Notes

• Examination reminder
  ▪ Next Tuesday. 75 minutes. Single-choices only. Closed-book exam.
  ▪ Not every question is covered in class. You have to read the book.
  ▪ You can choose online or paper exam

• Ask questions on Blackboard forum
  ▪ Discussions -> IS651-forum, not email from bb. You can subscribe it.
  ▪ You can ask if you need more explanations on homework, not your solutions
  ▪ Students are encouraged to help others on exercise questions
Homework/exercise Notes

• You might need to check “Additional Links” first before doing exercise/homework

• You can have an index.html at your folder to be more private.

• Grading is final one week after the deadline. Contract grader before that if you find problem and want regrading.

• Submit only url such as http://userpages.umbc.edu/~jianwu/ex1.xml, not file path on gl machine. No need to upload any files.

• You can resubmit if needed. Grader will grade the last submission before deadline.
Learning Outcomes

• After learning chapter 3, you should be able to
  ▪ Understand how web architecture works
  ▪ Understand HTTP request and response, and the differences between HTTP GET and HTTP POST method
  ▪ Understand and write simple HTML, CSS, JavaScript, AJAX and PHP scripts
XML Encoding

• Default encoding: UTF-8, which support international characters
• Demo: http://userpages.umbc.edu/~jianwu/is651/programs/chp2/
• Right click and select “View Page Source” to check encoding and DTD
Critical Thinking Questions of Last Chapter

• What are the main reasons for the evolution of distributed systems?

Reasons
1. Hardware evolution
2. Finer work division (identify general functionalities)
3. Flexibility (hide heterogeneity)
4. Standardization

• Some old techniques/hardware die out: Mainframe, CORBA, etc.

• Some still in use (internally): Client/Server architecture, Synchronous, J2EE
Critical Thinking Questions of Last Chapter (2)

• What do you think of the next generation of distributed systems?
  • Cloud computing (Everything as a Service)
  • Semantic Web?
  • Big Data?
Web Technologies

• Web Architecture
• HTTP, Browsers, URLs
• Client-side Techniques
• Server-side Techniques
• Important web site for the chapter/course: http://www.w3schools.com/
Web Architecture

<table>
<thead>
<tr>
<th>Presentation – Web Browser (client)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication – Web Server</td>
</tr>
<tr>
<td>Logic – Application Server</td>
</tr>
<tr>
<td>Storage – Database Server</td>
</tr>
</tbody>
</table>
Browsers, URLs

• URL Structure
  - `http://userpages.umbc.edu:80/~jianwu/is651/651.ref.s16.html#ch2`
    - protocol
    - host
    - port
    - path from web root
    - anchor
HTTP Request and Response

• HTTP Request
  • Method: GET, POST, etc.
  • Path: requested file under the web root directory
  • Entity body: data sent to server

• HTTP Response
  • Status code: standard code for the response
  • Phrase: an English version of the status code
  • Entity body: Data for web browser to display
HTTP Request and Response Demo

• $> \text{telnet zaad.is.umbc.edu 80}$
• $> \text{curl -v http://zaad.is.umbc.edu/index.html}$
Client-side Techniques

• HTML
  • Fundamental markup language for web pages
  • Define the content of web pages

• Cascading style sheets (CSS)
  • Used to set the presentational properties (or layout) of an HTML page: colors, fonts, layout, alignments, borders, etc.
  • It has its own syntax.

• JavaScript
  • Program the behavior of web pages
  • It is an object-oriented, dynamically typed scripting language that can be run by an interpreter inside the web browser and therefore included inside web page code.
  • It is quite different from Java: https://java.com/en/download/faq/java_javascript.xml
HTML

- **DOCTYPE**: defines the document type to be HTML
  - DOCTYPE is also used in XML.
  - XHTML (Extensible HTML): define an HTML as an XML document, stricter than HTML, well-formed XML.
- `<html>`: an HTML document
- `<head>`: information about the document. Javascripts and CSS are often defined here.
- `<body>`: the visible page content
- `<h1>`: the most important heading

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <h1>Hello World!</h1>
  </head>
  <body>
    <button type="button" onclick="">
      Set text color
    </button>
  </body>
</html>
```
Cascading Style Sheets (CSS)

• Syntax: selector {prop1,value1; prop2, value2; ...}

• Selector: select HTML elements based on element name, id, class, attribute, etc.

• Styling can be added to HTML elements in 3 ways: inline, internal, external.

• Cascading: a cascading order where the different types of stylesheets take priority and override a previous one.
  • The four stylesheet types with increasing priority: browser default, external, internal, and inline.
Document Object Model

• Document Object Model (DOM)
  • A cross-platform and language-independent standard to represent and interact with objects in HTML, XHTML, and XML documents.
  • When a web page is loaded, the web browser creates a DOM of the page organized in a tree structure, called the **DOM tree**.

• HTML DOM includes
  • The HTML elements as **objects**
  • The **properties** of all HTML elements
  • The **methods** to access all HTML elements
  • The **events** for all HTML elements
JavaScript

• A program language to dynamically change a web page based on its DOM
  • add, change, and remove HTML elements and attributes
  • change CSS styles
  • react to existing events
  • create new events

• Basic logic
  • Define event handler function
  • Associate an event with a function

• HTML DOM Events
  • Mouse event: onclick, oncontextmenu, ...
  • Keyboard Events: onkeydown, onkeyup, ...

