CpSc 462/662: Database Management Systems (DBMS) (TEXNH Approach)

HTML

James Wang

What is HTML?

- To publish information for global distribution, one needs a universally understood language.
- The publishing language used by the World Wide Web is HTML (from HyperText Markup Language).
- HTML gives authors the means to:
  - Publish online documents with headings, text, tables, lists, photos, etc.
  - Retrieve online information via hypertext links, at the click of a button.
  - Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
  - Include spread-sheets, video clips, sound clips, and other applications directly in their documents.

2

What is a HTML file?

- A file on the web server that follows the HTML specification.
- It contains small markup tags that tell the Web browser how to display the page .
- An HTML file must have an htm or html file extension and can be created using a simple text editor .
- Example:

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage. <b>This text is bold</b>
</body>
</html>
```

3

HTML Elements

- Each HTML file consists of a group elements, which are marked-up by tags.
- HTML includes element types that represent paragraphs, hypertext links, lists, tables, images, etc.
- Each *element type declaration* generally describes three parts: a start tag, content, and an end tag.
- HTML tags are surrounded by angle brackets '<' and '>' .
- HTML tags normally come in pairs like and .
- The first tag in a pair is the start tag, the second tag is the end tag. The text between the start and end tags is the element content
- HTML tags are not case sensitive.
- Some HTML element types have no content. For example, the line break element BR has no content; Such *empty* elements never have end tags.

4

Tag Attributes

- Tags can have attributes which provide additional information to an HTML element.
- Attributes always come in name/value pairs like this: name="value".
- Attributes are always specified in the start tag of an HTML element. For instance, <table border="0">.
- Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed.

5

Basic HTML tags

- Head and body:
  - <head> … </head> <body> … </body>
- Headings:
  - <h1> … </h1> <h2> … </h2> <h3> … </h3>
  - <h4> … </h4> <h5> … </h5> <h6> … </h6>
- Paragraph:
  - This is a paragraph.
  - HTML automatically adds an extra blank line before and after a paragraph.
- Line breaks:
  -
. This will force a line break but won't start a new paragraph.
- Horizontal rule:
  - <hr>. This will create a horizontal rule.
- Comments:
  - <!-- This is a comment. -->

6

Text Format Tags

Tag	Description
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<s>	Deprecated. Use <del> instead
<strike>	Deprecated. Use <del> instead
<u>	Deprecated. Use styles instead

7

"Computer Output" Tags

Tag	Description
<code>	Defines computer code text
<kbd>	Defines keyboard text
<samp>	Defines sample computer code
<tt>	Defines teletype text
<var>	Defines a variable
<pre>	Defines preformatted text
<listing>	Deprecated. Use <pre> instead
<plaintext>	Deprecated. Use <pre> instead
<xmp>	Deprecated. Use <pre> instead

8

Citations, Quotations, and Definition Tags

Tag	Description
<abbr>	Defines an abbreviation
<acronym>	Defines an acronym
<address>	Defines an address element
<bdo>	Defines the text direction
<blockquote>	Defines a long quotation
<q>	Defines a short quotation
<cite>	Defines a citation
<dfn>	Defines a definition term

9

HTML Character Entities

- **Some characters have a special meaning in HTML, like the less than sign (<) that defines the start of an HTML tag. If we want the browser to actually display these characters we must insert character entities in the HTML source.**
- **A character entity has three parts: an ampersand (&), an entity name or a # and an entity number, and finally a semicolon (;).**
  - To display a less than sign in an HTML document we must write: < or <
- **Non-breaking Space: The most common character entity in HTML is the non-breaking space.**
  - Normally HTML will truncate spaces in your text. If you write 10 spaces in your text HTML will remove 9 of them. To add spaces to your text, use the   character entity.
- **For other HTML character entities, please check the HTML specification:**
  http://www.w3.org/TR/html401/sgml/entities.html

10

HTML Links

- **How to create links in HTML document?**
  - HTML uses the <a> (anchor) tag and href attribute to create a link to another document.
  - An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc.
  - The syntax of creating an anchor:
    Text to be displayed
- **Target attribute: Target attribute specifies where you will open your new link.**
  Search
- **Name attribute: Use named anchor to label any text in an HTML document, so that we can create links that can jump directly into this labeled text .**
  Useful Tips Section
   Jump to the Useful Tips Section

11

HTML Links (cont.)

- **A link through an image: You can also use an image as a link:**

  <img border="0" src="buttonnext.gif" width="65" height="38">

- **Mail link in your web page:**

  Send Mail

12

HTML Frames

- **You can display several web pages in the same browser window using frame sets.**

