HyperText Markup Language (HTML)

What is HTML?

- HTML is the common language for publishing hypertext on the World Wide Web. It is a non-proprietary format based upon SGML.
- Can be created and processed by a wide range of tools, from simple plain text editors - you type it in from scratch- to sophisticated WYSIWYG authoring tools.

What is XHTML

- The Extensible HyperText Markup Language (XHTML™) is a family of current and future document types and modules that reproduce, subset, and extend HTML, reformulated in XML.
- XHTML Family document types are all XML-based, and ultimately are designed to work in conjunction with XML-based user agents.
- XHTML is the successor of HTML.

Markup Languages

- SGML:
  - Standard Generalized Markup Language
  - Mother of Markup Languages
- HTML
  - Most popular presentation language for web
- XML:
  - Draws heavily on the merits & shortcomings of HTML & SGML

Brief History of HTML

- 1989 CERN - WWW project proposal
- 1992 Laying the foundations
- 1993 NCSA Mosaic released, Web takes off
- IETF comes in:
  - July '93: Internet Draft for HTML+
  - Nov '95: RFC1866 - HTML 2.0
  - Nov '95 RFC1867 - Form-based file Upload
  - March '95 HTML 3.0 Draft stalls
  - May '96 RFC1942 - HTML Tables
  - Jan '97 RFC2070 - Internationalization of HTML
  - Late '96 IETF HTML WG closes
- World Wide Web Consortium (W3C – www.w3.org) Takes over:
  - Fall '95 W3C brings key vendors to negotiating table
  - Initial work on common object model: *<object>*
  - Followed by work on raising the baseline for HTML
  - Jan '97 W3C releases HTML 3.2 Recommendation
  - Dec '97 W3C releases HTML 4.0 Recommendation
  - May '98 W3C workshop on Future of HTML
  - Feb '99 XHTML 1.0 draft released
  - Feb 2004: Modularization of XHTML 1.0 - Second Edition
  - April 2004: XHTML-PRINT

Creating a HTML Page

- Requirements
  - Text or HTML Editor
  - Graphics editors
  - Browser (Netscape, Internet Explorer, Lynx, etc.)
- Focus
  - Usable and Eye-catching documents
  - Images in Web pages
  - Animation

HTML Basics

- HTML documents contain 4 things
  - Text
  - Tags
  - External Multimedia such as graphics, sound, movies, etc.
  - Scripts
- Example
  - <TAG> Your Text Here </TAG>
  - Types, used in pairs, or not in pairs
  - Tags can be nested
  - Tags have Attributes:

    <FONT size="+2" face="Times New Roman">Hi</FONT>

What are Tags?

- Mark text as
  - headings, paragraphs
  - formatting (physical, logical)
  - list
  - quotations, etc.
- Also for
  - creating hyperlinks
  - including images, making tables
  - fill-in forms, frames

HTML Document Structure

- Basic Structure

```
<HTML>
<HEAD>
<TITLE> KFUPM </TITLE>
</HEAD>
<BODY>
    ….. ….. …….
</BODY>
</HTML>
```

HTML Document Structure

- HTML= head + body
  - Body elements contain all the text and other material to be displayed
- Line breaks and indentation exist only for human readability
- Comment

```
<!--  this is a single-line comment  -->
<!--
also multi-line comment
-->
```

Example

```
<HTML>
<HEAD>
<TITLE>head/title</TITLE>
</HEAD>
<BODY> all elements of document
    <H1> Big heading </H1>
    <H6> Small heading </H6>
    <P> a para of text comes here </P>
</BODY>
</HTML>
```

Markup

- There are four kinds of markup elements in HTML:
  - ***structural* markup:** that describes the purpose of text (for example, <h1>Golf</h1> will cause a reader to treat "Golf" as a first-level heading),
  - ***presentational* markup:** that describes the visual appearance of text regardless of its function (for example, <b>boldface</b> will render **boldface** text) (Note that presentation markup is deprecated and is not recommended; authors should use CSS for presentation),
  - ***hypertext* markup:** that links parts of the document to other documents (for example, <a href="http://www.wikipedia.org/">Wikipedia</a> will render the word Wikipedia), and
  - ***widget* elements:** that create objects (for example, buttons and lists).

