20

Enhancing Forms

Filling out forms on the Web can be a trying experience for the user. Unclear labels, sloppy layouts, and hard-to-follow designs can all add unnecessary roadblocks to getting the user's full cooperation when it is most needed. In this lesson, you learn how to add clarifying structural elements like fieldsets to a form as well as how to lay out your form with tables and with CSS. You also get a peek of CSS form enhancements set forth in the HTML5 specification.

APPLYING FIELDSETS AND LEGENDS

When working with larger forms with lots of labels and form controls, it can be helpful to group sections by using `<fieldset>` and `<legend>` tags. These tags are placed within a form and add a border around a designated set of fields (hence, a ***fieldset***). The `<legend>` tag, which goes within the `<fieldset>` tag, provides a title that identifies the group. Here's an example:


```
<form method="post" action="">
  <fieldset>
    <legend>Personal details</legend>
    <p>
      <label for="Name"> Name:</label>
      <input type="text" name="name" id="Name" />
    </p>
    <p>
      <label for="Email">Email:</label>
      <input type="text" name="email" id="Email" />
    </p>
    <p>
      <label for="Tel">Telephone:</label>
      <input type="text" name="tel" id="Tel" />
    </p>
  </fieldset>
  <p>
    <input type="submit" value="Submit" />
  </p>
</form>
```

As shown in Figure 20-1, the legend is, by default, displayed within the border surrounding the fieldset. You can, of course, use CSS to modify both the border and the legend text; designers might, for example, assign a background color to the fieldset selector to further distinguish the form control group.

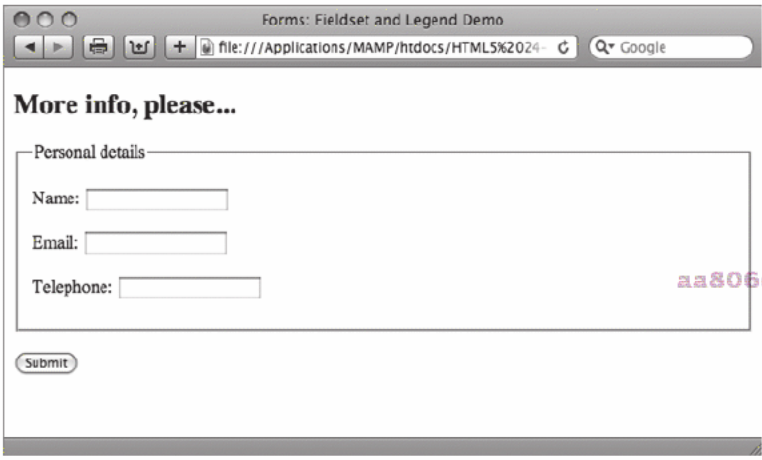


FIGURE 20-1

You can use as many fieldset/legend combinations as you would like in a form. For complex forms, they can certainly help guide the user to a successful form completion and submission.

TRY IT

In this Try It you learn how to add a fieldset and legend to a form.

Lesson Requirements

You will need the `tpa_saturn.html` file from the Lesson_20 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_20 folder, open `tpa_saturn.html`.
3. Put your cursor after the opening `<form>` tag and press Enter (Return).

4. Enter the following code:

```
<fieldset>
<legend>Your Info</legend>
```

5. Place your cursor after the closing `</p>` tag that follows the code `<input type="text" name="email" id="email">` and press Enter (Return).

6. Enter the following code:

```
</fieldset>
<fieldset>
<legend>Your entry</legend>
```

7. Place your cursor after the closing `</p>` tag that follows the final checkbox and press Enter (Return).

8. Enter the following code:

```
</fieldset>
```

9. Save your file.

10. In your browser, open `tpa_saturn.html` to view the rendered form with the fieldsets and legends as shown in Figure 20-2.

