You'll add a `form` element to the survey page within the main section of the page.

To add the `form` element:

1. Return to the **survey.htm** file in your text editor. Directly above the closing `</section>` tag, insert the following `form` element as shown in Figure 6-4:

```
<form id="survey" name="survey">
</form>
```

Inserting a form element

```
<section>
   <p>Required values are marked by an asterisk (*)</p>

   <form id="survey" name="survey">
   </form>

</section>
```

2. Save your changes to the file.

Interacting with a Web Server

Another set of form attributes specifies where to send the form data and how to send it. You indicate this information by adding the `action`, `method`, and `enctype` attributes t the `form` element, as follows

```
<form action="url" method="type" enctype="type"> ... </form>
```

where *`url`* specifies the filename and location of the program that processes the form, the `method` attribute specifies how Web browsers should send data to the server, and th `enctype` attribute specifies the format of the data stored in the fields.

The `method` attribute has two possible values: `get` and `post`. The **get method**, the default, appends the form data to the end of the URL specified in the `action` attribute. The **post method**, on the other hand, sends form data in a separate data stream. Each method has its uses. Web searches often use the get method because the search param- eters become part of the URL and thus can be bookmarked for future searching using th same parameters. However, this also can result in a long and cumbersome URL if sever fields and field values are attached to the URL, and it may even result in data being truncated if the URL text string becomes too long. There is also a security risk in hav- ing name/value pairs attached to a URL that easily can be read by others. Your Web site administrator can supply the necessary information about which of the two methods yo should use when accessing the scripts running on its server.

The `enctype` attribute determines how the form data should be encoded as it is sent to the server. Figure 6-5 describes the three most common encoding types.

Values of the enctype attribute

Value	Description
application/x-www-form-urlencoded	The default format. In this format, form data is transferred as a long text string in which spaces are replaced with the + character and nontext characters (such as tabs and line breaks) are replaced with their hexadecimal code values. Field names are separated from their field values with an = symbol.
multipart/form-data	Used when sending files to a server. In this format, spaces and nontext characters are preserved, and data elements are separated using delimiter lines. The action type of the `form` element must be set to `post` for this format.
text/plain	Form data is transferred as plain text with no encoding of spaces or nontext characters. This format is most often used when the action type of the `form` element is set to `mailto`.

Alice tells you that your survey form will be processed by the CGI script located at the URL *http://www.redballpizza.com/cgi-bin/survey* (a fictional address) using the post method. You do not have to specify a value for the `enctype` attribute because the default value of `application/x-www-form-urlencoded` is appropriate. Add this information to the `form` element now.

To add attributes to the `form` element:

1. Return to the **survey.htm** file and add the following attributes to the `form` element, as shown in Figure 6-6:

```
action="http://www.redballpizza.com/cgi-bin/survey"
method="post"
```

Setting the form attributes

```
<form id="survey" name="survey"
      action="http://www.redballpizza.com/cgi-bin/survey"
      method="post">
```

2. Save your changes to the file.

Because *www.redballpizza.com/cgi-bin/survey* does not correspond to a real CGI script running on the Web and thus cannot process the survey form you'll create in this tutorial, you'll add a JavaScript program named *formsubmit.js* to handle the form. The purpose of this program is to intercept the contents of the form before the browse attempts to contact the CGI script. The script also will report whether or not the data tained in the survey form has been correctly filled out. You'll add a link to this script

To link to the *formsubmit.js* JavaScript program:

1. Return to the **survey.htm** file in your text editor.
2. Go to the head section of the document. Directly below the `script` element that accesses the *modernizr.js* file, insert the following code, as shown in Figure 6-7:

```
<script src="formsubmit.js"></script>
```

Linking to the formsubmit.js file

```
<meta charset="UTF-8" />
<title>Customer Survey</title>
<script src="modernizr-1.5.js"></script>
<script src="formsubmit.js"></script>

<link href="rb.css" rel="stylesheet" />
```

3. Save your changes to the file.

Now that you've added the `form` element to the survey page, you can start populating the survey form with control elements and other form features. You'll start by adding field sets.

Creating a Field Set

A Web form like the survey form can have dozens of different fields. One way of organizing a form is to group similar fields into **field sets**. When rendered by a browser, a field set is usually displayed with a box enclosing the fields in the set. Field sets are created using the `fieldset` element

```
<fieldset id="id">
   controls
</fieldset>
```

where *id* identifies the field set and *controls* is the control elements associated with fields within the field set. The `id` value is not required, but it is useful in distinguishing one field set from another. Alice wants you to organize the form into two field sets named `custInfo` and `experience`.

REFERENCE

Creating a Field Set

- To create a field set, add the element

```
<fieldset id="id">
   controls
</fieldset>
```

to the form, where *id* identifies the field set and *controls* is the control elements associated with fields within the field set.

- To add a legend to a field set, nest the element

```
<legend>text</legend>
```

within the `fieldset` element, where *text* is the text of the legend.

