You create a text box input field by setting the `type` attribute value to `text`:

```
<input type="text" name="age" size=3>
```

The `name` attribute defines a unique identifier that allows you to retrieve the value entered into the field. It's important that you include that attribute. The `size` attribute allows you to set how large the form field appears on the web page. The default value is `20`, which is a bit much for entering an age, so I've changed it to `3`.


TIP

You can define a default value that appears in the form field using the `value` attribute. This feature is useful if you're trying to get your site visitor to update information that's already in your database. Just display the existing data as the default values for each form field.


TIP

The `disabled` attribute prevents you from entering data into the text field. It may sound weird to display a text field that you can't enter data into, but it has a purpose when you learn how to dynamically change the input fields using JavaScript later on in Book 3.

You can associate a label with a text box by using the *label element.* The input element should be enclosed in the label opening and closing tags:

```
<label>
   Last Name
   <input type="text" name="lastname">
</label>
```

With this format, you can use CSS to style and position both the label and the text box field at the same time.

Another feature is the ability to group input fields together into a *fieldset.* A fieldset creates a border area around the enclosed form fields to help separate them out in the web page. The format to use a fieldset is:

```
<fieldset>
   <legend>Enter your name</legend>
   <label>
      Last Name
      <input type="text" name="lastname">
   </label>
   <label>
      First Name
      <input type="text" name="firstname">
   </label>
</fieldset>
```

The legend element allows you to define text that appears in the fieldset border area. Figure 3-1 shows how this form looks.

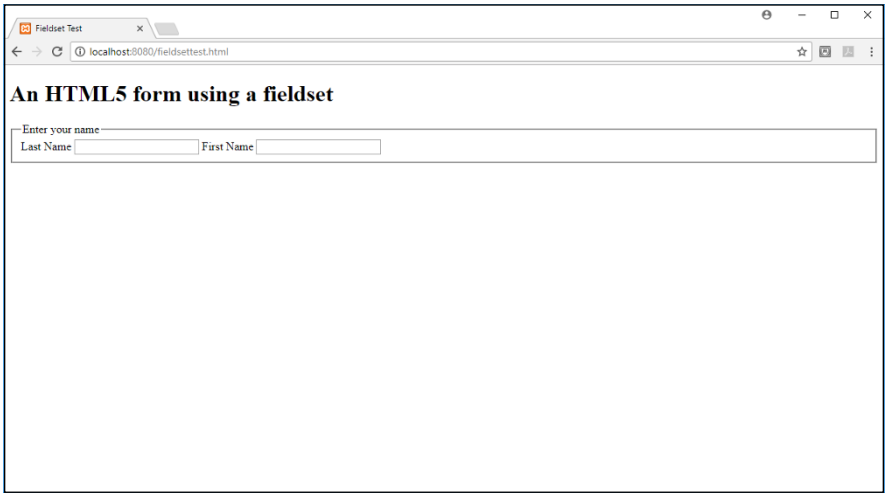

FIGURE 3-1: Using a fieldset to group form fields.

The nice thing about the fieldset is that you can assign it an `id` attribute and then apply specific styles to the entire group in CSS3.

Password entry

Many web applications require that site visitors enter sensitive information in the form, such as Social Security numbers (SSNs). The input element provides an easy way to hide that information from prying eyes trying to watch as visitors enter their data.

The `password` input field type instructs the browser to mask the characters as the site visitor enters them into the text box. Here's the format to create a password field:

```
<input type="password" name="ssn">
```

As your site visitor types data into the password form field, the browser masks the characters by displaying a neutral, nondescript character. Just how the characters are masked depends on the browser. Most browsers use bullet circles in the field.

Check boxes

Check boxes provide a simple yes-or-no response form field. The `checkbox` input type creates a simple square box that the site visitor can click. The check box field toggles with each click — from showing a check mark in the box to not showing a check mark in the box.

To define a `checkbox` input type, use the following format:

```
<input type="checkbox" name="fishing">
```

The `name` attribute defines the name that's passed along to the action URL when the site visitor submits the form. The value sent is a Boolean true/false value — true if the check box is marked, and false if the check box is not marked.

