1.264 Lecture 12

XHTML, CSS, Forms
Hypertext markup language (XHTML) exercise

```html
<!DOCTYPE ...
<html>
<head>
<title> Welcome to the Chemical System </title>
</head>
<body>
<h1> Welcome to the Chemical System </h1>
This system handles orders for chemicals. We handle many products. We comply with the latest US regulations.
<p>
The use of this system is subject to <a href=“MITRule.html”>MIT rules and regulations.</a></p>
</body>
</html>
```

Create site and page in DW; type this text in Code view (Example1.html)
XHTML

- Tags (e.g. `<h1>`) never display but direct the browser
- XHTML tag pairs (e.g. `<h1>` and `</h1>`) delimit section
- Some tags have attributes (e.g. `<a href="abc.html"> </a>`)  
- XHTML document begins with `<html>`, ends with `</html>`
  
  Two sections within document: head and body  
  Head has identifying information not displayed  
  Body is displayed, with formatting:  
    - Paragraph `<p>`  
    - Header levels 1 through 6 `<h1>` through `<h6>`  
    - Anchor `<a>`, placed around text or graphics; used for hyperlinks
- HTML and most of its formatting features will be deprecated eventually, replaced by XHTML and CSS (Cascading Style Sheets)
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Test Web page</title>
</head>
<body>
<p>Sample text</p>
</body>
</html>

• **XHTML is more structured form of HTML:**
  • Defined as XML document
  • Rules stricter:
    • All tags must be lower case
    • All tags must be closed
  • Quotation marks required for tag attributes
  • Dreamweaver generates XHTML, which we’ll use all term
Tag examples

- Control appearance of page
  - Not as precise as MS Word or other editors
  - Intended for pages that can be viewed on machines with very different graphics capabilities. Tags discourage specific assumptions.

<table>
<thead>
<tr>
<th>Opening tag</th>
<th>Closing tag</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;em&gt;</td>
<td>&lt;/em&gt;</td>
<td>Emphasis (often italic)</td>
</tr>
<tr>
<td>&lt;strong&gt;</td>
<td>&lt;/strong&gt;</td>
<td>Strong emphasis (often bold)</td>
</tr>
<tr>
<td>&lt;var&gt;</td>
<td>&lt;/var&gt;</td>
<td>Variable</td>
</tr>
<tr>
<td>&lt;cite&gt;</td>
<td>&lt;/cite&gt;</td>
<td>Citation (of an article or book)</td>
</tr>
<tr>
<td>&lt;address&gt;</td>
<td>&lt;/address&gt;</td>
<td>Address (identifying info at bottom of doc)</td>
</tr>
<tr>
<td>&lt;code&gt;</td>
<td>&lt;/code&gt;</td>
<td>Computer code fragment (tags ignored)</td>
</tr>
</tbody>
</table>

Add some of these to your Example1.html page
Tag examples, cont

- Special characters: `&gt;` is `>`
- Paragraph format: `<h2 align="center">`text`</h2>`
- Preformatted text: `<pre>` (lines up columns exactly, etc.)
- Horizontal lines: `<hr width="80%" align="left">`
- Lists: can be multi-level, etc.
  - Ordered (numbered) `<ol>`
  - Unordered (bulleted) `<ul>`
  - Definition (indented) `<dl>`
- Tables: `<table>`, `<td>` (data), `<tr>` (row), etc.

Add some of these to your Example1.html page
XHTML Filenames

- Relative names (UNIX conventions)
  - MITRule1.html: same folder (directory) as current doc
  - Laws/MITRule2.html: in subfolder (subdirectory) of current doc
  - ../MITRule3.html: in folder above current doc

- Absolute names (also UNIX)
  - /MITRule4.html: in root folder (Web root or local root)

- Document sections
  - MITRule1.html#sec1: points to target tag (sec1) in same doc
    Sections defined by <a name="sec1"> </a>

- Documents on other sites
  - <a href="http://www.mit.edu/index.html">
Images

<h1>Welcome to the Chemical Manufacturer system</h1>

<a href="picture.gif">
<img src="thumbnail_picture.gif" alt="Chemical services">
</a>

<p>Welcome to the Chemical System</p>

- This shows a small picture but allows the user to download the large picture
- The href can point to other documents about chemicals

Use two images on your PC to do this
Cascading Style Sheets (CSS)

