Reading Notes – Forms

Forms are the means by which users may input information into a web page.

HTML forms are helpless without some sort of script processing them in some manner, whether CGI/PerlPHP, or JavaScript; or by using a mailto action which sends an unformatted email.

Two type of scripts that HTML forms get processed by:

1. CGI/Perl or PHP scripts – process info and write it to database or send it via email to someone (store info)

2. JavaScript scripts – used for calculations or browser-specific activities (ie. changing page locations, etc). (calculate info)

Form tag and Form attributes

Description: this tag marks the boundaries of an HTML form. This tag MUST close. There may be more than one form on an HTML page. Each FORM tag will almost always have just ONE submit button (or simulated submit button) associated with it.  There will NEVER be any FORM tag WITHOUT a submit button (or simulated submit button).

<form>

<!-- one or more form elements, with their accompanying HTML context -->

<!-- a submit button -->

</form>

Two attributes for the FORM tag are used when connecting a form for unformatted email or to a CGI script: METHOD and ACTION.

Method - Description: this attribute defines the manner in which the form information is conveyed to a URL for providing unformatted email or conveyed to a URL containing a CGI script.

<form method="post">

<!-- all form element statements -->

</form>

Action - Description: the ACTION attribute of the FORM tag connects that form to a URL for providing "insecure" email, which means that the form data is NOT encrypted, but it does NOT mean that any harm will come to your computer or to the person's computer receiving the email of the form data.

<form method="post" action="mailto:srubin45@comcast.net"

enctype="text/plain">

<!-- all form element statements -->

</form>

Note: enctype="text/plain" should be included in your form statement when sending unformatted email. It specifies the MIME type of the data to be sent by the post method.

Input tag:

Description: the INPUT tag creates form elements within the confines of a FORM tag. Note: INPUT tags may ONLY be inserted as child elements of a FORM tag; they won't work outside of a FORM tag.

Form elements, such as radio buttons, checkboxes, and submit buttons, are created and inserted into a FORM using the INPUT tag. The INPUT tag doesn't require a closing tag.

Every INPUT tag, however, REQUIRES the TYPE attribute, which tells the browser what type of form element you want.

Attribute: **TYPE**  
Value: **text** (default), **radio**, **checkbox**, **password**, **submit**, **image**, **reset**, **button**, **hidden**, **file**  
Description: the TYPE attribute sets the INPUT tag to become the desired type of form element.  
Example (abbreviated): **<input type="submit" />** (makes a submit button)

Attribute: **NAME**  
Value: any name, NO SPACES IN THE NAME (and following the naming convention rules laid out in previous modules)  
Description: the NAME attribute of the INPUT tag identifies that INPUT for analysis in a script. Most INPUT tags require the NAME attribute.

Attribute: **VALUE**  
Value: any plain text (alphanumeric) information; spaces are permissible although not always recommended.  
Description: the VALUE attribute of the INPUT tag allows you to assign some piece of information to a named INPUT; this information can be retrieved by a script upon submission of the form. (Note that HTML5 does not require the ending / for the input statement).

Example (INPUT tag only):

<input type="checkbox" name="pets" value="dog" />

Checkboxes:

 The INPUT tag creates ONLY the form element itself; it does not create any context for the form element. Context for a form element must be created using ordinary HTML code and text.

<form>

<p>Choose one or more of your favorite pets: <br />

<input type="checkbox" name="pets" value="dog" /> Dog<br />

<input type="checkbox" name="pets" value="cat" /> Cat<br />

<input type="checkbox" name="pets" value="bird" /> Bird

</p>

</form>

Notice how all of the checkboxes in this group have the same NAME, while the VALUEs are all different?

An exception to this rule is when using a php script with checkboxes. There must be NOTHING else on the HTML page which uses that NAME, however; each named element on an HTML page must have a UNIQUE name (a named element being anything using the NAME attribute, regardless of the tag).

Submit Buttons:

<input type="submit" value="Submit Me, Please" />

The VALUE of a submit button is the text which appears on the button itself. Here is the above submit button in context with the set of checkboxes from the previous example:

Radio Buttons:

Radio buttons are coded identically to checkboxes, with one exception: the TYPE attribute must be set equal to **"radio"**.