```
<frameset cols="25%,75%">
    <frame src="frame_a.htm">
    <frame src="frame_b.htm">
</frameset>
```

- **A frameset can be embedded into another one.**

```
<frameset rows="50%,50%">
    <frame src="frame_a.htm">
    <frameset cols="25%,75%">
        <frame src="frame_b.htm">
        <frame src="frame_c.htm">
    </frameset>
</frameset>
```

13

More Frames

- **Given the following frame set.**

```
<frameset rows="50%,50%">
    <frame src="frame_a.htm" name = "frameA">
    <frameset cols="25%,75%">
        <frame src="frame_b.htm" name = "frameB">
        <frame src="frame_c.htm" name = "frameC">
    </frameset>
</frameset>
```

- **Display x.htm, y.htm and z.htm in frameA, frameB, and frameC respectively.**

```
<a href ="x.htm" target ="frameA">Frame a</a><br>
<a href ="y.htm" target ="frameB">Frame b</a><br>
<a href ="z.htm" target ="frameC">Frame c</a>
```

- **Inline frame.**

```
<iframe src="inframe.html"></iframe>
```

- **Break out of a frame.**

```
<a href="http://www.somewhere.com/" target="_top">Click here!</a>
```

14

HTML Tables

- **Tables are defined with the <table> tag.**
- **A table is further divided into rows using the <tr> tag.**
- **Each row is divided into data cells by the <td> tag.**
- **A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.**

```
<table border="1">
<tr>
    <td>row 1, cell 1</td>
    <td>row 1, cell 2</td>
</tr>
<tr>
    <td>row 2, cell 1</td>
    <td>row 2, cell 2</td>
</tr>
</table>
```

15

Table Properties

- **Within the <table> tag, you may add width, border, bordercolor, and cellpadding.**

```
<table width="95%" border="10" bordercolor="black"
    cellpadding="5">
```

- **You can also add attributes to the <td> tag.**

```
<td bgcolor="aqua" align="right" valign="top">
<td colspan="2" bgcolor="silver" align="center">
```

- **There are more table properties or elements defined in HTML reference. These include thead, tfoot, tbody, colgroup, col, etc. For more information about table, please read:**

  http://www.w3.org/TR/html401/struct/tables.html

- **Table heading: use <th> to specify headings.**
- **Embedding table:**
  - A table can be embedded in any data cell.

16

HTML Lists

- **Unordered list.**

```
<ul>
    <li>CPSC 462</li>
    <li>CPSC 662</li>
</ul>
```

- **Ordered List.**

```
<ol>
    <li>CPSC 462</li>
    <li>CPSC 662</li>
</ol>
```

- **Definition List:**

```
<dl>
<dt>CPSC 462/662</dt>
    <dd>DBMS Systems</dd>
<dt>CPSC 862</dt>
    <dd>DBMS System Design</dd>
</dl>
```

17

HTML Images

- **You need to use <img> tag and src attribute to display an image.**

```
<img src="tigerpaw.jpg" alt="Tiger Paw">
<img src="tigerpaw.jpg" alt="Tiger Paw">
```

- **Create an image map.**

```
<img src="james.gif" width="100" height="300" usemap="#mymap">
<map id="mymap" name="mymap">
    <area shape="circle" coords="150,50,10" alt="face"
        href="face.htm">
    <area shape="rect" coords="0,60,300,160" alt="body"
        href="body.htm">
</map>
```

18

Embed Flash Video Player

- **You can embed a flash video player in your web page.**

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,40,0" WIDTH="550"
HEIGHT="400" id="myMovieName">
<PARAM NAME=movie VALUE="myFlashMovie.swf">
<PARAM NAME=quality VALUE=high>
<PARAM NAME=bgcolor VALUE=#FFFFFF>
<EMBED src="/support/flash/ts/documents/myFlashMovie.swf" quality=high bgcolor=#FFFFFF WIDTH="550" HEIGHT="400"
NAME="myMovieName" ALIGN="" TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer"></EMBED>
</OBJECT>
```

