17

Forms

Up to now, all the (X)HTML you have learned has helped you communicate *your* ideas with your visitors. In this chapter, you'll learn how to create forms which enable your visitors to communicate with you.

There are two basic parts of a form: the collection of fields, labels, and buttons that the visitor sees on a page and hopefully fills out, and the processing script that takes that information and converts it into a format that you can read or tally. input process

Constructing a form's fields and buttons *(pages 254–276)* is straightforward and similar to creating any other part of the Web page. You can create text boxes, special password boxes, radio buttons, checkboxes, drop-down menus, larger text areas, and even clickable images. You will give each element a name that will serve as a label to identify the data once it is processed. I'll also show you how to format forms with CSS.

Processing the data from a form is only slightly more complicated. While in earlier editions I recommended using Perl to write CGI scripts, I now heartily recommend using PHP. It is easy and straightforward and perfectly suited to making Web pages interactive.

While both PHP and Perl are beyond the scope of this book, and even explaining how to use existing scripts stretches the limits a bit, I have provided some ready-made scripts to help you get started *(see pages 256 and 258)*.

Creating a Form

A form has three important parts: the `form` tag, which includes the URL of the script that will process the form; the form elements, like fields and menus; and the submit button which sends the data to the script on the server.

To create a form:

1. Type **<form method="post"**.
2. Type **action="script.url">** where *script.url* is the location on the server of the script that will run when the form is submitted *(see page 256)*.
3. Create the form's contents, as described on pages 262–280, including a submit button *(see page 272)* or active image *(see page 276)*.
4. Type **</form>** to complete the form.

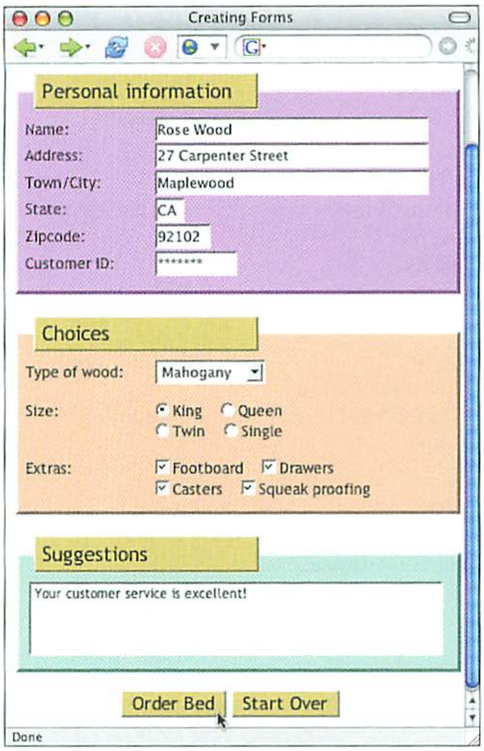
<form>
form elements
submit

```
<form method="post" action="showform.php">
<p class="legend">Personal information</p>
<fieldset id="personal">
<label>Name:</label><input type="text"
name="name" size="30" /> <br />
<label>Address:</label><input type="text"
name="address" size="30" /> <br />
<label>Town/City:</label><input type="text"
name="city" size="30" /> <br />
...
<p id="buttons"><input type="submit"
value="Order Bed"  /><input type="reset"
value="Start Over" /></p>
</form>
```

Figure 17.1 *Every form has three parts: the* `form` *tag, the actual form elements where the visitor enters information, and the submit button (or active image) that sends the collected information to the server.*

```
#form {font-family: "Trebuchet MS", Verdana,
sans-serif; width: 25em;}
h2 {margin: 0 0 0 0; padding: 0;}
p {margin: 0 0 1em 0; padding: 0; font-size: 90%;}
fieldset {background: #C361D2; border: none;
margin-bottom: 1em; width: 24em;
padding-top: 1.5em;}
p.legend {background: #DED983; color: black;
padding: .2em .3em; font-size: 1.2em;
border: 2px outset #DED983; position: relative;
margin-bottom: -1em; width: 10em;
margin-left: 1em; margin-top: 1em;}
#personal {background: #F3B4F5;
border:outset #f3b4f5;}
#choices {background: #F5D9B4;
border: outset #f5d9b4;}
```

Figure 17.2 *Here is a portion of the style sheet used to format the form. You can find the full style sheet on the Web site (see page 26).*


Creating Forms

Personal information

Name:	Rose Wood
Address:	27 Carpenter Street
Town/City:	Maplewood
State:	CA
Zipcode:	92102
Customer ID:	*******

Choices

Type of wood:	Mahogany
Size:	[✓ King] Queen Twin Single
Extras:	[✓ Footboard] [✓ Drawers] [✓ Casters] [✓ Squeak proofing]

Suggestions

Your customer service is excellent!

Figure 17.3 *Here is the complete form discussed in this chapter.*

✓ Tips

- You can download the *showform.php* script from my Web site *(see page 26)* and use it in step 2 to test your forms as you go through this chapter. It is also shown in Figure 17.4 on page 256.
- In order for your visitor to send you the data on the form, you'll need either a submit button or an active image. For more on submit buttons, consult *Creating the Submit Button* on page 272. For details on active images, consult *Using an Image to Submit Data* on page 276.
- You can use CSS *(see page 169)* or tables *(see page 227)* to lay out your form elements. The example that I demonstrate with illustrations throughout this chapter and in Figure 17.3 was created using strict XHTML and CSS.
- You can also use the `get` method to process information gathered with a form. However, since the `get` method limits the amount of data that you can collect at one time, I recommend using `post`.

Processing Forms

A form gathers the information from your visitor and the script processes that information. The script can log the information to a database on the server, send the information via email, or any number of other functions.

In this chapter, since the focus is on creating Web forms, we'll use a very simple PHP script to echo the data back to the visitor when they fill out and submit a form **(Figure 17.4)**. I'll also give you a script that you can use to submit a form's contents to your email address *(see page 258)*.

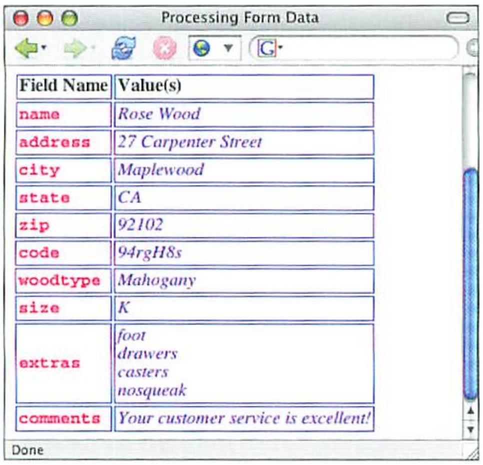
About PHP = Make page c interaction

PHP, which is a recursive abbreviation that stands for *PHP: Hypertext Preprocessor*, is an Open Source scripting language that was written specifically for making Web pages interactive. It is remarkably simple and straightforward. I wrote the scripts for this chapter after having worked with PHP for a very short time—though I was fortunate enough to have a copy of Larry Ullman's excellent *PHP for the World Wide Web: Visual QuickStart Guide, Second Edition*, which I highly recommend. While it's true that my scripts are not very complicated, that's sort of the point. I was able to get them to do what I needed without having to jump through a lot of hoops.

In addition to being easy to learn, PHP has a number of additional characteristics that make it ideal for processing (X)HTML forms. First of all, PHP is an *interpreted* or *scripting* language, which means that it does not need to be compiled first. You write it and off you go. In contrast with Perl scripts, you don't have to make PHP scripts executable or put them in any special place on your server. Indeed, although PHP scripts can be inde-

```
<!DOCTYPE html ... -transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Processing Form Data</title>
<style type="text/css">...</style>
</head><body>
<p>This is a very simple PHP script that outputs
...</p>
<table>
<tr><th>Field Name</th><th>Value(s)</th></tr>

<?php
if (empty($_POST)) {
   print "<p>No data was submitted.</p>";
} else {

foreach ($_POST as $key => $value) {
   if (get_magic_quotes_gpc()) $value=
stripslashes($value);
   if ($key=='extras') {
   if (is_array($_POST['extras']) ){
       print "<tr><td><code>$key</code>
</td><td>";
       foreach ($_POST['extras'] as $value) {
       print "<i>$value</i><br />";
       }
       print "</td></tr>";
   } else {
       print "<tr><td><code>$key</code>
</td><td> <i>$value</i></td></tr>\n";
       }
   } else {
   print "<tr><td><code>$key</code></td>
<td><i>$value</i></td></tr>\n";
   }
}
}
?>
</table>
</body>
</html>
```

Figure 17.4 *Here is the script used to process the forms in this chapter. Notice how the PHP script lives right in an (X)HTML page. (You can find a commented version of this script on my Web site.)*



Field Name	Value(s)
name	Rose Wood
address	27 Carpenter Street
city	Maplewood
state	CA
zip	92102
code	94rgH8s
woodtype	Mahogany
size	K
extras	foot drawers casters nosqueak
comments	Your customer service is excellent!