Because the check box field is just a box, you'll most likely want to associate a label with the check box field so your site visitors know what they're selecting. Often, check boxes are used in groups, so you can use the `fieldset` element:

```
<fieldset>
   <legend>Please select which sports you like</legend>
   <label>
      Baseball
      <input type="checkbox" name="baseball"><br>
   </label>
   <label>
      Basketball
      <input type="checkbox" name="basketball"><br>
   </label>
   <label>
      Football
      <input type="checkbox" name="football"><br>
   </label>
   <label>
      Hockey
      <input type="checkbox" name="hockey"><br>
   </label>
</fieldset>
```

Figure 3-2 shows how this form looks in the browser window.

FIGURE 3-2: Using check boxes in a fieldset.

You can also set a default state for the check box, but not by using the `value` attribute. Instead, you have to use the `checked` attribute:

```
<input type="checkbox" name="football" checked>
```

The `checked` attribute doesn't have a value associated with it. If it appears in the input element, the check box appears with a check mark in it.

Radio buttons

A similar interface to check boxes are *radio buttons.* Radio buttons allow you to select only one out of a group of options. You create radio buttons by using the `radio` input type:

```
<input type="radio" name="sports">
```

To group options together, you have to assign them all the same name attribute. Then the browser will allow your site visitors to select only one option from the group. That code would look like this:

```
<fieldset>
   <legend>Please select your favorite sport</legend>
   <label>
      Baseball
      <input type="radio" name="sport"><br>
   </label>
```

```
    <label>
       Basketball
       <input type="radio" name="sport"><br>
    </label>
    <label>
       Football
       <input type="radio" name="sport" ><br>
    </label>
    <label>
       Hockey
       <input type="radio" name="sport"><br>
    </label>
 </fieldset>
```

Figure 3-3 shows how the radio buttons appear on the web page.

FIGURE 3-3: Using radio buttons to make a selection from a group.

As your site visitor selects each option, the previously selected option is reset. Only one value is sent back to the server from the form field.

TIP

If you'd like to set a default value for the radio button group, add the `checked` attribute to that radio button element.

Hidden fields

Your application may need to pass data behind the scenes as part of the application control. Perhaps it's a product ID value related to an item the site visitor is purchasing or an employee ID number in a human resources application. Not all data that the form submits needs to be seen by the site visitor.

To accommodate that, HTML uses the `hidden` input type:

```
<input type="hidden" name="productid" value="121">
```

The hidden form field doesn't appear in the form itself, so you have to use the `value` attribute to assign a value to the form field that gets passed to the server. When the site visitor clicks the Submit button to submit the form data, any hidden form fields that are defined are sent along with the normal form field data.

File upload

If your application requires that your site visitors upload files, you'll want to explore the `file` input type. The `file` input type produces an input field with two parts:

- A text box to display the filename
- A Browse button to launch a file manager

In some browsers, you can manually type the filename in the text box, but many of the popular browsers prevent that. The Browse button appears next to the text box, allowing site visitors to search for the file to upload. The interface that's used for searching depends on the OS the browser is running on. On Windows workstations, clicking the Browse button launches the File Explorer tool. On macOS workstations, clicking the Browse button launches the Finder tool. Figure 3-4 shows how the field appears on the web page.

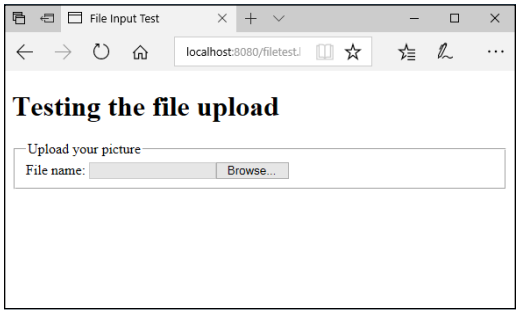

FIGURE 3-4: The file input type interface as shown in the Microsoft Edge browser.

The format of the file input field is:

```
<input type="file" name="upload">
```

That's simple enough! However, you need to take care of one more thing when using the file input field. By default, the form sets the `enctype` attribute for

encoding characters before they're uploaded. Most likely, your upload files will contain binary data, and encoding that data will corrupt it.

