4.4.4 specialized Controls

There are two important additional special-purpose form controls that are available in all browsers. The first of these is the <input type="hidden"> element, which will be covered in more detail in Chapter 13 on State Management. The other specialized form control is the <input type="file"> element, which is used to upload a file from the client to the server. The usage and user interface for this control are shown in Figure 4.24. The precise look for this control can vary from browser to browser, and platform to platform.

Notice that the <form> element must use the post method and that it must include the enctype="multipart/form-data" attribute as well. As we have seen in the section on query strings, form data is URL encoded (i.e., enctype="application/ x-www-form-urlencoded"). However, files cannot be transferred to the server using normal URL encoding, hence the need for the alternative enctype attribute.

Number and range

HTML5 introduced two new controls for the input of numeric values. When input via a standard text control, numbers typically require validation to ensure that the user has entered an actual number and, because the range of numbers is infinite, the entered number has to be checked to ensure it is not too small or too large.

The number and range controls provide a way to input numeric values that eliminate the need for client-side numeric validation (for security reasons you would still check the numbers for validity on the server). Figure 4.25 illustrates the usage and appearance of these numeric controls.

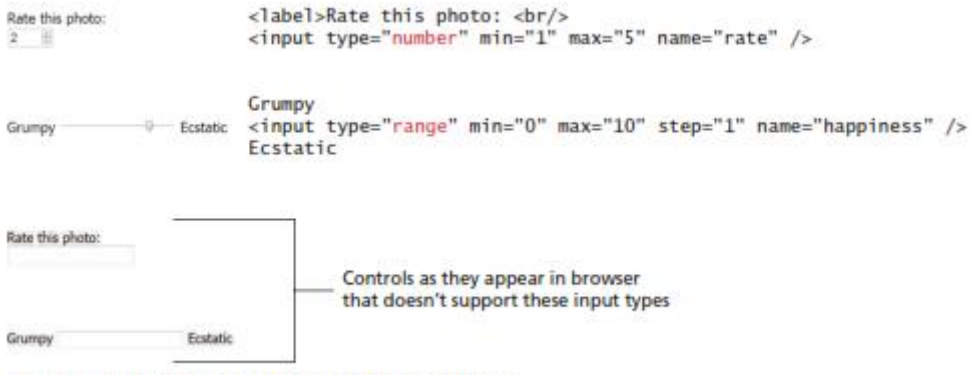

FIGURE 4.25 Number and range input controls

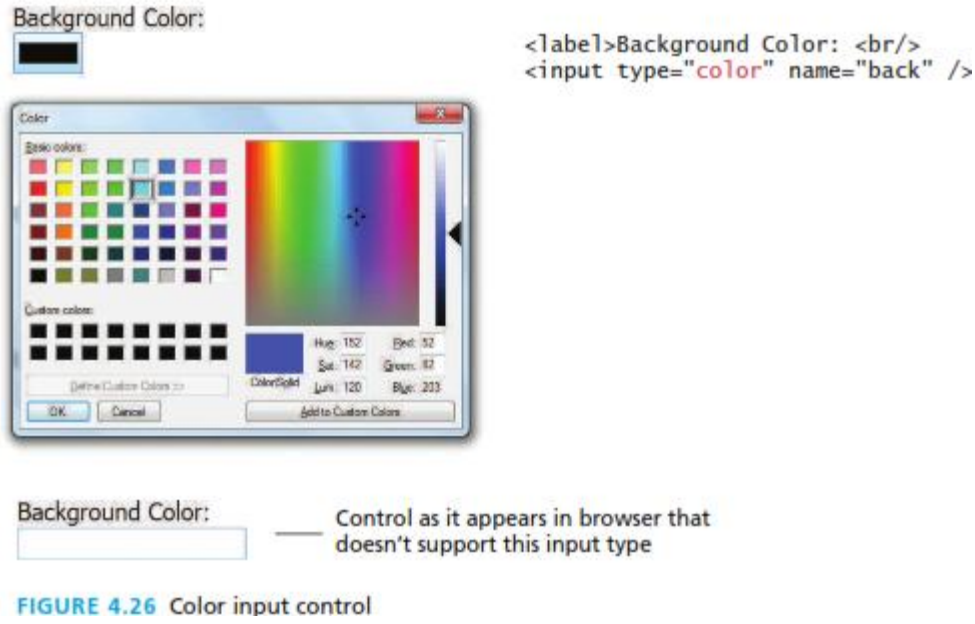

FIGURE 4.26 Color input control

Color

Not every web page needs the ability to get color data from the user, but when it is necessary, the HTML5 color control provides a convenient interface for the user, as shown in Figure 4.26. At the time of writing, only the latest versions of Chrome and Opera support this control.

4.4.5 Date and time Controls