Modularising HTML

Designed for easy use in subsetting HTML, the following modules are strictly based upon HTML 4.0.

- **Applet** — applet, param
- **Block phrasal** — address, blockquote, pre, h1-h6
- **Block presentational** — center, hr
- **Block structural** — div, p
- **Inline phrasal** — abbr, acronym, cite, code, dfn, em, kbd, q, samp, strong, var
- **Inline presentational** — b, basefont, big, font, i, s, small, strike, sub, sup, tt, u
- **Inline structural** — bdo, br, del, ins, span
- **Linking** — a, base, link
- **Lists** — dir, dl, dt, dd, ol, ul, li, menu
- **Simple forms** — form, input, select, option, textarea
- **Extended forms** — button, fieldset, label, legend, optgroup, option, select, textarea
- **Simple tables** — table, td, th, tr
- **Extended tables** — caption, col, colgroup, tbody, tfoot, thead
- **Images** — img
- **Image maps** — area, map
- **Objects** — object, param
- **Frames** — frameset, frame, iframe, noframes
- **Events** — onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
- **Metadata** — meta, title
- **Scripts** — noscript, script
- **Styles** — style element and attribute
- **Structure** — html, head, body

Applet

- Use <applet></applet> tag to include applets in your HTML document
- Attribute:
  - Name: applet name for scripting
  - Code: the name of the main class file that starts and runs the applet
  - Archive: classes can be compressed and archived in .zip format

Java Applet inclusion

- Compile the Java code (e.g., use javac)
  - example: javac Blinker
- Creates file with extension .class,
  - example Blinker.class
- Use the tags <APPLET> … </APPLET>
- Specify parameters such as speed, color (for background and text, etc.)

Block Phrasal

- Tags used for phrasing blocks of text like headers:
  - <h1></h1>, <h2>, …, <h6>.
  - <blockquote>: used for including quotations to text, used in a “reply” message in outlook express and other email clients that allow you to quote the sender when you reply.

Block Presentational

- Tags used for representing blocks of hypertext:
  - <center>: to position a certain block in the middle of the web page
  - <hr>: horizontal ruler, splits the webpage by a graphical horizontal ruler, *<u>has no closing tag</u>*, includes attributes:
    - color: presented in hexadecimal code or name, like color="#FFFFFF" or color="white".
    - width
    - noshade
    - align

Block Structural

- Structures blocks into layers and paragraphs, like <p> for a paragraph:
- <p></p>: will automatically format a new paragraph, thus leaving an empty line before it.
  - If you don’t want a new paragraph, just want a new line, use <br> (called line break, has no closing tag)
  - Has attributes like “Align” to align the text in the paragraph

Inline Presentational

- Present text
  - <B> bold </B> or use <strong>
  - <BIG> …. </BIG>
  - <SUB> Makes text subscripts </SUB>
  - <TT> emphasized text </TT>
  - <I> text in italics </I> or use <em>

Linking (A, LINK, BASE)

- To make a hyper link we use “anchor” signified by the tag <a></a>
- <link> and <base> are used to link the HTML document to another reference HTML file, used in the <head> section of the document
- Anchors: have attribues:
  - href: the page or Internet Service you want to link to
  - hreflang: the encoding language used at the target document, used as advisory to help the browser prepare itself for new encoding
  - name: to link within the document, encouraged in XHTML 1.0, called a Named Anchor
  - target: where to open the new page
    - _top:
    - _parent:
    - _blank:
    - _self:
    - or name of frame