Figure 17.5 *The script shown in Figure 17.4 outputs the name and value(s) for each field in a table in the browser window. You can try it out on (and download it from) my Web site.*

pendent text files, they are often written right inside the (X)HTML page itself, making PHP extremely convenient for Web designers.

Finally, because PHP was designed for the Web, it's good at the tasks that Web pages require and coordinates well with (X)HTML. There are hundreds of ready-made built-in functions that you can take advantage of. In this chapter we'll touch briefly on PHP's form processing tools. PHP's official site can be found at *http://www.php.net/*

Security

As always when you're sending information to the server, you need to be very careful with security. Never assume anything about your data. Just because you may have built safeguards into your form doesn't mean the bad guys won't create their own form that calls your script in order to send out millions of spam messages with it. Check your data explicitly and make sure that it is what it should be, with no extra bits lurking about.

Alternatives to PHP

There are many alternatives to PHP for processing forms. CGI scripts written in Perl are one common strategy, as are ASP, VisualBasic, and even AppleScript. You can find more information about Perl in my *Perl and CGI for the World Wide Web, Visual QuickStart Guide, Second Edition*, also published by Peachpit Press. A few of the examples in this chapter still rely on Perl scripts. You'll find the forms and accompanying scripts in the *Examples* section of my Web site *(see page 26)*.

Server side vs. Client side

PHP is a *server-side* language, which means that it is run on the computer that serves your Web pages (aptly called a *server*), not on your visitor's computer where the page is viewed. Indeed, it won't work at all if the script is not uploaded to a server. In addition, that server must have PHP installed for the script to be interpreted. Server-side languages are ideal for processing forms, sending email, and other functions that require a server.

Client-side languages, like JavaScript, work right inside the browser. They can do many tasks without interacting with the server at all. They are great for manipulating the browser window, checking that all the data has been entered before submitting a form, and other tasks that happen without the server (or before the server gets involved). You'll find more information on JavaScript in Chapters 19–20.

Sending Form Data via Email

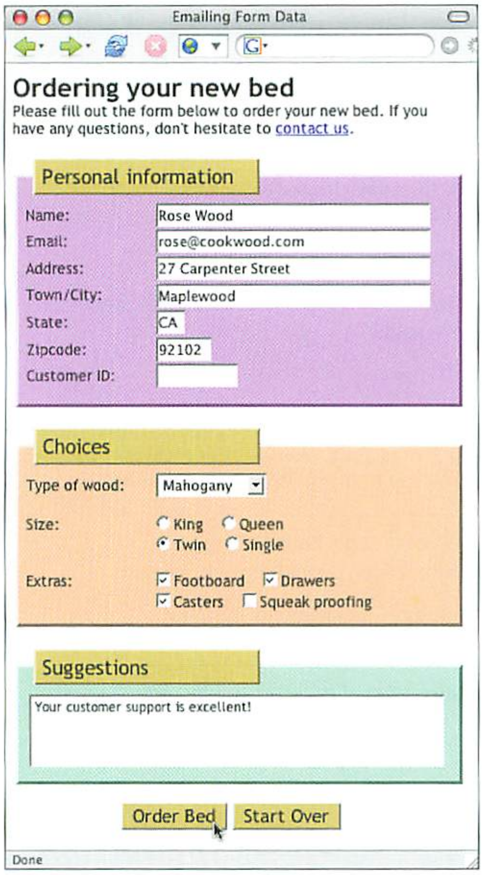
If you don't feel like messing with CGI scripts and can deal with not having your data perfectly formatted (or pre-processed by a script), you can have a visitor's data be sent to you via email.

To send form data via email:

1. Type **<form method="post"**.
2. Type **action="emailform.php"**, where *emailform.php* is the script that will send the form data to your email.
3. Type **>**.
4. Create the form's contents, as described on pages 262–280.
5. Type **</form>**.

```
...
<body>

<?php
//This is a very simple PHP script that ...

if (empty($_POST)) {
    print "<p>No data was submitted.</p>";
    print "</body></html>";
    exit();
}
function clear_user_input($value) {
    if (get_magic_quotes_gpc())
$value=stripslashes($value);
    $value= str_replace( "\n", '', trim($value));
    $value= str_replace( "\r", '', $value);
    return $value;
    }
if ($_POST['comments'] == 'Please share any
comments you have here') $_POST['comments'] =
'';
$body ="Here is the data that was submitted:\n";
|
foreach ($_POST as $key => $value) {
    $key = clear_user_input($key);
    $value = clear_user_input($value);
    if ($key=='extras') {

    if (is_array($_POST['extras']) ){
        $body .= "$key: ";
        $counter =1;
        foreach ($_POST['extras'] as $value) {
        //Add comma and space until last element
        if (sizeof($_POST['extras']) == $counter) {
        $body .= "$value\n";
        break;}
        else {
        $body .= "$value, ";
        $counter += 1;
        }}
        } else {
        $body .= "$key: $value\n";
        }
    } else {

    $body .= "$key: $value\n";
    }
}
extract($_POST);
$email = clear_user_input($email);
$name = clear_user_input($name);
$from='From: '. $email . "(" . $name . ")" . "\r\n"
. 'Bcc: alternate@yoursite.com' . "\r\n";

$subject = 'Bed Order from Web Site';

mail ('youremail@yoursite.com', $subject, $body,
$from);
?>
<p>Thanks for your order! We'll send your bed
right away.</p>
</body></html>
```

Figure 17.6 *Here is a script used to send form data via email. You can find a commented version of this script on my Web site.*



Ordering your new bed

Please fill out the form below to order your new bed. If you have any questions, don't hesitate to contact us.

Personal information

Name:	Rose Wood
Email:	rose@cookwood.com
Address:	27 Carpenter Street
Town/City:	Maplewood
State:	CA
Zipcode:	92102
Customer ID:	

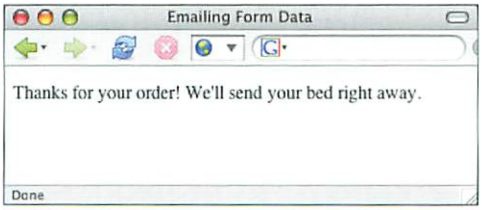
Choices

Type of wood:	Mahogany
Size:	King Queen [✓ Twin] Single
Extras:	[✓ Footboard] [✓ Drawers] [✓ Casters] Squeak proofing

Suggestions

Your customer support is excellent!

Figure 17.7 *This form is almost identical to the other except for an added email field.*

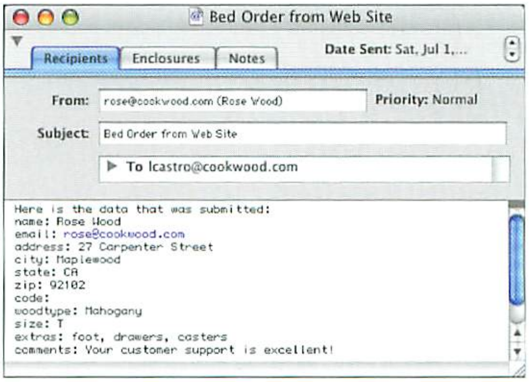


Thanks for your order! We'll send your bed right away.

Figure 17.8 *It's always a good idea to give your visitor feedback about what just happened—since they can't see the email wending its way to you.*

✓ Tips

- You might want to ask for the email address to be entered twice, in order to prevent typos from keeping you from receiving the form data. Then have the script compare the two fields and return an error if they're not identical.
- You can find the code for this script on my Web site *(see page 26)*. You are welcome to use it on your own site.
- In earlier editions of this book, I offered a technique that used `enctype="text/plain"` along with an `action` attribute set to an email address in order to receive form data via email. Unfortunately, it didn't work with some email programs (like Outlook) and so I substituted it for the PHP script shown here.
- If this script doesn't work on your server, it may be that your server doesn't have PHP installed. Contact your Web host and ask them (or check their Support pages).



Date Sent: Sat, Jul 1,...

From: rose@cookwood.com (Rose Wood)   Priority: Normal

Subject: Bed Order from Web Site

To lcastro@cookwood.com

```
Here is the data that was submitted:
name: Rose Wood
email: rose@cookwood.com
address: 27 Carpenter Street
city: Maplewood
state: CA
zip: 92102
code:
woodtype: Mahogany
size: T
extras: foot, drawers, casters
comments: Your customer support is excellent!
```

Figure 17.9 *Here is the email that was received after the form was submitted in Figure 17.7.*

Organizing the Form Elements

If you have a lot of information to fill out on a form, you can group related elements together to make the form easier to follow. The easier it is for your visitors to understand the form, the more likely they are to fill it out correctly.