```javascript
<script type="text/javascript">
    function displayResult() {
        // the DOM object is captured from the h1 element id.
        var x = document.getElementById('c1');
        // the x variable is now the DOM object and is styled using the object method style with the CSS property color and assignment to the value 'red.'
        x.style.color = "red";
    }
</script>

<button type="button" onclick="displayResult()">Set text color</button>
```
Asynchronous JavaScript and XML (AJAX)

• By default, JavaScript runs locally, manipulates the DOM without communicating with server.

• AJAX allows JavaScript to send asynchronous requests to a server, receive the response, and processes it without user interaction or a page reload.

• AJAX uses
  • (Internally) XMLHttpRequest object (to retrieve data from a web server)
  • JavaScript/DOM (to display/use the data)

• Example applications: Google Maps, Gmail, YouTube, and Facebook.
Server-Side Techniques (for Dynamic Web Pages)

• Common gateway interface (CGI)
  • CGI is a standard for communications between a web server and any programming language that has a CGI library.
  • One disadvantage is its poor performance. It forks a new process for each request, which is not scalable.

• Web server application programming interfaces (APIs)
  • Plug-ins for web servers that allow the web server process to spin off new threads for each request rather than a process, which much more lighter weight.
  • The scripts can be embedded in html using special template tags such as <%...%>
  • Script languages include PHP, JSP, ASP, etc.

• Java Servlet API
  • Allows a Java virtual machine to work as a plug-in to a web server.
  • A servlet is a java class that receives a request, then prepares and sends a response.
  • Normally work with Java Server Pages (JSP) together for dynamic web pages.
PHP

• Originally stands for Personal Home Page, but it is now a recursive backronym: PHP: Hypertext Preprocessor.

• PHP scripts are embedded using <? ?>. They can be part of a html or not.

• PHP scripts (optionally) read some inputs from client request, generate output as html content
  • Read input: $_GET, $_POST
  • Generate output: echo/print

• You won’t see php source code using “View Page Source” option
  ▪ Local web browser only get html content generated by PHP script
PHP Demo: HTML Form

• **Demo link**

• GET VS. POST
  • GET causes a querystring to be appended to the calling URL
  • POST puts the querystring in the HTTP entity body and not in the URL

```html
<!--The form action calls the PHP program form.php as a relative url.-->
<form action="form.php" method="get">
  <p>Choose a number between 1 and 6 for a random friend.</p>
  <!-- The name attribute of the input tag will be used in the PHP.-->
  Friend Number: <input type="text" name="friend" />
  <p>Give your random friend a last name.</p>
  Friend Last Name: <input type="text" name="lname" />
  <input type="submit" />
</form>
```
Jim, Tom, Sue
Hege, Tim, Qu

Data File

<?php

// php program starts here

$f=$_GET['friend'];  // read value for parameter 'friend' from link
$l=$_GET['lname'];   // read value for parameter 'lname' from link

$file = fopen("contacts.csv","r")  // read data file
$array=array();

while(!feof($file))  // loop until the end of file
{
  $a=fgetcsv($file, 1000);  // read a line from csv file
  $array=array_merge($array, $a);  // array merge
}

// output every element of array with <br/>

foreach($array as $x) echo $x."<br/>";

// construct output string based on inputs

echo "<p> My random friend is <strong>".$array[$f-1]." ".$l."</strong></p>";

fclose($file);  // close file
?>
</html>
### Web page VS Web service

<table>
<thead>
<tr>
<th></th>
<th>Web page</th>
<th>Web service</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td>client server</td>
<td>Built on client server architecture with additional layers and features</td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>for human to view business to consumer (B2C) e-commerce</td>
<td>for program to parse and understand results business to business (B2B) interactions automating business processes</td>
</tr>
<tr>
<td><strong>Commonalities</strong></td>
<td>Both are based on HTTP protocol</td>
<td>Both are hosted by a web server</td>
</tr>
<tr>
<td></td>
<td>Both are hosted by a web server</td>
<td>Both are web-based applications</td>
</tr>
<tr>
<td></td>
<td>Web technologies in this chapter are useful to both</td>
<td></td>
</tr>
</tbody>
</table>