

HTML 4.0 fill-out forms



John W. Shipman

2006-05-18 14:21

Table of Contents

1. Introduction to Web forms	1
1.1. How to get this publication	1
2. Form elements	2
3. Text fields	2
4. Password fields	2
5. Checkboxes	2
6. Radiobuttons	3
7. The submit button	3
8. The reset button	4
9. Pull-down menus	4
10. Scrolling menus	4
11. Multi-line text areas	5
12. Image maps	5
13. Relevant URLs	5

1. Introduction to Web forms

Unlike most HTML work, using a fill-out form requires programming: you must write a downstream *CGI script* to process the data from the form. This means that you must either be a programmer, or hire one. The programming can be done in any language that can evaluate environmental variables, and the programmer must understand how CGI scripts work (see the section Relevant URLs, below).

The interface between a form and the CGI script that handles the form is conceptually simple. When the user clicks on the *Submit* button on a Web form, the associated CGI script starts running, and receives the contents of the form as a set of (*name*, *value*) pairs. Each *name* designates one of the elements on the form, and the corresponding *value* describes what value the user has selected for that form element.

Forms design itself does not require programming, but the forms designer must coordinate with the script programmer on the exact set of names, and what the values mean.

1.1. How to get this publication

This publication is available in Web form¹ and also as a PDF document². Please forward any comments to tcc-doc@nmt.edu.

¹ <http://www.nmt.edu/tcc/help/pubs/htmlforms/>

² <http://www.nmt.edu/tcc/help/pubs/htmlforms.pdf>

2. Form elements

Any Web page can include a form between `<form>...</form>` tags. In general, a form looks like this in HTML:

```
<form method='post' action='url'>
  place form elements here
</form>
```

where **url** points to the downstream CGI script that processes the results from this form.

Each occurrence of an `<input>`, `<select>`, `<textarea>`, or other form element causes the browser to display a widget for capturing data entered on the form. These elements can be freely intermixed with normal HTML tags such as bullet lists, horizontal rules, and so forth.

3. Text fields

To include a field where the user can type a string of characters:

```
<input type="text" name="fieldName" value="defaultText"
  size="width" maxlength="capacity">
```

where:

fieldName

is an internal name that will be passed to the downstream CGI script as the name of this element.

defaultText

specifies text that will be placed in the text field before it is filled out. This attribute is optional; the default is an empty field.

width

specifies the size of the field on the screen. For example, use **size=30** for a field that occupies thirty characters' worth of space on the form. The default size is 20.

capacity

specifies the maximum number of characters that will be accepted in this field. The default is a field of unlimited capacity.

4. Password fields

A password field works the same as a text field, but characters entered are shown on the screen as asterisks ("*") so that onlookers cannot tell what is being typed:

```
<input type="password" name="fieldName" value="defaultText"
  size="width" maxlength="capacity">
```

where the values of the **name**, **value**, and **maxlength** attributes are the same as those of the text field above.

5. Checkboxes

A checkbox allows the user to select on-or-off options. It displays as a small square.

```
<input type="checkbox" name="fieldName" value="onText" checked>
```

where:

fieldName

is an internal name that will be passed to the downstream CGI script as the name of this element.

onText

will be passed to the downstream CGI script as the value of this field, if the user checks the box. If the box is not checked, no value will be passed on. This attribute is optional, and the default value is **value="on"**.

checked

is an optional attribute. By default, all checkboxes are initially unchecked (off). However, if you include the **checked** attribute, the checkbox will be set initially.

6. Radiobuttons

Radiobuttons are used in groups of two or more to allow the user to select one of a set of mutually exclusive choices. Setting one radiobutton in a group automatically clears the others in the group.

The radiobuttons in a group are defined separately, one per **input** tag. All radiobuttons in a group must, however, have the same **name="fieldName"** value.

```
<input type="radio" name="fieldName" value="choiceText" checked>
```

where:

fieldName

is an internal name that will be passed to the downstream CGI script as the name of this element. All radiobuttons in the same group must have the same value for this attribute.

choiceText

will be passed to the downstream CGI script as the value of this field, if this particular radiobutton is selected.

checked

is optional; specify this attribute in the radiobutton that you want to be set (on) when the form is initially displayed.