- Format XHTML more flexibly than XHTML tags
  HTML format tags are being deprecated
  Take less space
  Easier to apply across a Web site or part of a site
- Internal style sheets stored on Web page
  Apply only to that page, appear in <head> of document
- External style sheets stored as separate files
  Apply to all pages in site that reference it
  Contain only CSS commands, no html or XHTML
- Two types of style:
  Class styles: manually apply to entities such as ‘Company’
  Tag styles: automatically apply to tags such as <h1>
CSS Exercise- Internal Style Sheet

• Open Dreamweaver
  File-> New -> Basic Page, HTML (XHTML)
  Type your company name and a 1-sentence paragraph about it
  Text->CSS Styles-> New CSS Style
    Create .company CSS file (class, “this document only”)
    Font= “Courier New”, Courier, mono
    Font size= 24 px
    Set a light blue background color
  Select your company name on the Web page
  In property inspector, apply .company style
  Examine XHTML page with CSS in Code view:

    .company {
      font-family: "Courier New", Courier, mono;
      font-size: 24px;
      background-color: #00FF00;
    }

  Top of file:
  <p class="company">My company name</p>
CSS Exercise- External Style Sheet

Text->CSS Styles-> New CSS Style
  Create .chemparagraph CSS file (class, new style sheet)
  Save as chemparagraph.css (no leading period)
  Font= Arial, Helvetica, sans serif
  Font size= 16 px
  Set a light yellow background color
Select the paragraph from the Web page
In property inspector, apply .chemparagraph style
Examine CSS page in Code view:

```
.chemparagraph {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 16px;
  background-color: #FFFF66;
}
```

Top of HTML file:

```
<link href="chemparagraph.css" rel="stylesheet" type="text/css" />
```
CSS Exercise- Tag Style

Type another paragraph in the XHTML page with <h3> style
Text->CSS Styles-> New CSS Style
    Create h3 CSS file (tag, this document only)
    Font= Arial, Helvetica, sans serif
    Font size= 24 px
    Font color red, weight bold or bolder
Examine CSS page in Code view
Now create another <h3> header on the page and see what it looks like

Do the chapter 6 tutorial in [DW] for a more thorough example
Forms

- Web server and its pages are a communications channel between databases
- We first create communications between a user and a database
  More visible and intuitive, uses the same principles
  Based on XHTML pages and forms
- Database-database communications will be covered next
  Based on XML and HTTP, collectively called SOAP
  Machine-readable, can be validated by machines
  Some human is still used because there isn’t complete trust among trading partners, engineering collaborators, etc.
  We’ll have more to say about this under ‘Security’
- Machine-machine communications create database queries in SQL to fill in ‘XML forms’ and send them to the next database
XHTML forms

• Used as front ends to Web server programs
• Forms are user interface elements to collect data from user and transmit it to the server application program
• Forms are limited and slow, so there have been many enhancements attempted over the years:
  JavaScript (different than Java!) or VBScript can be used with XHTML forms to enhance their operation (called DHTML)
  Java applets can also provide a richer user interface, are more complex and have security features. Not used much.
  AJAX (Asynchronous Javascript and XML) is fashionable this year to improve Web application performance
  All of these run on the browser (on the client machine) and are user interface components
How XHTML forms transmit data

- Forms allow a series of controls to be placed on the page
  Controls are text boxes, dropdowns, radio buttons, check boxes...
  Each control has a name and a value
- Form data is sent when user presses ‘Submit’ button. Three options:
  Data is sent to URL with HTTP POST command as string of form:
  \[\text{Name1 Value1&Name2 Value2&...NameN ValueN}\]
  POST data is sent after the blank line after HTTP headers, as a string
  Data can alternatively be sent to URL with HTTP GET command, appended to end of GET string after a `?:`
  \[\text{GET/Index.html?Name1 Value1&Name2 Value2&...NameN ValueN ...}\]
  SOAP is an extension to HTTP to allow XML documents to be sent by POST (or GET) commands instead of plain text strings
  We’ll cover XML later but it allows validation and business rules
- Form’s ‘Submit’ button is associated with the URL of a server program that will process the input data
  Server programs have methods (function points!) to extract the data from the string and use it in the program
  See why we counted Web pages and estimated function points?
Browser-server forms interaction

Step 1: Browser requests form without any parameters
Step 2: Server fetches form, and returns empty input form
Step 3: User fills out input form and presses ‘submit’ button
Step 4: Browser packages form into query string and sends to server
Step 5: Program synthesizes a response document and returns it