- **For more and up-to-date information, please check:**

  http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_4150&sliceId=1

19

Embed Real Media Player

- **You can embed a real media player in your web page.**

```
<OBJECT id='rvocx' classid='clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA' width="320" height="240">
<param name='src' value="http://servername/path/to/media.file">
<param name='autostart' value="true">
<param name='controls' value='imagewindow'>
<param name='console' value='video'>
<param name='loop' value="true">
<EMBED src="http://servername/path/to/media.file" width="320" height="240" loop="true" type='audio/x-pn-realaudio-plugin' controls='imagewindow' console='video' autostart="true">
</EMBED>
</OBJECT>
```

- **For more and up-to-date information, please check:**

  http://service.real.com/help/library/guides/extend/embed.htm

20

Embed Windows Media Player

- **You can embed a windows media player in your web page.**

```
<OBJECT ID="MediaPlayer" WIDTH=320 HEIGHT=240
CLASSID="CLSID:22D6f312-B0F6-11D0-94AB-0080C74C7E95"
STANDBY="Loading Windows Media Player components..."
TYPE="application/x-oleobject"
CODEBASE="http://activex.microsoft.com/activex/controls/mplayer/en/nsmp2inf.cab#Version=6,4,7,1112">
<PARAM name="autoStart" value="True">
<PARAM name="filename" value="http://WebServer/MyFile.wvx">
<EMBED TYPE="application/x-mplayer2" SRC="http://WebServer/MyFile.wvx" NAME="MediaPlayer"
WIDTH=320 HEIGHT=240>
</EMBED>
</OBJECT>
```

- **For more and up-to-date information, please check:**

  http://msdn2.microsoft.com/en-us/library/ms983653.aspx

21

Embed QuickTime Player

- **You can embed a windows media player in your web page.**

```
<object classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B" width="320" height="256"
codebase="http://www.apple.com/qtactivex/qtplugin.cab#version=6,0,2,0" align="middle" >
<param name="src" value="sample.mov" />
<param name="autoplay" value="true" />
<embed src="sample.mov" width="320" height="256"
pluginspage=http://www.apple.com/quicktime/download/
align="middle" autoplay="true" bgcolor="black" >
</embed>
</object>
```

- **For more and up-to-date information, please check:**

  http://developer.apple.com/documentation/QuickTime/Conceptual/QTScripting_HTML/QTScripting_HTML_Document/chapter_1000_section_3.html

22

CSS – Manage Web Page Layout

- **What is CSS?**
  - CSS stands for Cascading Style Sheets
  - Styles define how to display HTML elements
  - Styles are normally stored in Style Sheets
  - Styles were added to HTML 4.0 to separate the content of HTML documents from the document's presentation layout
  - External Style Sheets are stored in CSS files so that you do not need to rewrite them in your new HTML file
  - Multiple style definitions will cascade into one

23

Style Cascading Order

- **Style sheets allow style information to be specified in many ways.**
  - inside a single HTML element
  - inside the <head> element of an HTML page
  - in an external CSS file.
  - multiple external style sheets can be referenced inside a single HTML document.
- **Cascading Order**
  - Inline style (inside an HTML element)
  - Internal style sheet (inside the <head> tag)
  - External style sheet
  - Browser default
    - Note: If the external style sheet link is placed below the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet.

24

CSS Syntax

selector {property: value}

- The selector is normally the HTML element/tag you wish to define
- the property is the attribute you wish to change, and each property can take a value.
- The property and value are separated by a colon, and surrounded by curly braces.

