19

Creating a Form

Forms turn the Web into a two-way medium. Without forms, website owners could never hear directly from their site visitors outside of other forms of communication. Forms are essential to surveys, polls, contact requests, and online shopping. In this lesson, you learn the basics of how forms are structured as well as the specifics for implementing the key form elements.

UNDERSTANDING FORMS

A form is basically made of four parts:

- The `<form>` tag
- Form controls, such as the `<input>` tag
- Labels, which identify the form elements
- A trigger, typically a button form element, that submits the form

A simple form, in code, would look like this:

```
<form>
  <label>Name:
    <input type="text" name="fullName"
/>
  </label>
  <input type="button" value="Submit" />
</form>
```

When displayed in the browser, the label, text form field, and button are presented all in a single line, as shown in Figure 19-1.

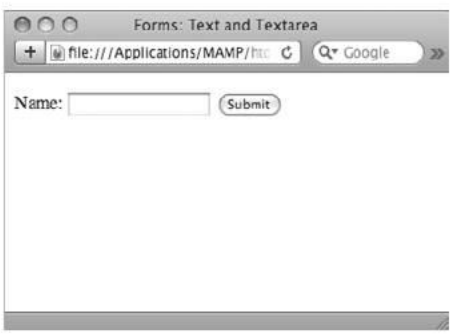

FIGURE 19-1

As you learn later in this lesson, you can use <p> tags, tables, and other HTML elements along with CSS to create a structure to position the form elements.

In the preceding code example, notice how the `<label>` tag wraps around the `<input>` tag, which defines their connection. Many modern web designers use a variation on this technique to connect a given label to a particular form control. This variation uses a `for` attribute in the `<label>` tag that points to an `id` attribute in the form control, like this:

```
<form>
  <label for="fullName">Name: </label>
  <input type="text" name="fullName" id="fullName" />
  <input type="button" value="Submit" />
</form>
```


The `for` attribute technique has a several benefits over the `<label>` tag wrapping method. First, it separates the two tags — `<label>` and `<input>` — which allows the tags to be positioned in separate table cells, a common design approach. In addition, the independent tags make it easier to apply CSS styles; with the `for` attribute technique, you could, for instance, add padding to a `label` tag selector to keep the form controls uniformly distant. Perhaps most importantly, the `for` attribute technique is far more accessible to assistive technology like screen readers.

One of the least understood aspects of website development is how the entries in a form are transmitted to the website owner or other designated party. It is important to understand that some form of server-side processing is necessary for form data to be delivered properly. Such processing usually takes the form of a script that runs on the server natively (in a high-end computer language like Perl) or on installed server applications such as PHP, .NET, or ColdFusion. The specific form processing script to be used is identified with the `action` attribute in the `<form>` tag, like this:

```
<form action="scripts/mailForm.php">
```


The action attribute requires a path to a file; this web address can be a relative or absolute URL. Another attribute, `method`, determines how the data is transmitted to the file noted in the `action` attribute. The `method` attribute accepts one of two values: `get` and `post`. If your code specifies a `method` of `get`, the data is passed via the URL. For example, the following code expands on the prior example:

```
<form action="scripts/mailForm.php" method="get">
  <label for="fullName">Name: </label>
  <input type="text" name="fullName" id="fullName" />
  <input type="button" value="Submit" />
</form>
```

When the user clicks the Submit button, the next web address displayed in the browser will be a combination of the `action` value as well as information from the form, like this:

```
http://www.mysite.com/scripts/mailForm.php?fullName=Joseph%20Lowery
```

The question mark after the name of the referenced page (`mailForm.php`) indicates that what follows is one or more name/value pairs. With forms, the name portion of the pair corresponds to a form control's `id` value, which, here, is `fullName`. The value that follows is what was entered in the form text field, in this case `Joseph Lowery`. The `%20` between the first and last names is a URL-encoded value for a space.

USING TEXT AND TEXTAREA FIELDS

Text fields come in two flavors. When an `<input>` tag's `type` attribute is set to `text`, a single-line text field is rendered in the browser, best used for a limited set of characters. Use the `<textarea>` tag when you want a more open-ended multi-line entry, capable of handling larger blocks of text.