To organize the form elements:

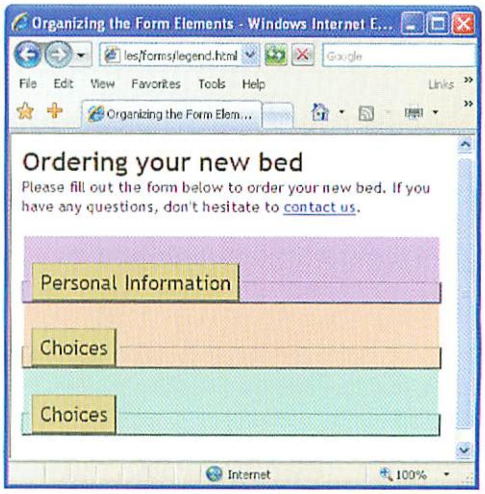
1. Below the `form` tag but above any form elements that you wish to have contained in the first group, type **<fieldset>**.
2. If desired, type **<legend**.
3. If desired, type **align="direction"** where *direction* is `left` or `right`.
4. Type **>**.
5. Type the text for the legend.
6. Type **</legend>** to complete the legend.
7. Create the form elements that should belong in the first group. For more information, see pages 262–276.
8. Type **</fieldset>** to complete the first group of form elements.
9. Repeat steps 1–8 for each group of form elements.

```
<form method="post" action="showform.php">
<fieldset id="personal"><legend>Personal
Information</legend>
</fieldset>
<fieldset id="choices"><legend>Choices
</legend>
</fieldset>
<fieldset id="suggestions"><legend>Suggestions
</legend>
</fieldset>
```

Figure 17.10 *I have added an* `id` *attribute to each* `fieldset` *element to facilitate applying styles to each group of form elements.*

```
fieldset {margin-bottom:1em;
width: 24em; padding-top: 1.5em;}
#personal {background: #f3b4f5;
border:outset #f3b4f5;}
```

Figure 17.11 *I gave all the fieldset elements some margin, width, and padding, and then applied a separate background color and outset border to each one.*


Organizing the Form Elements - Windows Internet E...

Ordering your new bed

Please fill out the form below to order your new bed. If you have any questions, don't hesitate to contact us.

Personal Information

Choices

Choices

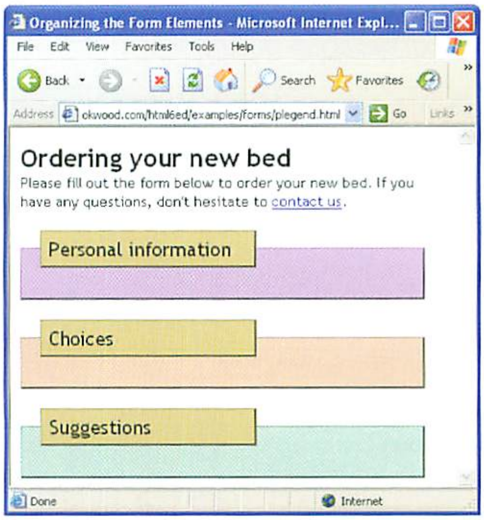
Figure 17.12 *Internet Explorer does strange things to the* `legend` *element, pulling the background from the* `fieldset` *up and around it.*

```
<form method="post" action="showform.php">
<p class="legend">Personal information</p>
<fieldset id="personal">
```

Figure 17.13 *Because of Internet Explorer's lack of support of the* `legend` *element, I recommend using a regular* `p` *tag with a* `class` *of* legend.

```
p.legend {background:#DED983; color: black;
border: 2px outset #DED983; padding: .2em .3em;
font-size: 1.2em; position: relative;
margin: 1em 0 -1em 1em; width: 10em;}
```

Figure 17.14 *Next I style the legend paragraph with a background and outset border. Then I give it a negative bottom margin and relative positioning to pull it on top of the fieldset.*


Organizing the Form Elements - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Search Favorites
Address okwood.com/html6ed/examples/forms/plegend.html Go Links
Done Internet

Ordering your new bed

Please fill out the form below to order your new bed. If you have any questions, don't hesitate to contact us.

Personal information

Choices

Suggestions

Figure 17.15 *Now the legends look good even in the recalcitrant IE 6.*

✓ Tips

- I think `fieldset` elements look great formatted with an outset border. Use **background: color; border: outset color**, where *color* is the same in both instances and is what you want for the background of the `fieldset`. I formatted the `legend` elements the same way.
- The `legend` element is not at all well supported by Internet Explorer. If you create one and then give your `fieldset` a background, the background extends up and around the `legend`, looking really ugly **(Figure 17.12)**. I recommend recreating the legend effect with an aptly styled `p` element.
- Organizing your form into fieldsets is completely optional.
- While the `align` attribute for `legend` has been deprecated, it's still supported by Firefox and IE (from 4 on). It's default value is `left`. There were supposedly `top` and `bottom` values as well, but I've never seen a browser support them.

Creating Text Boxes

Text boxes can contain one line of free-form text—that is, anything that the visitor wants to type—and are typically used for names, addresses, and the like.

To create a text box:

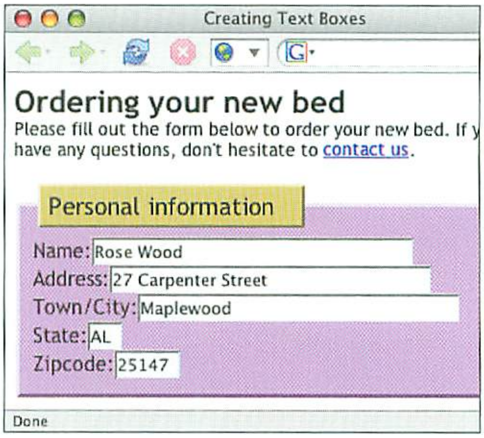
1. If desired, type the label that will identify the text box to your visitor (for example, **Name:**).
2. Type **<input type="text"**.
3. Type **name="label"**, where *label* is the text that will identify the input data to the server (and your script).
4. If desired, type **value="default"**, where *default* is the data that will initially be shown in the field and that will be sent to the server if the visitor doesn't type something else.
5. If desired, define the size of the box on your form by typing **size="n"**, replacing *n* with the desired width of the box, measured in characters.
6. If desired, type **maxlength="n"**, where *n* is the maximum number of characters that can be entered in the box.
7. Finish the text box by typing a final **/>**.

✓ Tips

- Even if your visitor skips the field (and you haven't set the default text with the `value` attribute), the `name` attribute is still sent to the server (with an undefined, empty `value`).
- The default for `size` is 20. However, visitors can type up to the limit imposed by the `maxlength` attribute. Still, for larger, multi-line entries, it's better to use text areas *(see page 269)*.

```
<form method="post" action="showform.php">
<p class="legend">Personal information</p>
<fieldset id="personal">
Name:<input type="text" name="name" size="30"
/> <br />
Address:<input type="text" name="address"
size="30" /> <br />
Town/City:<input type="text" name="city"
size="30" /> <br />
State:<input type="text" name="state" size="2"
maxlength="2" /><br />
Zipcode:<input type="text" name="zip" size="5"
maxlength="5" /> <br />
```

Figure 17.16 *While it's essential to set the* `name` *attribute for each text box, you only have to set the* `value` *attribute when you want to add default values for a text box.*



Ordering your new bed

Please fill out the form below to order your new bed. If y
have any questions, don't hesitate to contact us.

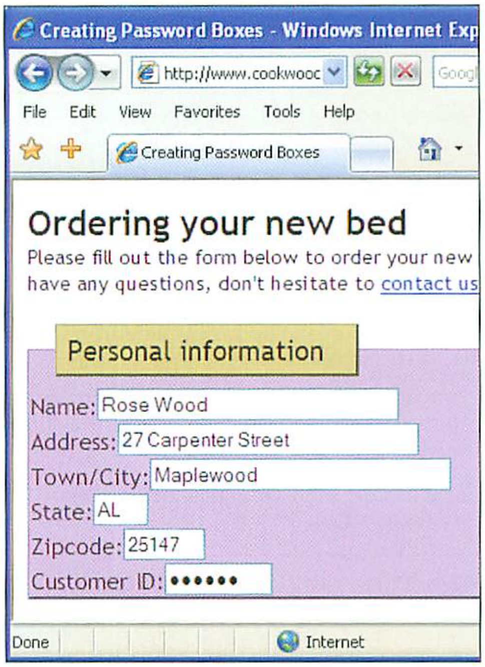
Personal information

Name: Rose Wood
Address: 27 Carpenter Street
Town/City: Maplewood
State: AL
Zipcode: 25147

Figure 17.17 *Text boxes can be different sizes to accommodate different types of fields. We'll straighten these up in just a moment (on page 264).*

```
Zipcode:<input type="text" name="zip" size="5"
maxlength="5" /> <br />
Customer ID:<input type="password"
name="code" size="8" />
```

Figure 17.18 *The* `name` *attribute identifies the password when you compile the data.*


Creating Password Boxes - Windows Internet Exp
http://www.cookwooc
File Edit View Favorites Tools Help
Creating Password Boxes
Done
Internet

Ordering your new bed

Please fill out the form below to order your new
have any questions, don't hesitate to contact us

Personal information

Name: Rose Wood
Address: 27 Carpenter Street
Town/City: Maplewood
State: AL
Zipcode: 25147
Customer ID: ••••••

Figure 17.19 *When the visitor enters a password in a form, the password is hidden with bullets or asterisks.*

Creating Password Boxes

The only difference between a password box and a text box is that whatever is typed in the former is hidden by bullets or asterisks. The information is *not* encrypted when sent to the server.