To insert a field set:

1. Within the `form` element in the **survey.htm** file, insert the following two field sets, as shown in Figure 6-8:

```
<fieldset id="custInfo">
</fieldset>

<fieldset id="experience">
</fieldset>
```

Inserting field sets

```
<form id="survey" name="survey"
      action="http://www.redballpizza.com/cgi-bin/survey"
      method="post">

   <fieldset id="custInfo">
   </fieldset>

   <fieldset id="experience">
   </fieldset>

</form>
```

2. Save your changes to the file.

Every field set can contain a legend describing its contents. The syntax of the `legend` element is

```
<legend>text</legend>
```

where `text` is the text of the legend. The `legend` element can contain only text and no other page elements. Based on Alice's sketch from Figure 6-2, you'll add the legend text *Customer Information* and *Share Your Experience at Red Ball Pizza* to the two field sets you just created.

To insert legends for the field sets:

1. Within the first field set in the *survey.htm* file, insert the following `legend` element:

```
<legend>Customer Information</legend>
```

2. In the second field set, insert the following `legend` element:

```
<legend>Share Your Experience at Red Ball Pizza</legend>
```

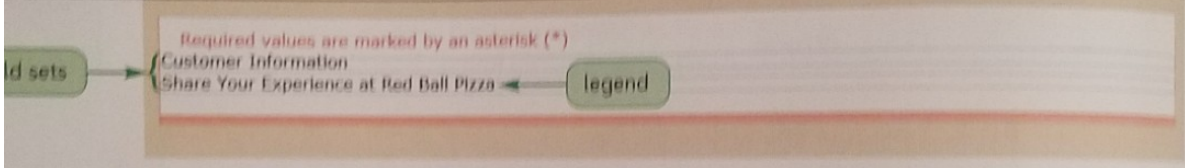
Figure 6-9 highlights the revised text in the HTML file.

ure 6-9 **Inserting field set legends**

```
<form id="survey" name="survey"
      action="http://www.redballpizza.com/cgi-bin/survey"
      method="post">

   <fieldset id="custInfo">
      <legend>Customer Information</legend>
   </fieldset>

   <fieldset id="experience">
      <legend>Share Your Experience at Red Ball Pizza</legend>
   </fieldset>

</form>
```

3. Save your changes to the file and then refresh the **survey.htm** file in your Web browser. Figure 6-10 shows the current appearance of the form.

ure 6-10 **Viewing field sets and legends**



Field sets are block elements that expand to accommodate their content. The field sets you added are currently empty, so they appear small and narrow on the survey page. By default, browsers display the legend text in the upper-left corner of the field set box. However, you can use CSS positioning styles to move the legend and format its appearance. Next, you'll add control elements and other content to the two field sets.

Creating Input Boxes

Most of the control elements in which users either type or select a data value are marked as `input` elements using the `<input>` tag

```
<input type="type" name="name" id="id" />
```

where *type* specifies the type of input control, the `name` attribute provides the name of the field associated with the control element, and the `id` attribute identifies the control element itself. When the form data is submitted to the server, the server program pairs the field name with the field value; thus you almost always need a `name` attribute if you are submitting the form to a server. The `id` attribute is required only when you need to reference the control element itself, as you might if you intend to apply a CSS style rule to modify the appearance of the control element. In this tutorial, you'll supply both a `name` and an `id` attribute for every control, and you'll set them to the same value.

HTML 4 supports 10 different input types, which are described in Figure 6-11; HTML5 adds 13 new input types that you'll explore later in this tutorial.

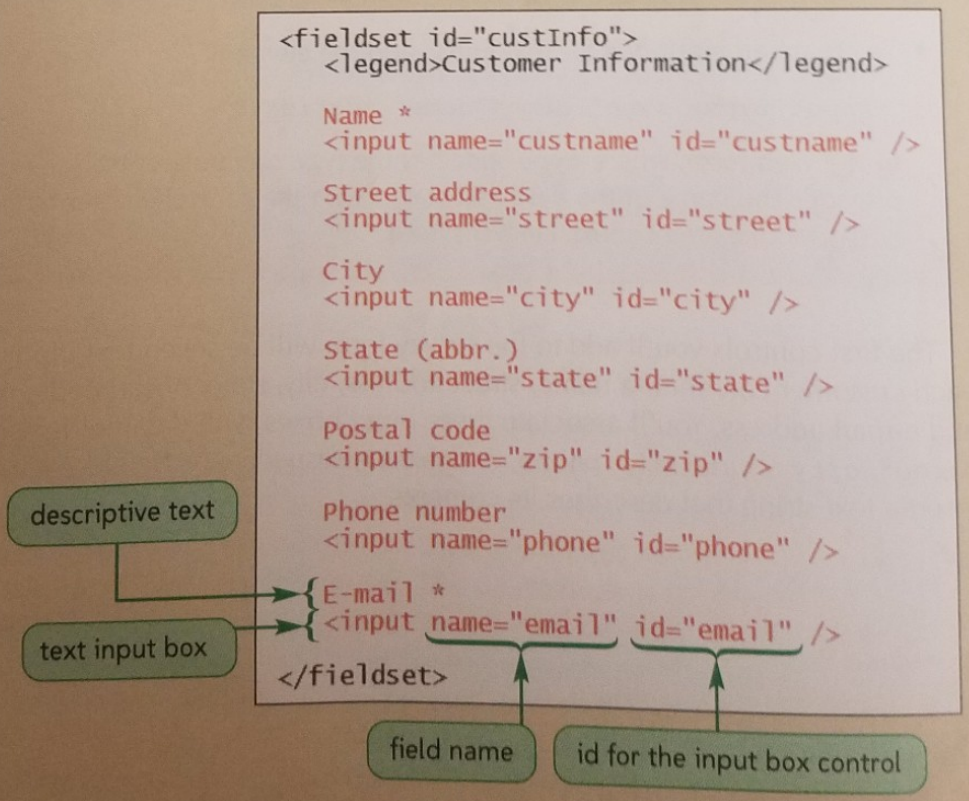
P
ribute is
wsers assume
control is a

To add the input boxes for the customer information:

1. Return to the **survey.htm** file. Within the `custInfo` field set, insert the following code as shown in Figure 6-12:

```
Name *
<input name="custname" id="custname" />

Street address
<input name="street" id="street" />

City
<input name="city" id="city" />

State (abbr.)
<input name="state" id="state" />

Postal code
<input name="zip" id="zip" />

Phone number
<input name="phone" id="phone" />

E-mail *
<input name="email" id="email" />
```

6-12 **Inserting input boxes to record customer information**



Note that the asterisks next to the Name and E-mail text entries tell users that these fields are required. Later on in this tutorial, you'll learn how to make it ma datory that users enter data into required fields.

2. Save your changes to the file.

Next, you'll add text input boxes for the date of the customer's visit to Red Ball Pizza and the receipt number of the order. The names for these data fields are `visitdate` and `receipt`, respectively.

To add the input boxes for the customer information:

1. Within the Share Your Experience at Red Ball Pizza field set, insert the following input boxes (see Figure 6-13):

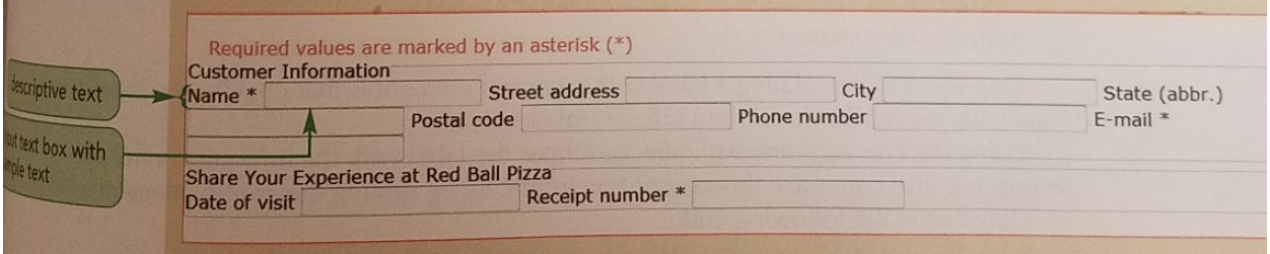
```
Date of visit
<input name="visitdate" id="visitdate" />

Receipt number *
<input name="receipt" id="receipt" />
```

Figure 6-13 Inserting input boxes to record the customer's experience

```
<fieldset id="experience">
   <legend>Share Your Experience at Red Ball Pizza</legend>

   Date of visit
   <input name="visitdate" id="visitdate" />

   Receipt number *
   <input name="receipt" id="receipt" />

</fieldset>
```

2. Save your changes to the file and then refresh **survey.htm** in your Web browser.
3. Test the controls by typing your ***first and last name*** in the `Name` input box. Figure 6-14 shows the newly inserted control elements in the form with sample text in the `custname` field.

Figure 6-14 Input boxes with sample customer name



Note that browsers treat all form control elements as text-level elements, so the input boxes that you created for the survey form appear within the same line rather than in separate blocks.

INSIGHT

Navigating Forms with Access Keys

You activate control elements like input boxes either by clicking them with your mouse button or by tabbing from one control element to another. As your forms get longer, you might want to give users the ability to jump to a particular input box. This can be done with an access key. An **access key** is a single key on the keyboard that you type in conjunction with the Alt key commonly used for Windows users, or the control key for Mac users, to jump to a spot in the Web page. You create an access key by adding the `accesskey` attribute to an element. For example, to create an access key for the `custName` input box, you would enter the following code:

```
<input name="custName" id="custName" accesskey="l" />
```

If a user types Alt+l (or control+l for Mac users), the cursor automatically moves to the `custName` input box. Note that you must use letters that are not reserved by your browser. For example, Alt+f is used by many browsers including Internet Explorer to access the File menu and thus should not be used for an access key.

Access keys also can be used with hypertext links and are particularly helpful to users with impaired motor skills who find it difficult to use a mouse.

Adding Field Labels

In the last set of steps, you entered descriptive text alongside the input boxes to indicate the purpose of each input box to users. However, nothing in the HTML code explicitly associates that text with the input box. To associate text with a control element, you enclose the descriptive text within a `label` element as follows

```
<label for="id">label text</label>
```

where *`id`* is the value of the `id` attribute of the control element associated with the label, and *`label text`* is the text of the label. For example, the following code associates the label text *Street address* with the `street` control element:

```
<label for="street">Street address</label>
<input id="street" />
```

One effect of associating a label with a control element is that clicking the label automatically moves the cursor into the control element.