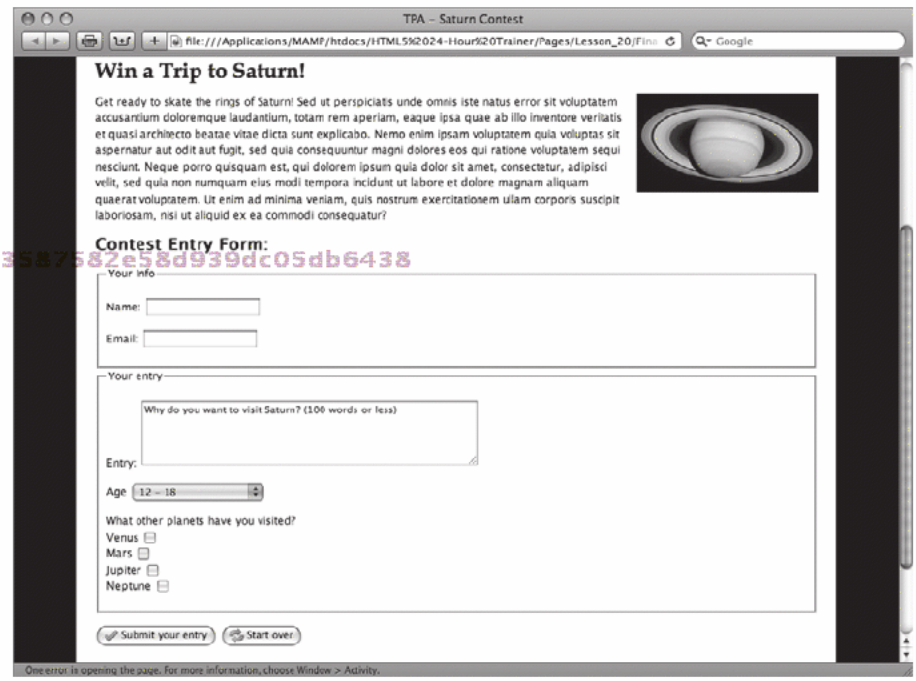


FIGURE 20-2

USING TABLES FOR FORM LAYOUT

For many years, web designers relied on the natural fit between forms and tables. A common layout placed labels in one column of a table and their associated form controls in the next. Frequently, the labels were right-aligned so that their connection to the adjacent controls were obvious. It's a very tried-and-true technique, and one that works well even today.

The basic table structure is to provide a row for each label/form control pair and a final row for the submit button. The key code aspect to remember is to place the entire table (or tables) within the form, like this:

```
<form method="post" action="">
<table>
  <tr>
    <td><label for="Name"> Name:</label></td>
    <td><input type="text" name="name" id="Name" /></td>
  </tr>
  <tr>
    <td><label for="Email">Email:</label></td>
    <td><input type="text" name="email" id="Email" /></td>
  </tr>
  <tr>
    <td><label for="Tel">Telephone:</label></td>
    <td><input type="text" name="tel" id="Tel" /></td>
  </tr>
  <tr>
    <td> </td>
    <td><input type="submit" value="Submit" /></td>
  </tr>
</table>
</form>
```


This approach, even without CSS styling, offers a very neat form appearance, as shown in Figure 20-3. If desired, you can add additional rows for a caption, summary, and details. Should you want to integrate a fieldset and legend, it is recommended that multiple tables be used.

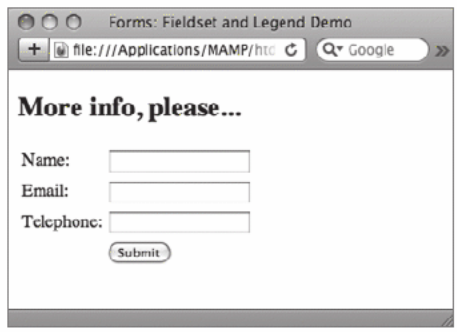


FIGURE 20-3

Right-aligning the label text is a two-stage process. First, you need to declare a custom CSS rule:

```
.labelText {
  text-align: right;
  padding-right: 3px;
}
```

