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

Starting Web Page → Server-side Code → New Web Page

Even if the new page is almost identical to the starting web page, the server engages in total re-computation, creation and transmission of new response.

POOR INTERACTION

Ajax

Client Communicates Asynchronously with Server

In response to some event, a JavaScript function creates XMLHttpRequest object and makes request to server. Browser does not redraw or reload.

The request has also defined which JavaScript function will be activated upon receiving the response. This function typically re-computes part of the page.
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

Ajax Example 1

```html
<body>
  Write your story here:
  <form action="nowhere" method="GET">
    <p>
      <textarea rows="10" cols="80" name="story"
        onkeyup="lastTimeFunction();">
      </textarea>
    </p>
    <p>
      <span id="lastTime">
        You have not typed anything in the above box yet
      </span>
    </p>
  </form>
</body>
```
Ajax Example 1 (cont’d)

```javascript
function lastTimeFunction() {
    var xmlHttp = new XMLHttpRequest;

    var responseHandler = function() {
        if (xmlHttp.readyState == 4) {
            document.getElementById("lastTime").innerHTML = "You last typed on " + xmlHttp.responseText;
        }
    }

    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

---

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
**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

---

**Browser Compatibility**

```javascript
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;
        }
    }
}
```
Enter XML: Auto-Completion of Contact Info

```html
<html>
<head><script src="selectCustomerXML.js"></script></head>
<body>
<form action="">
   Select a Customer:
   <select name="custs" onchange="showCust(this.value)"/>
      <option value="ALF">Alfred</option>
      <option value="JOH">John</option>
      <option value="WOL">Wolski</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>
```

Enter XML: Auto-Completion of Contact Info (cont’d)

```javascript
function showCust(str) {
   var xmlHttp = new XMLHttpRequest();
   var url="getCustomerXML.jsp";
   url = url + "?q=" + str;
   url = url + "&sid=" + Math.random();
   xmlHttp.onreadystatechange = stateChanged;
   xmlHttp.open("GET", url, true);
   xmlHttp.send(null);
}
```
Enter XML: Auto-Completion of Contact Info (cont’d)

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;
    }
}

Navigation into XML result using the XML Document Object Model (DOM)
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

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
  - Can be resolved with fragment identifier
- Non-crawlable web
- New opportunities for malicious hackers
- Complicates structure
A little memo and questions...

Browser side

- Browser catches event A on element E of the page
- The HTML may dictate that JavaScript function f is executed upon event A on E
- The function f modifies the HTML DOM
- Browser refreshes and shows revised page
- Ajax/XHR further allow JavaScript function to contact server and obtain new data

HTML DOM

What Should You Use When...

- Your application displays a table and you want the background of the rows of this table to turn red when you mouse over them
- Comment on pros and cons of:
  - Plain JavaScript
  - JavaScript communicating via XHR to the server which sends a revised “red row” and JavaScript changes old row with new row
What Should You Use When...

- Your application’s page shows a table with today’s College Basketball games with an update button next to each one of them.
- Upon clicking the update button, new rows appear showing the latest events of the particular game.