To solve that problem you need to set the `enctype` attribute in the `<form>` opening tag to use the `multipart/form-data` value:

```
<form method="POST" action="myhost.com/index.php" enctype="multipart/form-data">
```

This ensures that the binary data contained in the uploaded file is uploaded in binary format, but the data contained in the other form fields are properly encoded for upload.

Buttons

Button, button, who's got the button? That's just a silly child's game, but buttons are a crucial part of your web forms. Buttons allow your site visitor to trigger actions on the web page, from launching JavaScript programs to uploading the form data to the server.

There are three types of button input types available to use: `button`, `reset`, and `submit`.

Button

The `button` field type creates a generic button to trigger an event. When a site visitor clicks the button, nothing happens by default. The trick is to define an action using the `onclick` attribute:

```
<input type="button" name="launch" value="Click Me" onclick="myprogram()">
```

The `value` attribute defines what text appears in the button. The browser will automatically size the button to fit the text you specify. The `onclick` attribute defines a JavaScript function that the browser runs when you click the button.

Reset

The `reset` field type resets any values in the form data fields back to their original values — either to empty if no default value is defined or to the default value if it's defined:

```
<input type="reset" name="reset" value="Reset fields">
```

Submit

The `submit` input field type is a crucial part of most forms. It signals to the browser that it's time to upload the form field data values to the server:

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

By default, the button appears with *Submit* as the button label. You can change the button text using the `value` attribute. It's customary to place the Submit button at the bottom of a form, but that isn't required. You can place the Submit button anywhere between the opening `<form>` tag and the closing `</form>` tag.

Adding a Text Area

Text boxes are extremely versatile, but there's a limit to what they can do. If you need to enter large amounts of text, the text box scrolls to allow you to enter the text, but you lose sight of the text you previously typed.

The `textarea` element provides a larger interface for entering text. To create a text area, you use the following opening and closing tags:

```
<textarea name="story"></textarea>
```

That, by itself, though, won't give you what you're looking for. There are a few attributes that you'll want to use to define the text area. Table 3-3 shows the attributes you can use.

TABLE 3-3 The textarea Attributes

Attribute	Description
`cols`	Specifies the width of the text area in the web page
`disabled`	Grays out the text area so nothing can be typed
`name`	Specifies the form field name associated with the field
`readonly`	Locks the text area so nothing can be typed, but default text can be displayed
`rows`	Specifies the height of the text area in the web page

So, to create a text area that's 20 characters wide by 30 characters high, you'd use the following:

```
<textarea name="story" cols=20 rows=30></textarea>
```

Your site visitors can then type their text in the text area. If they type more than 30 rows of text, the browser will add a scrollbar to the right side of the text area and allow them to continue typing.


TIP

You'll notice that in my text area examples, there's nothing between the opening and closing `textarea` tags. That produces an empty text area. Any text that you place between the opening and closing tags appears as the default text in the text area.

Using Drop-Down Lists

Often, you want to limit the choices your site visitors have for a specific data field. To do that, you use a drop-down list. The drop-down list appears in the form as a single line, similar to a text box, but with a down arrow. If you click the down arrow, a box drops down with all the options available in it. You can then select one or more options from the drop-down list.

In the HTML5 world, this feature is called a `select` element. The `select` element consists of two parts:

- The `select` opening and closing tags to define the select element
- One or more option elements that define the allowed options

Here's an example of a simple `select` element (see Figure 3-5):

```
<select name="sports">
<option value="baseball">Baseball</option>
<option value="basketball">Basketball</option>
<option value="football">Football</option>
<option value="hockey">Hockey</option>
</select>
```