Take a look at the smaller text field first. The code for creating a basic text field is straightforward:

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

Although it's apparently redundant, it is best to include both the `name` and `id` attributes with the same value. The `name` attribute is required and should be both meaningful and unique on the page. The `id` attribute is important for accessibility, most notably providing a hook for the `<label>` tag's `for` attribute.

With CSS you can set the width, alignment, and font characteristics for a text field.

HTML5 brings a wide range of new attributes to the `<input>` tag. However, most are not supported across all browsers as of this writing. Some of the more interesting ones to keep an eye on are:

- `autocomplete`: When this attribute is set to `on`, browsers remember previous entries and will display them in a list when the user types the first couple of letters. If set to `off`, the entries are stored.
- `autofocus`: Allowed only once per form, it establishes the active form control when the page loads. Use the following syntax for the attribute: `autofocus="autofocus"`.
- `max`: Determines the maximum number of characters allowed.
- `min`: Sets the minimum number of characters allowed.
- `placeholder`: The value of this attribute is initially shown in the text field and then removed when the form control is given focus.
- `required`: Ensures that the form field has an entry when the form is submitted.

Currently, support for these attributes is most complete in Opera 10.x and Safari 5.x browsers.

Many other form controls share the `<input>` tag with the `text` type. Web designers often add a custom CSS class to their text fields to allow for more selective customization.

The code for inserting a multi-line `<textarea>` form control is quite different from that of a standard text field:

```
<textarea name="comments" id="comments" cols="50" rows="5"> Tell us about yourself
  in 100 words or less</textarea>
```

Unlike the `<input>` tag, `<textarea>` has both opening and closing tags. Any content within the `<textarea>` tag pair is displayed in the field itself as shown in Figure 19-2. The size of the textarea field can be set in two ways. HTML5 recognizes the `rows` and `cols` attributes, which define the number of lines (the height) and number of characters in each row (the width), respectively. Alternatively, you can create a CSS rule for the `textarea` selector with `width` and `height` properties.

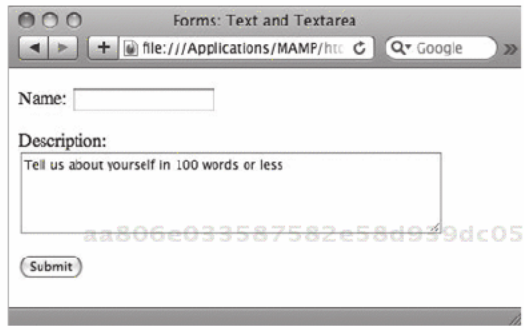

FIGURE 19-2

In HTML5, the `<textarea>` tag supports the `autofocus`, `placeholder`, and `required` attributes previously discussed. In addition, it has a few other attributes specific to itself:

- `maxlength`: Sets the number of characters permitted in the textarea.
- `wrap`: Determines how the text will be submitted. If `wrap="hard"`, line breaks are added at the `cols` value; if `wrap="soft"`, no breaks are added.

TRY IT

In this Try It you learn how to create a form with text and textarea fields.

Lesson Requirements

You will need the `tpa_saturn.html` file from the Lesson_19 folder, as well as a text editor and web browser.

You can download the code and resources for this lesson from the book's web page at `www.wrox.com`.

Step-by-Step

1. Open your text editor.
2. From the Lesson_19 folder, open `tpa_saturn.html`.

3. Put your cursor after the closing `</h2>` tag that contains the text `Contest Entry Form` and press Enter (Return).

4. Enter the following code:

```
<form name="contest" method="post" action="">
  <p>
    <label for="fullName">Name: </label>
    <input type="text" name="fullName" id="fullName">
  </p>
  <p>
    <label for="email">Email: </label>
    <input type="text" name="email" id="email">
  </p>
  <p>
    <label for="entry">Entry: </label>
    <textarea name="entry" id="entry" cols="50" rows="5">Why do you want to
      visit Saturn? (100 words or less)</textarea>
  </p>
  </form>
```

5. Save your file.

6. In your browser, open `tpa_saturn.html` to view the rendered form with the text fields and textarea as shown in Figure 19-3.