Next, you need to apply the `.labelText` class to the `<td>` tag for each of the cells that contain a `<label>` tag. Here's the code with the appropriate classes applied:

```
<form method="post" action="">
<table>
  <tr>
    <td class="labelText"><label for="Name"> Name:</label></td>
```

```
    <td><input type="text" name="name" id="Name" /></td>
  </tr>
  <tr>
    <td class="labelText"><label for="Email">Email:</label></td>
    <td><input type="text" name="email" id="Email" /></td>
  </tr>
  <tr>
    <td class="labelText"><label for="Tel">Telephone:</label></td>
    <td><input type="text" name="tel" id="Tel" /></td>
  </tr>
  <tr>
    <td> </td>
    <td><input type="submit" value="Submit" /></td>
  </tr>
</table>
</form>
```


When viewed in a browser (Figure 20-4), the labels move closer to their associated form fields, making it easy for users to follow the form at glance.

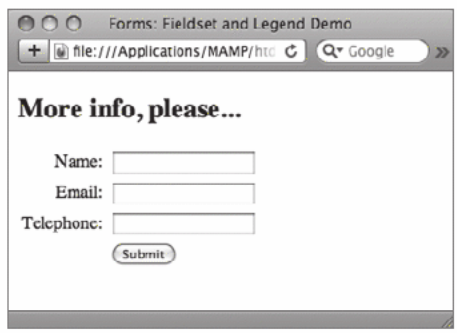


FIGURE 20-4

STYLING FORMS WITH CSS

Although tables offer a very straightforward layout option for forms, many web designers prefer a pure CSS approach. In addition to opening up a more colorful world of design possibilities, the two-column, right-aligned label look-and-feel can easily be replicated with just a few CSS rules.

Creating a Two-Column Layout

The key to recreating a two-column form layout in CSS is separating the `<label>` tag from the form control by using the `for` attribute as described in Lesson 19. Here's an example to refresh your memory:

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

Because the `<label>` tag is not wrapped around the form control, you can declare a CSS rule for the label selector that floats it to the left — and then align the text to the right within that floated width. For example:

```
label {
  width:100px;
  float:left;
  margin-right:10px;
  text-align:right;
  clear:left;
}
```

The `width` attribute ensures that all the labels will have the same distance to work with. A constant `margin-right` attribute keeps the form controls the same number of pixels (in this case) to the right. And, as shown in Figure 20-5, the `text-align` property works just as well here as in the table cells. Finally, the `clear:left` declaration stops the `float` property from extending to the next line.

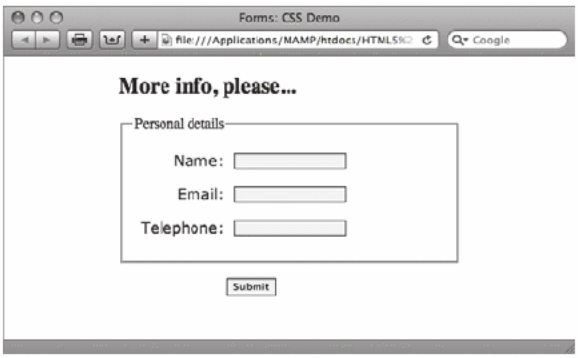


FIGURE 20-5

Styling Fieldsets and Legends

If your form includes one or more `<fieldset>` and `<legend>` tags, they provide very handy hooks on which to hang some distinctive CSS. You can easily add a background color and border to make both stand out, as well as padding and margins to keep the form controls easy to read. As shown in Figure 20-6, the following CSS rules give the fieldset selector a light-orange background, complete with rounded corners in modern browsers (note, the color is not visible in this grayscale figure):

```
fieldset {
  margin: 0;
  padding: .5em;
  background: #FF9900;
  border: 1px solid #000000;
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
}
legend {
  padding: .2em;
  background-color: #EBEBFF;
  font-weight: bold;
  color: #000000;
  border: 1px solid #000000;
}
```