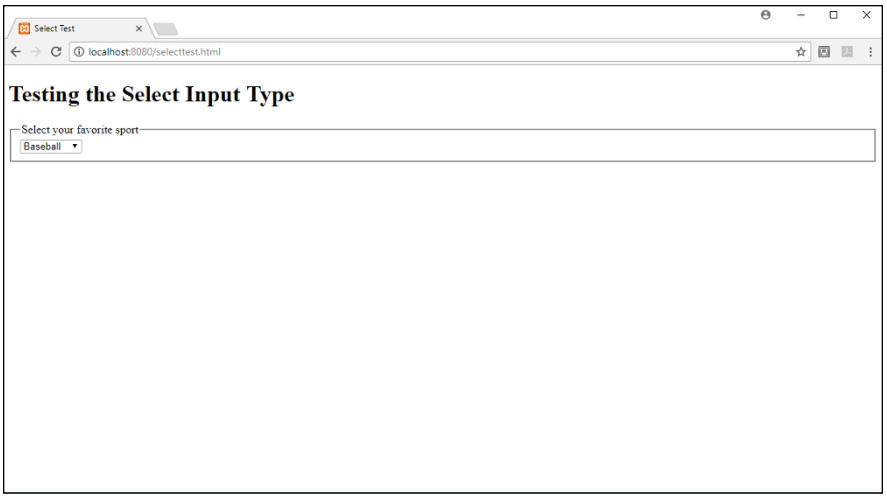

FIGURE 3-5: Using the `select` element.

With this format, the browser displays a single text box along with a down arrow indicating that there's a drop-down list to select from. When you click the arrow, you see the list.

If you prefer to have more of the options appear on the web page than just one, set the `size` attribute in the <select> tag:

```
<select name="sports" size="4">
```

This creates a list of options that you can scroll through, as shown in Figure 3-6.

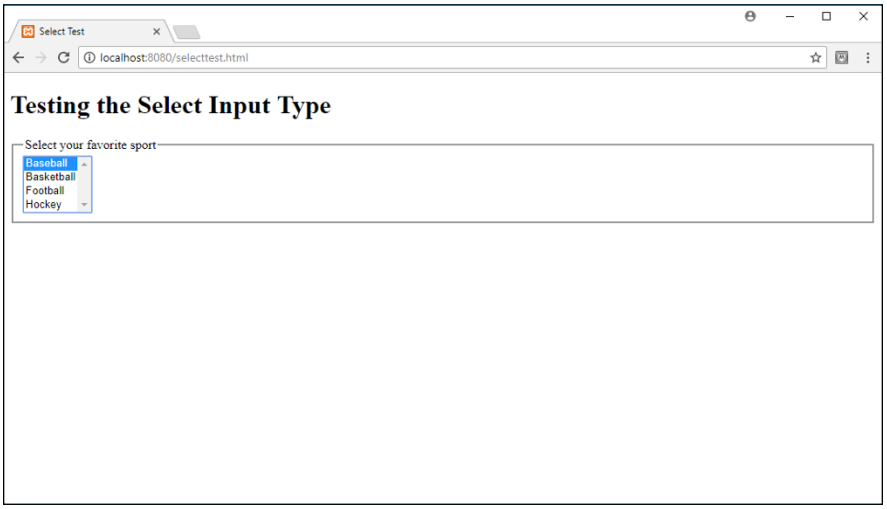

FIGURE 3-6: Displaying multiple options in the select element.

Each option element defines one item in the select list. The browser displays the text between the opening `<option>` and closing `</option>` tags, but it sends the `value` attribute of the item your site visitor selects to the server. This can come in handy if you want to use abbreviations or codes in your data, but you want to display the full text to the site visitors.


TIP

By default, the select element only allows the site visitor to select one value. You can change that behavior by setting the `multiple` attribute in the `<select>` opening tag.

Enhancing HTML5 Forms

The original HTML standards were pretty bare-bones with the form field options. These days web developers gather all types of information from forms. To help with that, the HTML5 standard defines some fancier form types that you can use. This section walks you through what those are.

Data lists

The `datalist` element is new to HTML5. It allows you to create an option list for drop-down lists that use the autocomplete feature, made popular by Google searching. As you start typing a value in the text box, the list that appears in the drop-down box narrows to only the values that match what you've typed.

