CITS1231 Web Technologies
Ajax and Web 2.0
Turning clunky website into interactive mashups
What is Ajax?

• Shorthand for Asynchronous JavaScript and XML.
• Coined by Jesse James Garrett of Adaptive Path.
• Helps transform clunky Web interfaces into interactive applications.
• Made popular by Google because of Google Suggest.
Google AJAX Search API

What is the Google AJAX Search API?

The Google AJAX Search API lets you put Google Search in your web pages with JavaScript. You can embed a simple, dynamic search box and display search results in your own web pages or use the results in innovative, programmatic ways. See the examples below for inspiration.

How do I start?

- Sign up for a [Google AJAX API key](#).
- Read the [developer documentation](#).
- Use the [wizards](#) to add search to your web site.
- Look at some [AJAX API community samples](#).

**Web Search**

Integrate Web Search, News Search, and Blog Search into your web site.

**Local Search**

Add Local Search results to your web site, or integrate them with your [Google Maps Search](#) API mashup.

**Multimedia Search**

Add YouTube Videos and Google Image Search results to your web site or blog.

Don't want to write code? Generate cut-and-paste HTML with our [AJAX Search wizards](#).
Google AJAX Search API

Welcome to Wei Liu’s homepage

I am a lecturer in the School of Computer Science and Software Engineering at the University of Western Australia.

Most of the time, I will be in my office, computer science building (campus map) room 2.18.

Postal Address: School of Computer Science and Software Engineering
                The University of Western Australia
                35 Stirling Highway, Crawley
                WA 6009

Phone: +61 8 6488 2095
Fax: +61 8 6488 1089
Email: weil@csse.uwa.edu.au
Office: CS Building 2.18

search this map

Today is: Sunday June 01, 2008
Page created by Wei Liu
Last Modified: 13:02:57 Friday May 30, 2008
Web 1.0 is Synchronous

- In Web 1.0, all requests are **synchronous**:
  - The client sends a request to the server.
  - The client waits for a response from the server.

- While waiting, user sees:
  - An hourglass (especially on Windows).
  - A spinning beachball (usually on Mac machines).
  - The application essentially freezes and sometimes the cursor changes.

- Web 1.0 applications:
  - Feel clunky or slow.
  - Lack of realtime interactivity.
  - Application essentially becomes unusable until response received.
  - If server busy or extensive processing required, wait maybe long.
Web 2.0 and Asynchronous Requests

• Web 2.0 introduces **asynchronous** requests.
  – The client sends a request to the server.
  – The client does not wait for a response from the server.
  – The user can continue to use the web page.
  – No spinning beachball or whirling hourglass.
  – No big application freeze.

• When server response finally arrives, some JavaScript function will execute in the background and updates your web page (see later).

• End result is a Web 2.0 application:
  – *doesn’t* feel clunky or slow.
  – *is* responsive, interactive, and feels faster.

• This is just one component of Web 2.0, but it’s a very important one.

• All the slick GUI components and Web design paradigms can’t overcome a slow, synchronous request/response model.
Web 2.0 Components

• HTML is used to build Web forms and identify fields for use in the rest of your application.

• JavaScript code is the core code running Ajax applications and it helps facilitate communication with server applications.

• DHTML, or Dynamic HTML, helps you update your page dynamically.

• DOM, the Document Object Model, will be used (through JavaScript code) to work with both the structure of your HTML and (in some cases) XML returned from the server.
**DHTML**

- DHTML is an acronym for Dynamic HTML.

- Dynamic HTML is a combination of JavaScript and DOM
  - Enables you to make a html page react/change to users’ actions.
  - Lets the user interact with the page.

- Technology that makes user interfaces possible
  - hide or show menus
  - change one dropdown select based on what was selected in another dropdown select
  - Integrate Gmail, or the new Yahoo mail beta.
Ajax versus DHTML

- **DHTML:**
  - is limited to using information loaded into browser.
  - Only way to interact with the server is to refresh the page.
- **Ajax:**
  - Interacts with the server asynchronously.
  - Loads new information into the current page.
- Good user interfaces should use DHTML and Ajax.
- DHTML will control what happens in the browser.
- Ajax will allow dynamic communication with the server.
- Updates can be made to a database or new information can be loaded without refreshing the page and wasting so much time recreating the page from scratch for every user action.
JavaScript XMLHttpRequest Class

- XMLHttpRequest class is the key to Ajax.

- The XMLHttpRequest object is supported in Internet Explorer 5.0+, Safari 1.2, Mozilla 1.0 / Firefox, Opera 8+, and Netscape 7.

