LESSON: HTML FORMS

DYNAMIC CONTENT: HTML FORMS

Categories of website

Basically there are two categories of websites, they are:

1. Static website
2. Dynamic website

Static Website

A static website is one with no interactivity and is usually just a presentation of information with content not expected to change frequently. Static websites (sometimes called a flat page) are primarily hard coded in Hypertext Markup Language (HTML). There are many static websites on the Internet. Static websites are the cheapest to develop and host, and many smaller companies still use these to get a web presence.

Dynamic Website

A dynamic website is an interactive website that may have frequently changing information. A dynamic website can involve any level of interactivity from a simple feedback form to a database that personalizes the website for each individual visitor. Dynamic web pages were first introduced in 1995 with the creation of JavaScript. Dynamic interactive content can be delivered using either Server-side technology (PHP, Perl, Active Server Pages – ASP, Coldfusion etc) or Client-side technology (JavaScript, Flash etc) that are used to develop dynamic websites. Too little interactivity on a website and users may lose interest; too much interactivity and they may feel overwhelmed.

Dynamic sites can be more expensive to develop initially, but the advantages are numerous. At a basic level, a dynamic website can give the website owner the ability to simply update and add new content to the site.

HTML FORMS AND FORM PROCESSING

Forms in HTML are much like forms in real life. HTML forms are used to pass data by the browser to a web server for processing. Typically a user fills out a form by filling in some fields with text and/or selecting from lists of options. The user must fill in a set of fields on the form and post it somewhere. This processing of incoming data is usually handled by a script or program written in php, java, perl, asp or another language that manipulates text, files, and information.

The forms themselves are not hard to code. They follow the same constructs as other HTML tags. What could be difficult is the program or script that takes the information submitted in a form and processes it.

HOW FORMS WORK

There are two parts to a working form. The first part is the form that you see on the page itself. Forms are made up of buttons, text fields, and pull-down menus (collectively known as form controls) used to collect information from the user. Forms may also contain text and other elements.

The other component of a web form is an application or script on the server that processes the information collected by the form and returns an appropriate response.

Be careful not to nest form elements or allow them to overlap. A form element must be closed before the next one begins.

Forms are added to web pages using the form element. The form element is a container for all the content of the form, including some number of form controls, such as text entry fields and buttons. In addition to being a container for form control elements, the form element has some attributes that are necessary for interacting with the form-processing program on the server.

The browser understands that when an HTML form is submitted to a web application, the values that the user input to the system must somehow accompany the request itself. The request is a normal HTTP request, there is no separate protocol for form submission. The designer of the HTML form must include information in the form itself that tells the browser where the submission should be sent, and whether to use an HTTP GET or POST request.

FORM ATTRIBUTES

Method attribute: The method attribute specifies how the information should be sent to the server. There are only two methods for sending this encoded data to the server: POST or GET indicated using the method attribute in the form element.

Action attribute: The action attribute provides the location (URL) of the application or script (sometimes called the action page) that will be used to process the form.

Get: This is the default action. If a URL is specified as the value of the action, a ? followed by the input is sent to the server or whatever processing agent is specified.

Post: In this case the input is embedded in the form and sent to the processing agent.

THE FORM TAG

All form elements must be placed between a <FORM> and </FORM> tags. There can be multiple forms within a single HTML document - each form is a separate entity and the contents of only one form are submitted as query (although a single form may contain many elements). The FORM tag has required attributes METHOD and ACTION.

A form can contain input elements like text fields, checkboxes, radio buttons, submit buttons and more. A form can also contain select lists and textarea elements.

The syntax for <form> tag is:

<FORM METHOD=”post” ACTION=””>

input elements

</FORM>

The METHOD attribute of a form tag specifies the HTTP request method that should be used when the contents of the form are submitted as part of an HTTP request. Typically the request method is either GET or POST.

The ACTION attribute specifies where the contents of the form should be sent. This is typically a URL (could be relative or absolute), although sometimes people use a mailto: URL so that when the user submits the form it is sent to as an email message to the specified email address.