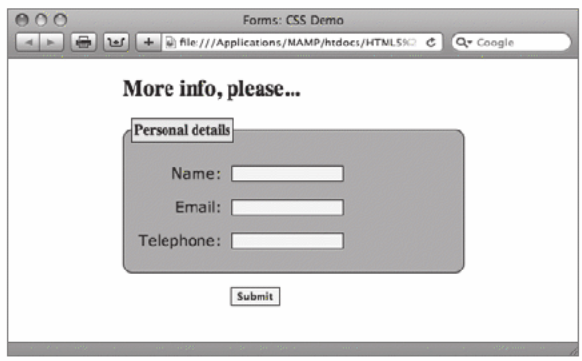



FIGURE 20-6

You may be wondering about the two somewhat odd looking properties in the `fieldset` *rule,* `-webkit-border-radius` *and* `-moz-border-radius`*. These properties were implemented by Safari and Mozilla (Firefox) to bring a rounded corner option to their browsers while the CSS 3 specification — which includes* `border-radius` *— is still in the formation phase. At this point in web design, it's best to include all three declarations for backward and forward compatibility.*

Working with Input Fields

Way back in Lesson 8, when discussing link styles, the `:focus` link state was mentioned. Although it can be used for text links, this particular state really comes into play with form controls. Whenever a user selects or clicks into a particular form control, such as a text field, that control is said to have *focus* and, thus, be in the `:focus` state. You can use this distinction to give your form controls two different styles: one when the field is selected and one when it is not. For example, if you want to change the text and background colors when a user clicks into a text field, here are two CSS rules you might use:


```
input {
  border: 1px solid #000000;
  font-weight: bold;
  background-color: #F5F5F5;
}
input:focus {
  font-weight: bold;
  color: #FFF;
  background-color: #0F0
}
```


The different (green) `background-color` and white text values defined in the `input:focus` rule should be readily apparent in Figure 20-7, even in grayscale.

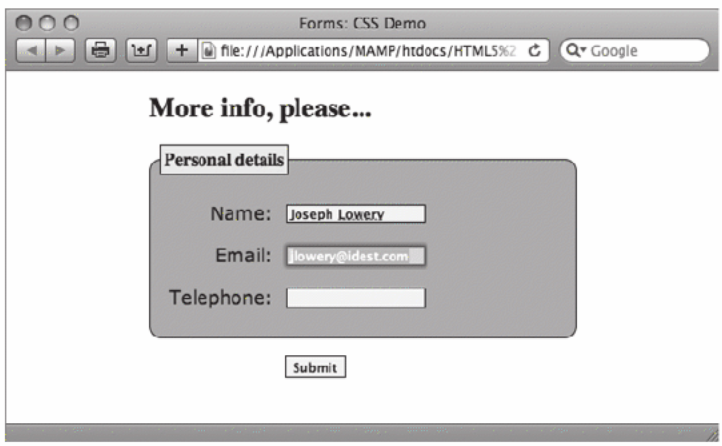


FIGURE 20-7

You'll recall the `input` selector affects many different types of form controls from text fields to checkboxes. You need to be careful when you create a CSS rule that targets all `<input>` tags that you don't inadvertently affect a particular form control. To avoid this problem, you can specify the type of form control with a more particular CSS selector with the attribute selector. Here's an example CSS rule intended to modify the submit button:

```
input[type="submit"] {
  margin:0 0 0 120px;
}
```

The square brackets indicate an attribute selector that targets an attribute in the tag, here `type="submit"`. You can easily create selectors for checkboxes and radio buttons using similar selectors.

