return new XMLHttpRequest();
  else if (window.ActiveXObject)
      return new ActiveXObject("Microsoft.XMLHTTP");
}
```

XMLHttpRequest

- `open(method, URL [,async?, user, password])`
- `send([data])`
- `abort()`
- `readyState:  unsent = 0`
  `opened = 1`
  `headers_received = 2`
  `loading = 3`
  `done = 4`
- `onreadystatechange: (Event)`

http://www.w3.org/TR/XMLHttpRequest/

XMLHttpRequest Object Properties

- **onreadystatechange property**

  stores the function that will process the response from a server (callback function).

```
xmlhttp.onreadystatechange=function()
{
        if(xmlhttp.readyState==4)
                {
                document.myForm.time.value=xmlhttp.responseText;
                }
}
```

XMLHttpRequest Response Properties

- `readyState`
  - 0 uninitialized
  - 1 loading
  - 2 loaded
  - 3 interactive
  - 4 complete
- `responseText:` Response as a string
- `responseXML:` Response as XML document object
- `Status: int (HTTP status code)`
  - Response status as a number (eg 200 for "OK")
- `statusText: string (HTTP status text)`
  - Status as a string (eg "Not Found")
- `getResponseHeader(string): string`
- `getAllResponseHeaders(): string`

HTTP Status Codes

- 1xx Informational
  - 100 Continue
- 2xx Successful
  - 200 OK
- 3xx Redirection
  - 301 Moved Permanently
  - 307 Temporary Redirect
- 4xx Client Error
  - 400 Bad Request
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
- 5xx Server Error
  - 500 Internal Server Error
  - 501 Not Implemented
  - 503 Service Unavailable

Sending Asynchronous Requests

```
sender = GetHttpRequest();
sender.open("GET", http://myserver/path/file.ext, true);
sender.send(data);
```

Receiving Asynchronous Responses

```
function processXMLResponse()
{
   if (sender.readyState == 4)// XMLHTTPRequest is complete
   {
        if (sender.status == 200)  // XMLHTTPRequest  is Successful
         {
            // Process the XML response
         }
   }
   ...
}

sender = GetHttpRequest();
sender.open("GET", http://myserver/path/file.ext, true);
sender.onreadystatechanged = processXMLResponse;
sender.send(data);
```

Processing the XML response

- to extract the `balance` from the incoming XML stream:
- `<balance> 1000 </balance>`

- You can use the following:
- `var balance = sender.responseXML.getElementById("balance").nodeValue;`

Updating the HTML document

```
function processXMLResponse()
{
  if (sender.readyState == 4) //XMLHTTPRequest is complete

  {
    document.getElementById("balance").innerHTML =
      sender.responseXML.getElementById("balance").nodeValue;

    ...
  }
}
```

AJAX Example

Address: file:///C|/aranala2/login.php

```
215         <h1>Login as:</h1>
216         <form action="verify.php" method="post">
217             <input type="radio" id="parent"  name="cat" value="parent"  onclick="CategoryChanged()"/><b>Parent</b>
218             <input type="radio" id="athlete" name="cat" value="athlete" onclick="CategoryChanged()"/><b>Athlete</b>
219             <br/>
220             Surname: <input id="surname" name="surname" type="text" size="30">
221             <input type="button" value="Lookup" onClick="FetchMatchingFirstNames()">
222
223             <div id="FirstnamePlaceholder"/>
224
225             <div id="VerificationPlaceholder"/>
226         </form>
```

44K / 1 sec | Western European

```
78
79          function FetchMatchingFirstNames()
80          {
81              var surname = document.getElementById("surname").value;
82              var category;
83              if (document.getElementById("parent").checked)
84                  category = 'parent';
85              else
86                  category = 'athlete';
87              var URL = "names.php?surname=" + surname + "&category=" + category + "&timestamp=" + Date();
88              var callback = function() {
89                  var name = document.getElementById('name');
90                  if (name)
91                      name.focus();
92              }
93              AJAXRequest("FirstnamePlaceholder", URL, callback);
94          }
95
96          function AJAXRequest(elementId, URL, callback)
97          {
98              var xhr = getXMLHttpRequest();
99              xhr.onreadystatechange  = function()
100             {
101                 if(xhr.readyState  == 4)
102                 {
103                     if(xhr.status  == 200)
104                     {
105                         document.getElementById(elementId).innerHTML =  xhr.responseText;
106                         if (callback)
107                             callback();
108                     }
109                     else
110                         alert(xhr.status);
111                 }
112             };
113
114             xhr.open("GET", URL, true);
115             xhr.send(null);
116         }
```

Dealing with Older Browsers

```
47          function getXMLHttpRequest()
48          {
49              if (typeof XMLHttpRequest != "undefined")
50                  return new XMLHttpRequest
51
52              try
53              {
54                  return new ActiveXObject("Msxml2.XMLHTTP.6.0");
55              }
56              catch (e)
57              {
58              }
59
60              try
61              {
62                  return new ActiveXObject("Msxml2.XMLHTTP.3.0");
63              }
64              catch (e)
65              {
66              }
67
68              try
69              {
70                  return new ActiveXObject("Msxml2.XMLHTTP");
71              }
72              catch (e)
73              {
74              }
75
76              throw new Error("This browser does not support XMLHttpRequest.");
77          }
```

XML vs JSON
(JavaScript Object Notation)