To create password boxes:

1. If desired, type the label that will identify the password box to your visitor (for example, **Enter password:**).
2. Type **<input type="password"**.
3. Type **name="label"**, where *label* is the text that will identify the input data to the server (and your script).
4. If desired, define the size of the box on your form by typing **size="n"**, replacing *n* with the desired width of the box, measured in characters.
5. If desired, type **maxlength="n"**, where *n* is the maximum number of characters that can be entered in the box.
6. Finish the text box by typing a final **/>**.

✓ Tips

- Even if nothing is entered in the password box, the `name` is still sent to the server (with an undefined `value`).
- You could set default text for `value` (as in step 4 on page 262), but that kind of defeats the purpose of a password.
- The only protection the password box offers is from folks peering over your visitor's shoulder as she types in her password. To really protect passwords you have to use them on a secure server.

Formally Labeling Form Parts

As you've seen, the explanatory information next to a form element is generally just plain text. For example, you might type "First name" before the text field where the visitor should type her name. (X)HTML provides a method for marking up labels so that you can formally link them to the associated element and use them for scripting or other purposes.

To formally label form parts:

1. Type **<label**.
2. If desired, type **for="idname">**, where *idname* is the value of the `id` attribute in the corresponding form element.
3. Type the contents of the label.
4. Type **</label>**.

✓ Tips

- If you use the `for` attribute, you must also add the `id` attribute to the associated form element's opening tag in order to mark it with a `label`. (Otherwise, the document will not validate.) For more details about the `id` attribute, consult *Naming Elements* on page 63.
- If you omit the `for` attribute, no `id` attribute is required in the element being labeled. The label and the element, in that case, are then associated by proximity, or perhaps by being placed in a common `div` element.
- Another labeling technique is to use the `title` attribute. For more information, consult *Labeling Elements in a Web Page* on page 68.
- You can use CSS to format your labels **(Figure 17.21)**.

```
<fieldset id="personal">
<label>Name:</label><input type="text"
name="name" size="30" /> <br />
<label>Address:</label><input type="text"
name="address" size="30" /> <br />
<label>Town/City:</label><input type="text"
name="city" size="30" /> <br />
<label>State:</label><input type="text"
name="state" size="2" maxlength="2" /><br />
<label>Zipcode:</label><input type="text"
name="zip" size="5" maxlength="5" /> <br />
<label>Customer ID:</label><input
type="password" name="code" size="8" />
</fieldset>
```

Figure 17.20 *Marking field labels in a formal way gives you an easy way to identify them in a CSS style sheet.*

```
#personal label {position: absolute; font-size: 90%;
padding-top: .2em; left: 20px;}
input {margin-left: 9em; margin-bottom:.2em;
line-height:1.4em; }
```

Figure 17.21 *Now I can sweep the labels out of the flow and align the text and password boxes to their right.*

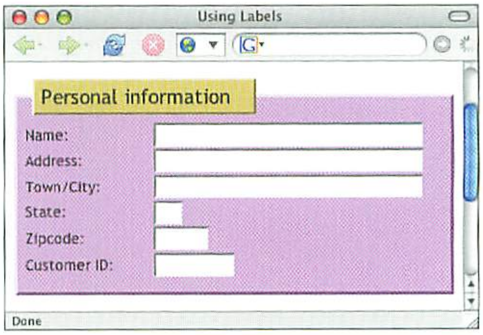


Figure 17.22 *The form is beginning to take shape.*

```
<fieldset id="choices">
<p id="size"><label>Size:</label><input
type="radio" name="size" value="K" />King
<input type="radio" name="size" value="Q"
/>Queen <br />
<input type="radio" name="size" value="T"
/>Twin
<input type="radio" name="size" value="S"
/>Single</p>
```

Figure 17.23 *The* `name` *attribute serves a dual purpose for radio buttons: it links the radio buttons in a given set and it identifies the value when it is sent to the script. The* `value` *attribute is crucial since the visitor has no way of typing a value.*

```
#choices label {position: absolute;
padding-top: .2em; left: 20px}
#size {font-size: 90%;}
input {margin-left: 9em;}
input+input {margin-left: 1em;}
br+input {margin-left: 9em;}
```

Figure 17.24 *This CSS positions the labels absolutely, just as in Figure 17.21. Then it gives the first radio button and any radio button after a* `br` *element a 9 em margin. The remaining radio buttons get 1 em margins.*

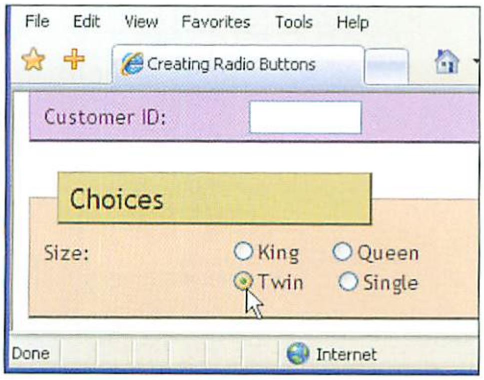


Figure 17.25 *The radio buttons themselves are created with the (X)HTML tags. The labels (King, Queen, etc.) are created with plain text alongside the (X)HTML tags.*

Creating Radio Buttons

Remember those old-time car radios with big black plastic buttons? Push one to listen to WFCR; push another for WRNX. You can never push two buttons at once. Radio buttons on forms work the same way (except you can't listen to the radio).

To create a radio button:

1. If desired, type the introductory text for your radio buttons. You might use something like **Select one of the following**.
2. Type **<input type="radio"**.
3. Type **name="radioset"**, where *radioset* both identifies the data sent to the script and also links the radio buttons together, ensuring that only one per set can be checked.
4. Type **value="data"**, where *data* is the text that will be sent to the server if the radio button is checked, either by you (in step 5) or by the visitor.
5. If desired, type **checked="checked"** to make the radio button active by default when the page is opened. You can only do this to one radio button in the set. (The **="checked"** is optional in HTML.)
6. Type the final **/>**.
7. Type the text that identifies the radio button to the visitor. This is often the same as `value`, but doesn't have to be.
8. Repeat steps 2–7 for each radio button in the set.

✓ Tip

■ If you don't set the `value` attribute, the word "on" is sent to the script. It's not particularly useful since you can't tell which button in the set was pressed.

Creating Menus

Menus are perfect for offering your visitors a choice from a given set of options.

To create menus:

1. If desired, type the text that will describe your menu.
2. Type **<select**.
3. Type **name="label"**, where *label* will identify the data collected from the menu when it is sent to the server.
4. If desired, type **size="n"**, where *n* represents the height (in lines) of the menu.
5. If desired, type **multiple="multiple"** to allow your visitor to select more than one menu option (with Ctrl or Command). (The **="multiple"** is optional in HTML.)
6. Type **>**.
7. Type **<option**.
8. If desired, type **selected="selected"** if you want the option to be selected by default. (The **="selected"** is optional in HTML.)
9. Type **value="label"**, where *label* identifies the data that will be sent to the server if the option is selected.
10. If desired, type **label="menu option"**, where *menu option* is the word that should appear in the menu.
11. Type **>**.
12. Type the option name as you wish it to appear in the menu.
13. Type **</option>**.
14. Repeat steps 7–13 for each option.
15. Type **</select>**.

```
<input type="radio" name="size" value="T"
/>Twin
<input type="radio" name="size" value="S"
/>Single</p>
<p id="woodtype"><label>Type of
wood:</label><select name="woodtype" >
<option value="Mahogany">Mahogany</option>
<option value="Maplewood">Maplewood
</option>
<option value="Pine">Pine</option>
<option value="Cherry">Cherry</option>
</select></p>
</fieldset>
```

Figure 17.26 *Menus are made up of two (X)HTML tags:* `select` *and* `option`*. You set the common* `name` *attribute in the* `select` *tag and the individual* `value` *attribute in each of the* `option` *tags.*

```
select {margin-left: 9em;}
```

Figure 17.27 *We'll use CSS again to push the menu over in line with the other fields.*

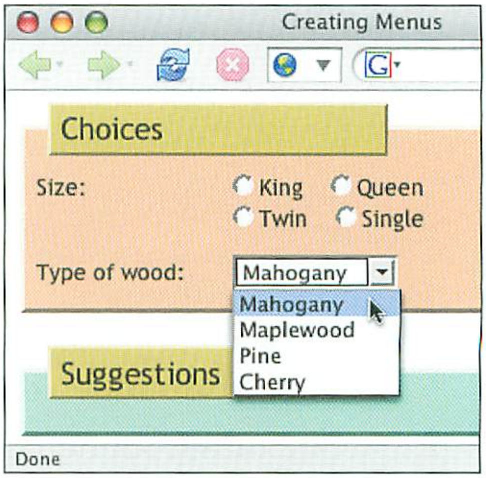


Figure 17.28 *A visitor will not be able to select* nothing *in a menu unless you set the* `size` *attribute. The default selection is either the first option in the menu or the one you've set as* `selected` *in step 8.*

```
<p id="woodtype"><label>Type of
wood:</label><select name="woodtype" >
<optgroup label="Hard woods">
<option value="Mahogany">Mahogany</option>
<option value="Maplewood">Maplewood
</option>
<option value="Cherry">Cherry</option>
</optgroup>
<optgroup label="Soft Woods">
<option value="Pine">Pine</option>
<option value="Fir">Fir</option>
</optgroup>
</select></p>
```

