

CSE 135

Client-Side Programming Ajax (unaided)

Interactive Applications Using Ajax

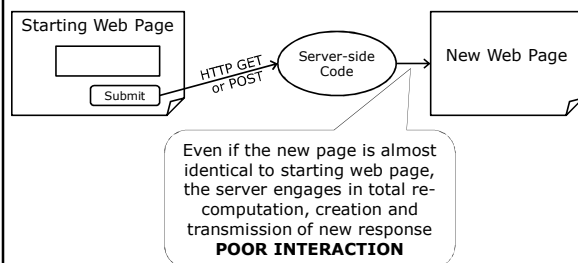
- Ajax is a set of technologies that collectively enable interactive applications
 - Hallmark: Part of a page changing asynchronously
- Ajax = Asynchronous JavaScript and XML
- Extensive use in Gmail, Yahoo! Finance, etc.

Only one really new primitive:

- JavaScript uses a (client side) **XMLHttpRequest** to asynchronously communicate with the server
- <http://www.w3.org/TR/XMLHttpRequest/>

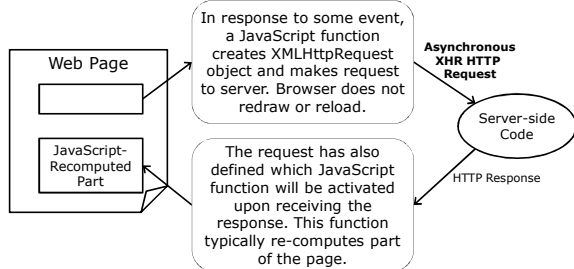
So Far in Class...

Client Communicates Synchronously with Server



Ajax

Client Communicates Asynchronously with Server



4

New JavaScript Material

- You can assign functions to variables and object properties
 - We will assign the function that handles the HTTP response to a property of XMLHttpRequest object

5

Ajax Example 1

```
<body>
  Write your story here:
  <form action="nowhere" method="GET">
    <p/>
    <textarea rows="10" cols="80" name="story"
      onkeyup="lastTimeFunction();" />
    </textarea>
    <p/>
    <span id="lastTime">
      You have not typed anything in the above box yet
    </span>
  </form>
</body>
```

6

date.jsp used (that's all)

```
<%= new java.util.Date() %>
```

7

Ajax Example 1 (cont'd)

```
function lastTimeFunction() {  
    var xmlhttp = new XMLHttpRequest;  
  
    var responseHandler = function() {  
        if (xmlhttp.readyState == 4)  
            document.getElementById("lastTi  
                "You last typed on " + xmlhttp  
        }  
        xmlhttp.onreadystatechange = responseHandler;  
  
        xmlhttp.open("GET", "date.jsp", true)  
        xmlhttp.send(null);  
    }  
}
```

The value of the variable is a function and NOT the result of a function call (think of C++ function pointers)

Call the function stored in this property whenever the server produces the HTTP response

3rd argument is asynchronous communication flag (versions with user & password also avail)

Initiates request. If it was POST, argument would be body

8

XMLHttpRequest

The **readyState** Property of XMLHttpRequest

- 0 : request not initialized yet
- 1 : request is set up
- 2 : request has been sent
- 3 : request is in process
- 4 : request is complete

9

XMLHttpRequest (cont'd)

```
// request
• open(DOMString method, DOMString url, boolean async,
  DOMString? user, DOMString? password);
• setRequestHeader(DOMString header, DOMString value);

// response
• unsigned short status;           // holds the HTTP status code
• DOMString statusText;           // holds the HTTP status text
• DOMString responseText;         // DOMString
• Document responseXML;           // Document
```

10

Browser Compatibility

```
var xmlhttp;
try {
  xmlhttp = new XMLHttpRequest(); // Firefox, Opera, Safari
} catch (e) {
  // Internet Explorer
  try {
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    try {
      xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (e) {
      alert("Your browser does not support Ajax!");
      return false;
    }
  }
}
```

11

How to transfer complex data from the server to the client? (Strings not enough) The 'x' in Ajax : XML

```
<html>
<head><script src="selectCustomerXML.js"></script></head>
<body>
  <form action="">
    Select a Customer:
    <select name="custs" onchange="showCust(this.value)">
      <option value="AV">Art Vandelay</option>
      <option value="JP">Jim Progress</option>
      <option value="ND">Nick Dummy</option>
    </select>
  </form>
  <b><span id="company"></span></b><br />
  <span id="contact"></span><br />
  <span id="address"></span>
  <span id="city"></span><br />
  <span id="country"></span>
</body>
</html>
```

12

```
function showCust(str) {
    var xmlhttp = new XMLHttpRequest();
    var url="getCustomerXML.jsp";
    url = url + "?q=" + str;
    xmlhttp.onreadystatechange = stateChanged;
    xmlhttp.open("GET", url, true);
    xmlhttp.send(null);
}
```