Form Elements (Fields)

Between the <FORM> and </FORM> tags you define the text and fields that make up the form. You can include HTML tags inside a form to format the text, and there are a number of new tags that are used to define form fields.

A form in XHTML or HTML is by far the most common way to use a form online. The following elements can make up the user-inputting portion of a form:

· Text fields

  _ text

  _ password

  _ textarea

· Radio Buttons

· Checkboxes

· Select

  _ Drop-down list

· Buttons

  _ Submit Button

  _ Reset Button (clear)

INPUT FIELDS

Input fields allow the user to type in a string value as input or to click on buttons or menus to select specific options. These fields are all created with the INPUT tag, the required attribute TYPE indicates what specific kind of input field is being created.

The INPUT tag also has a required attribute named NAME that establishes the name of the field being created. This name is important, since it will be sent to the script program along with the value the user provides. Within a single form, every input must have a unique name.

Below is a description of many of the types of input fields, see any HTML reference for a complete description.

TEXT INPUT FIELD

TEXT is the most common type of form input field used for entering/typing a single word or line of text. It is added to the form using the input element with its type attribute set to text. There are some additional attributes that are used together with TEXT input field.

Name: The name attribute is required for identifying the variable name.

Value: The value attribute specifies initial/default value text that appears in the field when the form is loaded. When you reset a form, it returns to this value.

Size: By default, browsers display a text-entry box that is 20 characters wide, but you can change the size of the box drawn by the browser using the size attribute.

Maxlength: By default, users can type an unlimited number of characters in a text field regardless of its size (the display scrolls to the right if the text exceeds the character width of the box). You can set a maximum character limit using the maxlength attribute if the forms processing program you are using requires it.

Note: Value, Size and Maxlength attributes are optional.

Example :

<FORM METHOD=”post” ACTION=””>

Your Name: <INPUT TYPE=TEXT NAME="Name" SIZE=”50”>

Your Age: <INPUT TYPE=TEXT NAME="Age" SIZE=”50”>

</FORM>

Output/Display in a Browser:

Your Name:

Your Age:

PASSWORD TEXT ENTRY FIELD

A password field works just like a text entry field, except the characters are obscured from view using asterisk (*) or bullet (•) characters, or another character determined by the browser.

Example:

<FORM METHOD=”post” ACTION=””>

Login Screen

Username: <input type="TEXT" name="usr">

Password: <input type="password" name="pwd">

<input type=submit value="login">

</FORM>

Output/Display in a Browser:

Login Screen

Username:

Password: **********

login

MULTILINE TEXTAREA FORM FIELDS

At times, you'll want your users to be able to enter more than just one line of text. For these instances, use the textarea element that is replaced by a multi-line, scrollable text entry box when displayed by the browser.

Unlike the empty input element, the textarea element has content between its opening and closing tags.

The content of the textarea element is the intial content of the text box when the form is displayed in the browser. In addition to the required name attribute, the textarea element uses the following attributes:

Rows: Specifies the number of lines of text the area should display. Scrollbars will be provided if the user types more text than fits in the allotted space.

Cols: Specifies the width of the text area measured in number of characters.

You can create a multiline text field with the TEXTAREA tag - you don't use the INPUT tag to create this kind of field. The TEXTAREA tag requires the NAME attribute and supports the attributes ROWS and COLS (to define the size of the box drawn on the screen). Unlike the INPUT tag - the TEXTAREA tag has an end tag, so you need to include a </TEXTAREA > tag. Everything included between the <TEXTAREA> and </TEXTAREA> tags is the initial value of the multiline text box. The user can delete, edit or add to this initial text.