Figure 17.29 *Each submenu has a title, specified in the* `label` *attribute of the* `optgroup` *tag, and a series of options (defined with* `option` *tags and regular text).*

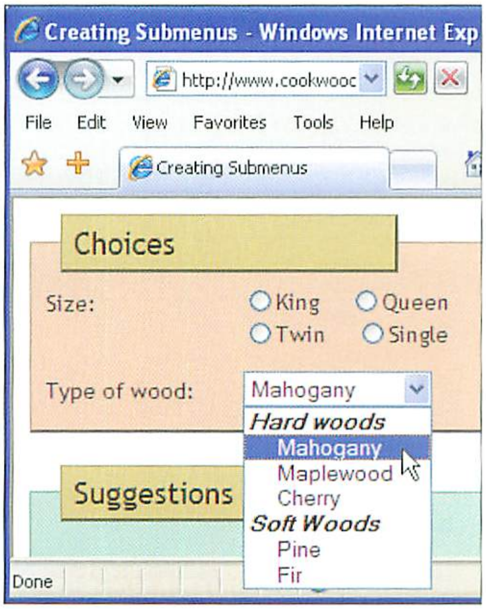


Figure 17.30 *Browsers generally don't create true submenus, but rather group the items in a single menu with subgroups.*

If you have a particularly large menu with many options, you may want to group the options into categories.

To group menu options:

1. Create a menu as described on page 266.
2. Before the first `option` tag in the first group that you wish to place together in a submenu, type **<optgroup**.
3. Type **label="submenutitle">**, where *submenutitle* is the header for the submenu.
4. After the last `option` tag in the group, type **</optgroup>**.
5. Repeat steps 2–4 for each submenu.

✓ Tips

- If you add the `size` attribute in step 4, the menu appears more like a list, and there is no automatically selected option (unless you use `selected`—see step 8).
- If `size` *(see step 4)* is bigger than the number of options, visitors can deselect all values by clicking in the empty space.
- The closing `option` tag in step 13 is optional in HTML 4 (but not in XHTML). You can also use the abbreviated `selected` and `multiple` in HTML, whereas XHTML requires `selected="selected"` and `multiple="multiple"`.

Creating Checkboxes

While radio buttons can accept only one answer per set, a visitor can check as many checkboxes in a set as they like. Like radio buttons, checkboxes are linked by the value of the `name` attribute.

To create checkboxes:

1. If desired, type the introductory text (something like **Select one or more of the following**) for your checkboxes.
2. Type **<input type="checkbox"**. (Notice there is no space in the word *checkbox*.)
3. Type **name="boxset"**, where *boxset* both identifies the data sent to the script and also links the checkboxes together.
4. Type **value="data"**, where *data* is the text that will be sent to the server if the checkbox is marked (either by the visitor, or by you as described in step 5).
5. Type **checked="checked"** to make the checkbox checked by default when the page is opened. You (or the visitor) may check as many checkboxes as desired. (The **="checked"** is optional in HTML.)
6. Type **/>** to complete the checkbox.
7. Type the text that identifies the checkbox to the user. This is often the same as the `value`, but doesn't have to be.
8. Repeat steps 2–7 for each checkbox in the set.

✓ Tip

- If you use PHP, you can automatically create an array (called `$_POST['boxset']`) out of the checkbox values by using **name="boxset[ ]"** in step 3, where *boxset* identifies the data sent to the script.

```
<p id="extras"><label>Extras:</label>
<input type="checkbox" name="extras[]"
value="foot" />Footboard
<input type="checkbox" name="extras[]"
value="drawers" checked="checked" />Drawers
<br />
<input type="checkbox" name="extras[]"
value="casters" />Casters
<input type="checkbox" name="extras[]"
value="nosqueak" />Squeak proofing <br /></p>
</fieldset>
```

Figure 17.31 *Notice how the label text (not highlighted) does not need to match the* `value` *attribute. That's because the label text identifies the checkboxes to the visitor in the browser while the* `value` *identifies the data to the script. The empty brackets are for PHP (see tip).*

```
#extras {font-size: 90%;}
input {margin-left: 9em;}
input+input {margin-left: 1em;}
br+input {margin-left: 9em;}
```

Figure 17.32 *The CSS is very similar to what we used for the radio buttons: give the first checkbox and any checkbox that follows a* `br` *element a 9 em margin. Remaining checkboxes get only 1 em.*

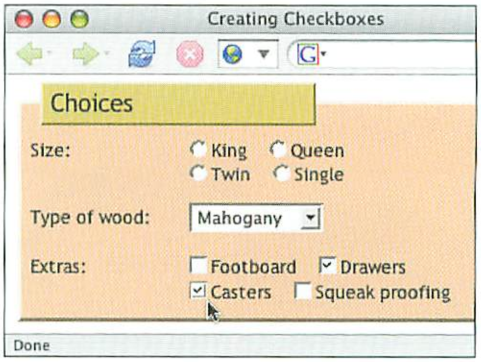


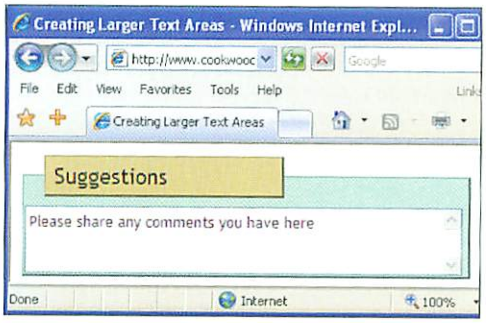
Figure 17.33 *The visitor can check as many boxes as necessary. Each corresponding value will be sent to the script, together with the checkbox set's name.*

```
<p class="legend">Suggestions</p>
<fieldset id="suggestions">
<textarea name="comments" rows="3"
cols="40">Please share any comments you have
here</textarea>
</fieldset>
```

Figure 17.34 *The* `value` *attribute is not used with the* `textarea` *tag. Default values are set by adding text between the opening and closing tags (as in "Please share..." shown here).*

```
textarea {font: .8em "Trebuchet MS", Verdana,
sans-serif; width: 29em; padding: .2em;}
```

Figure 17.35 *For some reason, I had to apply the* `font` *property directly to the* `textarea` *element to get it to take hold. It did not inherit from the* `body`.


Creating Larger Text Areas - Windows Internet Expl...
http://www.cookwooc
Google
File Edit View Favorites Tools Help
Links
Creating Larger Text Areas
Done
Internet
100%

Suggestions

Please share any comments you have here

Figure 17.36 *The visitor can override the default text simply by typing over it.*

Creating Larger Text Areas

In some cases, you want to give the visitor more room to write. Unlike text boxes *(see page 262)*, text areas may be as large as your page, and will expand as needed if the person enters more text than can fit in the display area. They're perfect for eliciting questions and comments.

To create larger text areas:

1. If desired, type the explanatory text that will identify the text area.
2. Type **<textarea**.
3. Type **name="label"**, where *label* is the text that will identify the input data to the server (and your script).
4. Type **rows="n"**, where *n* is the height of the text area in rows.
5. Type **cols="n"**, where *n* is the width of the text area in characters.
6. Type **>**.
7. Type the default text, if any, for the text area. No formatting is allowed here.
8. Type **</textarea>** to complete the text area.

✓ Tips

- There is no use for the `value` attribute with text areas.
- Visitors can enter up to 32,700 characters in a text area. Scroll bars will appear when necessary.
- Both the `rows` and `cols` attributes are required.

Allowing Visitors to Upload Files

If the information you need from the folks filling out your form is complicated, you might want to have them upload an entire file to your server.

To allow visitors to upload files:

1. Type **<form method="post" enctype= "multipart/form-data"**. The `enctype` attribute ensures that the file is uploaded in the proper format.
2. Next, type **action= "upload.url">,** where *upload.url* is the URL of the script that processes incoming files. You'll need a special script for this.
3. Type the caption for the file upload area so your visitors know what to do. Something like **What file would you like to upload?** would work well.
4. Type **<input type="file"** to create a file upload box and a Browse button.
5. Type **name="title"**, where *title* identifies to the server the files being uploaded.
6. If desired, type **size="n"**, where *n* is the width, in characters, of the field in which the visitor will enter the path and file name.
7. Type the final **/>**.
8. Complete the form as usual, including the submit button and final `</form>` tag.