Asking the user to enter a date or time is a relatively common web development task. Like with numbers, dates and times often need validation when gathering this information from a regular text input control. From a user's perspective, entering dates can be tricky as well: you probably have wondered at some point in time when entering a date into a web form, what format to enter it in, whether the day comes before the month, whether the month should be entered as an abbreviation or a number, and so on. The new date and time controls in HTML try to make it easier for users to input these tricky date and time values.

Table 4.5 lists the various HTML5 date and time controls. Their usage and appearance in the browser are shown in Figure 4.27.

Type	Description
date	Creates a general date input control. The format for the date is "yyyy-mm-dd."
time	Creates a time input control. The format for the time is "HH:MM:SS," for hours:minutes:seconds.
datetime	Creates a control in which the user can enter a date and time.
datetime-local	Creates a control in which the user can enter a date and time without specifying a time zone.
month	Creates a control in which the user can enter a month in a year. The format is "yyyy-mm."
week	Creates a control in which the user can specify a week in a year. The format is "yyyy-W##."

TABLE 4.5 HTML5 Date and Time Controls

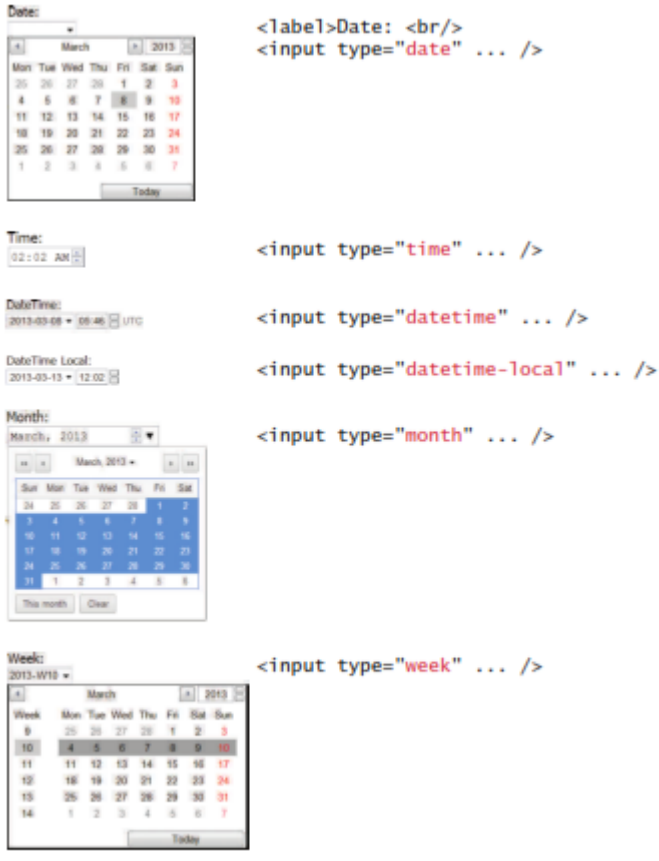

FIGURE 4.27 Date and time controls

4.5 table and Form accessibility

In Chapter 2, you were reminded that not all web users are able to view the content on web pages in the same manner. Users with sight disabilities, for instance, experi- ence the web using voice reading software. Color blind users might have trouble differentiating certain colors in proximity; users with muscle control problems may have difficulty using a mouse, while older users may have trouble with small text and image sizes. The term **web accessibility** refers to the assistive technologies, various features of HTML that work with those technologies, and different coding and design practices that can make a site more usable for people with visual, mobil- ity, auditory, and cognitive disabilities.