Example:

<FORM METHOD="post" ACTION="">

Please enter your address in the space provided:

<TEXTAREA NAME=address COLS=40 ROWS=5>

Your address goes here…

</TEXTAREA>

</FORM>

SIMPLE DROP-DOWN LIST (SELECT FORM FIELDS)

The SELECT and OPTION tags can be used to create pull-down menus and scrolling lists of choices. The SELECT tag must include a NAME attribute (this is the name of the form field sent by the browser). The SELECT tag supports the optional attribute SIZE which controls the number of options displayed at once. If the SIZE attribute is omitted or set to the value 1, the browser will display a pull-down menu. If the attribute has a value greater than 1, the browser will display a scrolling list of options.

Between the <SELECT> tag and the corresponding end tag </SELECT> there can be number of OPTION tags

- one for each choice you want displayed. The text that follows the OPTION tag is the text that will be displayed by the browser. The value sent by the browser if a choice is selected can be specified with a

VALUE attribute, if none is specified the value sent will be the text that follows the OPTION tag.

Example 1:

<FORM METHOD=”post” ACTION=””>

<select name="Universities">

<option value="MU">Moi University</option>

<option value="UoN">University of Nairobi</option>

<option value="KU">Kenyatta University</option>

<option value="JKUAT">Jomo Kenyatta University</option>

<option value="EU">Egerton University</option>

<option value="MASU">Maseno University</option>

</select>

</FORM>

Drop-Down List with a pre-selected value

How to create a drop-down list with a pre-selected value using selected attribute.

Example 2:

<FORM METHOD=”post” ACTION=””>

<select name="cars">

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="pre" selected="selected">Premio</option>

<option value="audi">Audi</option>

</select>

</FORM>

Example 3:

Here is another example, this one will appear as a scrolling list (the SIZE attribute controls the size), and the

OPTION values are set inside the OPTION tags:

<FORM METHOD=”post” ACTION=””>

Select a Public University:

<select name="Universities" size=3>

<option value="MU">Moi University</option>

<option value="UoN">University of Nairobi</option>

<option value="KU">Kenyatta University</option>

<option value="JKUAT">Jomo Kenyatta University</option>

<option value="EU">Egerton University</option>

<option value="MASU">Maseno University</option>

</select>

</FORM>

RADIO AND CHECKBOX BUTTONS

Both checkbox and radio buttons make it simple for your users/visitors to choose from a number of provided options. They are added using the input element. However, each of them serve distinct functions.

Radio buttons

A form control made up of a collection of radio buttons is appropriate when only one option from the group is permitted, or, in other words, when the selections are mutually exclusive (such as Yes or No, or Male or Female). When one radio button is "on," all of the others must be "off". Radio buttons let a user select ONLY ONE one of a limited number of choices:

The way the browser knows which radio buttons are part of a group of choices is by looking at the NAME attribute. All radio button inputs that form a group must have the same NAME, and each should have a different value for the VALUE attribute (this is how the script program will find out which was picked).

Note: When a user clicks on a radio-button, it becomes checked, and all other radio-buttons with equal name become unchecked.

Radio buttons are added to a form with the input element with the type attribute set to radio. The name attribute is required. Here is the syntax for a minimal radio button:

<input type="radio" name="variable">

Example 1:

<FORM METHOD="post" ACTION="">

<input type="radio" name="gender" value="male" /> Male

<input type="radio" name="gender" value="female" /> Female

</FORM>

Example 2 with an initial checked option using checked attribute:

<FORM METHOD="post" ACTION="">

Select Mode of Study:

<INPUT TYPE=RADIO NAME=study Value=full> Full Time

<INPUT TYPE=RADIO NAME= study Value=part> Part Time

<INPUT TYPE=RADIO CHECKED NAME= study Value=school> School Based

</FORM>

Checkbox buttons

Inputs of type CHECKBOX present a user with an item that can be selected or deselected. This makes them the right choice for lists in which more than one selection is okay. Each checkbox has a name and a value and can be initially selected/deselected. The difference from a radio button, as we've already noted, is that more than one checkbox may be checked at a time. The value of every checked button will be sent to the server when the form is submitted. When checkboxes are grouped together, it is possible to select as many or as few from the group as desired.

Checkboxes are added using the input element with its type set to checkbox. As with radio buttons, you create groups of checkboxes by assigning them the same name value.