✓ Tips

- The `size` attribute is optional, but since most paths and file names are pretty long, it's a good idea to set it at 40 or 50. The default is 20.
- You can't use the `get` method for forms that allow uploading.

```
<form method="post" enctype="multipart/form-data" action="http://www.cookwood.com/cgi-bin/perl2e/uploading/uploading.cgi">

<h2>What files are you sending?</h2>

<p><input type="file" name="uploadfile" size="30" />

</form>
```

Figure 17.37 *To allow visitors to upload files, you must make sure to set the proper* `enctype` *attribute, as well as create the* `file` *type* `input` *element.*

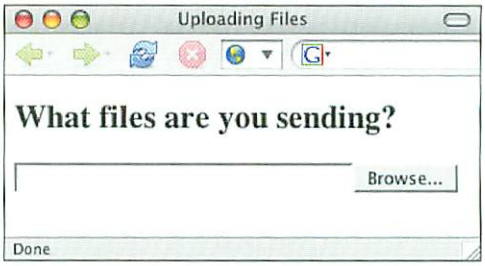


Figure 17.38 *When you create a file upload area, both a field where the visitor can type the path to the file and a Browse button (so the visitor can use an Open dialog box to choose the file) automatically appear on your page.*

```
<form method=post action="whatever.php">
<input type="hidden" name="name"
value="$name" />
<input type=submit value="submit data" />
```

Figure 17.39 *When you create a hidden field, you use the variables from your script to set the value of the field to what the visitor originally entered.*

Creating Hidden Fields

Hidden fields are generated by the processing script to store information gathered from an earlier form so that it can be combined with the present form's data.

To create hidden fields:

1. Type **<input type="hidden"**.
2. Type **name="label"**, where *label* is a short description of the information to be stored.
3. Type **value="data"**, where *data* is the information itself that is to be stored. It is often a variable from the form processing script **(Figure 17.39)**.
4. Type **/>**.

✓ Tips

- It doesn't matter where the hidden fields appear in your form since they won't appear in the browser anyway. As long as they are within the opening and closing `form` tags, you're OK.
- To create an element that will be submitted with the rest of the data when the visitor clicks the submit button but that is also *visible* to the visitor, create a regular form element and use the `readonly` attribute *(see page 280)*.

When to use a hidden field?

Imagine you've got a form and you want to be able to give your visitors a chance to review what they've entered before they submit it. Your processing script can show them the submitted data and at the same time create a form with hidden fields containing the same data. If the visitor wants to edit the data, they simply go back. But if they want to submit the data, the hidden fields will already be filled out, saving them the task of typing the data in again.

Creating the Submit Button

All the information that your visitors enter won't be any good to you unless they send it to the server. You should always create a submit button for your forms so that the visitor can deliver the information to you. (You can also use images to submit form data—see page 276.)

To create a submit button:

1. Type **<input type="submit"**.
2. If desired, type **value="submit message"** where *submit message* is the text that will appear in the button.
3. Type the final **/>**.

✔ Tips

- If you leave out the `value` attribute, the submit button will be labeled *Submit Query*, by default.
- The name-value pair for the submit button is only sent to the script if you set the `name` attribute. Therefore, if you omit the `name` attribute, you won't have to deal with the extra, usually superfluous submit data.
- On the other hand, you can create multiple submit buttons (with both the `name` and `value` attributes) and then write your script to react according to which submit button the visitor presses.

```
<p id="buttons"><input type="submit"
value="Order Bed"  /></p>
```

Figure 17.40 *If you leave out the* `name` *attribute, the name-value pair for the submit button will not be passed to the script. Since you usually don't need this information, that's a good thing.*

```
input {background: #DED983; font:1.2em
"Trebuchet MS", Verdana, sans-serif;}

#buttons {text-align: center;}
```

Figure 17.41 *I apply a background and font formatting to the submit button. For more information on selecting by attribute, see page 147.*

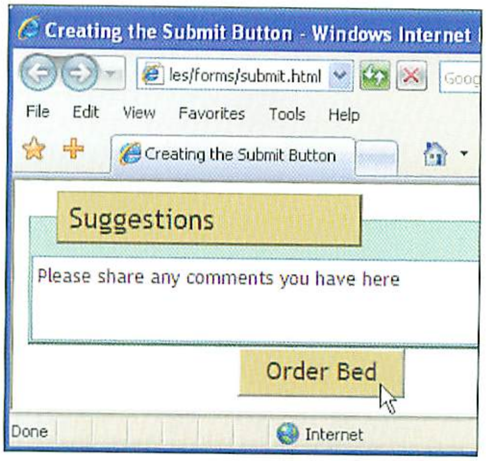

Creating the Submit Button - Windows Internet
les/forms/submit.html
File Edit View Favorites Tools Help
Creating the Submit Button
Suggestions
Please share any comments you have here
Order Bed
Done
Internet

Figure 17.42 *The submit button activates the script that collects the data from the form. You can personalize the button's contents with the* `value` *attribute. (The phrase* Order Bed *is clearer for your visitors than the default text* Submit Query*).*

```
<input type="radio" name="cats" value="X"
accesskey="x" />Xixona<br />

<input type="radio" name="cats" value="L"
accesskey="l" />Llumeta<br />

<input type="radio" name="cats" value="A"
accesskey="a" />All of them (Don't

make me choose!)

<p id="buttons"><button type="submit"><img
src="check.gif" width="40" height="40" alt="Vote
button" /> Vote</button>
```

Figure 17.43 *You can create a submit button with an image by using the* `button` *tag.*

```
body {background: #FEFEE7;}

h2 {margin: 1em 0 0; padding: 0;}

button {font: 48px "Trebuchet MS", "Verdana",
sans-serif; background: #F9CC7D;
border: outset #F9CC7D;}


p#buttons {white-space: nowrap;}
```

Figure 17.44 *I gave the buttons more body by applying an outset border that is the same color as the background in the CSS.*

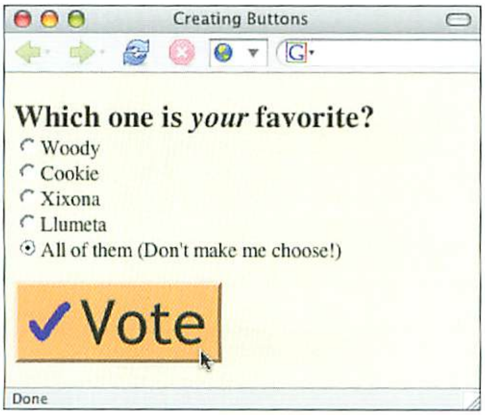

Creating Buttons
Which one is *your* favorite?
Woody
Cookie
Xixona
Llumeta
All of them (Don't make me choose!)
✓Vote
Done

Figure 17.45 *The (X)HTML code for a submit button with an image is a little more complicated, but looks so good. (Of course, it would help if I could actually draw.)*

(X)HTML's `button` element lets you create prettier submit buttons. You can add an image, change the font, or even change the background color. That'll get them to submit that form!

To create a submit button with an image:

1. Type **<button type="submit">**.
2. Type the text, if any, that should appear on the left side of the image in the button.
3. Type **<img src="image.url"** where *image.url* is the name of the image that will appear on the button.
4. Type **alt="alternate text"**, where *alternate text* is what appears if the image doesn't.
5. If desired, add any other image attributes.
6. Type **/>** to complete the image.
7. Type the text, if any, that should appear on the right side of the image in the button.
8. Type **</button>**.

✓ Tips

- If you have multiple submit buttons, you can give a `name` and `value` attribute to each one so that your script can tell which one was pressed.
- You can also use the `button` tag to create a submit button without an image. Just skip steps 3–6.
- You can use CSS to style buttons.
- For information on creating buttons with scripts, consult *Creating a Button that Executes a Script* on page 316.

Resetting the Form

If humans could fill out forms perfectly on the first try, there would be no erasers on pencils and no backspace key on your computer keyboard. You can give your visitors a reset button so that they can start over with a fresh form (including all the default values you've set).

To create a reset button:

1. Type **<input type="reset"**.
2. If desired, type **value="reset message"** where *reset message* is the text that appears in the button. The default reset message is *Reset*.
3. Type **/>**.

✔ Tips

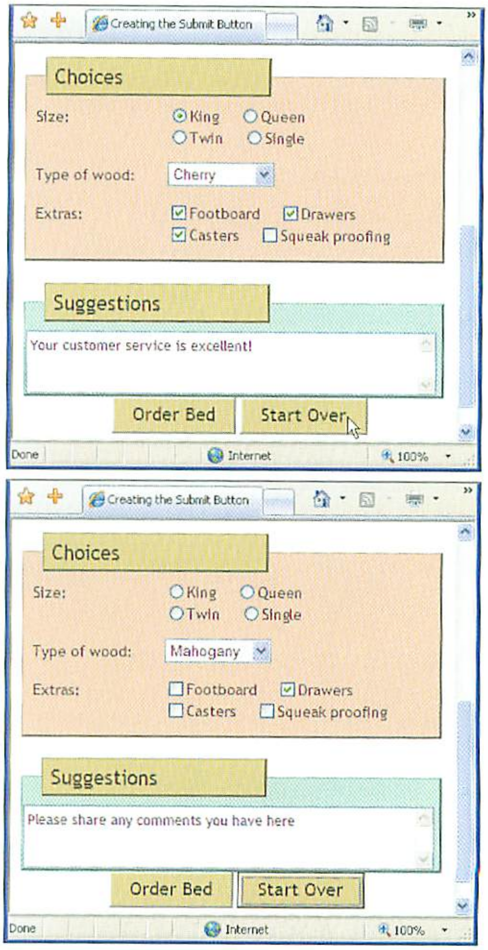
- The name-value pair for the reset button is only sent to the script if you set the `name` attribute. Therefore, if you omit the `name` attribute, you won't have to deal with the completely superfluous reset data—which is usually something like "reset, Reset".
- You could add the `name` attribute to a reset button for scripting purposes *(see page 316)*.

```
<input type="reset" value="Start Over" />
```