Radio buttons behave differently from checkboxes in one important respect: the user may check only ONE from a group of radio buttons, whereas they may check as many as they like from a group of checkboxes. That is the essential nature of radio buttons.

It is important that all radio buttons in a group have the same NAME!

Once you've checked all the buttons once, you can't UNCHECK them, nor can you get them to behave like proper radio buttons.

Method Attribute:

You will notice that a question mark (?) has been added to the end of the URL for the current page, and see so-called "name-value pairs" appended onto the end of the URL following the question mark (in the example above, the name-value pairs are **"pets=cat"** and **"colors=red"**).

When a FORM is submitted, the various form elements in the FORM may be converted into name-value pairs derived from the values of their NAME and VALUE attributes, e.g. **NAME=VALUE**. When there is more than one

name-value pair in a FORM, the ampersand (&) character is inserted between the pairs. These name-value pairs may be looked at by a script analyzing a FORM.

The text comprised of the question mark followed by name-value pairs in the web browser's location bar is called a "query string".

get and post:

When a FORM has its METHOD attribute set equal to **"get"** (the default value), clicking on the submit button causes the browser to extract all of the chosen name-value pairs from the FORM; these name-value pairs are then converted into a query string which is appended onto the end of the URL in the web browser's location bar.

When a FORM has its METHOD attribute set equal to **"post"**, the form information is submitted in a manner which does not need further explanation at this time. With **method="post"**, the submission process is invisible to the user. –secure: use with mailto and sensitive data

Input tag: Text and Password

type=”text”

the NAME attribute, and the SIZE attribute; the VALUE attribute is not necessary with a text INPUT field.

Attribute: **SIZE**  
Value: an integer representing the number of characters wide the TEXT INPUT field will be (using the browser's default monospace font, usually Courier or Courier New 10 point).  
Description: The SIZE attribute of the INPUT tag determines the width of a text input field.

<input type="text" size="30" name="fred" />

If you submit this form, you will notice that the name-value pair in the query string is**"firstName=somename"** (where **"somename"** is whatever you have typed). The VALUE of a text INPUT field is whatever the user types into the text field; this is why you don't need to set a VALUE attribute for a text INPUT field.

Attribute: **MAXLENGTH**  
Value: an integer representing the maximum number of characters the user may type into this particular text INPUT field.  
Description: the MAXLENGTH attribute restricts the number of characters a user may type into a text INPUT field.

<form>

<p>Login Name:<br>

<input type="password" size="30" maxlength="30" name="thePassword" /></p>

<input type="submit" value="Submit Me, Please" />

</form>

Remember, the password INPUT field doesn't actually have any real security attached to it beyond obscuring the letters with bullets. Real security requires a secure server connection using the "https" scheme and an appropriate server. HTML forms themselves have NO security built into them.

Checked and Reset

Radio buttons and checkboxes may be pre-checked by using the CHECKED attribute of the INPUT tag.

<input type="checkbox" name="pets" value="cat" checked="checked" />

**Remember**: only ONE radio button may be pre-checked, while as many checkboxes as you like may be pre-checked.

Reset Input:

<input type="reset" value="Clear Me, Please" />

clears info on page (resets to default)

Image and Button:

The IMAGE INPUT allows us to substitute a GIF or JPEG for a regular submit button, while the BUTTON INPUT allows us to create a submit-like button that does nothing at all. The uses for the IMAGE INPUT seem obvious, but the uses for the BUTTON INPUT are only open to those who use JavaScript scripts to add functionality to plain HTML elements.

The image INPUT requires that the TYPE attribute for the INPUT tag be set equal to **"image"**. In addition to the TYPE attribute, you also need to use the SRC, WIDTH, and HEIGHT attributes of the IMG tag.

<input type="image" src="mod8ggo.gif" width="50" height="50" />

To prevent the appearance of the border, you must set the BORDER attribute of the INPUT tag equal to **"0"** (zero).

The image INPUT may NOT be used as a substitute for anything other than the submit button, however.