The data list feature requires three parts:

- An `<input>` tag that defines the data list
- A `datalist` element that defines the list
- One or more `<option>` tags that define the list values

A complete data list looks like this:

```
<input list="sports">
<datalist id="sports">
    <option value="Baseball">
    <option value="Basketball">
    <option value="Football">
    <option value="Hockey">
</datalist>
```

The `list` attribute in the `<input>` tag refers to the data list `id` attribute value for the data list to use. This allows you to define multiple data lists in your form. Figure 3-7 shows how the data list looks in action.

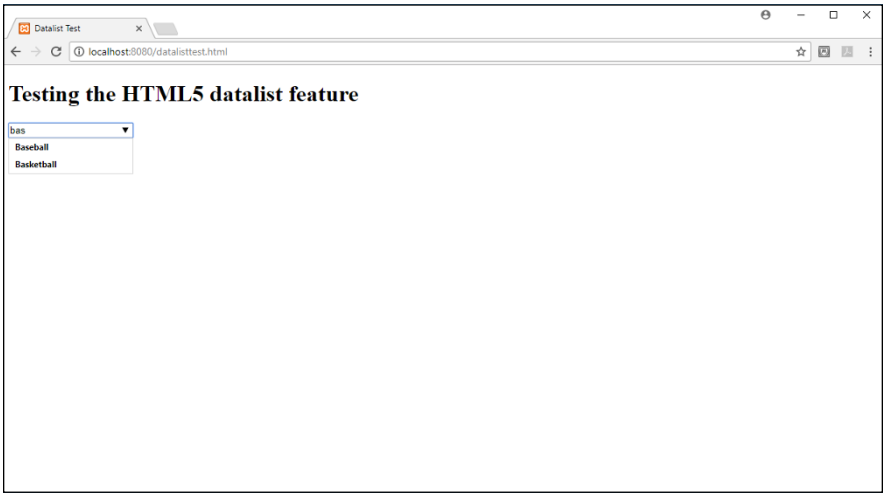

FIGURE 3-7: Using a data list in the web page.

In this example, as I typed the characters, the matching data list values appeared in the drop-down box, limiting my choices. Notice that the match is case insensitive and that the match is made anywhere in the text string of the option values.

Additional input fields

One of the more exciting features in the HTML5 standard form additions are the additions to the `<input>` tag. HTML5 defines 13 additional input element types:

- `color`: Produces a color palette for the site visitor to select a color. Returns the RGB color value associated with the selected color.
- `date`: Produces a graphical month calendar to select a date. Returns the selected year, month, and day values.
- `datetime`: Produces a graphical month calendar to select a date and a text box to select the time. Returns the selected year, month, date, hour, minute, second, and fraction-of-a-second values, along with the time zone.
- `datetime-local`: Produces the same form field as the `datetime` input type, but doesn't return a time zone.
- `email`: For inputting a single email address or a comma-separated list of email addresses.

- `month`: Produces a graphical month calendar. Returns the year and month selected.
- `number`: Produces a spin box for increasing or decreasing a numeric value in a text box. Returns the numeric value selected.
- `range`: Produces a slider to select a value from a range. You define the range using the `min` and `max` attributes in the tag. Returns the numeric value selected.
- `search`: Produces a text box that some browsers style like a search box (such as with a magnifying glass icon). Returns the value entered into the text box.
- `tel`: Produces a standard text box for entering a telephone number. Some browsers may validate the format of the text entered to ensure it matches a telephone number format. Returns the value entered into the text box.
- `time`: Produces a time selector that shows two numeric values, along with a spin box for increasing or decreasing the values. The numeric values indicate 1 through 12 for the hour and 0 through 59 for the minutes. Returns the values selected in a time format.
- `url`: Produces a text box for entering a text URL. Some browsers may validate the URL format entered. Returns the text entered into the text box.
- `week`: Produces a graphical calendar to select a week number for a specified year. Returns the year and the week number selected.