- **Examples:**

```
body {color: black}
p {font-family: "sans serif"}
p {text-align:center; color:red}
p { text-align: center;
   color: black;
   font-family: arial }
```

25

Group and Class

- **You can group selectors. Separate each selector with a comma.**

```
h1,h2,h3,h4,h5,h6 { color: green }
```

- **With the class selector you can define different styles for the same type of HTML element.**

Style class:

```
p.right {text-align: right}
p.center {text-align: center}
```

HTML:

```
<p class="right"> This paragraph will be right-aligned. </p>
<p class="center"> This paragraph will be center-aligned. </p>
```

26

Class without Tag

- **You can also omit the tag name in the selector to define a style that will be used by all HTML elements that have a certain class.**

```
.center {text-align: center}
```

```
<h1 class="center"> This heading will be center-aligned </h1>
<p class="center"> This paragraph will also be center-aligned.
</p>
```

27

Styles based on attributes

- **The style rule below will match all input elements that have a type attribute with a value of "text":**

```
input[type="text"] {background-color: blue}
```

- **You can also define styles for HTML elements with the id selector. The id selector is defined as a #.**
  - The style rule below will match the element that has an id attribute with a value of "green":

```
#green {color: green}
```

  - The style rule below will match the p element that has an id with a value of "para1":

```
p#para1 { text-align: center; color: red }
```

28

CSS Comments

- **Comments are used to explain your code, and may help you when you edit the source code at a later date.**
- **A comment will be ignored by browsers.**
- **A CSS comment begins with "/*", and ends with "*/", like this:**

```
/* This is a comment */
p { text-align: center;
   /* This is another comment */
   color: black;
   font-family: arial
}
```

29

Adding Style Sheets to HTML

- **External Style Sheet**
  - An external style sheet is ideal when the style is applied to many pages. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

  - An external style sheet file should not contain any html tags and should be saved with a .css extension.

```
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
```

30

Adding style sheets to HTML (cont.)

- **Internal Style Sheet**
  - An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section by using the <style> tag, like this:

```
<head>
<style type="text/css">
   hr {color: sienna}
   p {margin-left: 20px}
   body {background-image: url("images/back40.gif")}
</style>
</head>
```

31

Adding style sheets to HTML (cont.)

- **Inline Styles**
  - An inline style loses many of the advantages of style sheets by mixing content with presentation.
  - Use this method sparingly, such as when a style is to be applied to a single occurrence of an element.
  - To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property.

```
<p style="color: sienna; margin-left: 20px">
    This is a paragraph
</p>
```

32

HTML Form

- **An HTML form is a section of a document containing normal content, markup, special elements called *controls* (checkboxes, radio buttons, menus, etc.), and labels on those controls.**
- **Users generally "complete" a form by modifying its controls (entering text, selecting menu items, etc.)**
- **Users submit the form to an agent for processing (e.g., to a Web server, to a mail server, etc.).**
- **Users interact with forms through named *controls*.**

33

Form Example

```
<FORM action="http://somesite.com/prog/adduser" method="post">
  <P>
  <LABEL for="firstname">First name: </LABEL>
        <INPUT type="text" id="firstname"><BR>
  <LABEL for="lastname">Last name: </LABEL>
        <INPUT type="text" id="lastname"><BR>
  <LABEL for="email">email: </LABEL>
        <INPUT type="text" id="email"><BR>
  <INPUT type="radio" name="sex" value="Male"> Male<BR>
  <INPUT type="radio" name="sex" value="Female"> Female<BR>
  <INPUT type="submit" value="Send"> <INPUT type="reset">
  </P>
</FORM>
```

First name:
Last name:
email:
Male
Female
Send Reset

34

Controls

- **A control's *"control name"* is given by its name attribute. The scope of the name attribute for a control within a FORM element is the FORM element.**
- **Each control has both an initial value and a current value, both of which are character strings.**
- **The control's *"current value"* is first set to the initial value. Thereafter, the control's current value may be modified through user interaction and scripts.**
- **A control's initial value does not change. Thus, when a form is reset, each control's current value is reset to its initial value.**
- **When a form is submitted for processing, some controls have their name paired with their current value and these pairs are submitted with the form. Those controls for which name/value pairs are submitted are called successful controls.**

35

Control Types

- **buttons:**
  - **submit buttons:** When activated, a submit button submits a form. A form may contain more than one submit button.
  - **reset buttons:** When activated, a reset button resets all controls to their initial values.
  - **push buttons:** Push buttons have no default behavior.
- **checkboxes**
  - Checkboxes (and radio buttons) are on/off switches that may be toggled by the user.
  - Several checkboxes in a form may share the same control name.