Values for TARGET attribute

- **_blank:** The user agent (i.e. Internet browser) should load the designated document in a new, unnamed window.
- **_self:** The user agent should load the document in the same frame as the element that refers to this target.
- **_parent:** The user agent should load the document into the immediate FRAMESET parent of the current frame. This value is equivalent to _self if the current frame has no parent.
- **_top:** The user agent should load the document into the full, original window (thus canceling all other frames). This value is equivalent to _self if the current frame has no parent

Linking with <base>

- No closing tag
- Attribute:
  - **href :** This attribute specifies an absolute URL that acts as the base URL for resolving relative URLs.
  - **target**: target frame information
- In HTML, links and references to external images, applets, form-processing programs, style sheets, etc. are always specified by a URL. Relative URLs are resolved according to a base URL, which may come from a variety of sources. The BASE element allows authors to specify a document's base URL explicitly.
- When present, the BASE element must appear in the HEAD section of an HTML document, before any element that refers to an external source. The path information specified by the BASE element only affects URLs in the document where the element appears.
- For example, given the following <base>declaration and <a> declaration:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
     <HTML>
      <HEAD>
      <TITLE>Our Products</TITLE>
      <BASE href="http://www.aviary.com/products/intro.html">
      </HEAD>
      <BODY>
      <P>Have you seen our <A href="../cages/birds.gif">Bird Cages</A>?
      <P>What about our <A href="bird.jpg">Birds</A>?
      </BODY> </HTML>
```

the relative URL "../cages/birds.gif" would resolve to: http://www.aviary.com/cages/birds.gif
and the relative URL “birds.jpg" would resolve to: http://www.aviary.com/products/bird.jpg

Lists

- To generate menus or lists we use
  - <menu> and <li>
  - or <ul> and <li> (preferred)
  - Example:

```
<ul>
        <li>DB-9</li>
        <li>DB-12</li>
        <li>DB-25</li>
</ul>
```

- This example will generate an unordered list of items, for ordered lists use <ol> instead of <ul>
- One important attribute is *type {circle, disc or square}* which is used to specify the type of bullet you want to use. This attribute has been deprecated in HTML 4 in favor of the *list-style-type* style sheet attribute.
- or you can use your own bullet by using *list-style-image* style sheet attribute

Tables

- These are tags used to generate and manipulate tables, rows and columns.
- Use <table></table> to create a table
- Use <tr></tr> to format rows inside tables and <td></td> for cells in the rows
- Indentation in HTML document is encouraged especially when it comes to tables, example:

```
<table>
        <tr>
                <td>column1,row1</td>
                <td>column2,row1</td>
        </tr>
        <tr>
                <td>column1,row2</td>
                <td>column2,row2</td>
        </tr>
</table>
```

Tables – cont’d

- Attributes:
  - *background*: specifies an image file used as backdrop to the table
  - *bgcolor*: specifies background color of the table
  - *border*: specifies the thickness in pixels of the table border, the browser will render it as a 3D border
  - *cellspacing*: specifies space size between cells in the table
  - *width and height*: specify dimensions of the table other than the defaults given by the browser.

Images

- Use the tag <img> to include images (or animated GIFs) in your HTML document (no closing tag), example:

<img src=“myphoto.jpg” alt=“my photo” \>

- Attributes:
  - *src*: the source file of the image
  - *width and height*: specify the dimensions of the image, you might want to display it in different dimensions like thumbnails.
  - *border*: border thickness in pixels around the image
  - *usemap*: to specify name of the image map associated with this image, ex. usemap=“#navigation”
  - *alt*: alternate text that will appear when the user points at the image with the mouse

Some notes on Images

- Loading of images is made faster by telling the browser the size of the image
- Size is specified in pixels
- You can link by using images
  - Can have pictures with no borders
- You can use thumbnail images to link to larger images
- Making clickable images (image maps)

Image Maps

- Enable users to click on parts of images (e.g., click on a state or country in a map)
- HTML tag used is <map></map> and <area> is used to specify what are the clickable areas (hotspots) in an image map
- areas have default shapes of a rectangle, a circle or a polygon
- Attributes for <map>:
  - name: specifies name of image map to be referenced in the <img> tag

Image Maps – cont’d

- Attributes for <area>:
  - href: what link should the browser follow when user clicks on this area
  - shape: shape of area
    - circle
    - rectangle
    - poly or polygon
  - target: where the target site should open
    - _top
    - _parent
    - _blank
    - _self
    - or name of frame
  - coords: specifies coordinates of area
- It is advised that you use a special software tool to generate image maps and areas, especially when it comes to computing coordinates

Image Maps – cont’d

- Example:

```
<img scr=“images\logo.gif” alt=“scroll to the bottom for navigation links” height=“300” width=“250”
      usemap=“#navigation” >
<map name=“navigation”>
        <area shape=“rect” coords=“0,0,100,100” href=“products.html”>
        <area shape=“rect” coords=“0,100,300,100” href=“support.html”>
</map>
```

Objects

- Use the tag <object></object> to include any external objects to your HTML document, like flash files, video files, sound files…etc.
- An important attribute is *classid* which is used to direct the browser to load the program, an applet or a plugin class file.
- You must obtain the *classid* value from a supplier of an ActiveX control
- Another tag used with object inclusion is <param> (no closing tag) which is used to pass parameters to the object is needed

Objects - example

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,29,
    0" width="570" height="75">

        <param name="movie" value="file:///C|/Inetpub/wwwroot/flash/Movie1.swf">
        <param name="quality" value="high">
        <embed src="file:///C|/Inetpub/wwwroot/flash/Movie1.swf" quality="high"
            pluginspage="http://www.macromedia.com/go/getflashplayer" type="application/x-shockwave-
            flash" width="570" height="75"></embed>

</object>
```

Events

- Events are scripts that will be executed when a certain event takes place, like: onMouseOver, onClick, onLoad
- Events are used inside tags, just like attributes

  e.g., <body onLoad="document.form1.field1.focus()">….</body>

Scripts

- To include scripts inside an HTML document, we use the tag <script></script>
- Attributes:
  - language: sets the scripting language you want to use in writing the script, like c#, javascript, vb, php, jscript, vbscript
  - source: import script writing in another document like a .js file or another website by specifying the URL
  - type: an advisory about the MIME-content type of the script, to tell the browser what scripting engine to use.

Scripts - example

```
<script language="javascript" type="text/javascript">
        function wrong_pass() {
                alert("please enter correct password!");
        }
</script>
```

Scripts – cont’d

- Some browsers don’t support scripting and cannot render scripts, in this case you might want to inform the user that this page contains scripts and it will not work on this browser, for this we use the tag <noscript></noscript>

e.g.,

<noscript>

This document contains programming that requires a scriptable browser, such as Microsoft Internet Explorer or Netscape Navigator. You may not have full access to this page’s power at this time.

</noscript>

This message will appear to the user if the browser does not support scripting, but it will not appear if the browser does support scripting.

Frames

- Frames split the webpage into different regions for more clarity and accessibility to the user.
- To make frames first we must define a frameset using the tag <framset> </framset>
- Using this tag (element) we can specify how many frames we want and what is the layout of these frames
- Then we define the properties of every frame using the <frame> tag (no closing tag) nested inside the <frameset>

<frameset> attributes

- *border*: a 3-D border is displayed by default, to change the size of this border use this attribute
- *cols*: defines the number and sizes or proportions of column arrangement of frames in the frameset, you can use absolute pixel size or use percentage, but the total percentage must sum up to 100%, like cols="25%,50%,25%", you can also use wildcard asterisk (*)
- *rows*: like cols, but used for row arrangements.
- *frameborder*: a boolean value indicating whether you want a divider between frames or not

