


# CpSc 4620/6620: Database Management Systems (DBMS) (TEXNH Approach)


## HTML Forms

James Wang



# 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*.




# 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




# 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*.




# 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.



# 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.



## 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 HTTP.
- object controls**
  - Authors may insert generic objects in forms such that associated values are submitted along with other controls.

7




## 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 %ContentType; #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>
```


8



## 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.

9




## 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 --
  checked (checked) #IMPLIED -- for radio buttons and check boxes --
  disabled (disabled) #IMPLIED -- unavailable in this context --
  readonly (readonly) #IMPLIED -- for text and password --
  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 %ContentType; #IMPLIED -- list of MIME types for file upload --
>
```


10



## 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.

11




## 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.

12



## 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 --
  >


```

```

<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>


```



13



## 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.


14

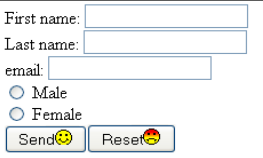



## Example


```

<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>

```




15



## The TEXTAREA element

```

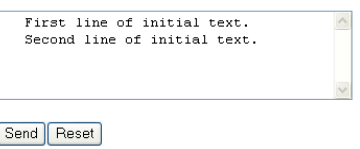
<ELEMENT TEXTAREA - - (#PCDATA) -- multi-line text field -->
<!ATTLIST TEXTAREA
  %attrs;          -- %coreattrs, %i18n, %events --
  name             CDATA          #IMPLIED
  rows             NUMBER        #REQUIRED
  cols             NUMBER        #REQUIRED
  disabled         (disabled)    #IMPLIED -- 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 --
  >


```


```

<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>
    <INPUT type="submit" value="Send"><INPUT type="reset">
  </P>
</FORM>

```






16



## 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.


17

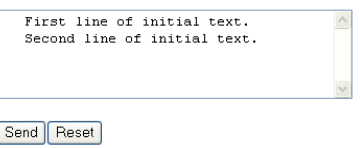



## Example


```

<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>
    <INPUT type="submit" value="Send"><INPUT type="reset">
  </P>
</FORM>

```




18



## 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>
```

19




## 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 [ISO 10646] character set.*

20




## 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.


21



## 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.

22



## More inputs, attribute types, and APIs in HTML5

23




## More input types in HTML 5

- HTML 5 has introduced more input types:**


search	Text with no line breaks	Search field or combo box
tel	Text with no line breaks	A text field or combo box
url	An absolute URL	A text field or combo box
email	An e-mail address or list of e-mail addresses	A text field or combo box
date	A date (year, month, day) with no time zone	A date control
month	A date consisting of a year and a month with no time zone	A month control
week	A date consisting of a week-year number and a week number with no time zone	A week control
time	A time (hour, minute, seconds, fractional seconds) with no time zone	A time control
datetime-local	A date and time (year, month, day, hour, minute, second, fraction of a second) with no timezone offset	A date and time control
number	A numerical value	A text field or combo box or spinner control
range	A numerical value, with the extra semantic that the exact value is not important.	A slider control or similar
color	An sRGB color with 8-bit red, green, and blue components	A color well
image	A coordinate, relative to a particular image's size, with the extra semantic that it must be the last value selected and initiates form submission	Either a clickable image, or a button

24




## Attributes for Types in HTML 5

- HTML 5 has defined more attribute types:
  - The **maxlength** and **minlength** attributes
  - The **size** attribute
  - The **readonly** attribute
  - The **required** attribute
  - The **multiple** attribute
  - The **pattern** attribute
  - The **min** and **max** attributes
  - The **step** attribute
  - The **list** attribute
  - The **placeholder** attribute




25




## Common input APIs

- input . value [ = value ]**
  - Returns the current value of the form control. Can be set, to change the value. Throws an "InvalidStateError" DOMException if it is set to any value other than the empty string when the control is a File Upload control.
- input . checked [ = value ]**
  - Returns the current checkedness of the form control. Can be set, to change the checkedness.
- input . files**
  - Returns a FileList object listing the selected files of the form control. Returns null if the control isn't a file control.




26




## Common input APIs (cont.)

- input . valueAsDate [ = value ]**
  - Returns a Date object representing the form control's value, if applicable; otherwise, returns null. Can be set, to change the value. Throws an "InvalidStateError" DOMException if the control isn't date- or time-based.
- input . valueAsNumber [ = value ]**
  - Returns a number representing the form control's value, if applicable; otherwise, returns NaN. Can be set, to change the value. Setting this to NaN will set the underlying value to the empty string. Throws an "InvalidStateError" DOMException if the control is neither date- or time-based nor numeric.




27




## Common input APIs (cont.)

- input . stepUp( [ n ] )**  
**input . stepDown( [ n ] )**
  - Changes the form control's value by the value given in the step attribute, multiplied by n. The default value for n is 1. Throws "InvalidStateError" DOMException if the control is neither date- or time-based nor numeric, or if the step attribute's value is "any".
- input . list**
  - Returns the datalist element indicated by the list attribute.




28



## References

- <http://www.w3.org/TR/REC-html40/interact/forms.html>
- <https://www.w3.org/TR/html52/sec-forms.html#sec-forms>
- <http://www.w3schools.com/html/>



29