13

An example XML response by the server to the client

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<company>
  <comp>Vandelay Industries</comp>
  <cont>Inc.</cont>
  <addr>9500 Gilman Drive</addr>
  <city>La Jolla</city>
  <cntr>USA</cntr>
</company>
```

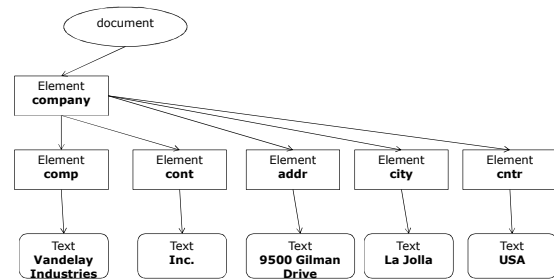
14

A dummy jsp producing XML

```
<% response.setContentType("text/xml") ; %>
<%
String person=request.getParameter("person") ;
if (person!=null && person.equals("AV"))
{
    %>
    <company>
      <compname>Vandelay Industries</compname>
      <contname>Inc.</contname>
      <address>9500 Gilman Drive</address>
      <city>La Jolla</city>
      <country>USA</country>
    </company>
    <%
    }
else if (person!=null && person.equals("JP"))
{
    %>
    <company>
      <compname>Acme Industries</compname>
```

15

How to access XML: It is not just text; it is a DOM object



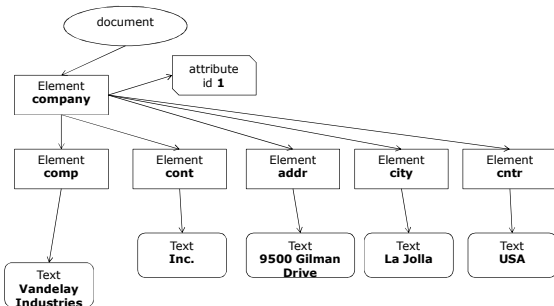
16

An example including attributes

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<company id='1'>
  <comp>Vandelay Industries</comp>
  <cont>Inc.</cont>
  <addr>9500 Gilman Drive</addr>
  <city>La Jolla</city>
  <cntr>USA</cntr>
</company>
```

17

How to access XML: It is not just text; it is a DOM object



18

The response handler code navigates in the XML DOM object to pick data

```
function stateChanged() {
  if (xmlHttp.readyState==4) {
    var xmlDoc=xmlHttp.responseXML.documentElement;
    document.getElementById("company").innerHTML =
      xmlDoc.getElementsByTagName("comp")[0].childNodes[0].nodeValue;
    document.getElementById("contact").innerHTML =
      xmlDoc.getElementsByTagName("cont")[0].childNodes[0].nodeValue;
    document.getElementById("address").innerHTML =
      xmlDoc.getElementsByTagName("addr")[0].childNodes[0].nodeValue;
    document.getElementById("city").innerHTML =
      xmlDoc.getElementsByTagName("city")[0].childNodes[0].nodeValue;
    document.getElementById("country").innerHTML =
      xmlDoc.getElementsByTagName("cntr")[0].childNodes[0].nodeValue;
  }
}

xmlDoc.getElementsByTagName("comp")[0]
xmlDoc.getElementsByTagName("comp")[0].childNodes[0]
xmlDoc.getElementsByTagName("comp")[0].childNodes[0].nodeValue
```

Picks the document node

Picks the first element e whose tag is 'comp'

Picks the first child t of e

Picks the value of t

The Ajax page incremental update pattern

Browser

HTML
DOM

JavaScript
Components

Event triggers JavaScript f

Collect arguments from page
Optionally make some page changes
Make XHR object
Define response handler h in XHR

```
function h //the response handler
Receive string or XML or JSON
describing diff
Change HTML DOM or Javascript
components to reflect diff
```

XHR request to jsp j prepping diff

Application Server

Jsp j that prepares the diff

Book-keeping, change session/db state

Compute the diff (i.e., data that describe how the page changes).
Format as string, XML, JSON

Students example revisited: XHTML compliance & why it matters

Our Step 1 code was not XHTML compliant (form element directly within tr).
Most browsers get confused by non-XHTML compliant incremental updates.

```
<table id="studentsTbl" border="1">
  <tbody id="students">
    <tr>
      <th>ID</th>
      <th>PID</th>
      <th>First Name</th>
      <th>Middle Name</th>
      <th>Last Name</th>
    </tr>
    <tr>
      <th>&nbsp;</th>
      <th><input id="pid" value="" name="pid" size="10"/></th>
      <th><input id="first" value="" name="first" size="15"/></th>
      <th><input id="middle" value="" name="middle" size="15"/></th>
      <th><input id="last" value="" name="last" size="15"/></th>
      <th><input onClick="studentAction(null,'insert');" type="button" value="Insert"/></th>
    </tr>
```