Using the `for` attribute explicitly associates the label with the relevant control element. You also can make this association implicitly by nesting the control element within the label as in the following code:

```
<label>
   Street address
   <input id="street" />
</label>
```

Notice that you do not need to include a `for` attribute when you nest the control element within the label element.

Which approach you take depends on how you want to lay out a form's contents. When you use the `for` attribute, you can place the label text anywhere within the Web page and it still will be associated with the control element. However, by nesting the control element within the label, you can treat both the control element and its label as a single object, which can make form layout easier because you can move both the label text and the control element as a single unit around the page. Depending on the layout of your Web form, you might use both approaches.

REFERENCE

Creating a Field Label

- To explicitly associate a text label with a control element, use the `label` element and the `for` attribute

```
<label for="id">label text</label>
```

where *id* is the id of the control element.

- To implicitly associate a text label with a control element, nest the control element within the `label` element as follows

```
<label>
   label text
   control
</label>
```

where *control* is the control element. You do not have to include a `for` attribute.

You'll use the `label` element and the `for` attribute to associate the descriptive text you've entered in the survey form with the relevant input boxes.

To apply the field labels:

1. Return to the **survey.htm** file in your text editor.
2. Go to the customer information field set and enclose the text string *Name* * within a `label` element, associating it with the `custname` input box as follows:

```
<label for="custname">Name *</label>
```

3. Repeat this process for the remaining descriptive text strings in the two field sets, using the `for` attribute to associate each label with the corresponding input box. Figure 6-15 shows the revised code in the file, highlighting the different values of the `for` attribute.

Adding form labels

```
<fieldset id="custInfo">
   <legend>Customer Information</legend>

   <label for="custname">Name *</label>
   <input name="custname" id="custname" />

   <label for="street">Street address</label>
   <input name="street" id="street" />

   <label for="city">City</label>
   <input name="city" id="city" />

   <label for="state">State (abbr.)</label>
   <input name="state" id="state" />

   <label for="zip">Postal code</label>
   <input name="zip" id="zip" />

   <label for="phone">Phone number</label>
   <input name="phone" id="phone" />

   <label for="email">E-mail *</label>
   <input name="email" id="email" />

</fieldset>

<fieldset id="experience">
   <legend>Share Your Experience at Red Ball Pizza</legend>

   <label for="visitdate">Date of visit</label>
   <input name="visitdate" id="visitdate" />

   <label for="receipt">Receipt number *</label>
   <input name="receipt" id="receipt" />

</fieldset>
```

4. Save your changes to the file and then refresh the **survey.htm** file in your Web browser.
5. Test the labels by clicking each label and verifying that the cursor appears with the corresponding control element.

INSIGHT

Setting the `autocomplete` *Attribute*

Many browsers include an autocomplete feature that automatically fills in input form values if they are based on previously filled out forms. For example, a user who routinely fills in his or her street address in a multitude of Web forms can enable the browser to remember the information and to insert it automatically when the browser encounters another street address field in another form.

The autocomplete feature is a useful time-saver in most cases, but it also can be a security risk for a personal computer located in a public place. After all, you may not want to have a private credit card number or password automatically filled in by a browser on a computer that other people will be using.

One way to prevent this problem is by implementing the HTML5 `autocomplete` attribute. The `autocomplete` attribute can be added to a `form` or `input` element to turn off (or turn on) the form complete feature. For example, the `input` element

```
<input name="creditcardnumber" autocomplete="false" />
```

prevents browsers from automatically filling in the `creditcardnumber` field, even if they have credit card data handy. Currently, the `autocomplete` attribute is supported by Opera and Firefox, but not by Safari, Chrome, or Internet Explorer. However, you may want to consider using it if you want to add another level of security to confidential information.

Applying a Style Sheet to a Web Form

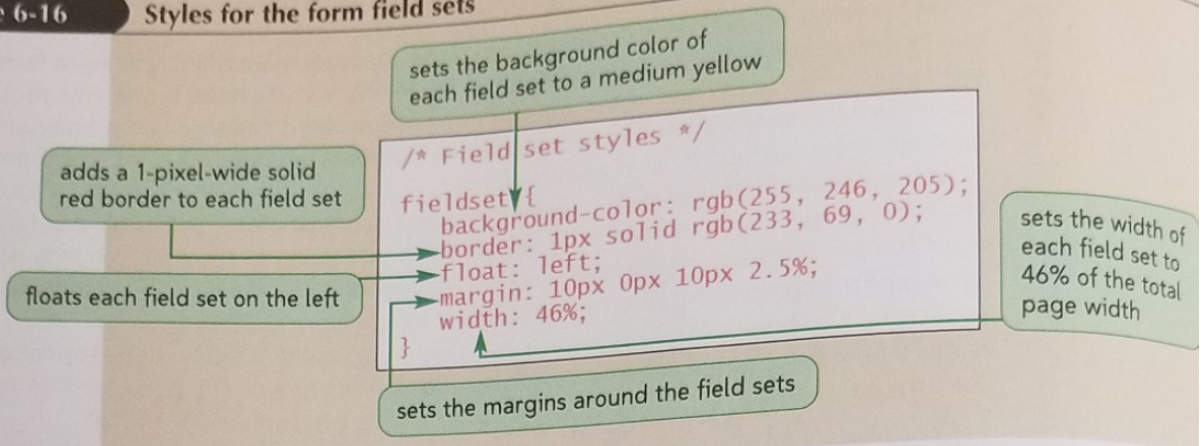
Alice stops by to see your progress on the survey form. In its current state, the form is difficult to read and she wants you to replace the default styles for the form with your own customized style sheet. First, she wants the two field sets to be floated side-by-side within the main section of the page and resized to be about half the width of the Web page. You'll add a style rule to an external style sheet file to modify the appearance of these field sets.