In order to improve the accessibility of websites, the W3C created the **Web Accessibility Initiative (WAI)** in 1997. The WAI produces guidelines and recom- mendations, as well as organizing different working groups on different accessibility issues. One of its most helpful documents is the Web Content Accessibility Guidelines, which is available at **http://www.w3.org/WAI/intro/wcag.php**.
Perhaps the most important guidelines in that document are:

- *Provide text alternatives for any nontext content so that it can be changed into other forms people need, such as large print, braille, speech, symbols, or simpler language*.
- *Create content that can be presented in different ways (for example simpler layout) without losing information or structure*.
  - *Make all functionality available from a keyboard*.
- *Provide ways to help users navigate, find content, and determine where they are*.

The guidelines provide detailed recommendations on how to achieve this advice. This section will look at how one can improve the accessibility of tables and forms, two HTML structures that are often plagued by a variety of accessibility issues.

4.5.1 accessible tables

HTML tables can be quite frustrating from an accessibility standpoint. Users who rely on visual readers can find pages with many tables especially difficult to use. One vital way to improve the situation is to only use tables for tabular data, not for layout. Using the following accessibility features for tables in HTML can also improve the experience for those users:

1. Describe the table's content using the <caption> element (see Figure 4.6).
This provides the user with the ability to discover what the table is about before having to listen to the content of each and every cell in the table. If you have an especially long description for the table, consider putting the table within a <figure> element and use the <figcaption> element to provide the description.

2. Connect the cells with a textual description in the header. While it is easy for a sighted user to quickly see what row or column a given data cell is in, for users relying on visual readers, this is not an easy task.

It is quite revealing to listen to reader software recite the contents of a table that has not made these connections. It sounds like this: "row 3, cell 4: 45.56; row 3, cell 5: Canada; row 3, cell 6: 25,000; etc." However, if these connections have been made, it sounds instead like this: "row 3, Average: 45.56; row 3, Country: Canada; row 3, City Count: 25,000; etc.," which is a significant improvement.

Listing 4. 1 illustrates how to use the scope attribute to connect cells with their headers.

```
<table>
<caption>Famous Paintings</caption>
<tr>
<th scope="col">Title</th>
<th scope="col">Artist</th>
<th scope="col">Year</th>
<th scope="col">Width</th>
<th scope="col">Height</th>
</tr>
<tr>
<td>The Death of Marat</td>
<td>Jacques-Louis David</td>
<td>1793</td>
<td>162cm</td>
<td>128cm</td>
</tr>
<tr>
<td>Burial at ornans</td>
```

(*continued*)

```
<td>Gustave Courbet</td>
<td>1849</td>
<td>314cm</td>
<td>663cm</td>
</tr>
</table>
```

Listing 4.1 Connecting cells with headers

4.5.2 accessible Forms

HTML forms are also potentially problematic from an accessibility standpoint. If you remember the advice from the WAI about providing keyboard alternatives and text alternatives, your forms should be much less of a problem.

The forms in this chapter already made use of the <fieldset>, <legend>, and <label> elements, which provide a connection between the input elements in the form and their actual meaning. In other words, these controls add semantic content to the form.

While the browser does provide some unique formatting to the <fieldset> and <legend> elements, their main purpose is to logically group related form input elements together with the <legend> providing a type of caption for those elements. You can of course use CSS to style (or even remove the default styling) these elements.

The <label> element has no special formatting (though we can use CSS to do so). Each <label> element should be associated with a single input element. You can make this association explicit by using the for attribute, as shown in Figure 4.28. Doing so means that if the user clicks on or taps the <label> text, that control will

```
<label for="f-title">Title: </label>

<input type="text" name="title" id="f-title"/>


<label for="f-country">Country: </label>

<select name="where" id="f-country">
<option>Choose a country</option>
                                    <option>Canada</option>
                                    <option>Finland</option>
<option>United States</option>
                                                </select>
```

Figure 4.28 Associating labels and input elements