36

Control Types (Conti.)

- **radio buttons**
  - Radio buttons are like checkboxes except that when several share the same control name, they are mutually exclusive: when one is switched "on", all others with the same name are switched "off".
- **menus**
  - Menus offer users options from which to choose.
- **text input**
  - The INPUT element creates a single-line input control.
  - The TEXTAREA element creates a multi-line input control.

37

Control Types (Conti.)

- **file select**
  - This control type allows the user to select files so that their contents may be submitted with a form.
- **hidden controls**
  - Authors may create controls that are not rendered but whose values are submitted with a form.
  - Authors generally use this control type to store information between client/server exchanges that would otherwise be lost due to the stateless nature of HTT .
- **object controls**
  - Authors may insert generic objects in forms such that associated values are submitted along with other controls.

38

The FORM element

```
<!ELEMENT FORM - - (%block;|SCRIPT)+ -(FORM) -- interactive form -->
<!ATTLIST FORM
  %attrs;                                -- %coreattrs, %i18n, %events --
  action      %URI;          #REQUIRED -- server-side form handler --
  method      (GET|POST)     GET       -- HTTP method used to submit the form--
  enctype     %ContentType;  "application/x-www-form-urlencoded"
  accept      %ContentTypes; #IMPLIED  -- list of MIME types for file upload --
  name        CDATA          #IMPLIED  -- name of form for scripting --
  onsubmit    %Script;       #IMPLIED  -- the form was submitted --
  onreset     %Script;       #IMPLIED  -- the form was reset --
  accept-charset %Charsets;  #IMPLIED  -- list of supported charsets --
  >
```

- **Example:**

```
<FORM action="http://somesite.com/prog/adduser" method="post">
  ...form contents...
</FORM>
```

39

FORM Attributes

- **action = *uri* [CT]**
  - This attribute specifies a form processing agent.
- **method = get|post [CI]**
  - This attribute specifies which HTTP method will be used to submit the form data set. Possible (case-insensitive) values are "get" (the default) and "post".
- **enctype = *content-type* [CI]**
  - This attribute specifies the content type used to submit the form to the server (when the value of method is "post").
- **accept-charset = *charset list* [CI]**
  - This attribute specifies the list of character encodings for input data that is accepted by the server processing this form.
- **accept = *content-type-list* [CI]**
  - This attribute specifies a comma-separated list of content types that a server processing this form will handle correctly.
- **name = *cdata* [CI]**
  - This attribute names the element so that it may be referred to from style sheets or scripts.

40

The INPUT element

```
<!ENTITY % InputType
  "(TEXT | PASSWORD | CHECKBOX |
    RADIO | SUBMIT | RESET |
    FILE | HIDDEN | IMAGE | BUTTON)"
  >

<!-- attribute name required for all but submit and reset -->
<!ELEMENT INPUT - O EMPTY              -- form control -->
<!ATTLIST INPUT
  %attrs;                              -- %coreattrs, %i18n, %events --
  type        %InputType;    TEXT      -- what kind of widget is needed --
  name        CDATA          #IMPLIED  -- submit as part of form --
  value       CDATA          #IMPLIED  -- Specify for radio buttons and checkboxes --
  src         %URI;          #IMPLIED  -- use client-side image map --
  checked     (checked)      #IMPLIED  -- for radio buttons and check boxes --
  disabled    (disabled)     #IMPLIED  -- unavailable in this context --
  readonly    (readonly)     #IMPLIED  -- for text and passwd --
  size        CDATA          #IMPLIED  -- specific to each type of field --
  maxlength   NUMBER         #IMPLIED  -- max chars for text fields --
  src         %URI;          #IMPLIED  -- for fields with images --
  alt         CDATA          #IMPLIED  -- short description --
  usemap      %URI;          #IMPLIED  -- use client-side image map --
  ismap       (ismap)        #IMPLIED  -- use server-side image map --
  tabindex    NUMBER         #IMPLIED  -- position in tabbing order --
  accesskey   %Character;    #IMPLIED  -- accessibility key character --
  onfocus     %Script;       #IMPLIED  -- the element got the focus --
  onblur      %Script;       #IMPLIED  -- the element lost the focus --
  onselect    %Script;       #IMPLIED  -- some text was selected --
  onchange    %Script;       #IMPLIED  -- the element value was changed --
  accept      %ContentTypes; #IMPLIED  -- list of MIME types for file upload --
  >
```