Figure 17.46 *You can use the* `value` *attribute to set any text you wish for the reset button.*

```
input {background: #DED983; font: 1.2em
"Trebuchet MS", Verdana, sans-serif;}
```

Figure 17.47 *The CSS from Figure 17.41 on page 272 already applies to the Reset button. No additions are necessary.*



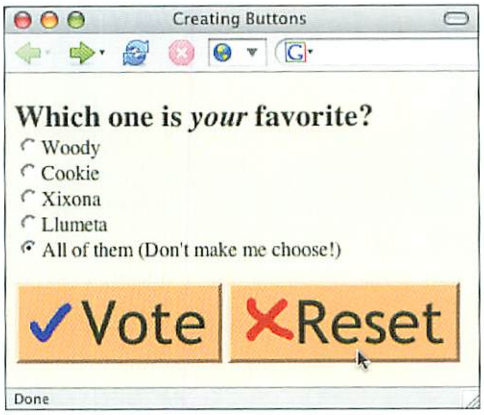
Size: [✓ King] Queen Twin Single
Type of wood: Cherry
Extras: [✓ Footboard] [✓ Drawers] [✓ Casters] Squeak proofing
Your customer service is excellent!

Size: King Queen Twin Single
Type of wood: Mahogany
Extras: Footboard [✓ Drawers] Casters Squeak proofing
Please share any comments you have here

Figure 17.48 *If your visitor clicks the reset button, all the fields are set to their default values.*

```
<input type="radio" name="cats" value="X"
accesskey="x" />Xixona<br />

<input type="radio" name="cats" value="L"
accesskey="l" />Llumeta<br />

<input type="radio" name="cats" value="A"
accesskey="a" />All of them (Don't make me
choose!)

<p id="buttons"><button type="submit" name=
"submit" value="submit"><img src="check.gif"
width="40" height="40" alt="Vote button" />
Vote</button>

<button type="reset"> <img src="reset.gif"
width="40" height="40" alt="Reset button"
/>Reset</button></p>
```

Figure 17.49 *Make sure you set the* `type` *to* `reset`. *Otherwise, the button won't actually do anything at all.*



Which one is *your* favorite?

- Woody
- Cookie
- Xixona
- Llumeta
- [✓ All of them (Don't make me choose!)]

✓Vote ✗Reset

Figure 17.50 *Now both the submit and reset buttons really stand out. (It really works, by the way. Try it on my Web site—see page 26.)*

You can add images, font choices, and even a background color to your reset button.

To create a reset button with an image:

1. Type **<button type="reset">**.
2. Type the text, if any, that should appear on the left side of the image in the button.
3. Type **<img src="image.url"**, where *image.url* is the name of the image that will appear on the button.
4. Type **alt="alternate text"**, where *alternate text* is what appears if the image doesn't.
5. If desired, add any other image attributes.
6. Type **/>** to complete the image.
7. Type the text, if any, that should appear on the right side of the image in the button.
8. Type **</button>**.

✓ **Tips**

- You can also use the `button` tag to create a reset button without an image. Just skip steps 3–6.
- For information on creating buttons with scripts, consult *Creating a Button that Executes a Script* on page 316.
- Current browsers support the `button` tag quite well. Older browsers do not, despite it being a standard part of (X)HTML.
- I can't think of any good reason to add the `value` attribute to a reset button, but you might want to add a `name` attribute for scripting purposes.

Using an Image to Submit Data

You may use an image—called an *active image*—as a combination input element and submit button. In addition to submitting the data from the other fields in the form, a click on the image sends the current mouse coordinates to the server in two name-value pairs.

To use an image to submit data:

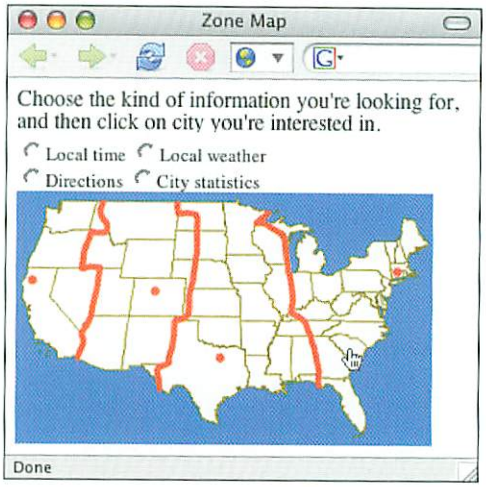
1. Create a GIF or JPEG image.
2. Type **<input type="image"**.
3. Type **src="image.url"**, where *image.url* is the location of the image on the server.
4. Type **name="label"**, where *label* will be appended by *x* and *y* and will identify the x and y coordinates sent to the server, when the visitor clicks the image.
5. Type **alt="description"**, where *description* will appear if the image does not.
6. Type the final **/>** to finish the active image definition for the form.

✓ Tips

- Setting the `value` attribute has no effect. The values are set to the mouse coordinates automatically.
- In a Perl script, the names are set to *label.x* and *label.y* for the x and y coordinates respectively, where *label* matches what you used in step 4 above, *x* is the horizontal distance in pixels from the image's left edge, and *y* is the vertical distance in pixels from the image's top edge. In a PHP script, the names are changed to *label_x* and *label_y* instead since PHP doesn't accept periods in variable names.
- You can find the PHP script used in this example on my Web site *(see page 26)*.

```
<input type="radio" name="infotype"
value="directions" />Directions

<input type="radio" name="infotype"
value="statistics" />City statistics <br /></small>

<input type="image" src="zonemap.gif"
name="coord" alt="US Map, Click to submit data"
/>

</form>
```

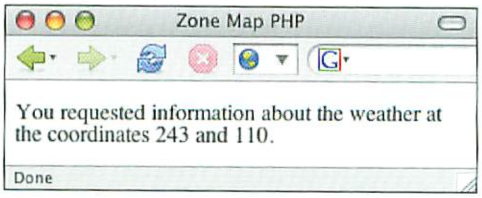
Figure 17.51 *If you use an active image, you don't need a submit button.*



Choose the kind of information you're looking for, and then click on city you're interested in.

Local time · Local weather · Directions · City statistics

Figure 17.52 *The same form can have both regular fields (like the radio buttons) and an image map. When the visitor clicks the map, all of the data is sent to the script.*



You requested information about the weather at the coordinates 243 and 110.

Figure 17.53 *This script outputs the coordinates from the mouse click in the image along with the radio button value.*

```
<body>
<a href="moreinfo.html" tabindex="4">About our company</a>
<h2>Please tell us more about yourself:</h2>
<form method="post" action="showform.php">
<p>Name: <input type="text" name="firstname" size="20" tabindex="1" /></p>
<p>Email address: <input type="text" name="email" tabindex="2" /></p>
<p>Hobbies: <input type="text" name="hobbies" tabindex="3" size="25" /></p>
<p><input type="submit" value="Tell us" /></p>
</form>
</body>
```

Figure 17.54 *By setting the* `tabindex`, *you control the order in which your visitor can tab through the fields.*

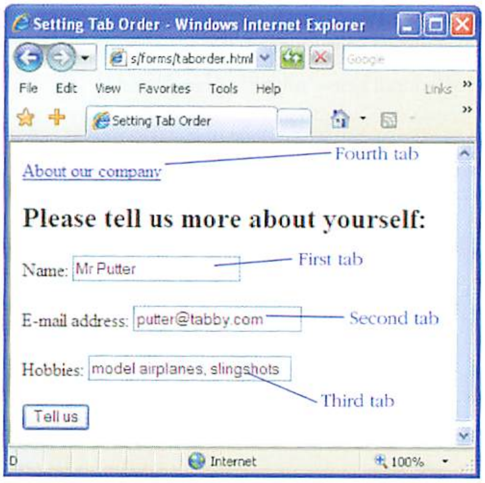

Setting Tab Order - Windows Internet Explorer
s/forms/taborder.html
File Edit View Favorites Tools Help
Setting Tab Order
About our company
Fourth tab
Please tell us more about yourself:
Name: Mr Putter
First tab
E-mail address: putter@tabby.com
Second tab
Hobbies: model airplanes, slingshots
Third tab
Tell us
Internet
100%

Figure 17.55 *With forms on a page that begins with a link, you may want to change the tab order so that the first tab takes you to the first field, not the first link.*

Setting the Tab Order in a Form

By pressing the Tab key, visitors can move the focus through the fields in your form from top to bottom (and then select the desired one by pressing Return). Depending on your form's layout, you may prefer to set the tab order yourself so that the visitor fills out all the fields in a particular group before going on to the next group.

To set the tab order:

In the form element's tag, type **tabindex="n"**, where *n* is the number that indicates the tab order.