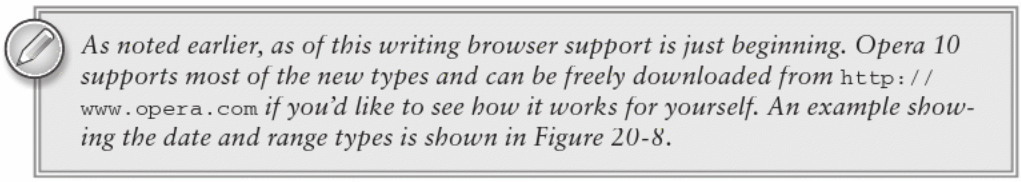
UNDERSTANDING ADDITIONAL HTML5 FORM ENHANCEMENTS

One of the major areas addressed in HTML5 is forms. In addition to the `required`, `autocomplete`, `autofocus`, and other attributes covered in Lesson 19, many — 13, in fact — new types have been added to the `<input>` tag. Though there is not full cross-browser compatibility for these new types yet, support is included in many of the latest browser versions with more on the way.

Perhaps best of all, all of these new `type` attributes degrade gracefully because the default `type` value is `text`. In other words, if a browser does not recognize the new `url` type, it handles it as if it were text. Here's a quick overview of the newly available types:

- `color`: Displays a color picker. Unfortunately, as of this writing, no browser has implemented the `color` type.
- `date`: Displays a calendar and adds the selected date in the field as a text string.
- `datetime`: Displays a calendar as well as a time field with up and down arrows.
- `datetime-local`: Displays a calendar as well as a time field with up and down arrows without a time zone.
- `time`: Displays a time field with up and down arrows.
- `week`: Displays a calendar and, when a date is selected, inserts the number of the week (1 to 52) as well as the year.
- `month`: Displays a calendar and, when a date is selected, inserts the number of the month (1 to 12) as well as the year.
- `number`: Displays a stepper control (up and down arrows). Available attributes include `min`, `max`, `step`, and `value`.
- `range`: Displays a slider control. Available attributes include `min`, `max`, `step`, and `value`.
- `email`: Validates the entered value as an e-mail address.
- `search`: Includes a clear search icon.

- `tel`: Validates the entered value as a telephone number.
- `url`: Validates the entered value as a web address.



As noted earlier, as of this writing browser support is just beginning. Opera 10 supports most of the new types and can be freely downloaded from `http://www.opera.com` *if you'd like to see how it works for yourself. An example showing the date and range types is shown in Figure 20-8.*

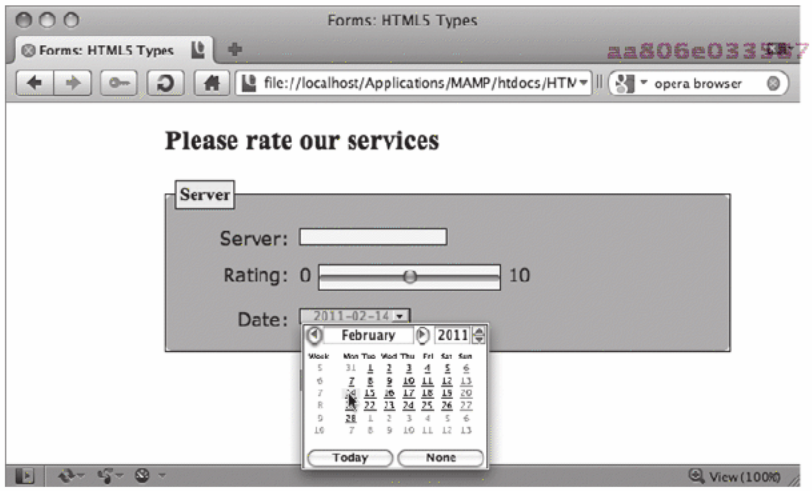


FIGURE 20-8

TRY IT

In this Try It you learn how to style a form with CSS.

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_20 folder, open the previously saved `tpa_saturn.html`.