Every dynamic validation, fill-in requires server intervention.
- If later choices depend on earlier choices, etc.
- AJAX, Javascript, applets are used for this
- Pages can be XML or XHTML
## XHTML tags for forms

<table>
<thead>
<tr>
<th>Tag</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;form&gt;</td>
<td></td>
<td>Start a form</td>
</tr>
<tr>
<td>&lt;input type=</td>
<td>text</td>
<td>Single line of text entry</td>
</tr>
<tr>
<td></td>
<td>password</td>
<td>Single line password entry</td>
</tr>
<tr>
<td></td>
<td>file</td>
<td>File to upload, with ‘Browse’ button</td>
</tr>
<tr>
<td></td>
<td>checkbox</td>
<td>Checkbox</td>
</tr>
<tr>
<td></td>
<td>radio</td>
<td>Radio button (option box)</td>
</tr>
<tr>
<td></td>
<td>image</td>
<td>Image acting as button</td>
</tr>
<tr>
<td></td>
<td>hidden</td>
<td>track user, store predefined input, etc.</td>
</tr>
<tr>
<td></td>
<td>submit</td>
<td>Submit button for form</td>
</tr>
<tr>
<td></td>
<td>reset</td>
<td>Button to restore default values</td>
</tr>
<tr>
<td>&lt;select&gt;</td>
<td></td>
<td>List box or combo box</td>
</tr>
<tr>
<td>&lt;option&gt;</td>
<td></td>
<td>Item in scrolling list or popup menu</td>
</tr>
<tr>
<td>&lt;textarea&gt;</td>
<td></td>
<td>Start multiple-line text entry field</td>
</tr>
</tbody>
</table>
XHTML forms demo

- Create site that uses server technology
- Create an Active Server Pages (ASP) page, not a .html page with File->New: Page. Name it test.asp
- Connect to SQL Server database on your machine or database server:
  
  Application Panel: Databases: +
  Create new Data Source Name using ODBC
  Database must allow SQL Svr authentication, not just Windows
  Need to have created login in SSE (username, pwd)
  Set default db table to the MIT1264 table
  Include username and password in ODBC DSN
  Connect to SQL Server, lecture database (MIT1264 or similar)
  (If db connection doesn’t work, debug with TAs in recitation)
- Build form to insert data in, Offices table
  Insert->Application Objects-> Insert Record -> ...Wizard
  Insert a new part; only part number and vendor are needed
- Look at the XHTML produced
- Save and then test in browser
XHTML Forms Exercise

• To look at the fundamentals of a form:
  – Create a new XHTML page
  – Choose the Forms insert bar
  – Insert form
  – Insert two text boxes
  – Type labels in front of them: Part ID and Vendor
  – Insert a button, label it ‘Insert’

• You’d then have to write some code (or use a DW wizard) to create a POST action and connect it to a database table
Data transmittal across Web

• These forms are sending user entered data
  Is it an HTTP POST or HTTP GET?
• What do you think the URL is?
  What machine? What application on the machine?
• How is the form data being sent?
  Text string? XML (self-describing document)?
  Which do you think would be better? Why?
• Would it be nice if the data were sent the same way, regardless of whether it was user-entered or came from a database on the ‘client’ machine?
  (The client is the machine sending the data; it may only be a client for the purposes of this transaction. In others it may be a server, receiving data, in a supply chain/engineering collaboration.)
Summary

• XHTML forms are used to transmit data from human users to computers
  POST or GET are used, with data in strings
  Not validated or structured
• Scripts (AJAX, JavaScript) are used for dynamic data entry
  Security risks of active content and data disclosure
  No access to database on server for sophisticated checks
• Active content (DHTML) hasn’t been needed so far
• What we do want is:
  Data transfer to servers, from other computers or humans in a consistent way
  Validation and correctness of data
  Ability to have good security
• XML is the current solution
  Uniform, validated format for human and machine data
HTTP and XHTML

- **HTTP**
  - Is only direct form of interaction between browser and server
  - Was an extremely perceptive extension of email, ftp protocols by Tim Berners-Lee to enable Web browsers
  - Request-response paradigm
  - Connection made for each request/response pair
  - Most popular protocol on Internet, fairly stable definition

- **XHTML**
  - Text description language, based on tags
  - High level description rather than specific formatting
  - XHTML has forms, which are the basis for submitting data to databases.
  - XML is also used for data exchange, along with HTTP extensions