To create the form style sheet:

1. Use your text editor to open the **formstxt.css** file from the tutorial.06\tutorial folder. Enter ***your name*** and ***the date*** in the comment section of the file, and then save it as **forms.css** in the same folder.
2. Below the comment section, add the following style rule as shown in Figure 6-16:

```
/* Field set styles */

fieldset {
   background-color: rgb(255, 246, 205);
   border: 1px solid rgb(233, 69, 0);
   float: left;
   margin: 10px 0px 10px 2.5%;
   width: 46%;
}
```

Figure 6-16 Styles for the form field sets



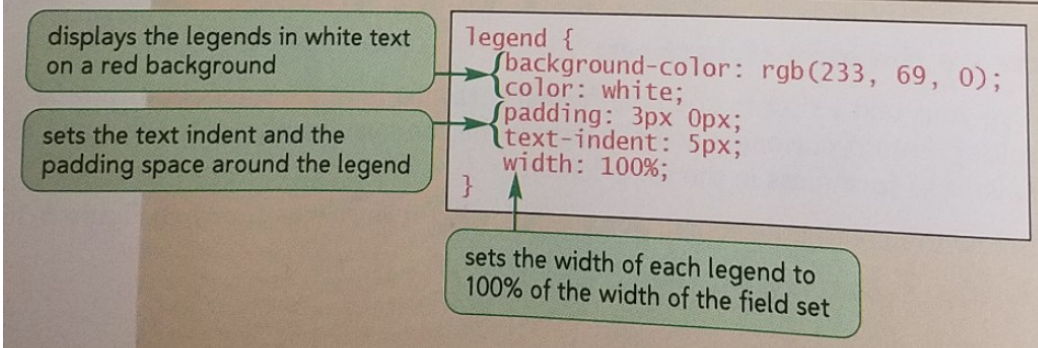
Next, you'll change the style of the field set legends, defining each legend to appear as white text on a medium-red background.

To add a style rule for the field set legends:

1. Directly below the style for the field set selector, add the following style rule as shown in Figure 6-17:

```
legend {
   background-color: rgb(233, 69, 0);
   color: white;
   padding: 3px 0px;
   text-indent: 5px;
   width: 100%;
}
```

Figure 6-17 Styles for the field set legends



2. Save your changes to the file.

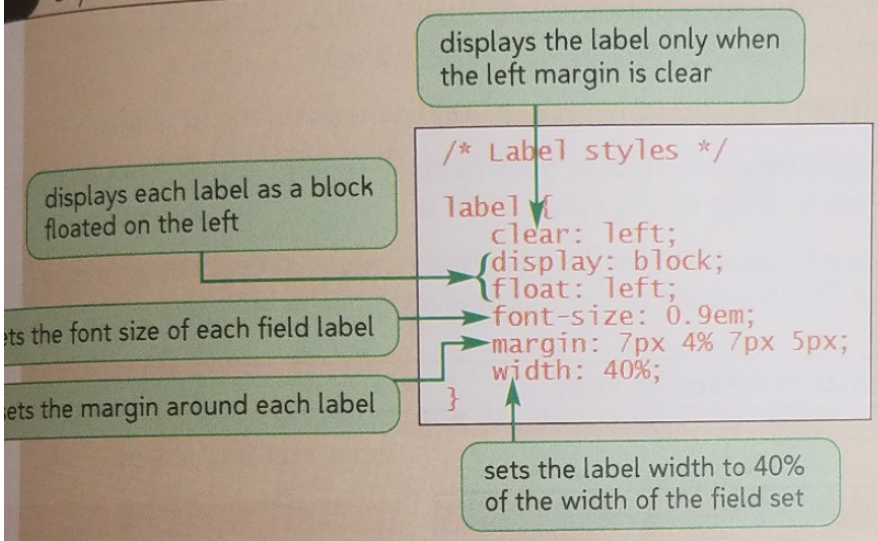
Finally, since the default style for labels and input controls is to display them inline, they appear to run together on the form. Alice suggests that you instead display them as blocks and float them side-by-side within the two field sets to make them easier to read

To define a style for the labels and input controls:

1. At the bottom of the style sheet, add the following style rule for the form labels (see Figure 6-18):

```
/* Label styles */

label {
   clear: left;
   display: block;
   float: left;
   font-size: 0.9em;
   margin: 7px 4% 7px 5px;
   width: 40%;
}
```