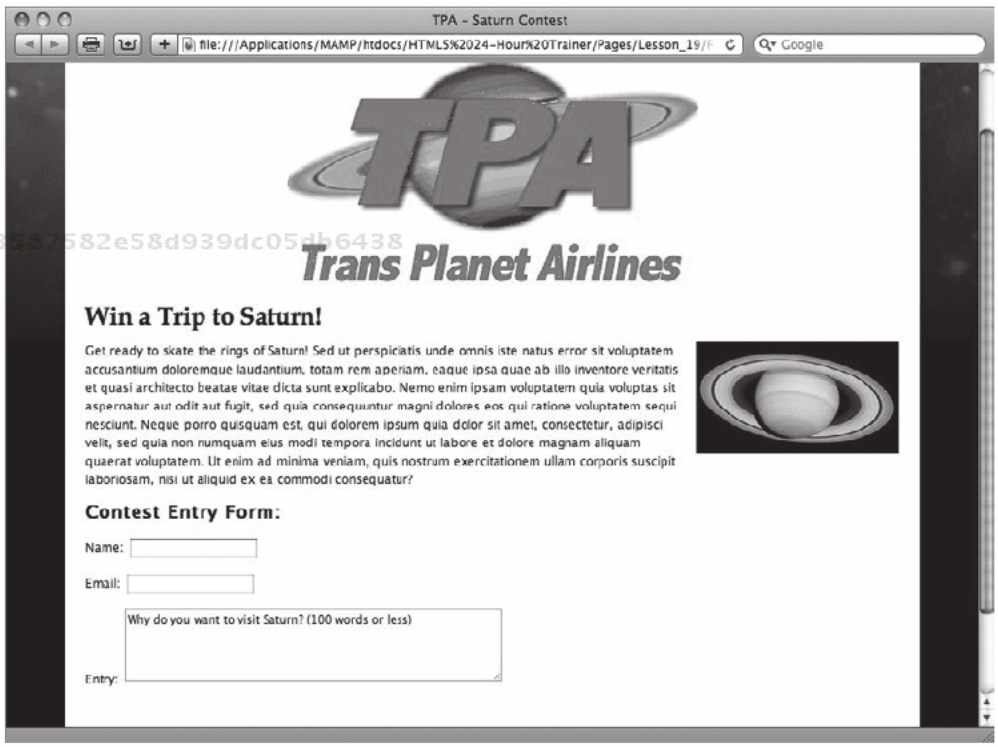

FIGURE 19-3

WORKING WITH RADIO BUTTONS

Radio buttons allow the user to choose one item from two or more options. Users can switch their choices, and the previously chosen option is deselected so that only one option is selected. To make it possible for browsers to understand which radio buttons are part of the same group, the `name` attribute must be the same for all the options. Here's a simple example with two options:

```
<input type="radio" name="gender" id="male" value="male" />
<label for="male">Male</label>
<input type="radio" name="gender" id="female" value="female" />
<label for="female">Female</label>
```

In this example, the `name` attribute is defined as `gender` for both `<input>` tags, whereas the `id` and `value` attributes are different. As with the text and textarea form controls, the `id` attribute is used by the `<label>` for identification. The `value` attribute contains the text string to be submitted if the associated radio button is selected. For example, if someone chooses the Male radio button option, the value sent is `male`.

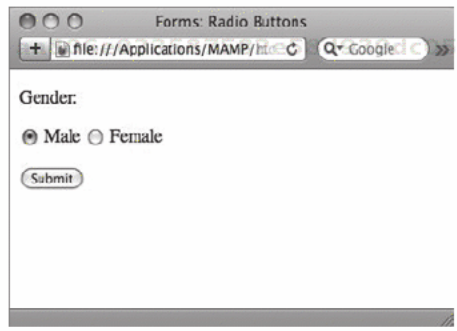

FIGURE 19-4

Common practice is to place the label to the right of the radio button, as shown in Figure 19-4. How you group radio buttons is determined by the design and the number of options in a group.