These produce some pretty amazing input fields in your web pages! The only downside is that different browsers may use different methods to produce these form fields. Let's walk through an example to create a test program so you can see how your browsers handle the new input fields:

1. **Open your favorite text editor, program editor, or integrated development environment (IDE) package.**
2. **Enter the following code:**

```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Input Types Test</title>
</head>
<body>
<h1>Testing the HTML5 Input Types</h1>
<fieldset>
<legend>HTML5 Input Fields</legend>
<label>
Color Selector
<input type="color" name="colortest">
</label><br>
```

```
<label>
Date Selector
<input type="date" name="datetest">
</label><br>
<label>
DateTime Selector
<input type="datetime" name="datetimetest">
</label><br>
<label>
DateTime-Local Selector
<input type="datetime-local" name="datetimelocaltest">
</label><br>
<label>
Email Selector
<input type="email" name="emailtest">
</label><br>
<label>
Month Selector
<input type="month" name="monthtest">
</label><br>
<label>
Number Selector
<input type="number" name="numbertest">
</label><br>
<label>
Range Selector
<input type="range" min=0 max=100 name="rangetest">
</label><br>
<label>
Search Selector
<input type="search" name="searchtest">
</label><br>
<label>
Telephone Selector
<input type="tel" name="teltest">
</label><br>
<label>
Time Selector
<input type="time" name="timetest">
</label><br>
<label>
URL Selector
<input type="url" name="urltest">
</label><br>
```

```
<label>
Week Selector
<input type="week" name="weektest">
</label>
</body>
</html>
```

3. **Save the file as `inputtypestest.html` in the `DocumentRoot` folder for your web server (such as `c:\xampp\htdocs` for XAMPP in Windows, or `/Application/XAMPP/htdocs` for XAMPP in macOS).**

4. **Start your web server.**

5. **Open a browser and enter the following URL:**

```
http://localhost:8080/inputtypestest.html
```

6. **Close the browser window and shut down the web server.**

The `inputtypestest.html` file is a great way to see how the new HTML5 input types look in different browsers. Figure 3-8 shows how they look in the Google Chrome browser.

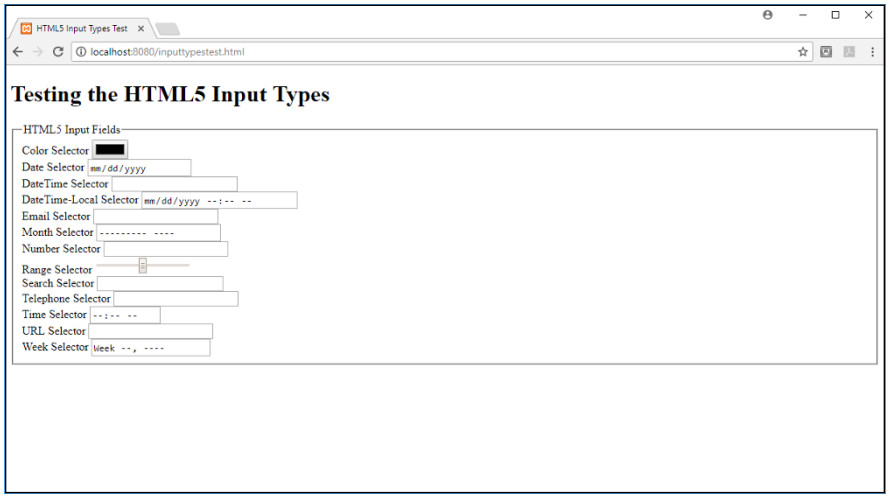

FIGURE 3-8: Viewing the `inputtypestest.html` output in the Google Chrome browser.

TIP

If you have a mobile device handy, try testing the `inputtypestest.html` web page on your mobile device. Mobile devices use virtual keyboards that appear on the screen when you click in an input form field. Most mobile devices will customize the keyboard depending on which type of input form field you click in. For example, in the `tel` input type, the mobile device may only display a numeric keypad

for entering the phone number; for the `email` input type, the mobile device may display a keyboard with a .com button.

Using HTML5 Data Validation

Accepting data from unknown website visitors is a dangerous thing. However, dynamic web applications must have user interaction to work. The conundrum is how to do both.