41

INPUT Attributes

- **type = text|password|checkbox|radio|submit|reset|file|hidden|image|button [CI]**
  - This attribute specifies the type of control to create.
- **name = cdata [CI]**
  - This attribute assigns the control name.
- **value = cdata [CA]**
  - This attribute specifies the initial value of the control.
- **size = cdata [CN]**
  - This attribute tells the user agent the initial width of the control.
- **maxlength = number [CN]**
  - When the type attribute has the value "text" or "password", this attribute specifies the maximum number of characters the user may enter.
- **checked [CI]**
  - When the type attribute has the value "radio" or "checkbox", this boolean attribute specifies that the button is on.
- **src = uri [CT]**
  - When the type attribute has the value "image", this attribute specifies the location of the image to be used to decorate the graphical submit button.

42

Control types created with INPUT

The control type defined by the INPUT element depends on the value of the type attribute:

- **text:** Creates a single-line text input control.
- **password:**
  - Like "text", but the input text is rendered in such a way as to hide the characters (e.g., a series of asterisks).
- **checkbox:** Creates a checkbox.
- **radio:** Creates a radio button.
- **submit:** Creates a submit button.
- **image:** Creates a graphical submit button..
- **reset:** Creates a reset button.
- **button:**
  - Creates a push button. User agents should use the value of the value attribute as the button's label.
- **hidden:** Creates a hidden control.
- **file:**
  - Creates a file select control. User agents may use the value of the value attribute as the initial file name.

43

The BUTTON element

```
<!ELEMENT BUTTON - -
    (%flow;)* -(A|%formctrl;|FORM|FIELDSET)
    -- push button -->
<!ATTLIST BUTTON
  %attrs;                    -- %coreattrs, %i18n, %events --
  name       CDATA           #IMPLIED
  value      CDATA           #IMPLIED  -- sent to server when submitted --
  type       (button|submit|reset) submit -- for use as form button --
  disabled   (disabled)      #IMPLIED  -- unavailable in this context --
  tabindex   NUMBER          #IMPLIED  -- position in tabbing order --
  accesskey  %Character;     #IMPLIED  -- accessibility key character --
  onfocus    %Script;        #IMPLIED  -- the element got the focus --
  onblur     %Script;        #IMPLIED  -- the element lost the focus --
  >
```

44

BUTTON Attributes

- **name = *cdata* [CI]**
  - **This attribute assigns the control name.**
- **value = *cdata* [CS]**
  - **This attribute assigns the initial value to the button.**
- **type = submit|button|reset [CI]**
  - **This attribute declares the type of the button. Possible values:**
    - **submit: Creates a submit button. This is the default value.**
    - **reset: Creates a reset button.**
    - **button: Creates a push button.**

45

Example

```
<FORM action="http://somesite.com/prog/adduser" method="post">
  <P>
  First name: <INPUT type="text" name="firstname"><BR>
  Last name: <INPUT type="text" name="lastname"><BR>
  email: <INPUT type="text" name="email"><BR>
  <INPUT type="radio" name="sex" value="Male"> Male<BR>
  <INPUT type="radio" name="sex" value="Female"> Female<BR>
  <BUTTON name="submit" value="submit" type="submit">
  Send<IMG src="/icons/wow.gif" alt="wow"></BUTTON>
  <BUTTON name="reset" type="reset">
  Reset<IMG src="/icons/oops.gif" alt="oops"></BUTTON>
  </P>
</FORM>
```