- You need to become intimately familiar with XMLHttpRequest.
# XMLHttpRequest: attributes

<table>
<thead>
<tr>
<th>attributes</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>readyState</td>
<td>the code successively changes value from 0 to 4.</td>
</tr>
<tr>
<td>status</td>
<td>200 is OK; 404 if the page is not found.</td>
</tr>
<tr>
<td>responseText</td>
<td>holds loaded data as a string of characters.</td>
</tr>
<tr>
<td>responseXml</td>
<td>holds an XML loaded file, DOM's methods can be used to extract data.</td>
</tr>
<tr>
<td>onreadystatechange</td>
<td>takes a function name as value that is invoked when the readystatechange event is dispatched.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>state</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The request is not initialized</td>
</tr>
<tr>
<td>1</td>
<td>The request has been set up</td>
</tr>
<tr>
<td>2</td>
<td>The request has been sent</td>
</tr>
<tr>
<td>3</td>
<td>The request is in process</td>
</tr>
<tr>
<td>4</td>
<td>The request is complete</td>
</tr>
</tbody>
</table>
## XMLHttpRequest: methods

<table>
<thead>
<tr>
<th>methods</th>
<th>description</th>
</tr>
</thead>
</table>
| open(mode, url, boolean) | **mode:** type of request, GET or POST  
**url:** the location of the file, with a path.  
**boolean:** true (asynchronous) / false (synchronous).  
optionally, a login and a password may be added to arguments. |
| send("string") | null for a GET command. |
Create XMLHttpRequest Object

```javascript
if (window.XMLHttpRequest) {
    // Firefox, Safari, ...
    xhr = new XMLHttpRequest();
} else if (window.ActiveXObject) {
    // Internet Explorer
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
```

- Object detection for Firefox, Safari
- Create XMLHttpRequest object
- Object detection for IE
- Create XMLHttpRequest object
Aside: JavaScript try…catch

- The try…catch statement allows one to “try” execute some code and to “catch” any errors for special handling.

- Syntax:

```javascript
try {
    // execute code here
} catch(err) {
    // handle errors here
}
```
Example: try…catch

- Alternative to previous object detection:

```javascript
try {
    // Trying Internet Explorer
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
} catch(err) {
    // handle errors here
    xhr = new XMLHttpRequest();
}
```

- Will succeed if IE browser
- Failure if not IE browser
- Succeeds for Firefox, Safari
function createXHR() {
    var request = false;
    try {
        request = new ActiveXObject('Msxml2.XMLHTTP');
    } catch (err2) {
        try {
            request = new ActiveXObject('Microsoft.XMLHTTP');
        } catch (err3) {
            try {
                request = new XMLHttpRequest();
            } catch (err1) {
                request = false;
            }
        }
    }
    return request;
}
Typical Steps for Ajax Code

1. Get whatever data you need from the Web form.
2. Set up a function for the client to run and prepare a place holder to receive the server response.
3. Build the URL to connect to.
4. Open a connection to the server.
5. Send the request.
Step 1: Collect Data

```html
<html>
<head>
    <title>My First Ajax Code</title>
</head>
<body>
    <form name="myForm">
        Name: <input type="text" name="username" />
        Time: <input type="text" name="time" />
    </form>
</body>
</html>
```
Step 2: Create XMLHttpRequest Object

```html
<html>
<head>
    <title>My First Ajax Code</title>
    <script type="text/javascript" src="ajax.js"></script>
    <script type="text/javascript">
        function ajaxTime(){
            var xhr = createXHR();
        }
    </script>
</head>
<body>
    <form name="myForm">
        Name: <input type="text" name="username" onkeyup="ajaxTime()" />
        Time: <input type="text" name="time" />
    </form>
</body>
</html>
```

From previous slide, ajax.js contains createXHR()

As user edits name, ajaxTime() is called.

Create XMLHttpRequest
Step 3: Use XMLHttpRequest

```html
<html>
  <head>
    <title>My First Ajax Code</title>
    <script type="text/javascript" src="ajax.js"></script>
    <script type="text/javascript">
      function ajaxTime()
      {
        var xhr = createXHR();

        // Set up callback function when server response arrives ...
        // see later slide ...

        xhr.open("GET","time.php",true);
        xhr.send(null);
      }
    </script>
  </head>
  <body>
    <form name="myForm">
      Name: <input type="text" name="username" onkeyup = "ajaxTime()" />
      Time: <input type="text" name="time" />
    </form>
  </body>
</html>
```

Open connection to server

Send (GET) request to server
xhr.open("GET","time.php",true);

• A connection is opened.
• The method of connection is indicated (GET).
• The URL to connect to is “time.php”.
• The final parameter:
  – If false, code waits until a response was received (Web 1.0).
  – If true
    • requests asynchronous connection (Web 2.0, Ajax).
    • users can still use the form (and even call other JavaScript methods) while the server is processing this request.
Step 3 Continued:

```
<html>
  <head>
    <title>My First Ajax Code</title>
    <script type="text/javascript" src="ajax.js"></script>
    <script type="text/javascript">
      function ajaxTime(){
        var xhr = createXHR();
        xhr.onreadystatechange = function(){
          if(xhr.readyState==4){
            document.myForm.time.value=xhr.responseText;
          }
        }
        xhr.open("GET","time.php",true);
        xhr.send(null);
      }
    </script>
  </head>
  <body>
    <form name="myForm">
      Name: <input type="text" name="username" onkeyup = "ajaxTime()" />
      Time: <input type="text" name="time" />
    </form>
  </body>
</html>
```