Parameter collection: id's for the insert part and calls to “f” studentAction

```
<tr>
  <th>&nbsp;</th>
  <th><input id="pid" value="" name="pid" size="10"/></th>
  <th><input id="first" value="" name="first" size="15"/></th>
  <th><input id="middle" value="" name="middle" size="15"/></th>
  <th><input id="last" value="" name="last" size="15"/></th>
  <th><input onClick="studentAction(null,'insert');" type="button" value="Insert"/>
</tr>

<td><input onClick="studentAction(<%= id %>,'update');" type="button" value="Update">
```

22

Parameter collection: Synthesize id's that will make it easy to spot updated values during parameter collection

```
<tr id="<%= ((Student)pair.getValue()).getPID() %>">
<td> <%= id %> </td>
<td> <input id="pid_<%= id %>" value="<%= ((Student)pair.getValue()).getPID() %>"
  name="pid" size="15"/>
</td>
<td>
<input id="first_<%= id %>" value="<%= ((Student)pair.getValue()).getFirstName() %>"
  name="first" size="15"/>
</td>
```

23

Parameter collection: the Javascript part

```
function studentAction(id, actionType) {
  var params = new Array();
  var pid = null;

  var elements = ["pid", "first", "middle", "last"];

  for (var i = 0; i < elements.length; i++) {
    var key = elements[i];
    var ele = key;

    if (id != null) ele = ele + "_" + id;

    var element = document.getElementById(ele);
    var value = element.value;

    if (actionType === "insert") element.value = null;

    if (key === "pid") pid = value;
    params.push(key + "=" + value);
  }

  if (id != null) params.push("id=" + id);
  params.push("action=" + actionType);
  makeRequest(params, actionType, pid);
}
```

24

```

var response = eval('(' + responseDoc + ')');
switch (actionType){
  case "insert":
    if (response.success) {
      var table = document.getElementById("studentsTbl");

      var row = table.insertRow(table.rows.length);
      row.id = response.pid;

      var html = '<td>' + response.id + '</td><td><input id="pid_' + response.id + '" value='
      row.innerHTML = html;
      document.getElementById('response').innerHTML = "Insert Complete";
    }
}

```

25

JavaScript Object Notation (JSON)

- Self-describing notation (as XML), tuned for Javascript
- Offered by Groupon, Twitter etc

Attribute name Array value

Object start Object end

Text value

Number value

```

{"movielist": ["Friday the 13th",
               "Friday the 13th Part 2",
               "Friday the 13th Part III",
               "Friday the 13th: The Final Chapter",
               "Friday the 13th: A New Beginning"]}
{"reviewer": "Pam", "stars": 3, "text": "Pretty good, but could have used more Jason"}
{"title": "Friday the 13th",
 "year": 1980,
 "reviews": [{"reviewer": "Pam", "stars": 3, "text": "Pretty good, but could have used more Jason"},
              {"reviewer": "Alice", "stars": 4, "text": "The end was good, but a little unsettling"}
            ]
}

```

26

Receiving a JSON response and converting it into a JavaScript object

```

// assume server has sent a JSON-encoded text
var responseDoc = xmlhttp.responseText;
var response = eval('(' + responseDoc + ')');

```

27

Creating a JSON response in Java

```
{ "success" : true,  
  "id" : 1,  
  "pid" : 2222,  
  "first" : "Yannis",  
  "middle" : "G",  
  "last" : "Papakonstantinou"  
}  
  
JSONObject result = new JSONObject();  
result.put("success",true);  
result.put("id",pair.getKey());  
result.put("pid",pid);  
result.put("first",student.getFirstName());  
result.put("middle",student.getMiddleName());  
result.put("last",student.getLastName());  
out.print(result);
```

28

Common Use Cases of Ajax

- Today's busy pages have multiple almost independent sections
- Reduce load by updating only the relevant section
- Quick response to inputs
 - "Illusion" that the page is faster even when it is not, simply because there is always something on screen

29

Common Ajax Downsides

- The "revisions" do not automatically register with browser's history
 - Back button behaves weirdly
- GET is good for bookmarking; Ajax is bad
 - Mitigation: fragment identifier
- Non-crawlable web
- New opportunities for malicious hackers
- Complicates structure
- As of now, very poor interaction with server-side frameworks (Struts, Spring) and their custom tags
 - Ruby's partial templates probably the best extension of an MVC compliant framework for Ajax
- As of now, poor interaction with non-HTML-standard features supported by browsers

30

Why is Ajax so hard?

- Ajax introduces difficulties of distributed computation
 - part of computation in Javascript, part in Java
 - Need to make requests from Javascript to server, vice versa “package” results of small server-side functions into http response and XML
- Ajax computations are often not fully compatible across browsers
 - Eg, “hacks” that operate on load, do not operate after an Ajax update of the page
- Half-baked, pre-paradigmatic technology

31

The difficulties of (unaided) Ajax

- The students example revisited to include Ajax
 - Fasten your seatbelts, it will be a bumpy ride
 - Browser oddities revealed, low level coding
- Project advice
- Ajax frameworks
- Make a contribution to the web & db lab’s FORWARD framework for Ajax applications
 - Natl Science Foundation REU support possible

32