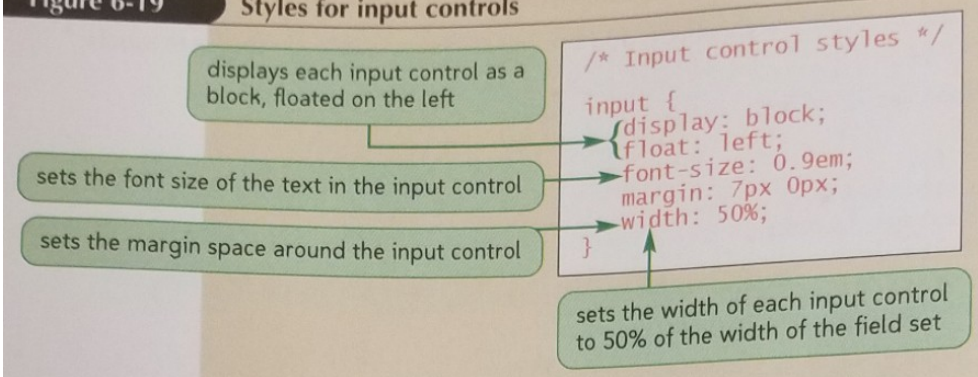
Styles for the field labels



2. Below the style rule you just created, add the following style rule for input controls (see Figure 6-19):

```
/* Input control styles */

input {
   display: block;
   float: left;
   font-size: 0.9em;
   margin: 7px 0px;
   width: 50%;
}
```

Figure 6-19 **Styles for input controls**



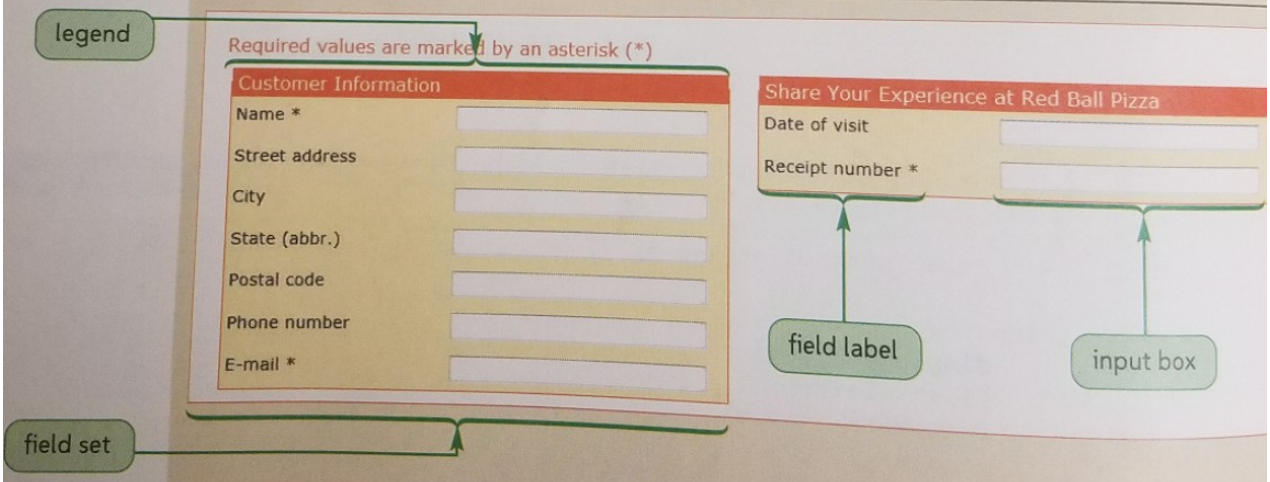
3. Save your changes to the **forms.css** file and then return to the **survey.htm** file in your text editor.
4. Below the `link` element that links the file to the *rb.css* style sheet, add the following element to link to the *forms.css* style sheet:

```
<link href="forms.css" rel="stylesheet" />
```

Now you'll view the effect of your fieldset, legend, label, and input styles on the appearance of the survey form.

5. Save your changes to the file and then refresh **survey.htm** in your Web browser. Figure 6-20 shows the revised appearance of the Web form.

Figure 6-20 **Revised format of the survey form**



Trouble? If your form does not match that shown in Figure 6-20, check your style sheet code against that shown in Figures 6-16 through 6-19. Make sure you have not mistyped a style property or forgotten to separate one style from another with a semicolon.

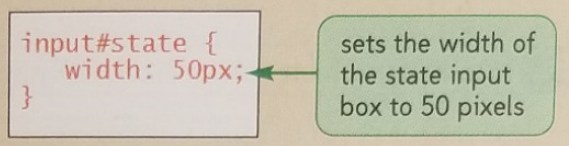
Not all of the input boxes need to have a width of 50%. The input box for the `state` field only needs to be large enough to display the two-letter abbreviation of the state name. You'll reduce the width of this input box now.

To reduce the width of the `state` field input box:

1. Return to the **forms.css** file in your text editor and add the following style rule at the bottom of the file (see Figure 6-21):

```
input#state {
   width: 50px;
}
```

21 **Setting the width of the state input box**



2. Save your changes to the file and then refresh **survey.htm** in your Web browser. Verify that the width of the input box for the `state` field has been reduced.