<input type="checkbox" > defines a checkbox.

Example 1:

<FORM METHOD="post" ACTION="">

<input type="checkbox" name="vehicle" value="Bike"> I have a bike

<input type="checkbox" name="vehicle" value="Car"> I have a car

</FORM>

Note: To set a particular checkbox to be initially checked use the attribute CHECKED inside that INPUT Tag.

Example 2:

<FORM METHOD="post" ACTION="">

Select the Public Universities you are applying:

<INPUT TYPE=CHECKBOX CHECKED NAME=MU> Moi University

<INPUT TYPE=CHECKBOX NAME=UON> University of Nairobi

<INPUT TYPE=CHECKBOX NAME=KU>Kenyatta University

<INPUT TYPE=CHECKBOX NAME=JKUAT> Jomo Kenyatta University

<INPUT TYPE=CHECKBOX NAME=EGER> Egerton University

<INPUT TYPE=CHECKBOX NAME=MAS> Maseno University

<INPUT TYPE=CHECKBOX NAME=MMUST> Masinde Muliro University

</FORM>

FORM SUBMISSION (SUBMIT AND RESET BUTTONS)

There are a number of different kinds of buttons that can be added to web forms. The most fundamental is the submit and Reset buttons. Both submit and reset buttons are added using the input element. Because these buttons have specific functions that do not include the entry of data, they are the only form control elements that do not require the name attribute.

By default, the submit button displays with the label “Submit” or “Submit Query” and the reset button is labeled “Reset.” Change the text on the button using the value attribute.

Submit Buttons

When clicked, the submit button immediately sends the collected form data to the server for processing.

Another type of input field is the SUBMIT type, this tells the browser to draw a button. When the user clicks on the Submit button, the browser knows to submit the contents of the form to the URL specified as the ACTION in the form tag.

Submit inputs support the attribute VALUE which is the string you want displayed in the button. If you don't include a VALUE attribute, the browser will put the string "Submit" in the button. Note that the NAME attribute is not required for a submit input.

Example:

<FORM METHOD=”post” ACTION=””>

Your Name: <INPUT TYPE=TEXT NAME=Name>

Your Age: <INPUT TYPE=TEXT NAME=Age>

Your Gender: <INPUT TYPE="RADIO" NAME="gender" VALUE="male" > Male

<INPUT TYPE="RADIO" NAME="gender" VALUE="female" > Female

<INPUT TYPE=SUBMIT VALUE=Submit>

<INPUT TYPE=RESET VALUE="Clear Form">

</FORM>

How SUBMIT Button Works

When the user presses on a submit button the following happens:

· The browser uses the FORM method and action attributes to construct a request.

· A query string is built using the (name,value) pairs from each form element. Form field name and values are concatenated together with an '=' separating each name and value, and '&' separating the name/value pairs. The query string would look something like:

name1=value1&name2=value2&name3=value3.

There are a few rules to keep in mind:

· For each checkbox selected the name,value pair is sent.

· For all checkboxes that are not selected - nothing is sent.

· A single name,value pair is sent for each group of radio buttons.

· Unnamed elements like submit and clear buttons don't generate a name/value pair to be part of the query.

Reset Buttons

The Reset button goes through the form and resets all the form controls to the state they were in when the form loaded. An input of type RESET tells the browser to create a button that the user can press to clear all form fields (set to the default values). You can specify the text to appear in the button with the VALUE attribute.

Example:

<FORM METHOD=”post” ACTION=””>

Your Name: <INPUT TYPE=TEXT NAME=Name>

Your Age: <INPUT TYPE=TEXT NAME=Age>

Your Gender: <INPUT TYPE="RADIO" NAME="gender" VALUE="male" > Male

<INPUT TYPE="RADIO" NAME="gender" VALUE="female" > Female

<INPUT TYPE=SUBMIT VALUE=Submit>

<INPUT TYPE=RESET VALUE=Reset>

</FORM>