It is possible to preselect a radio button in a group by adding the attribute `checked`, like this:

```
<input type="radio" name="gender" id="female" value="female" checked="checked" />
```

If you didn't want the radio button to be checked you would set the `checked` attribute to an empty string, that is, `checked=""`, or remove the attribute entirely.

OFFERING CHECKBOX OPTIONS

Unlike radio buttons, checkbox form controls allow the user to select as many options as desired, not just one. Although checkboxes often appear near each other, they are not grouped by the name or other attribute.

```
<input type="checkbox" name="redCheckbox" id="redCheckbox" value="red" />
<label for="redCheckbox">Red</label>
<input type="checkbox" name="greenCheckbox" id="greenCheckbox" value="green" />
<label for="greenCheckbox">Green</label>
  <input type="checkbox" name="blueCheckbox" id="blueCheckbox" value="blue" />
  <label for="blueCheckbox">Blue</label>
```

Again, the `value` attribute contains the information to be transmitted if the checkbox is selected. Like radio buttons, the label is typically placed after the checkbox (Figure 19-5).

Preselecting checkboxes is handled the same way as radio buttons, by using the `checked` attribute. Say you wanted to have the Green option already checked when the user first sees the page. Here's how the `<input>` tag for a selected checkbox would be coded:

```
<input type="checkbox" name="greenCheckbox"
id="greenCheckbox" value="green"
  checked="checked" />
```

Obviously, unlike radio buttons, you can have as many checkboxes checked as necessary.

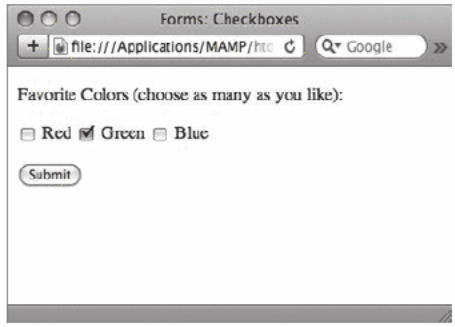

FIGURE 19-5

IMPLEMENTING SELECT LISTS

Select lists — also known as drop-down lists — provide another way for users to make selections. Select lists are extremely flexible and can be set up to emulate either radio buttons (with a single mutually exclusive choice) or checkboxes (with multiple selections).

To code a select list, you'll need two separate tags, similar to ordered and unordered lists. The outer tag is the `<select>` tag, which contains the `name` attribute and, optionally, an `id` attribute. Each item in a select list form control is coded with an `<option>` tag. The text in between the opening and closing `<option>` tag pair is what is displayed in the drop-down list. When a user chooses a particular select list item, the content of the `value` attribute is conveyed as the choice for the select list.

Take a look at some example code:

```
<select name="region" id="region">
  <option value="ne" selected="selected">Northeast</option>
  <option value="se">Southest</option>
  <option value="mw">Midwest</option>
  <option value="sw">Southwest</option>
  <option value="w">West</option>
</select>
```


When this select list is clicked by the user, the list drops down to display the options as shown in Figure 19-6. The first option, Northeast, is visible in the list when the list is closed.

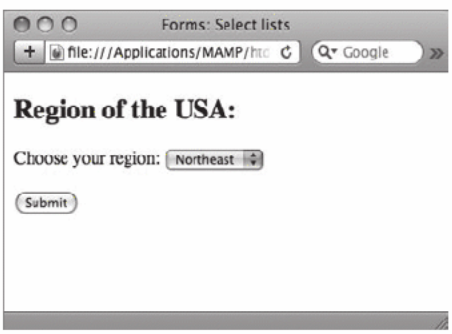

FIGURE 19-6

By default, the select list form control acts like a radio button in that it allows one mutually exclusive choice from many. To change the behavior to be like checkboxes, you add the `multiple` attribute to the `<select>` tag, like this:

```
<select name="region" id="region" multiple="multiple" size="5">
```

When you add the `multiple` attribute, the select list transforms from a drop-down list to a fully visible menu of selections as shown in Figure 19-7. To make multiple selections, the user must press Ctrl on the PC, Command on the Mac, or — for contiguous selections — Shift on either platform.