<frame> attributes

- *frameborder*: a boolean value indicating whether you want a divider (border) for the frame or not, overrides *frameborder* attribute in <frameset> tag
- *name*: the name of a frame will be used to reference to it, especially when you want to open a URL in a specific frame.
- *noresize*: the user will not be able to move the divider of the frame or resize it
- *scrolling*: boolean value of whether you want a scroll bar for the frame or not, can be “yes”| ”no”| ”auto”
- *src*: what URL or HTML page will displayed in this frame

Frames - example

- We want to make a set of 3 frames, one header for company logo, one menu on the left and one main frame as body to the right
- We want the header to cover over the menu and the body

```
<frameset rows="16%,*" cols="*" frameborder="NO" border="0"
    framespacing="0">
  <frame src="header.htm" name="topFrame" scrolling="NO" noresize >
  <frameset cols="17%,*" frameborder="NO" border="0" framespacing="0">
    <frame src="menu.html" name="leftFrame" scrolling="NO" noresize>
    <frame src=“body.asp" name="MainFrame">
  </frameset>
</frameset>

<noframes>Your browser does not support frames.<br>To view this website, you
    will need a different browser</noframes>
```

Note on Frames

- Frames should be defined before the <body></body> part of the HTML document
- Usually:

```
<html>
<head>
…
</head>

<frameset…>
    <frame…\>
</frameset>

<body>
…..
</body>
</html>
```

Separation of content and Style in HTML

- Efforts of the web development community have led to a new thinking in the way a web document should be written; XHTML epitomizes this effort.
- Standards stress using markup which suggests the structure of the document, like headings, paragraphs, block quoted text, and tables, instead of using markup which is written for visual purposes only, like <font>, <b> (bold), and <i> (italics).
- Such presentational code has been removed from the HTML 4.01 Strict and XHTML specifications in favor of CSS solutions. CSS provides a way to separate the HTML structure from the content's presentation

Document Type Definition (DTD)

- All HTML documents should start with a Document Type Definition (or DTD) declaration. For example:

  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

  This defines a document that conforms to the Strict DTD of HTML 4.01, which is purely structural, leaving formatting to Cascading Style Sheets.

- Other DTDs, including Loose, Transitional, and Frameset, define different rules for the use of the language.

Styles

- Styles are assigned to different HTML elements using the *style* attribute, which applies to mostly all HTML tags
- You are discouraged from writing a style for every element.
- Instead you are encouraged to create a separate cascading style sheet file (.css) and linking your HTML document to this file using the <link> tag in the header of the HTML file, e.g., <link href=“mystylesheet.css" rel="stylesheet" type="text/css">
- Then you can apply the style to any HTML element by simply referring to it using the *class* attribute
- This is very versatile and makes it very easy for the developer in case you want to apply the same style to different elements in the web page

Cascading Style Sheets

- A style must be defined using <style></style> tag
- You can use this tag and define as many styles as you want and put them all in a single file and save it as .css, this will be a cascading style sheet, to which you can link from within your HTML document.

CSS

- Style sheets in CSS are made up of rules, a rule has 3 parts:
  - Selector: where you want to apply the rule
  - Property: what property you want to affect
  - Value: what is the value you want for this property

e.g., font { color:#d8da3d }

CSS – grouping properties

- You can put more than one property in a rule, but you have to separate them by semi-colon:

e.g., font {color:#d8da3d; text-decoration:underline}

or

```
font
    {color:#d8da3d;
     text-decoration:underline
    }
```

CSS – grouping selectors

- You can also put more than one selector for the same style. Separate them by comma.
- Example:

```
Font, p, body
    {color:#d8da3d;
     text-decoration:underline
    }
```

Class Selectors

- Define your own classes and apply them to HTML elements in your HTML document.
- Example:

  In CSS:

  .warningFont { color: red;}

  In HTML:

  <p class=“warningFont”>Invalid Password</p>

Comments

- You can add comments in your style sheets using /*…..*/

Contextual Selectors

- ***Contextual selectors*** are merely strings of two or more simple selectors separated by white space. These selectors can be assigned normal properties and, due to the rules of cascading order, they will take precedence over simple selectors. For example, the contextual selector in:

```
P EM { background: yellow }
```

is **P EM**. This rule says that emphasized text within a paragraph should have a yellow background; emphasized text in a heading <h1> would be unaffected

Pseudo-classes in CSS

- Pseudoclasses are subclasses added to the property of a style to distinguish different cases of the property
- Like a link <a>, we can set a different style for a link when it is visited or active or when the mouse hovers over it
- Exmaple, the anchor pseudoclass:

```
<style type="text/css">
    a {color:#000000}
    a:visited {color:#0000ff}
    a:active {color:#000fff}
    a:hover {color:#ff0000; text-decoraction:none}
</style>
```

Pseudo-elements

- Pseudo-elements include:
  - First line pseudo-element, and
  - First letter pseudo-element.

- Example:

```
P:first-line {
        font-variant: small-caps;
        font-weight: bold
       }
P:first-letter {
     font-size: 300%;
     float: left
}
```

CSS – cont’d

- Using style sheets you can pre-assign the properties of an HTML tag (like in the example before) or create new classes of style.
- Later in your HTML file you can reference these classes in the *class* attribute of a certain tag to apply the selected style.
- Example

```
<style type=“text\css”>
        .main_table {background-color: #CC3366}
        .nav_table {background-color: #FFFF66}
</style>
….inside HTML:
<table class=“main_table”>….</table>
…
```

Forms

- What are they used for
  - Surveys
  - Collect addresses of visitors to your Homepage
  - Allow people to register for something
- Features
  - Submitted by mail
  - Security (Passwords)
  - Checkboxes and Radio buttons
  - Area for Text and Comments
- Require an application environment like ASP, PHP, JSP, CGI or ColdFusion on the server-side to process data coming from the form submission

Forms – cont’d

- To make a form you must first use the <form></form> tag and then enclose all the form elements inside the form block
- Attributes:
  - **action**: specifies the URL to be accessed when the form is submitted, usually it is a file which contains ASP or CGI scripts to process the input data
  - **autocomplete**: boolean value to enable form auto-complete feature provided by some browsers. Must be accompanied with *vcard-name* attribute in the type **input** elements of the form
  - **name**: specifies form name to be referenced
  - **method**: can be either “GET” or “POST”, specifies how the input data will be passed to the file specified in *action* attribute
    - *GET*: appends the data to the *action* URL in what is referred to as Query String
    - *POST*: data will be sent as transaction message body, i.e., will not appear to the user in the URL

Form Controls

- These are elements of a form used to gather user input.
- The mostly used and mostly important control is the <input> element (no closing tag)
- Using this tag you can define different input types by manipulating the *type* attribute of this element.

<input>

- Attributes:
  - **Name**: name of this control to be referred to later
  - **Checked**: indicates if the control should appear checked or not when the page loads (for check boxes)
  - **Disabled**: a disabled control element cannot receive focus and cannot be activated by the user (will appear grayed out on the screen), like an inactive button
  - **Maxlength**: the maximum number of characters a user is allowed to enter into a certain *text* field
  - **Readonly**: a *text* field marked as readonly cannot be edited by user, although scripts can modify the content.
  - **Size:** the width of a *text* field, you are encouraged to use CSS to specify widths of text fields and buttons
  - **Src:** the location of the image if the control element is of *image* type
  - **Value**: pre-assigns a value to an input element, important for drop down menus, jump menus and checkboxes. Applies to:
    - Button
    - Checkbox
    - Hidden
    - Radio
    - Submit

<input> - cont'd

The most important attribute is:

- **type:** an input control field can be:
  - Button: submit, reset, none (must use onClick event)
  - Checkbox:can be on or off
  - File: must use enctype="multipart/form-data“ attribute inside the <input> tag
  - Hidden: a hidden text field to carry some data, treat like a variable
  - Image: graphical button
  - Password: a text field but will present asterisks instead of chatacters
  - Radio: a radio button, on\off
  - Reset: clear button, will return all values to defaults
  - Submit: submits the form data and calls the *action* attribute
  - Text: an input text field
  - Label: output text

Multi-Line Input

- Use <textarea></textarea>
- Can be used to input multi-line or output multi-line like notes
- Example

```
<textarea rows="5" cols="50" name="notes">use this area
   for extra notes</textarea>
```

List \ Drop-down menu

- This is a list that allows multiple selections, and will display only 2 lines of data:

```
<select name="majors" size="2" multiple>
        <option value="1">COE</option>
        <option value="2“>ICS</option>
        <option value="3">SE</option>
        <option value="4">MIS</option>
</select>
```

- To make it a dropdown menu remove the *size* and *multiple* attributes
- Use the attribute *selected* with the <option> tag to pre-select an item from the list/menu

Finally…

- Issues with HTML:

Merits:

- Very easy to use & learn
- Presentation technology
- It is the most popular

Shortcomings:

- NOT a data technology
- Poor Searching
- There is no Intelligence of content/data
- We loose meaning association with content
- Data cannot be represented hierarchically
- Limited set of tags

Challenges facing HTML

- Prevalence of sloppy markup practices
- New kinds of browsers: Digital TVs, handhelds, phones and cars
- Pressure to subset HTML for simple clients
- Pressure to extend HTML for richer clients
- Combining HTML with other tag sets: Math, Vector Graphics, E-commerce, Metadata, ...

XHTML

- XHTML: The Extensible Hypertext Markup Language
- A reformulation of HTML in XML with namespaces for HTML 4.0 strict, transitional and frameset DTDs
- Modularises HTML for subsetting/combining with other tag-sets
- Document Profiles provide basis for interoperability guarantees
- Next generation forms features offering improved match to database and workflow applications

HTML vs. XHTML

- Whereas HTML was an application of SGML, a very flexible markup language, XHTML is an application of XML, a more restrictive subset of SGML
- The changes from HTML to transitional XHTML are minor, and are mainly to increase conformance with XML.
- The most important change is the requirement that all tags be well-formed.
- XHTML uses *lower-case* for tags and attributes: This is in direct contrast to established traditions which began around the time of HTML 2.0, when most people preferred uppercase tags.
- In XHTML, all attributes, even numerical ones, must be quoted. (This was mandatory in HTML as well, but often ignored.)
- All elements must also be closed, including empty elements such as img and br. This can be done by adding a closing slash to the start tag: <img … /> and <br />.
- Attribute minimization (e.g., <option selected>) is also prohibited.

XML

- HTML was created as an application of SGML - the Standard Generalized Markup Language (ISO 8879:1986)
- XML is a descendant of SGML, which is easier to implement
- XML requires you to:
  - make tags case-sensitive
  - include end tags e.g. </p> and </li>
  - add a / to empty tags, e.g. <br /> and <hr />
  - quote all attribute values, e.g. <img src="karen.jpg" />
- These make it practical to parse well-formed XML without *apriori* knowledge of the tags
- Old browsers can render XHTML 1.0 provided you follow simple guidelines

The instructor has provided an introduction and tutorial on XML in the "presentations" section of the website on WebCT

References

- W3C HTML Homepage

http://www.w3.org/MarkUp/

- XHTML: The Extensible Hypertext Markup Language by Dave Raggett, at W3C LA event in Stockholm, 24 March 1999

http://www.w3.org/Talks/1999/03/24-stockholm-xhtml/

- “HTML 4.01 Specification”, W3C Recommendation 24 December 1999

http://www.w3.org/TR/html4/

- W3C CSS Homepage

http://www.w3.org/Style/CSS/

- O’Reilly HTML Reference