**HTML Forms**

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.), before submitting the form to an agent for processing (e.g., to a Web server, to a mail server, etc.) A form is defined with the `<form>` tag.

This form has all the required elements for a form: To make this form work, we'll add the `<FORM ACTION="...">` attribute, which indicates the CGI program to send the form data to. We'll also use NAME in the `<INPUT ...>` tags, and add a Submit button:

```
<FORM ACTION="/cgi-bin/mycgi.pl">

Start the form here. The ACTION attribute, which is required with every `<FORM ...>` tag, is used with CGI, discussed below.

<INPUT NAME="color">

Data entry field. `<INPUT ...>` creates most types of form fields, but `<TEXTAREA ...>` and `<SELECT ...>` also create certain types.

<INPUT TYPE=SUBMIT>

This is a second input which adds a submit button.

</FORM>

End the form here.
```

Most forms require either the ACTION or NAME attributes to do anything meaningful. (The `<FORM ...>` attribute is always required.)

**CGI: The Common Gateway Interface**

Essentially, CGI is the connection (or interface) between a form on a Web page and the Web server. Web pages cannot interact directly with the reader. In fact, until JavaScript came along, Web pages had no way of interpreting reader reaction except through interaction with the server they were running on. This interaction is done through scripts and programs that use common gateway interface to create interactive programs on your Web pages.

**Controls (Input Types)**

Users interact with forms through named 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. Please consult the definition of each control for information about initial values and possible constraints on values imposed by the control. In general, a control's "initial value" may be specified with the control element's value attribute.
Controls, examples:

- **Text.** User may enter text
  - TEXTAREA (all one word): This tells the form command above that here is a form item: This one will be a text area box.
  - NAME: The information the reader puts in this box will be sent to the online database denoted by whatever name you use.
  - ROWS tells the computer how many rows of text it will accept.
  - COLS tells the computer how many characters will be in each row.

- **Radio button.** These small circular buttons can be used in polls or information forms to ask the user their preference. When you set up a group of them, you can only select one choice.
  - INPUT: This tells the computer "I am a form element"
  - TYPE: This tells the computer what type of form item it will be. In this case, it's a radio button.
  - NAME: This names the category the button is in on your form page. Let's say you have six different choices under one heading. Like six animals all under the heading, "Favorite Animal." "Animal" would be the category. It's the heading of the group of radio buttons.
  - VALUE is the name assigned to the button. Like in the Animal example above, you have six buttons each labeled with six different animals. Well, you would give one the value of dog, cat, wombat, etc.

- **Checkbox group.** Groups of check boxes are similar to radio buttons except they are not grouped, so multiple boxes can be selected at the same time. The checkbox is an exact clone of the radio button except for two distinct features:
  - The item it places on the page is square and it is marked with a check when chosen.
  - You can check as many as you'd like.

- **Drop-down Select Boxes.** These are a cool way to get a user to select an option. They perform the same thing as radio buttons, it’s just the way they look that’s different. Most of the options available are not in view until the user gets intimate with the box and clicks on it. The rest of the options will then pop-up below the box.
  - SELECT tells the computer another form is going here. This time it's a SELECT or Pop-Up form.
  - NAME: Same as above. This is the heading of the form item. It denotes how the results of the reader will arrive at your e-mail box. In this case it will say; "Favorite_Color=" and then the reader's choice.
o SIZE denotes the size of the box. Here, 1 means one line or item is shown. Try putting two there if you'd like to see what it does. I prefer just one. More than one item tends to defeat the purpose of the Pop-Up Box.

o OPTION SELECTED denotes which option will appear in the box. Note on the page that "Blue" is visible.

o OPTION denotes another choice that will be visible when you click on the item.

o /SELECT ends the drop-down.

- **Submit and Reset Buttons.** Once the reader has filled in all the information you want, you need a finishing button to click on to send it all to your email address (or wherever you’ve said at the start). You can also clear all the info in the form out with the reset button.

  o <INPUT TYPE="submit">

  o <INPUT TYPE="reset">

### Reference: Form Tags

- `<form>` Defines a form for user input
- `<input>` Defines an input field
- `<textarea>` Defines a text-area (a multi-line text input control)
- `<label>` Defines a label to a control
- `<fieldset>` Defines a fieldset. This element allows authors to group thematically related controls and labels. Grouping controls makes it easier for users to understand their purpose while simultaneously facilitating tabbing navigation for visual user agents and speech navigation for speech-oriented user agents. The proper use of this element makes documents more accessible.

  o `<legend>` Defines a caption for a fieldset. This element allows authors to assign a caption to a fieldset. The legend improves accessibility when the fieldset is rendered non-Visually.

- `<select>` Defines a selectable list (a drop-down box)
- `<optgroup>` Defines an option group
- `<option>` Defines an option in the drop-down box
- `<button>` Defines a push button
"Mailto:" Forms

A common question is if a form can be created that sends its results to an email address instead of to a CGI. The short answer is "No". Some browsers do support this idea, but the results are URL-encoded and so are difficult to read. Many browsers don't support mail forms at all.

The best way to have form results emailed to you is to use a Common Gateway Interface (CGI) script. There are many free "form-mailer" CGI programs available. There's a good chance your web server already has a form-mailer CGI ready to use -- many web service providers include this as a freebie for their client. Check with your web administrator.