```xml
<categories>
  <category id="0" name="Vegetables">
    <products>
      <product>
        <name>Onion</name>
        <price>.75</price>
      </product>
      <product>
        <name>Carrot</name>
        <price>.50</price>
      </product>
      <product>
        <name>Eggplant</name>
        <price>1.50</price>
      </product>
    </products>
  </category>
  <category id="1" name="Fruit">
    <products>
      <product>
        <name>Orange</name>
        <price>1.25</price>
      </product>
      <product>
        <name>Apple</name>
        <price>1.30</price>
      </product>
      <product>
        <name>Tomato</name>
        <price>.40</price>
      </product>
    </products
  </category>
</categories>
```

```json
[
    {"id": "0", "name":"Vegetables", "products":
    [{"name": "Onion", "price": .75},
    {"name": "Carrot", "price":.50},
    {"name": "Eggplant", "price":1.50} ]},

    {"id": "1", "name":"Fruit", "products":
    [{"name": "Orange", "price": 1.25},
    {"name": "Apple", "price":1.30},
    {"name": "Tomato", "price":.40} ]},
]
```

AJAX Libraries/Frameworks

- Provide:
  - Cross-browser support
  - DOM utilities and higher level abstractions
  - New widgets/controls
  - Animation
- Client Side frameworks
  - Libraries implemented in JavaScript and used by application JavaScript executing within the browser.
- Server Side frameworks
  - Server-side controls automatically inject AJAX functionality into generated client-side content.

Some AJAX Libraries/Frameworks

- Client Side frameworks
  - JQuery (http://jquery.com)
  - Prototype (http://www.prototypejs.org)
  - Dojo (http://dojotoolkit.org/)
  - Yahoo UI (http://developer.yahoo.com/yui)
- Server Side frameworks
  - ASP.NET AJAX (http://ajax.asp.net)
  - Google Web Toolkit (http://code.google.com/webtoolkit/)
  - Java Server Faces with AJAX

  http://docs.oracle.com/javaee/6/tutorial/doc/gkiow.html

AJAX Library Issues

- Library vs Framework? (e.g. Dojo Framework)
- Language? (e.g. Google Web Toolkit = Java)
- Library size?
  - Download/start-up time.
  - Modular? Only download what you need?

ASP.NET AJAX

- Client-side JavaScript AJAX Libraries
  - Microsoft supports and ships JQuery with VS.
  - Microsoft also have their own AJAX JavaScript library
- AJAX Server Controls
  - ScriptManager, UpdatePanel, UpdateProgress, Timer
  - Partial Page Rendering
- AJAX Control Toolkit
  - http://www.asp.net/ajax/ajaxcontroltoolkit/samples/
  - Accordion, Animation, Calendar, HTMLEditor, ListSearch, NumericUpDown, RoundedCorners, Slider, SlideShow, ...

AJAX Server Controls Example

Client Objects & Events (No Events)

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>AJAX with ASP.NET</title>
</head>
<body>
    <form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server"/>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
        <ContentTemplate>
            <p id="messagewindow">
                <asp:Repeater ID="Repeater1" DataSourceID="SqlDataSource1" runat="server" EnableViewState="False">
                    <ItemTemplate>
                        <b><%# Eval("user") %></b>:<%# Eval("msg") %><br />
                    </ItemTemplate>
                </asp:Repeater>
                <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ConnectionString2 %>"
                    SelectCommand="SELECT Top 10 [user], [msg] FROM messages WHERE time &gt; 0 ORDER BY id DESC">
                </asp:SqlDataSource>
                <asp:Timer ID="Timer1" runat="server" Interval="4000">
                </asp:Timer>
            </p>
        </ContentTemplate>
        <Triggers>
            <asp:AsyncPostBackTrigger ControlID="OKButton" />
        </Triggers>
    </asp:UpdatePanel>
    <asp:UpdateProgress ID="UpdateProgress1" runat="server" AssociatedUpdatePanelID="UpdatePanel1">
        <ProgressTemplate>
            Loading ...
        </ProgressTemplate>
    </asp:UpdateProgress>
    Name: <input type="text" id="author" runat="server" />
    Message: <input type="text" id="msg" runat="server" />
    <asp:Button ID="OKButton" runat="server" Text="ok" OnClick="Button1_Click" />
    </form>
</body>
</html>
```

UpdatePanel server control

Content of Update

Event that triggers the update

Difficulties for AJAX

- Page history and book marking.
- Accessibility for vision impaired.
- Harder to implement:
  - Potentially lots of messy JavaScript and DOM manipulation.
  - Coping with variation between browsers.

References

- http://en.wikipedia.org/wiki/Ajax_framework
- http://en.wikipedia.org/wiki/List_of_Ajax_frameworks
- http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks
- http://jqueryui.com/demos/
- http://jquery.com/
- http://www.asp.net/ajax
- http://video.google.com/videoplay?docid=7503545790237259771#
- http://ajaxpatterns.org/
- http://msdn.microsoft.com/en-us/library/bb398874.aspx
- http://www.asp.net/ajax/ajaxcontroltoolkit/samples/
- http://weblogs.asp.net/scottgu/archive/2008/09/28/jquery-and-microsoft.aspx
- http://msdn.microsoft.com/en-us/magazine/cc163300.aspx
- http://msdn.microsoft.com/en-us/magazine/dd722809.aspx