Note

If none of the radiobuttons in a group have the **checked** attribute, none of the radiobuttons will be set initially. If the user of the form doesn't then select any of them, the downstream CGI script will not receive a (name,value) pair for that group. To avoid this situation, always specify one of the radiobuttons as **checked**; once one of the radiobuttons has been set in a form, there is no way for the user to clear them all.

7. The submit button

Every form must have a button, typically located at the end of the form, that the user clicks to signify that the form has been completely filled out:

```
<input type="submit" value="label">
```

The **value** attribute is optional. If supplied, it specifies the text of the **label** on the submit button. The default text is **value="Submit query"**.

8. The reset button

You can provide a button that restores the form to its initial contents:

```
<input type="reset" value="label">
```

As with the submit button, the **value** attribute is optional and specifies the text that appears on the button. The default text is **value="Reset"**.

When a user clicks the reset button on a form, all the form elements are reset to the same value they had when the page is first loaded. Text fields, for example, are reset to their **value** attribute; fields with no **value** attribute become blank. Each radiobutton group is cleared except for the radiobutton that has the **checked** attribute. Checkboxes with the **checked** attribute become set, and those without that attribute are cleared, and so forth.

9. Pull-down menus

To provide a pull-down menu that gives the user one of several choices:

```
<select name="fieldName">
  <option selected>first option
  <option>second option
  ...
</select>
```

The entire pull-down menu must be bracketed by `<select>...</select>` tags, and each option's text must be preceded by an `<option>` tag. One of the options (it need not be the first one) may be preceded by an `<option selected>` tag; that option will be the default.

Only plain text can appear after each `<option>` tag; do not attempt to imbed HTML tags within an option.

When the form data is sent to the downstream CGI script, the name part of the (name,value) pair for a pull-down menu comes from the **name** attribute of the `<select>` tag, and the value part is the text of the `<option>` choice selected by the user.

10. Scrolling menus

You can also provide a menu that shows the choices as a scrollable list:

```
<select name="fieldName" size="nVisible" multiple>
  <option selected>first option
  <option>second option
  ...
</select>
```

where:

fieldName

is an internal name that will be passed to the downstream CGI script as the name of this element.

nVisible

is a number that specifies how many choices will be visible at one time. For example, **size=8** would specify a scrolling list with room for eight entries at once.

multiple

is an optional attribute; if specified, it allows more than one choice to be selected at a time.

If you use the **multiple** attribute, you can have more than one `<option selected>` choice in the list of options.

11. Multi-line text areas

The `<textarea>` tag allows you to specify an area where any number of lines of text can be entered:

```
<textarea name="fieldName" rows="nRows" cols="nCols">
  default text
</textarea>
```

where:

fieldName

is an internal name that will be passed to the downstream CGI script as the name of this element.

nRows

is the number of rows in the window. This does not limit the amount of text that can be entered; it merely specifies the window size. The default value is 2.

nCols

is the number of text columns in the window. The default value is 20.

default text

The text inside the `<textarea>` element appears initially inside the text area on the form. This is a good place to put instructions such as "Enter your comments here."

12. Image maps

You can use a graphical image instead of a submit button. When a user clicks on that image, the CGI script will be started up and informed of the of the exact pixel coordinates where they clicked. Use a tag with this general form:

```
<input type="image" name="fieldName" src="imageFile">
```

where the **imageFile** is the name of an image file such as a .jpg file.

When the user clicks, your CGI script will be launched. It will receive two name-value pairs identifying the location where the user clicked. The x coordinate will be given as name **fieldName.x** and the y coordinate as **fieldName.y**.

For example, suppose you used this tag:

```
<input type="image" name="dartboard" src="bill.jpg">
```

Suppose a user clicks on the pixel that is located 319 to the right of the left side, and 114 pixels down from the top edge. Your CGI script would be activated and passed two name-value pairs, **dartboard.x** with value 319, and **dartboard.y** with value 114.

13. Relevant URLs

- See the basic documentation for HTML 4.0 fill-out forms³
- Some simple examples⁴.

³ <http://www.w3.org/TR/REC-html40/interact/forms.html>

⁴ <http://www.nmt.edu/tcc/help/lang/python/forms/example-1.html>

- To learn how to process the data that comes out of a form, see: [The Common Gateway Interface](#)⁵
- See the principle starting point for official HTML documentation⁶. Among many other useful references, this URL points to the current standard.

⁵ <http://hoohoo.ncsa.uiuc.edu/cgi/>

⁶ <http://www.w3.org/MarkUp/>