3. Put your cursor before the closing `</style>` tag in the `<head>` section and press Enter (Return).
4. Enter the following code:

```
input, textarea, select {
  border: 1px solid #000000;
  margin-top: -5px;
}
input:focus {
  font-weight: bold;
  color: #F00;
}
label {
  width:100px;
  float:left;
  margin-right:10px;
  text-align:right;
  clear:left;
}
input[type="checkbox"] {
  margin:2px 0 0 0;
}
fieldset {
  background: #F8B9BC;
  margin-bottom: 15px;
  -webkit-border-radius: 8px;
  -moz-border-radius: 8px;
  border-radius: 8px;
}
legend {
  background: #FFF;
  border: 1px solid #F70816;
  padding: 5px;
  font-weight: bold;
  -webkit-border-radius: 8px;
  -moz-border-radius: 8px;
  border-radius: 8px;
}
```


5. Save your file.
6. In your browser, open `tpa_saturn.html` to view the rendered form with the new styling, as shown in Figure 20-9.

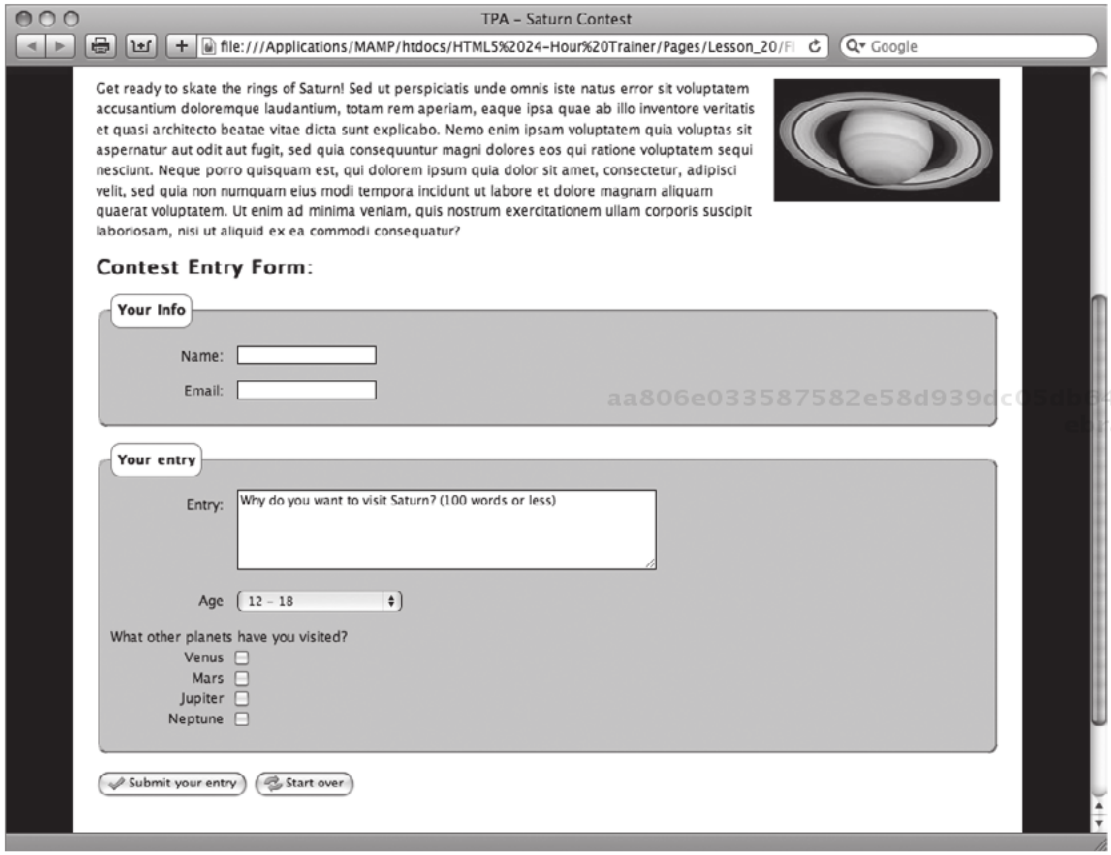


FIGURE 20-9

Please select a video from Lesson 20 on the DVD with the print book, or watch online at `www.wrox.com/go/html5video` *to see an example of the following:*

- Adding a fieldset and legend
- Styling a form with CSS

SECTION VIII
Enhancing HTML with JavaScript