Set up callback function to handle server response when it arrives

Wait until response is complete (readyState=4)

Update time on form
On the server side: time.php

```php
<?php
    echo date('l jS \of F Y h:i:s A');
?>
```

Prints something like:
Monday 8th of August 2005 03:12:46 PM
This string is return as response to Browser.
An example with XML - RSS

```html
<html>
<head>
    <script type="text/javascript" src="getrss.js"></script>
</head>
<body>
    <form>
        Select an RSS-Feed:
        <select onchange="showRSS(this.value)"
            ><option value="Google">Google News</option>
            <option value="MSNBC">MSNBC News</option>
        </select>
    </form>
    <div id="rssOutput"> <b>RSS Feed will be listed here.</b></div>
    <script>
        showRSS();
        Getrss.js contains showRSS()
    </script>
</body>
</html>
```

When user changes selection showRSS() is called

showRSS() will update this
getrss.js

```javascript
var xmlHttp;
function showRSS(str)
{
    xmlHttp=createXHR();
    if (xmlHttp==false)
    {
        alert("Browser does not support HTTP Request");
        return;
    }
    var url="getrss.php";
    url=url+"?q="+str;
    url=url+"&sid="+Math.random();
    xmlHttp.onreadystatechange=stateChanged;
    xmlHttp.open("GET",url,true);
    xmlHttp.send(null);
}
function stateChanged()
{
    if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete")
    {
        document.getElementById("rssOutput").innerHTML=xmlHttp.responseText;
    }
}
```

str is name of RSS feed
Create XMLHttpRequest
Form url for request
Use random number in url parameter to break cache
When response complete, update the rssOutput field
getrss.php (1)

```php
<?php
    //get the q parameter from URL
    //containing the feed name
    $q=$_GET['q'];
    if ($q=="Google") {
    }
    elseif ($q=="MSNBC") {
        $xml="http://rss.msnbc.msn.com/id/3032091/device/rss/rss.xml";
    }
    $xmlDoc = new DOMDocument();
    $xmlDoc->load($xml);
    $xmlDoc->load($xml);
```
getrss.php (2)

// get elements from "<channel>"
$channel=$xmlDoc->getElementsByTagName('channel')->item(0);
$channel_title = $channel->getElementsByTagName('title')
    ->item(0)->childNodes->item(0)->nodeValue;
$channel_link = $channel->getElementsByTagName('link')
    ->item(0)->childNodes->item(0)->nodeValue;
$channel_desc = $channel->getElementsByTagName('description')
    ->item(0)->childNodes->item(0)->nodeValue;

// output elements from "<channel>
echo("<p><a href=" . $channel_link. ">" . $channel_title . "</a>"
    . $channel_desc . "</p>");
getrss.php (3)

//get and output "<item>" elements
$x=$xmlDoc->getElementsByTagName('item');
for ($i=0; $i<=2; $i++) {

$item_title=$x->item($i)->getElementsByTagName('title')
       ->item(0)->childNodes->item(0)->nodeValue;
$item_link=$x->item($i)->getElementsByTagName('link')
       ->item(0)->childNodes->item(0)->nodeValue;
$item_desc=$x->item($i)->getElementsByTagName('description')
       ->item(0)->childNodes->item(0)->nodeValue;

echo ('"<p><a href=' . $item_link . '"">' . $item_title . '"</a>"');
echo ('"<br />"');
echo ('"</p>"');
}
?>
Typical Steps for Ajax code

1. Get whatever data you need from the Web form.
2. Set up a function for the client to run and prepare a place holder to receive the server response.
3. Build the URL to connect to.
4. Open a connection to the server.
5. Send the request.
Drawbacks of Ajax

- If JavaScript is not activated, Ajax can't work.
- The user must be asked to set JavaScript from within options of the browser, with the "<noscript>" tag.
- Since data to display are loaded dynamically, they are not part of the page, and the keywords inside are not used by search engines.
- The asynchronous mode may change the page after some delay (when the server response comes back), this may be confusing to some users.
- The back button may be deactivated (this is not the case in examples provided here). This can be overcome with extra code.
Ajax Frameworks

• The best technology to build dynamic web pages is Ajax.

• JavaScript code embedded into the HTML page is used to send requests to the server.

• At the server side, some processing is required to handle them, find the info or store the data.

• Specialized frameworks exist to provide the Ajax engine and associated client and server side functions.

• The framework has always a JavaScript part, and sometimes a server side part in another scripting language.

• Ajax frameworks exist in various programming languages and in all environments.
Types of Ajax Frameworks

- Javascript Libraries (no server-side processing)
- PHP Frameworks
- Java Frameworks
- .Net Frameworks
- ColdFusion Frameworks
- Ajax and XUL

Reference: http://www.xul.fr/ajax-frameworks.html
References

- Ajax Tutorial by Brett McLaughlin  
  http://www.yaldex.com/wAjax/AjaxTutorial.html

- Ajax Tutorial at xul.fr  

- Ajax Tutorial at w3schools.com  
  http://www.w3schools.com/Ajax/Default.Asp