Another way to set the width of an input box is by adding the following `size` attribute to the `input` element

```
size="chars"
```

where *chars* is the width of the input box in characters. For example, the HTML code

```
<input name="state" size="2" />
```

sets the size of the input box for the state field to two characters in width. Note that this is not an exact measure because the width of individual characters varies depending on the typeface and font style.

Defining Default Values and Placeholders

More than 90% of Red Ball Pizza customers come from Ormond Beach in Florida. Rather than forcing these customers to enter the same city and state information in the survey form, Alice wants you to specify Ormond Beach, FL as the default city and state values. To define a default value, you add the attribute

```
value="value"
```

to any form control, where `value` is the default field value that initially will appear in the control when the form is opened by the browser. For example, the following input control sets the default value of the `city` field to *Ormond Beach*:

```
<input name="city" value="Ormond Beach" />
```

REFERENCE

Defining a Default Field Value

- To define the default value of a field, add the attribute

```
value="value"
```

to the control element, where *value* is the default value assumed by a browser unless a user enters a different value.

You'll add the `value` attribute to the `city` and `state` input boxes, setting the default values of the fields to *Ormond Beach* and *FL*, respectively.

To set the default `city` and `state` field values:

1. Return to the **survey.htm** file in your text editor and scroll down to the `input` element for the `city` field. Add the attribute `value="Ormond Beach"` to the `<input>` tag.
2. Add the attribute `value="FL"` to the `<input>` tag for the `state` field. Figure 6-22 highlights the revised HTML code for the two input boxes.

ure 6-22 Defining default values for the city and state fields

default value for the city field

```
<label for="city">City</label>
<input name="city" id="city" value="Ormond Beach" />

<label for="state">State (abbr.)</label>
<input name="state" id="state" value="FL" />
```

default value for the state field

3. Save your changes to the file and then refresh **survey.htm** in your Web browser. Verify that the input boxes for the `city` and `state` fields show the text values *Ormond Beach* and *FL*, respectively.

Starting with HTML5, you can also populate your input boxes with placeholders. A **placeholder** is a text string that appears within the control element and provides users with information about the kind of information accepted by the field. To create a placeholder, add the attribute

```
placeholder="text"
```

to the control element, where *text* is the text of the placeholder. For example, the following `input` element for the `phone` field provides guidance to users about the format in which phone numbers can be entered:

```
<input name="phone" placeholder="(nnn) nnn-nnnn" />
```

When a browser displays the form, the text *(nnn) nnn-nnnn* will appear grayed out in the phone input box. This text will indicate to a customer reading the form that he or she should enter a phone number, including both the area code and the seven-digit number.

Unlike a default field value, a placeholder is not stored in the data field and is not sent to the server as a field value. The placeholder automatically disappears as soon as a user selects the input box. At the time of this writing, all current browsers except Internet Explorer support placeholders.

Alice wants you to add placeholders to the `custname`, `postal code`, `phone number`, and `receipt` input boxes; the placeholders should provide information to customers about the types of data they should enter and the expected formats. You'll add the placeholders now.

To create placeholders for the survey form:

1. Return to the **survey.htm** file in your text editor and scroll to the `input` element for the `custname` field. Add the following attribute to the `<input>` tag:

   `placeholder="first and last name"`

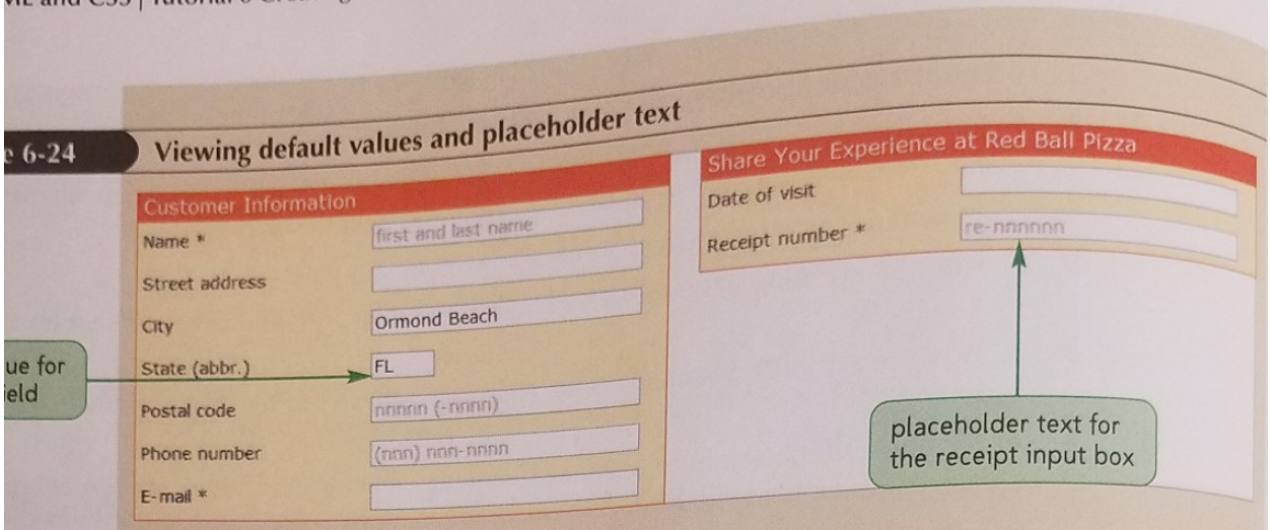
2. Add the attribute `placeholder="nnnnn (-nnnn)"` to the `input` element for the `zip` field.
3. Add the attribute `placeholder="(nnn) nnn-nnnn"` to the `input` element for the `phone` field.
4. Finally, add the attribute `placeholder="re-nnnnnn"` to the `input` element for the `receipt` field. Figure 6-23 highlights the newly added code in the survey form.