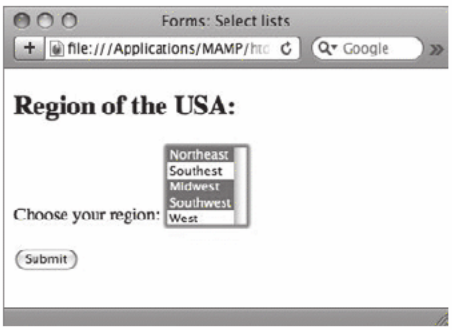

FIGURE 19-7

The `size` attribute determines how large the visible menu should be. If you choose less than the number of entries, a scroll bar appears so the user can select their option(s) from the entire list.

TRY IT

In this Try It you learn how to add a select list to an online form.

Lesson Requirements

You will need the `tpa_saturn.html` file from the previous exercise, as well as a text editor and web browser.

Step-by-Step

1. Open your text editor.
2. From the Lesson_19 folder, open the previously saved `tpa_saturn.html`.
3. Put your cursor at the end of the opening `</p>` tag after the closing `</textarea>` tag and press Enter (Return).
4. Enter the following code:

```
<p>
  <label for="age">Age</label>
  <select name="age" id="age">
    <option value="Under_12">Under 12 not allowed</option>
    <option value="12_18" selected>12 - 18</option>
```

```
    <option value="19_25">19 - 25</option>
    <option value="26_40">26 - 40</option>
    <option value="40 - 60">40 - 60</option>
    <option value="61_100">61 - 100</option>
    <option value="Over_100">Over 100</option>
  </select>
  </p>
```

5. Press Enter (Return) to create a new line and enter the following code:

```
<p>What other planets have you visited?<br>
<label>
<input type="checkbox" name="planets" value="venus" id="planets_0">
  Venus</label>
<label>
  <input type="checkbox" name="planets" value="mars" id="planets_1">
  Mars</label>
<label>
  <input type="checkbox" name="planets" value="jupiter" id="planets_2">
  Jupiter</label>
<label>
  <input type="checkbox" name="planets" value="neptune" id="planets_3">
  Neptune</label>
</p>
```