One method is to use *data validation*, which is the process of verifying that the data your site visitors enter into the form fields is correct. There are two ways to tackle that process:

- On the server, with server-side programming code
- In the client browser, using HTML, CSS, and JavaScript

In Book 4, Chapter 4, I cover all the bases on using server-side programming to validate form data. However, waiting until the browser has uploaded the data to the server to validate it can be somewhat cumbersome. By that time, the site visitor has already entered all the form data. Returning a web page making the site visitor re-enter all that data just because of one typo is not a good way to retain customers.

This is where client-side data validation comes in handy. The more data you can validate in the browser as the site visitor enters it, the better the chance you have of receiving valid data in the first place.

Holding your place

HTML5 helps that process with a few additional features. One such feature is the `placeholder` attribute for the `input` element. The `placeholder` attribute appears as gray text inside the form field and can provide a suggested format for the data to enter:

```
<label>
Enter your daytime phone number:
<input type="tel" name="num" placeholder="(nnn)nnn-nnnn">
</label>
```

The browser displays the placeholder value inside the input form field, but as gray text, as shown in Figure 3-9.

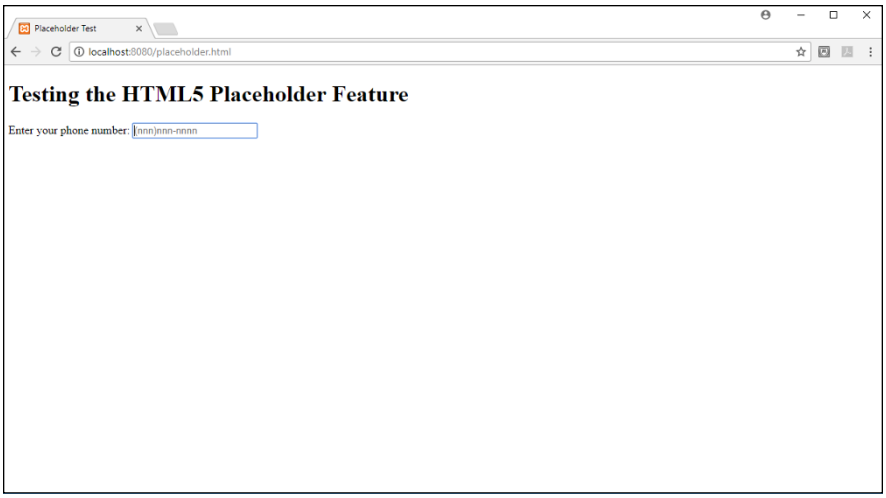

FIGURE 3-9: Using the placeholder HTML5 attribute.

As you start typing text in the input field, the placeholder text disappears.

Making certain data required

Another data validation attribute added by HTML5 is the `required` attribute:

```
<input type="text" name="lastname" required="required">
```

The `required` attribute marks the form field so that the browser won't upload the form if that field is empty. Some browsers will display an error message indicating which required form field(s) are empty.

Validating data types

Not only do the additional HTML5 input types produce different types of input fields, but you can also use them to validate data. Browsers that support the new HTML5 data types will mark input form fields that contain data not in the proper format with the invalid state.

CSS provides pseudo-class rules to style elements based on their state (see Book 2, Chapter 2). You use the `invalid` and `valid` pseudo-class states to style input fields with invalid data differently from input fields with valid data. This helps make the fields with invalid data stand out in the form.

Here's a quick example you can try to test this feature:

1. **Open your favorite text editor, program editor, or IDE package.**

2. **Type the following code:**

```
<!DOCTYPE html>
<html>
<head>
<title>Testing for Invalid Data</title>
</head>
<style>

input:invalid {
    background-color: red;
}

input:valid {
    background-color: lightgreen;
}
</style>
<body>
<h1>Testing for invalid data</h1>
<fieldset>
<legend>You must be over 18 to participate</legend>
<label>
Age:
<input type="number" name="age" min="18">
</label>
</fieldset>
</body>
</html>
```

3. **Save the file as** `invaliddatatest.html` **in the** `DocumentRoot` **folder for your web server (**`c:\xampp\htdocs` **for XAMPP on Windows or** `/Applications/XAMPP/htdocs` **for XAMPP on macOS).**
4. **Start the Apache web server from XAMPP.**
5. **Open a browser and enter the following URL:**

```
http://localhost:8080/invaliddatatest.html
```

6. **Close the browser, and stop the XAMPP web server.**

When the invaliddatatest.html form first appears, the age data field will be empty and colored green. If you use the spinner icons on the right side of the text box, the numbers will start at 18, and the text box will stay green. However, if you try to manually enter an age less than 18, the text box immediately turns red.