Button input:

The button INPUT is coded identically to the submit and reset buttons; the only difference is that the button INPUT has no specific function, but acts as a generic button for use in conjunction with a script.

<form>

<input type="button" value="Do-Nothing Button" />

</form>

In addition to the button INPUT, there are two final variations of the INPUT tag, **FILE** and **HIDDEN**. The file INPUT allows the programmer to create a text field which may be used to find and submit files to a server. The hidden INPUT is invisible to the user entirely and has many uses. We will see a good use of the INPUT tag and **HIDDEN** for PHP in assignment 8.

Select and Option tags:

The SELECT and OPTION tags are used to create pulldown menus and multiple-choice scrollboxes for forms

To create a pulldown menu, one must first lay a SELECT tag within a FORM.

The SELECT tag has one important attribute, NAME, which is just like the NAME attribute of the INPUT tag.

<form>

<select name="colors">

</select>

</form>

ou must create one OPTION tag for every desired option in the pulldown menu.

<form>

<select name="colors">

<option>Red</option>

</select>

</form>

If the value attribute is NOT specified, the value of the option is set to the content of the <option> tag

 You may pre-select any one of the other options in the pulldown menu by adding the SELECTED attribute to the appropriate OPTION tag.

Attribute: **SELECTED**  
Value: **SELECTED**  
Description: whichever OPTION tag contains the SELECTED attribute becomes the default choice in the pulldown menu. Note that XHTML requires a value of "selected", whereas HTML requires no value.

<option value="blue" selected="selected">

The VALUE attribute of the OPTION tag does NOT have to be the same value as the text appearing in the pulldown menu itself. This is a particularly useful feature when creating pulldown menus which will be used for navigation on a website.

Scrollbox

 two attributes to the SELECT tag: SIZE and MULTIPLE. The SIZE attribute converts the SELECT tag into a scrollbox, while the MULTIPLE attribute allows the user to choose more than one item at a time.

Attribute: **SIZE**  
Value: an integer representing the number of lines tall the scrollbox will be; this integer should always be at least **"4"** (it may be greater)

<select name="fred" size="4">

Attribute: **MULTIPLE**  
Value: none  
Description: allows a scrollbox to record multiple choices by the user; this functionality requires that the user hold down the "Command" key (Mac) or the "Control" key (PC) while clicking in order to make non-contiguous multiple choices. Holding down the "shift" key while clicking allows the user to choose a contiguous range of multiple choices.

<select name="fred" size="4" multiple="multiple">

Just as with the regular pulldown menu, the SELECTED attribute may be added to any OPTION tag which you wish to make pre-selected; unlike the pulldown menu, more than one OPTION tag may be pre-selected.

<option value="violet" selected="selected">Violet</option>

TextArea tag:

The TEXTAREA tag creates a multiple-line text input field within a FORM; here is an example.

TEXTAREA requires three attributes: NAME, COLS, and ROWS. NAME names the TEXTAREA so that it can be accessed by the script, COLS determines the width of the TEXTAREA, and ROWS determines the height of the TEXTAREA. The TEXTAREA tag always closes.

<form>

<p>Tell me your life story:<br>

<textarea name="lifeStory" cols="40" rows="5">

Type your life

story here…

</textarea></p>

<input type="submit" value="Submit This Puppy" />

</form>

 The VALUE of a TEXTAREA is whatever text is placed between the opening and closing TEXTAREA tags.

HTML 5

The new input types:

* email
* url
* number
* range
* Date pickers (date, month, week, time, datetime, datetime-local)
* search
* color

The new form elements:

* datalist
* keygen
* output

New form attributes:

* autocomplete
* novalidate

New input attributes:

* autocomplete
* autofocus
* form
* form overrides (formaction, formenctype, formmethod, formnovalidate, formtarget)
* height and width
* list
* min, max and step
* multiple
* pattern (regexp)
* placeholder
* required

**NOTE: In XHTML all form attributes are required to have values. In HTML5, attributes that are either 'on' or 'off' can be specified with no value.**  
  
Example:   
  
XHTML: <input type="checkbox" name="email" checked="checked" />  
HTML5: <input type="checkbox" name="email" checked>