6. Save your file.

7. In your browser, open `tpa_saturn.html` to view the select list, as shown in Figure 19-8.

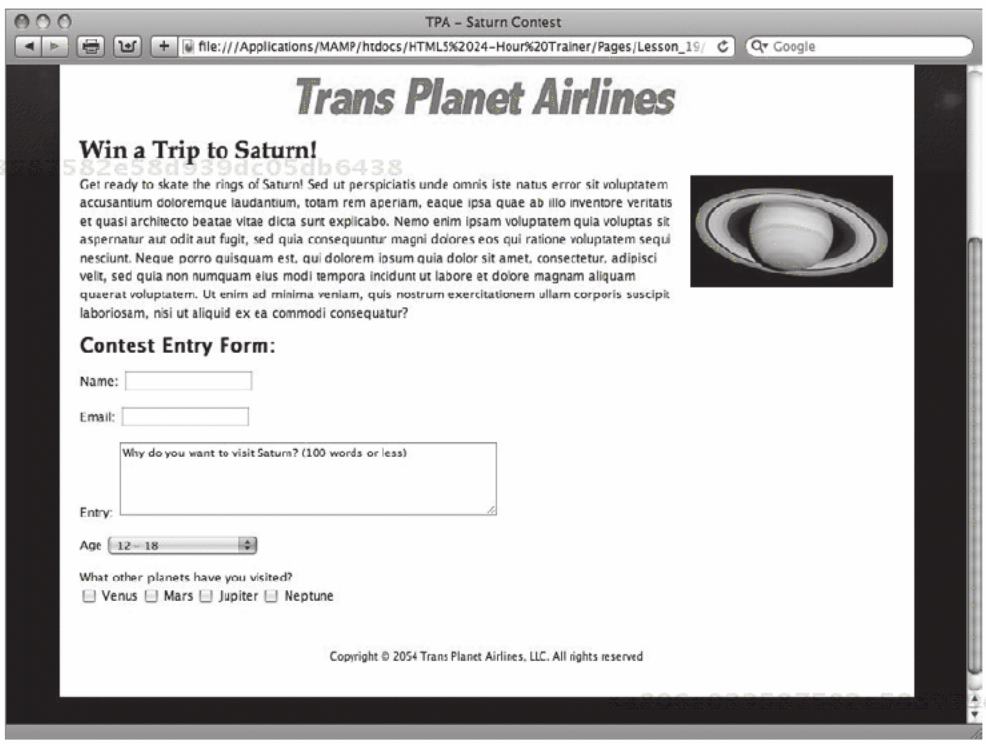

FIGURE 19-8

USING HIDDEN FORM CONTROLS

As you might suspect from the name, a hidden form control is not visible to the user. If forms are used to gain feedback from the user, what's the point of a hidden field? Quite often website owners work best when they know the context of the information supplied by the users. Say a site has two different forms on different pages, each of which asks for comments on the site's services. One form is for the general public, and the other is for current customers who are logged in. Both forms store their information in the same database. How can the data from the two groups be distinguished? By using a hidden form control, of course.

The hidden form control is another `<input>` tag type, which is coded like this:

```
<input type="hidden" name="Customer_Type" value="General Public" />
```


Because this form control is not displayed, there is no need for a label and thus, no need for an `id` attribute. You can have as many hidden form controls in your form as needed. Moreover, as long as the `<input>` tag is within the `<form>` tag, it can be placed anywhere.

From a coder's perspective, I prefer to group all of my hidden form controls after the rest of the form elements, just before the closing `</form>` *tag.*

INSERTING FORM BUTTONS

As mentioned in the beginning of this lesson, one of the key elements of every form is some sort of trigger to submit the form and all the collected information. Most frequently, this trigger takes the form of a button form control.

You have two ways to create HTML form buttons. You can use the faithful standby the `<input>` tag, or you can use the `<button>` tag. With `<input>`, you choose the appropriate `type` attribute, either `submit`, `reset`, or `button`:

```
<input type="submit" name="submitButton" value="Submit your form" />
```

With the `<input>` style button, the `value` attribute defines the label for the button, which appears in the button itself, as shown in Figure 19-9. As you probably have guessed, the `submit` type triggers the form and initiates the process to deliver the data. The `reset` type clears all the entries in the form, setting it to its default state. Finally, the `button` value for the `type` attribute allows the button to act as a general trigger, usually to activate some JavaScript.

Unlike the `<input>` tag, the `<button>` tag is not an empty tag — in other words, you need opening and closing tags with content in between to use the `<button>` tag. A `type` attribute is also needed in a `<button>` tag: the same three available for the button-related `<input>` tag: `submit`, `reset`, and `button`. The label for the button is entered as its content, like this:

```
<button type="submit" name="submitButton">Submit your form</button>
```

When rendered in a modern browser, this code creates a button similar to the one created with the previous `<input>` example code. So what's the difference between the two approaches? With a `<button>` tag, you can add other HTML elements as content, including images. For example:

```
<button type="submit" name="submitButton">
  <img src="images/checkmark.png" width="16" height="16" />
  Submit your form
</button>
```

As you can see in Figure 19-10, this code cleanly integrates a checkmark with the button text. More changes can be applied via CSS.

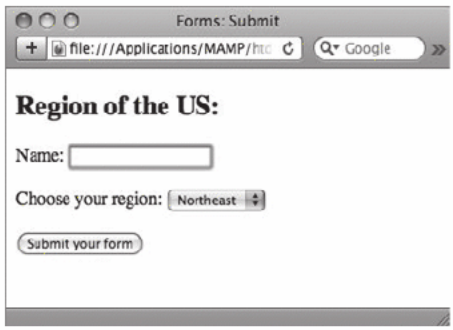

FIGURE 19-9

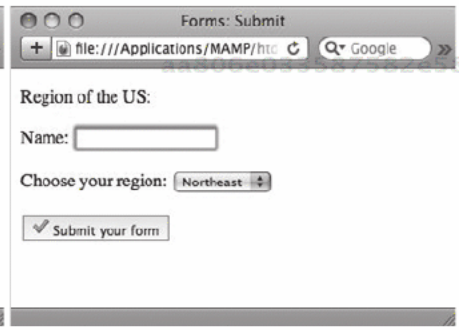

FIGURE 19-10

The one downside of using the `<button>` tag is that it is not supported in older versions of Internet Explorer, specifically IE 6 and below.