Adding placeholders to the survey form

```
<fieldset id="custInfo">
   <legend>Customer Information</legend>

   <label for="custname">Name *</label>
   <input name="custname" id="custname"
          placeholder="first and last name" />

   <label for="street">Street address</label>
   <input name="street" id="street" />

   <label for="city">City</label>
   <input name="city" id="city" value="Ormond Beach" />

   <label for="state">State (abbr.)</label>
   <input name="state" id="state" value="FL" />

   <label for="zip">Postal code</label>
   <input name="zip" id="zip"
          placeholder="nnnnn (-nnnn)" />

   <label for="phone">Phone number</label>
   <input name="phone" id="phone"
          placeholder="(nnn) nnn-nnnn" />

   <label for="email">E-mail *</label>
   <input name="email" id="email" />

</fieldset>

<fieldset id="experience">
   <legend>Share Your Experience at Red Ball Pizza</legend>

   <label for="visitdate">Date of visit</label>
   <input name="visitdate" id="visitdate" />

   <label for="receipt">Receipt number *</label>
   <input name="receipt" id="receipt"
          placeholder="re-nnnnnn"/>

</fieldset>
```

5. Save your changes to the file and then refresh **survey.htm** in your Web browser. As shown in Figure 6-24, placeholder text has been added to the `custname`, `zip`, `phone`, and `receipt` input boxes. Notice that placeholder text is distinguished from default text by appearing in a grayed-out font.

e 6-24 **Viewing default values and placeholder text**



Trouble? If you are using Internet Explorer or Firefox, you might not see any placeholders in the survey form.

6. Click in the input boxes that contain placeholders. Notice that the placeholders disappear as soon as you click each input box.

PROSKILLS

Decision Making: Using HTML5 Form Attributes

HTML5 offers several useful features such as the `placeholder` attribute, but the specification is still new and not universally supported in the browser market. This poses a problem for Web designers who must decide whether or not to use such attributes.

One school of thought holds that differences in appearance and functionality between one browser and the next confuse users and make it more difficult to manage the operations of a Web form. Thus, a feature like the HTML5 `placeholder` attribute should not be used. If a Web designer needs placeholders in a Web form, they should be created using a JavaScript program that can be applied uniformly across browsers and browser versions.

The opposing view holds that the best design is one that uses each browser to its utmost capabilities, and proposes that a Web designer cheats users when the designer decides to forgo an HTML feature that enhances the user experience, like the `placeholder` attribute. Moreover, the Web will only gain in the long run as more HTML5 features are employed because increasing their use will encourage more rapid support of HTML5 across the browser market.

To decide between these two approaches, you must evaluate whether the HTML5 feature you're adding is critical to understanding and using your Web form. Do placeholders add useful, but not essential, information for users? If so, there is little to be lost by adding them. However, even if you want all users to have placeholders in their forms, applying the `placeholder` attribute still can be useful as a quick and easy way of developing a prototype of your form for testing. Later, as you reach the final phase of development, you can replace it with a JavaScript program that extends placeholders to all users.

You've finished the initial stage of developing the survey form. Alice is pleased with the form's appearance and content. In the next session, you'll extend the form by adding new fields and control elements, including selection lists, option buttons, and check boxes.

Input box data types

Type	Displays	General Appearance
button	A button that can be clicked to perform an action from a script	Run Program
checkbox	A check box that can be clicked by the user	
file	A Browse button to locate and select a file	C:\survey.htm Browse...
hidden	A hidden field, not viewable on the form	
image	An inline image that can be clicked to perform an action from a script	
password	An input box that hides text entered by the user	•••••••
radio	An option button that can be clicked by the user	
reset	A button that resets the form when clicked	Cancel Form
submit	A button that submits the form when clicked	Submit Form
text	An input box that displays text entered by the user	Alice Nichols

REFERENCE

Creating an Input Control

- To create an input box for text entry, add the element

```
<input type="type" name="name" id="id" />
```

to the Web form, where *type* specifies the type of input control, the `name` attribute provides the name of the field associated with the control element, and the `id` attribute identifies the control element itself.

The first controls you'll add to the survey form will be seven text input boxes in which each customer can enter a name, street address, city, state, postal code, phone number, and e-mail address. You'll associate these input boxes with data fields named *custname*, *street*, *city*, *state*, *zip*, *phone*, and *email*, respectively. Before each input box you'll insert a text string that describes its contents.