Notes

• New schedule at course website:
  https://userpages.umbc.edu/~jianwu/is651/651.syll.s19.html

• General
  § Discussion 1 and homework 1 are now graded.
  § Grading is final one week after the deadline. Contract me before that if you find any problem and want regrading.

• In-class discussion
  § Do not use external resources: computer, phone, etc.
  § Write at most one page for each question
  § Brief bullet points are preferred, unless specified otherwise

• Ask questions on Piazza
  § You can ask if you need more explanations on homework, not your solutions
  § Ask your question early
  § Students are encouraged to help others on exercise questions
  § Private posts to All Instructors will get responses faster
Exam Reminder

• Next Monday (Feb. 25). 75 minutes. Multi-choices only. Closed-book exam.
• Not every question is covered in class. Read the book and know how to do exercise/homework.
• Online exam: bring a computer to the classroom.
• Grading rule: (summation of partial point for each correct choice) - (summation of partial point for each incorrect choice). Minimal point is 0
  ▪ E.g.: A question has 5 choices (a, b and d are correct). If your answer is a, d and e, you get (2/3-1/2) of the total point.
• Send me up to 5 good questions in your opinion, I’ll use top ones in the exam.
  ▪ Via private posts to All Instructors. Can be a group effort. You can add some explanation.
• If you cannot get a computer for the exam, contact me ASAP (before Friday).
Discussion #1

• We explained many advantages of SOA. What are disadvantages of SOA?
  ▪ Additional layer brings additional overhead.
  ▪ Not recommended for real time or graphical user interface (GUI) based applications.
  ▪ Complex service management.

• Interesting opinions
  ▪ Security issue
Homework Notes

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

• Submit only url such as http://userpages.umbc.edu/~jianwu/ex1.xml,
  ▪ Not file path on gl machine. I cannot access your folders!
  ▪ No need to upload any files.
  ▪ Please do not submit by email.

• Homework grading policy on late submission and resubmission
  ▪ 1 point deduction for no submission before deadline
  ▪ Additional 2 point deduction for no submission before next week’s class
  ▪ Additional 1.5 point deduction for resubmission within one week after next week’s class
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
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><strong>Presentation</strong> – Web Browser (client)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Communication</strong> – Web Server</td>
</tr>
<tr>
<td><strong>Logic</strong> – Application Server</td>
</tr>
<tr>
<td><strong>Storage</strong> – Database Server</td>
</tr>
</tbody>
</table>
Browsers, URLs

• URL Structure
  ▪ \texttt{http://userpages.umbc.edu:80/~jianwu/is651/651.ref.s19.html#ch2}
    o protocol
    o host
    o port
    o path from web root
    o 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

• Use Curl command to see request and response message
  ▪ $> curl -v -k https://swe.umbc.edu/~jianwu/test.html

• Guess what will happen with
  ▪ $> curl -v -k http://swe.umbc.edu/~jianwu/test.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

```xml
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
  </head>
  <body>
    <h1>Hello World!</h1>
    <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

```html
<html>
<head>
  <style type="text/css">
    h1 {color: red;}
  </style>
</head>
<body>
  <h1 id='c1' style='color: blue'>Hello World!</h1>
  <button type="button" onclick="">
    Set text color
  </button>
  <h1>H1 heading</h1>
  <h2>H2 heading</h2>
</body>
</html>
```
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, ...

```html
<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>
```
HTML JavaScript CSS Demo

• Demo link: http://userpages.umbc.edu/~jianwu/is651/programs/ch3/jxexample.html
  ▪ Right click to see page source

• You can write your own on gl server or w3schools
  ▪ http://www.w3schools.com/html/tryit.asp?filename=tryhtml_default
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)

• Advantage: more efficient, no need to reload the entire page

• Example applications
  ▪ Embed Google Maps and YouTube in your web page
jQuery

• jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML.

• It supports
  ▪ DOM manipulation
  ▪ AJAX support
  ▪ ...

• Import jQuery
  ▪ Direct download from http://jquery.com/download/
  ▪ Include it from a CDN (Content Delivery Network), such as Google and Microsoft

• jQuery has its own syntax on how to select (query) HTML elements and manipulate them
AJAX and jQuery Demo

- Ajax demo: http://userpages.umbc.edu/~jianwu/is651/programs/ch3/ajax.html
- Ajax google map demo: http://userpages.umbc.edu/~jianwu/is651/programs/ch3/ajax-google-map.html
- jQueryAjax demo: http://userpages.umbc.edu/~jianwu/is651/programs/ch3/jqueryAjax.html
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 light-weight
  ▪ The scripts can be embedded in html using special template tags such as <%...%>
  ▪ Script languages include PHP, JSP, ASP, etc.

• Java Servlet API: a special type of server 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:
  http://userpages.umbc.edu/~jianwu/is651/programs/ch3/form.html

• 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>
```
PHP Demo: PHP Script

• form.php

```php
<?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)); //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>
```

Data File

Jim, Tom, Sue
Hege, Tim, Qu
Demonstration for ex2

• Demo link:
  http://userpages.umbc.edu/~jianwu/is651/exercises/ch3/ex/ch3ex.php

• Source code:
  http://userpages.umbc.edu/~jianwu/is651/exercises/ch3/ex/ch3ex.text
Discussion #2

• 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
5. …