If your images don't align with the text in the `<button>` tag, adjust the graphics' position with the CSS `vertical-align` property. Quite often, setting that property to `middle` does the trick.

TRY IT

In this Try It you learn how to add buttons to your form.

Lesson Requirements

You will need the `tpa_saturn.html` file from the previous exercise, as well as a text editor and web browser.

Step-by-Step

1. Open your text editor.
2. From the Lesson_19 folder, open `tpa_saturn.html` saved in the previous exercise.

3. Put your cursor before the closing `</style>` tag within the `<head>` section and press Enter (Return).

4. Enter the following code:

```
button img {
  vertical-align: middle;
}
```

5. Put your cursor at the end of the closing `</p>` tag after the final checkbox and press Enter (Return).

6. Enter the following code:

```
<p>
    <button type="submit" name="submitButton">
    <img src="images/tick.png" width="16" height="16" /> Submit your entry
    </button>
    <button type="reset" name="resetButton">
    <img src="images/arrow_refresh.png" width="16" height="16" /> Start over
    </button>
  </p>
```

7. Save your file.

8. In your browser, open `tpa_jupiter.html` to view the rendered table with the new buttons as shown in Figure 19-11.

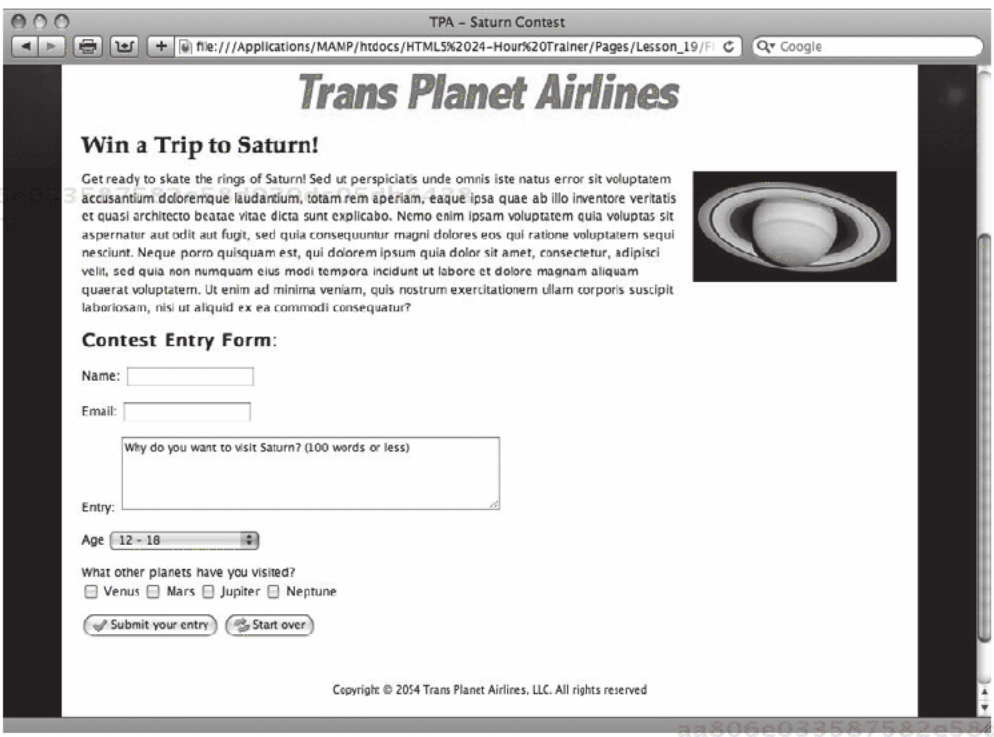

FIGURE 19-11

Please select a video from Lesson 19 on the DVD with the print book, or watch online at `www.wrox.com/go/html5video` *to see an example of the following:*

- Adding text and textarea fields
- Inserting radio buttons, checkboxes, and select lists
- Including form buttons