First name:
Last name:
email:
Male
Female
Send Reset

46

The TEXTAREA element

```
<!ELEMENT TEXTAREA - - (#PCDATA)      -- multi-line text field -->
<!ATTLIST TEXTAREA
  %attrs;                    -- %coreattrs, %i18n, %events --
  name       CDATA           #IMPLIED
  rows       NUMBER          #REQUIRED
  cols       NUMBER          #REQUIRED
  disabled   (disabled)      #IMPLIED  -- unavailable in this context --
  readonly   (readonly)      #IMPLIED
  tabindex   NUMBER          #IMPLIED  -- position in tabbing order --
  accesskey  %Character;     #IMPLIED  -- accessibility key character --
  onfocus    %Script;        #IMPLIED  -- the element got the focus --
  onblur     %Script;        #IMPLIED  -- the element lost the focus --
  onselect   %Script;        #IMPLIED  -- some text was selected --
  onchange   %Script;        #IMPLIED  -- the element value was changed --
  >
```

47

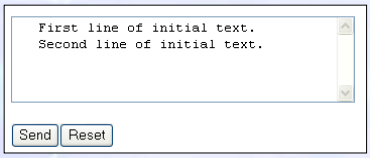
TEXTAREA Attributes

- **name = *cdata* [CI]**
  - This attribute assigns the control name.
- **rows = *number* [CN]**
  - This attribute specifies the number of visible text lines. Users should be able to enter more lines than this, so user agents should provide some means to scroll through the contents of the control when the contents extend beyond the visible area.
- **cols = *number* [CN]**
  - This attribute specifies the visible width in average character widths. Users should be able to enter longer lines than this, so user agents should provide some means to scroll through the contents of the control when the contents extend beyond the visible area. User agents may wrap visible text lines to keep long lines visible without the need for scrolling.

48

Example

```
<FORM action="http://somesite.com/prog/text-read" method="post">
  <P>
  <TEXTAREA name="thetext" rows="20" cols="80">
  First line of initial text.
  Second line of initial text.
  </TEXTAREA>
  </P>
  <P>
  <INPUT type="submit" value="Send"><INPUT type="reset">
  </P>
</FORM>
```



The ISINDEX element

- **ISINDEX is deprecated. This element creates a single-line text input control. Authors should use the INPUT element to create text input controls.**
- **DEPRECATED EXAMPLE:**

The following ISINDEX declaration:

```
<ISINDEX prompt="Enter your search phrase: ">
```

could be rewritten with INPUT as follows:

```
<FORM action="..." method="post">
<P>Enter your search phrase: <INPUT type="text"></P>
</FORM>
```


Form submission method

- **The method attribute of the FORM element specifies the HTTP method used to send the form to the processing agent. This attribute may take two values:**
  - **get:** With the HTTP "get" method, the form data set is appended to the URI specified by the action attribute (with a question-mark ("?") as separator) and this new URI is sent to the processing agent.
  - **post:** With the HTTP "post" method, the form data set is included in the body of the form and sent to the processing agent.
- **The "get" method should be used when the form is idempotent (i.e., causes no side-effects). Many database searches have no visible side-effects and make ideal applications for the "get" method.**
- **If the service associated with the processing of a form causes side effects (for example, if the form modifies a database or subscription to a service), the "post" method should be used.**

Note: *The "get" method restricts form data set values to ASCII characters. Only the "post" method (with enctype="multipart/form-data") is specified to cover the entire [ISO10646] character set.*

Successful controls

- **A *successful control* is "valid" for submission.**
- **Every successful control has its control name paired with its current value as part of the submitted form data set.**
- **A successful control must be defined within a FORM element and must have a control name.**
- **If a control doesn't have a current value when the form is submitted, user agents are not required to treat it as a successful control.**
- **Hidden controls and controls that are not rendered because of style sheet settings may still be successful.**
  - For example

```
<FORM action="..." method="post">
<INPUT type="password" style="display:none"
    name="invisible-password"
    value="mypassword">
</FORM>
```

value of "invisible-password" will be submitted with the form.

Processing form data

- **Step one: Identify the successful controls**
- **Step two: Build a form data set**
  - A *form data set* is a sequence of control-name/current-value pairs constructed from successful controls
- **Step three: Encode the form data set**
  - The form data set is then encoded according to the content type specified by the enctype attribute of the FORM element.
- **Step four: Submit the encoded form data set**
  - Finally, the encoded data is sent to the processing agent designated by the action attribute using the protocol specified by the method attribute.

References

- **http://www.w3.org/TR/html401/**
- **http://www.w3.org/TR/REC-html40/interact/forms.html**
- **http://www.w3schools.com/**