✓ Tips

- *Getting the focus* means the form element is selected but not activated. Activation requires pressing the Return key (or a keyboard shortcut—see page 278).
- The value for `tabindex` can be any number between 0 and 32767.
- By default, the tab order depends on the order of the elements in the (X)HTML code. When you change the tab order, the lower numbered elements get the focus first, followed by higher numbered elements.
- In a form, you can assign tab order to text fields, password fields, checkboxes, radio buttons, text areas, menus, and buttons.
- You can also assign tab order to links *(see page 113)* and client-side image maps *(see page 117)*.
- OK, I cannot tell a lie. Where the first Tab keystroke lands you depends on your browser and how it's configured. On IE 7 it took me 9 tabs to get to the *first* one.

Adding Keyboard Shortcuts

Keyboard shortcuts let your visitors select and activate links without using a mouse.

To add a keyboard shortcut to a form element:

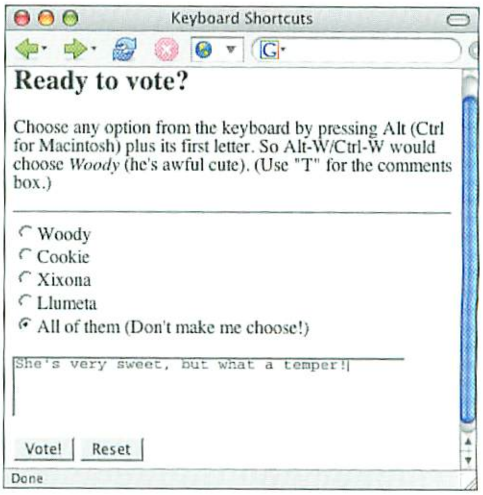
1. Inside the form element's tag, type **accesskey="**.
2. Type the keyboard shortcut (any letter or number).
3. Type the final **"**.
4. If desired, add information about the keyboard shortcut to the text so that the visitor knows that it exists.

✔ Tips

- Keyboard shortcuts are case-insensitive.
- On Windows systems, to invoke the keyboard shortcut, visitors use the Alt key plus the letter you've assigned. On Macs, visitors use the Control key.
- Explorer for Windows has supported keyboard shortcuts since version 4. Firefox and other Gecko browsers support them as well. Opera has its own keyboard navigation system and will ignore yours.
- When a visitor uses a keyboard shortcut it not only gives the element the focus, but actually activates it. In the case of radio buttons and checkboxes, this means the item is selected. If it's a text box, the cursor is placed inside (after any existing text). If it's a button, the button is activated.

```
<p>Choose any option from the keyboard by
pressing Alt (Ctrl for Macintosh) plus its first letter.
So Alt-W/Ctrl-W would choose
<em>Woody</em> (he's awful cute). (Use "T" for
the comments box.)</p><hr />

<form action="http://www.cookwood.com/cgi-
bin/vote.cgi" method="post">

<input type="radio" name="cats" value="W"
accesskey="w" />Woody<br />

<input type="radio" name="cats" value="C"
accesskey="c" />Cookie<br />

<input type="radio" name="cats" value="X"
accesskey="x" />Xixona<br />

<input type="radio" name="cats" value="L"
accesskey="l" />Llumeta<br />

<input type="radio" name="cats" value="A"
accesskey="a" />All of them (Don't make me
choose!)

<p><textarea name="comments" cols="40"
rows="3" accesskey="t">Any special
comments?</textarea></p>

<p><input type="submit" value="Vote!"
accesskey="v" />
```

Figure 17.56 *Keyboard shortcuts can be especially helpful on mobile devices.*



Ready to vote?

Choose any option from the keyboard by pressing Alt (Ctrl for Macintosh) plus its first letter. So Alt-W/Ctrl-W would choose *Woody* (he's awful cute). (Use "T" for the comments box.)

- ○ Woody
- ○ Cookie
- ○ Xixona
- ○ Llumeta
- ● All of them (Don't make me choose!)

She's very sweet, but what a temper!

Figure 17.57 *Pressing Ctrl-T (Alt-T on Windows) puts the cursor in the comments box.*

```
<input type="radio" name="cats" value="L"
accesskey="l" onclick=
"document.vote.submit.disabled=false" />Llumeta
<br />

<input type="radio" name="cats" value="A"
accesskey="a" onclick=
"document.vote.submit.disabled=false" />All of
them (Don't make me choose!)

<p><input name="submit" type="submit"
value="Vote!" accesskey="v" disabled="disabled"
/>

<input type="reset" value="Reset" accesskey="r"
onclick="document.vote.submit.disabled=true"
/></p>
```

Figure 17.58 *Here, I use JavaScript and the* `disabled` *attribute to make the submit button inaccessible until other options are selected.*

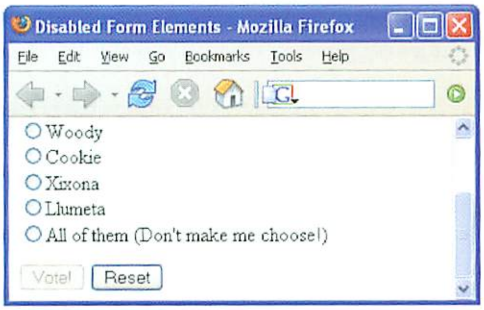

Disabled Form Elements - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Woody
Cookie
Xixona
Llumeta
All of them (Don't make me choose!)
Vote!
Reset

Figure 17.59 *When the visitor first views the form, nothing is selected and the submit button is disabled.*

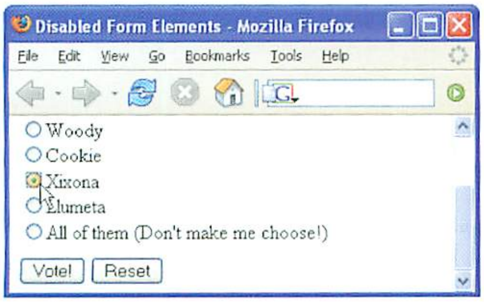

Disabled Form Elements - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Woody
Cookie
Xixona
Llumeta
All of them (Don't make me choose!)
Vote!
Reset

Figure 17.60 *When the visitor chooses an item, the submit button is no longer disabled (thanks to the JavaScript in Figure 17.58).*

Disabling Form Elements

In some cases, you may not want visitors to use certain parts of your form. For example, you might want to disable a submit button until all the required fields have been filled out.

To disable a form element:

In the form element's tag, type **disabled="disabled"**.

✓ Tips

- In HTML, you can just use `disabled` by itself. XHTML requires the redundant value.
- You can change the contents of a disabled form element with a script. For more information on scripting, consult Chapter 19, *Scripts*. You'll also need some JavaScript expertise. The very simple way I've added here is to add **onclick= "document.vote.submit.disabled=false"** to each radio button (where *vote* is the value of the form's `name` attribute, *submit* is the value of the disabled button's `name` attribute, and *disabled* is the attribute in that button whose value I want to change to *false*). So when one of the radio buttons is clicked, the Vote button will be enabled.
- If you disable a form element, its keyboard shortcut is also disabled. For more information on keyboard shortcuts, consult *Adding Keyboard Shortcuts* on page 278.

Keeping Elements from Being Changed

Sometimes it may be necessary to automatically set the contents of a form element and keep the visitor from changing it. For example, you could have the visitor confirm information, or you could show a past history of transactions and then submit that information again with the new data collected. You can do this by making the element "read-only".

To keep elements from being changed:

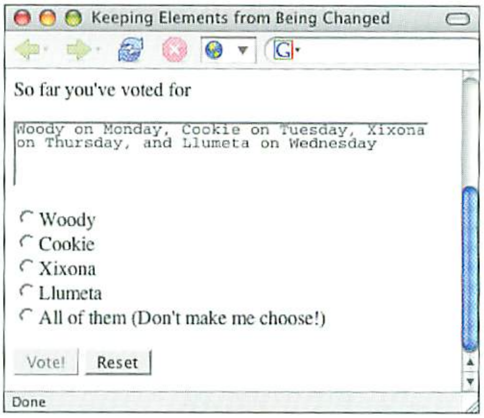
Type **readonly="readonly"** in the form element's tag.

✓ Tips

- In HTML, you can just use `readonly` by itself. XHTML requires the redundant value.
- You can use the `readonly` attribute in text boxes, password boxes, checkboxes, radio buttons, and text areas.
- Setting the `readonly` attribute is something like using a hidden field without making it hidden. For more information on hidden fields, consult *Creating Hidden Fields* on page 271.

```
<p><textarea name="votehistory" cols="40"
rows="3" readonly="readonly">Woody on
Monday, Cookie on Tuesday, Xixona on Thursday,
and Llumeta on Wednesday</textarea></p>
```

Figure 17.61 *Add the* `readonly` *attribute to any form element that you want to show to your visitors but that you don't want them to change.*



So far you've voted for

```
Woody on Monday, Cookie on Tuesday, Xixona
on Thursday, and Llumeta on Wednesday
```

- ○ Woody
- ○ Cookie
- ○ Xixona
- ○ Llumeta
- ○ All of them (Don't make me choose!)

Figure 17.62 *In this example, the visitor's prior votes are displayed in the read-only area. They can be viewed—but not changed—by the visitor and then submitted with the new vote.*