IN THIS CHAPTER

- Rounding corners
- Working with border images
- Exploring new colors
- Using gradients
- Lurking in the shadows
- Working with fonts
- Answering media queries

Chapter 4
Advanced CSS3

The previous two chapters show you how to use the combination of HTML5 and CSS to create content and style it for your web pages. CSS3 provides some more advanced features, allowing you to do even *more* styling for your web pages. This chapter walks you through some of the more exciting features from CSS3 that you can use to liven up your site.

Rounding Your Corners

In Book 2, Chapter 3, I explain how to build online forms using HTML5. However, by default, HTML forms are somewhat boring, even after adding some CSS styling.

The default styling used by browsers to display text boxes, buttons, and text areas in forms produces nothing but square boxes, which gets pretty boring. The original CSS standard didn't do anything to solve the problem, other than possibly adding some color to the square boxes. Cubism may be good for some styles of paintings, but that layout doesn't work in forms and can bore your website visitors.

One of the features that had been hotly sought after in the browser world has been the ability to use rounded corners for form elements. The simple act of rounding

the square boxes just a bit can liven up the form. Many individual browsers added the rounded corners feature on their own, separate from the CSS standard. Unfortunately, as you may guess, different browsers used different methods for implementing rounded corners. Trying to write a style that would work across all browsers became both difficult and confusing. But because using rounded corners became so popular, that feature was finally added to CSS3 as a standard.

The new `border-radius` style property allows you to round off the sharp edges from elements on the web page. It does that by allowing you to define the radius of an imaginary circle used to create the rounded corners. You can just shave a little off the edge by using a small radius value, or you can create a full ellipse by completely rounding all four corners with a large radius value. Figure 4-1 shows an example of applying the `border-radius` property to a few form elements.

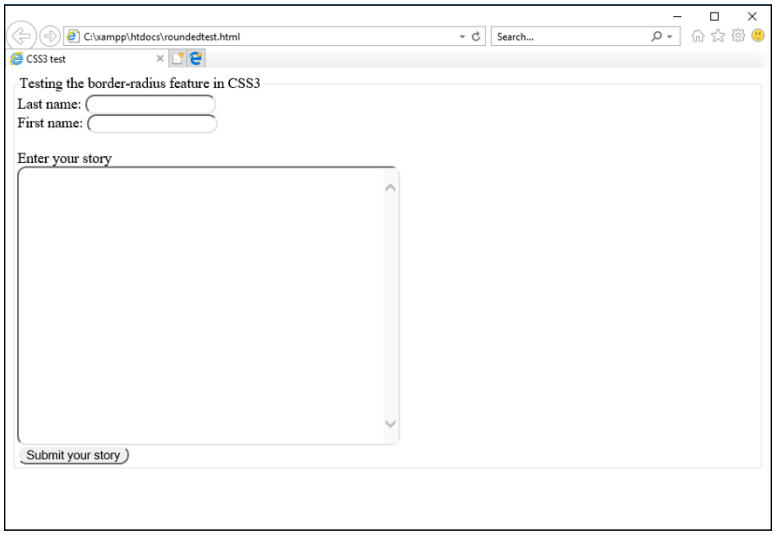

FIGURE 4-1: Using the `border-radius` property to create rounded corners.

Notice that the input text boxes, the text area, and even the Submit button are rounded instead of the standard squares. That makes quite a difference in the appearance of the web form.

What can get confusing with the `border-radius` property, though, is that there are four different formats for using it — with one, two, three, or four parameters. The following single parameter sets the radius of all four corners to 10 pixels:

```
border-radius: 10px;
```

The following two parameters set the radius of the top-left and bottom-right corners to 10 pixels, but the top-right and bottom-left corners to 5 pixels: