

3. Introduction to HTML5: Part 2

Form ever follows function.

—Louis Sullivan

I listen and give input only if somebody asks.

—Barbara Bush

Objectives

In this chapter you'll:

- Build a form using the new HTML5 `input` types.
- Specify an `input` element in a form as the one that should receive the focus by default.
- Use self-validating `input` elements.
- Specify temporary placeholder text in various `input` elements
- Use autocomplete `input` elements that help users reenter text that they've previously entered in a form.
- Use a `datalist` to specify a list of values that can be entered in an `input` element and to autocomplete entries as the user types.
- Use HTML5's new page-structure elements to delineate parts of a page, including headers, sections, figures, articles, footers and more.

Outline

3.1 Introduction

3.2 New HTML5 Form `input` Types

[3.2.1 input Type color](#)

[3.2.2 input Type date](#)

[3.2.3 input Type datetime](#)

[3.2.4 input Type datetime-local](#)

[3.2.5 input Type email](#)

[3.2.6 input Type month](#)

[3.2.7 input Type number](#)

[3.2.8 input Type range](#)

[3.2.9 input Type search](#)

[3.2.10 input Type tel](#)

[3.2.11 input Type time](#)

[3.2.12 input Type url](#)

[3.2.13 input Type week](#)

[3.3 input and datalist Elements and autocomplete Attribute](#)

[3.3.1 input Element autocomplete Attribute](#)

[3.3.2 datalist Element](#)

[3.4 Page-Structure Elements](#)

[3.4.1 header Element](#)

[3.4.2 nav Element](#)

[3.4.3 figure Element and figcaption Element](#)

[3.4.4 article Element](#)

[3.4.5 summary Element and details Element](#)

[3.4.6 section Element](#)

[3.4.7 aside Element](#)

[3.4.8 meter Element](#)

[3.4.9 footer Element](#)

[3.4.10 Text-Level Semantics: mark Element and wbr Element](#)

[Summary](#) | [Self-Review Exercises](#) | [Answers to Self-Review Exercises](#) | [Exercises](#)

3.1. Introduction

We now continue our presentation of HTML5 by discussing various new features, including:

- new `input` element types for colors, dates, times, e-mail addresses, numbers, ranges of integer values, telephone numbers, URLs, search queries, months and weeks—browsers that don't support these `input` types simply render them as standard text `input` elements
- autocompletion capabilities that help users quickly re-enter text that they've previously entered in a form
- `datalist`s for providing lists of allowed values that a user can enter in an `input` element and for autocompleting those values as the user types
- page-structure elements that enable you to delineate and give meaning to the parts of a page, such as headers, navigation areas, footers, sections, articles, asides, summaries/details, figures, figure captions and more

Support for the features presented in this chapter varies among browsers, so for our sample outputs we've used several browsers. We'll

discuss many more new HTML5 features throughout the remaining chapters.

3.2. New HTML5 Form input Types

[Figure 3.1](#) demonstrates HTML5's new form input types. These are not yet universally supported by all browsers. In this example, we provide sample outputs from a variety of browsers so that you can see how the input types behave in each.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 3.1: newforminputtypes.html -->
4  <!-- New HTML5 form input types and attributes. -->
5  <html>
6    <head>
7      <meta charset="utf-8">
8      <title>New HTML5 Input Types</title>
9    </head>
10
11   <body>
12     <h1>New HTML5 Input Types Demo</h1>
13     <p>This form demonstrates the new HTML5 input types
14       and the placeholder, required and autofocus attributes.
15     </p>
16
17     <form method = "post" action = "http://www.deitel.com">
18       <p>
19         <label>Color:
20           <input type = "color" autofocus />
21           (Hexadecimal code such as #ADD8E6)
22         </label>
23       </p>
24       <p>
25         <label>Date:
```

```
26      <input type = "date" />
27      (yyyy-mm-dd)
28  </label>
29 </p>
30 <p>
31     <label>Datetime:
32         <input type = "datetime" />
33         (yyyy-mm-ddThh:mm+ff:gg, such as 2012-01-27T03:15)
34     </label>
35 </p>
36 <p>
37     <label>Datetime-local:
38         <input type = "datetime-local" />
39         (yyyy-mm-ddThh:mm, such as 2012-01-27T03:15)
40     </label>
41 </p>
42 <p>
43     <label>Email:
44         <input type = "email" placeholder = "name@domain.com"
45             required /> (name@domain.com)
46     </label>
47 </p>
48 <p>
49     <label>Month:
50         <input type = "month" /> (yyyy-mm)
51     </label>
52 </p>
53 <p>
54     <label>Number:
55         <input type = "number"
56             min = "0"
57             max = "7"
58             step = "1"
59             value = "4" />
60     </label> (Enter a number between 0 and 7)
61 </p>
```

```
62     <p>
63         <label>Range:
64         0 <input type = "range"
65             min = "0"
66             max = "20"
67             value = "10" /> 20
68     </label>
69 </p>
70 <p>
71     <label>Search:
72         <input type = "search" placeholder = "search query" />
73     </label> (Enter your search query here.)
74 </p>
75 <p>
76     <label>Tel:
77         <input type = "tel" placeholder = "(###) ###-####"
78             pattern = "\\(\\d{3}\\) +\\d{3}-\\d{4}" required />
79             (###) ###-####
80     </label>
81 </p>
82 <p>
83     <label>Time:
84         <input type = "time" /> (hh:mm:ss.ff)
85     </label>
86 </p>
87 <p>
88     <label>URL:
89         <input type = "url"
90             placeholder = "http://www.domainname.com" />
91             (http://www.domainname.com)
92     </label>
93 </p>
94 <p>
95     <label>Week:
96         <input type = "week" />
97         (yyyy-Wnn, such as 2012-W01)
```

```
98      </label>
99  </p>
100  <p>
101      <input type = "submit" value = "Submit" />
102      <input type = "reset" value = "Clear" />
103  </p>
104  </form>
105 </body>
106 </html>
```

Fig. 3.1. New HTML5 form input types and attributes.

3.2.1. input **Type** color

The **color input type** ([Fig. 3.1](#), lines 20–21) enables the user to enter a color. At the time of this writing, most browsers render the **color input type** as a text field in which the user can enter a hexadecimal code or a color name. In the future, when you click a **color input**, browsers will likely display a *color picker* similar to the Microsoft Windows color dialog shown in [Fig. 3.2](#).

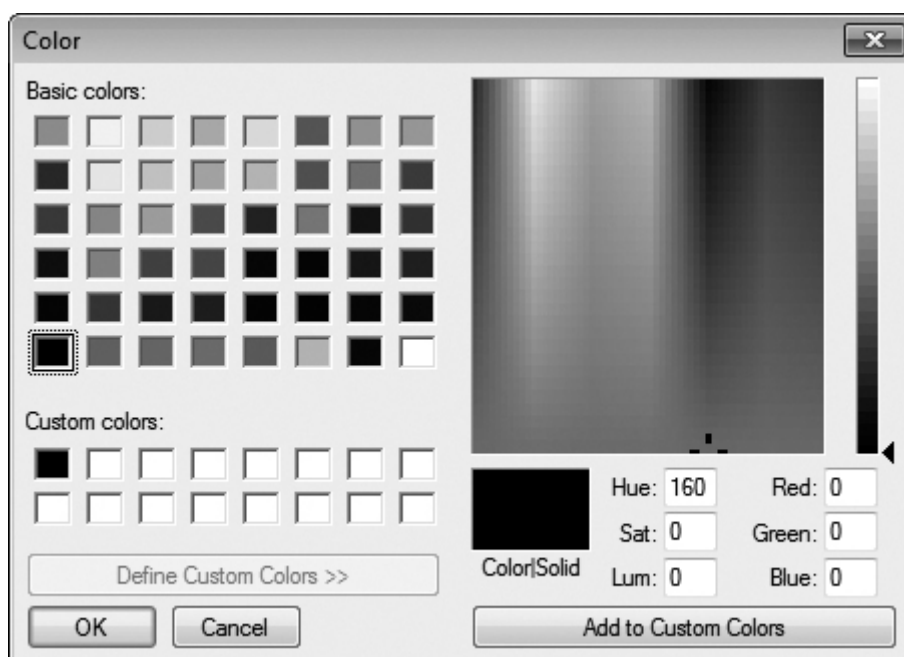


Fig. 3.2. A dialog for choosing colors.

autofocus **Attribute**

The **autofocus** attribute ([Fig. 3.1](#), line 20)—an optional attribute that can be used in only one input element on a form—automatically gives the focus to the input element, allowing the user to begin typing in that element immediately. [Figure 3.3](#) shows autofocus on the color element—the first input element in our form—as rendered in Chrome. You do not need to include autofocus in your forms.

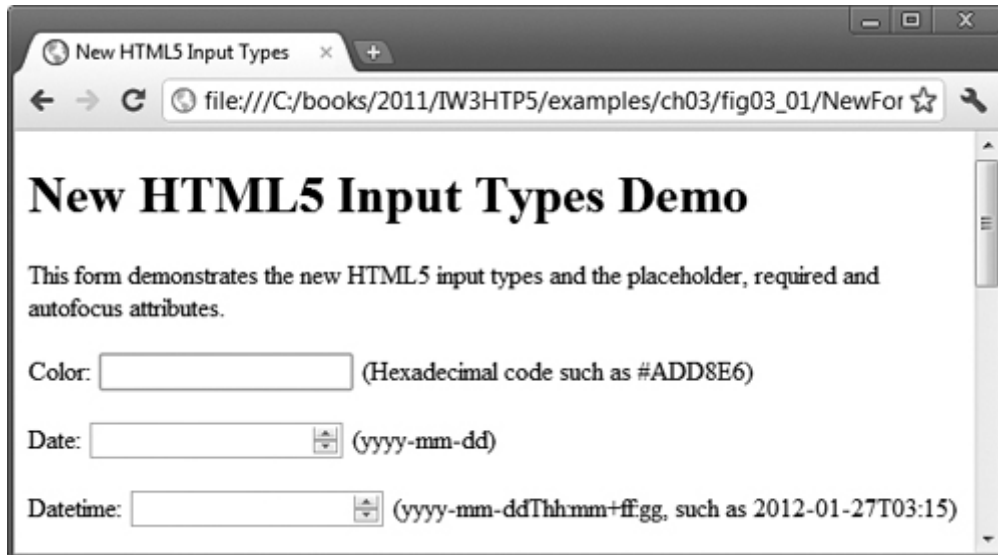


Fig. 3.3. Autofocus in the color input element using Chrome.

Validation

Traditionally it's been difficult to validate user input, such as ensuring that an e-mail address, URL, date or time is entered in the proper format. The new HTML 5 input types are *self validating* on the client side, eliminating the need to add complicated JavaScript code to your web pages to validate user input, reducing the amount of invalid data submitted and consequently reducing Internet traffic between the server and the client to correct invalid input. *The server should still validate all user input.*

When a user enters data into a form then submits the form (in this example, by clicking the **Submit** button), the browser immediately checks the self-validating elements to ensure that the data is correct. For example, if a user enters an incorrect hexadecimal color value when using a browser that renders the color elements as a text field (e.g., Chrome), a callout pointing to the element will appear, indicating that an invalid value was entered ([Fig. 3.4](#)). [Figure 3.5](#) lists each of the new HTML5 input types

and provides examples of the proper formats required for each type of data to be valid.

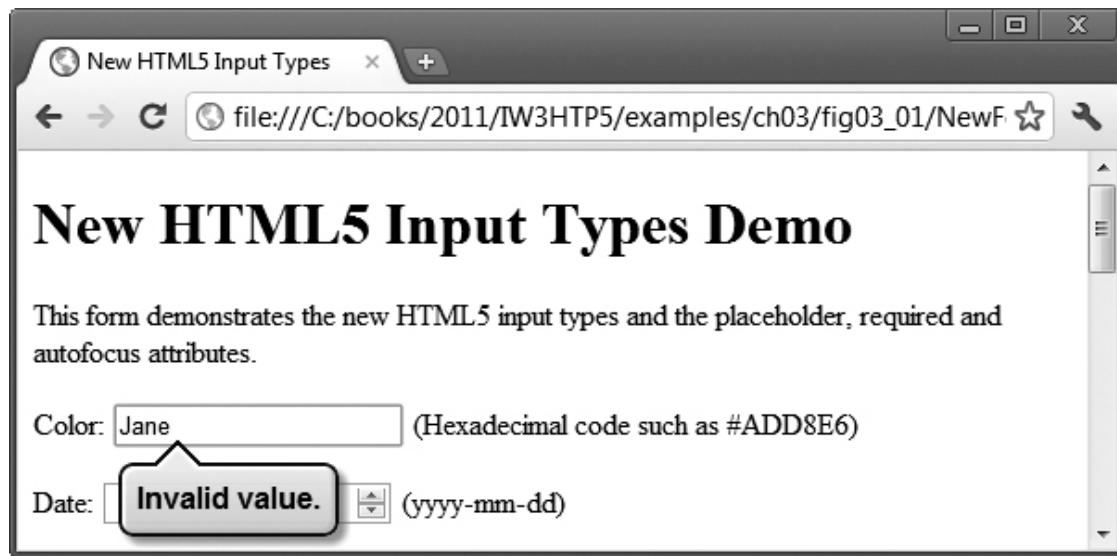


Fig. 3.4. Validating a color input in Chrome.

input type	Format
color	Hexadecimal code
date	yyyy-mm-dd
datetime	yyyy-mm-dd
datetime-local	yyyy-mm-ddThh:mm
month	yyyy-mm
number	Any numerical value
email	name@domain.com
url	http://www.domainname.com
time	hh:mm
week	yyyy-Wnn

Fig. 3.5. Self-validating input types.

If you want to bypass validation, you can add the **formnovalidate** attribute to input type submit in line 101:

```
<input type = "submit" value = "Submit" formnovalidate />
```

3.2.2. input Type date

The **date input type** (lines 26–27) enables the user to enter a date in the form yyyy-mm-dd. Firefox and Internet Explorer display a text field in

which a user can enter a date such as 2012-01-27. Chrome and Safari display a **spinner control**—a text field with an up-down arrow () on the right side—allowing the user to select a date by clicking the up or down arrow. The start date is the *current date*. Opera displays a calendar from which you can choose a date. In the future, when the user clicks a date input, browsers are likely to display a date control similar to the Microsoft Windows one shown in [Fig. 3.6](#).



Fig. 3.6. A date chooser control.

3.2.3. input Type datetime

The **datetime input type** (lines 32–33) enables the user to enter a date (year, month, day), time (hour, minute, second, fraction of a second) and the time zone set to UTC (Coordinated Universal Time or Universal Time, Coordinated). Currently, most of the browsers render **datetime** as a text field; Chrome renders an up-down control and Opera renders a date and time control. For more information on the **datetime** input type, visit:

www.w3.org/TR/html5/states-of-the-type-attribute.html#date-and-time-state

3.2.4. input Type datetime-local

The **datetime-local input type** (lines 38–39) enables the user to enter the date and time in a *single* control. The data is entered as year, month, day, hour, minute, second and fraction of a second. Internet Explorer,

Firefox and Safari all display a text field. Opera displays a date and time control. For more information on the `datetime-local` input type, visit:

www.w3.org/TR/html5/states-of-the-type-attribute.html#local-date-and-time-state

3.2.5. input Type email

The **email input type** (lines 44–45) enables the user to enter an e-mail address or a list of e-mail addresses separated by commas (if the `multiple` attribute is specified). Currently, all of the browsers display a text field. If the user enters an *invalid* e-mail address (i.e., the text entered is *not* in the proper format) and clicks the **Submit** button, a callout asking the user to enter an e-mail address is rendered pointing to the input element ([Fig. 3.7](#)). HTML5 does not check whether an e-mail address entered by the user actually exists—rather it just validates that the e-mail address is in the *proper format*.



Fig. 3.7. Validating an e-mail address in Chrome.

placeholder Attribute

The **placeholder attribute** (lines 44, 72 and 77) allows you to place temporary text in a text field. Generally, placeholder text is *light gray* and provides an example of the text and/or text format the user should enter ([Fig. 3.8](#)). When the *focus* is placed in the text field (i.e., the cursor is in the text field), the placeholder text disappears—it's not “submitted” when the user clicks the **Submit** button (unless the user types the same text).

a) Text field with gray placeholder text



b) placeholder text disappears when the text field gets the focus



Fig. 3.8. placeholder text disappears when the input element gets the focus.

HTML5 supports placeholder text for only six input types—text, search, url, tel, email and password. Because the user's browser might not support placeholder text, we've added descriptive text to the right of each input element.

required Attribute

The **required attribute** (lines 45 and 78) forces the user to enter a value before submitting the form. You can add **required** to any of the input types. In this example, the user *must* enter an e-mail address and a telephone number before being able to submit the form. For example, if the user fails to enter an e-mail address and clicks the **Submit** button, a call-out pointing to the empty element appears, asking the user to enter the information ([Fig. 3.9](#)).

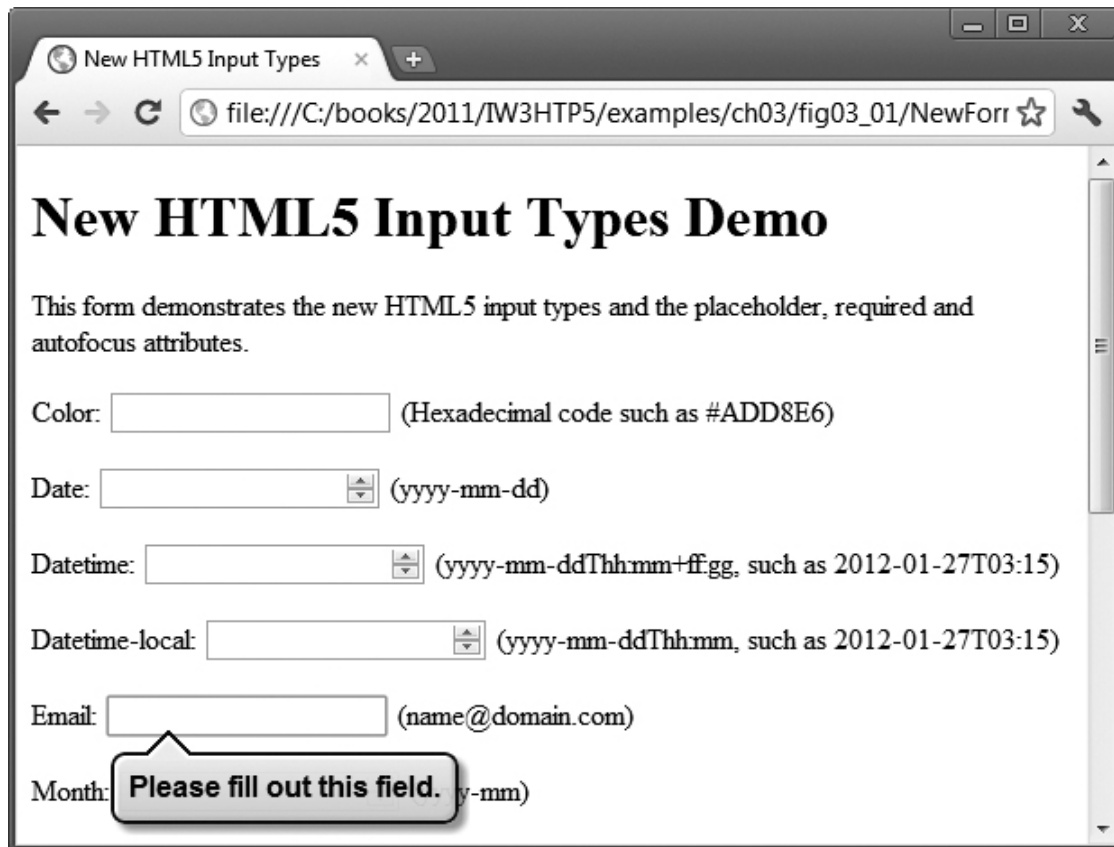


Fig. 3.9. Demonstrating the `required` attribute in Chrome.

3.2.6. `input` Type `month`

The **month input type** (line 50) enables the user to enter a year and month in the format `yyyy-mm`, such as `2012-01`. If the user enters the data in an improper format (e.g., `January 2012`) and submits the form, a callout stating that an invalid value was entered appears.

3.2.7. `input` Type `number`

The **number input type** (lines 55–59) enables the user to enter a numerical value—mobile browsers typically display a numeric keypad for this input type. Internet Explorer, Firefox and Safari display a text field in which the user can enter a number. Chrome and Opera render a spinner control for adjusting the number. The `min` attribute sets the minimum valid number, in this case `"0"`. The `max` attribute sets the maximum valid number, which we set to `"7"`. The `step` attribute determines the increment in which the numbers increase. For example, we set the `step` to `"1"`, so the number in the spinner control increases or decreases by one each time the up or down arrow, respectively, in the spinner control is clicked. If you change the `step` attribute to `"2"`, the number in the

spinner control will increase or decrease by two each time the up or down arrow, respectively, is clicked. The `value` attribute sets the initial value displayed in the form (Fig. 3.10). The spinner control includes only the valid numbers. If the user attempts to enter an invalid value by typing in the text field, a callout pointing to the `number` input element will instruct the user to enter a valid value.

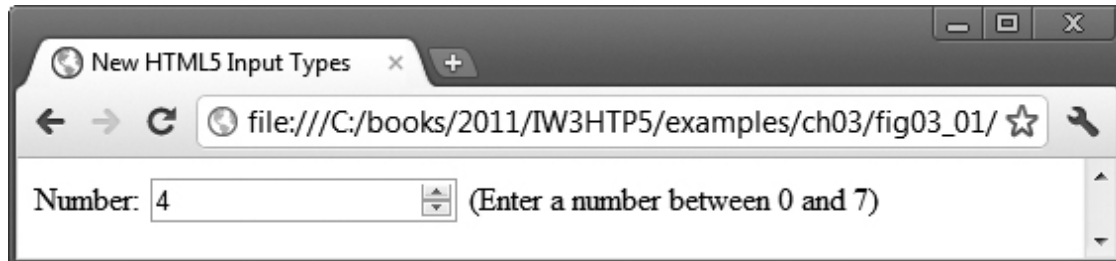


Fig. 3.10. `input` type `number` with a `value` attribute of 4 as rendered in Chrome.

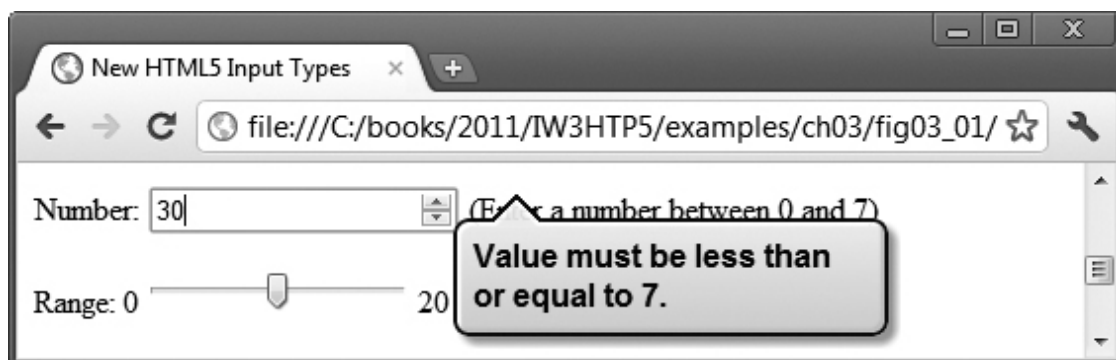


Fig. 3.11. Chrome checking for a valid number.

3.2.8. `input` Type `range`

The **range input type** (lines 64–67) appears as a *slider* control in Chrome, Safari and Opera (Fig. 3.12). You can set the minimum and maximum and specify a value. In our example, the `min` attribute is "0", the `max` attribute is "20" and the `value` attribute is "10", so the slider appears near the center of the range when the document is rendered. The `range` input type is *inherently self-validating* when it is rendered by the browser as a slider control, because *the user is unable to move the slider outside the bounds of the minimum or maximum value*. A range input is more useful if the user can see the current value changing while dragging the thumb—this can be accomplished with JavaScript, as you'll learn later in the book.

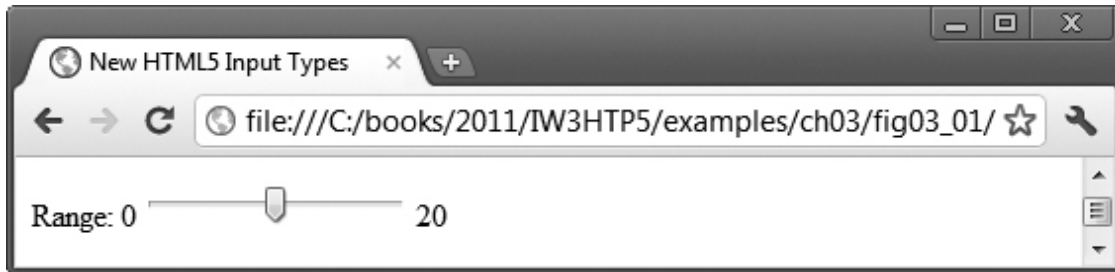


Fig. 3.12. range slider with a value attribute of 10 as rendered in Chrome.

3.2.9. input Type search

The **search input type** (line 72) provides a search field for entering a query. This input element is functionally equivalent to an input of type text. When the user begins to type in the search field, Chrome and Safari display an X that can be clicked to clear the field ([Fig. 3.13](#)).

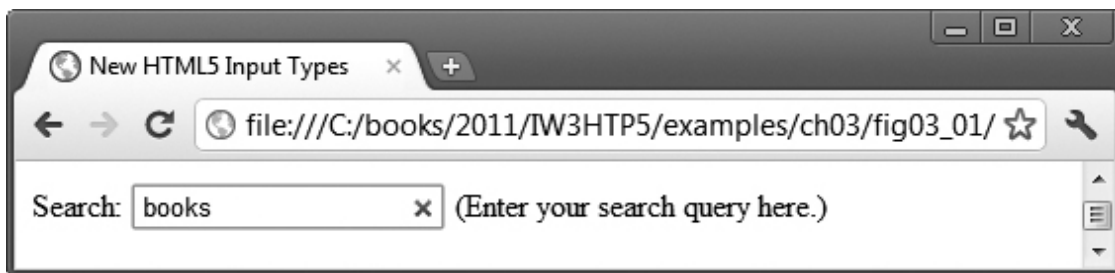


Fig. 3.13. Entering a search query in Chrome.

3.2.10. input Type tel

The **tel input type** (lines 77–79) enables the user to enter a telephone number—mobile browsers typically display a keypad specific to entering phone numbers for this input type. At the time of this writing, the tel input type is rendered as a text field in all of the browsers. The length and format of telephone numbers varies greatly based on location, making validation quite complex. HTML5 does *not* self validate the tel input type. To ensure that the user enters a phone number in a proper format, we've added a pattern attribute (line 79) that uses a *regular expression* to determine whether the number is in the format:

(555) 555-5555

When the user enters a phone number in the wrong format, a callout appears requesting the proper format, pointing to the `tel` input element (Fig. 3.14). Visit www.regexlib.com for a search engine that helps you find already implemented regular expressions that you can use to validate inputs.



Fig. 3.14. Validating a phone number using the `pattern` attribute in the `tel` input type.

3.2.11. input Type `time`

The `time` input type (line 84) enables the user to enter an hour, minute, seconds and fraction of second (Fig. 3.15). The HTML5 specification indicates that a time must have two digits representing the hour, followed by a colon (:), and two digits representing the minute. Optionally, you can also include a colon followed by two digits representing the seconds and a period followed by one or more digits representing a fraction of a second (shown as `fff` in our sample text to the right of the time input element in Fig. 3.15).



Fig. 3.15. `time` input as rendered in Chrome.

3.2.12. input Type `url`

The **url input type** (lines 89–91) enables the user to enter a URL. The element is rendered as a text field, and the proper format is <http://www.deitel.com>. If the user enters an improperly formatted URL (e.g., www.deitel.com or www.deitelcom), the URL will *not* validate (Fig. 3.16). HTML5 does not check whether the URL entered is valid; rather it validates that the URL entered is in the proper format.

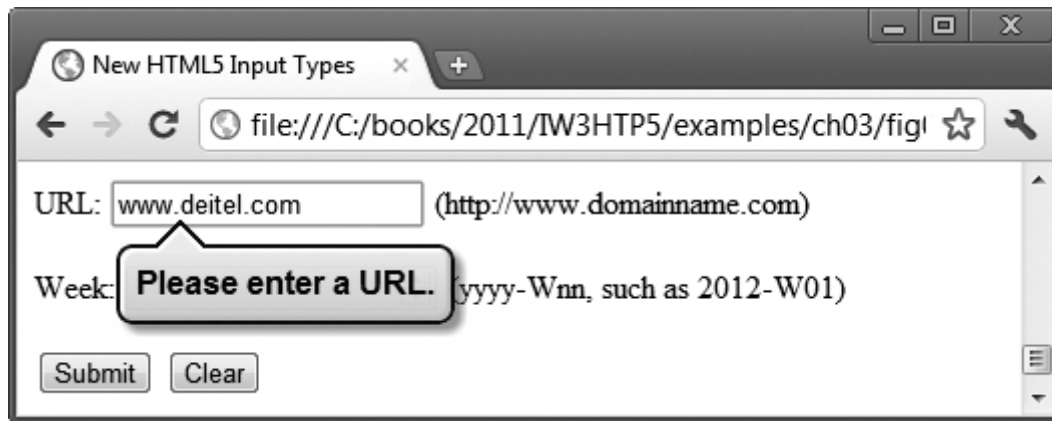


Fig. 3.16. Validating a URL in Chrome.

3.2.13. input Type week

The **week input type** enables the user to select a year and week number in the format `yyyy-Wnn`, where `nn` is 01–53—for example, `2012-W01` represents the first week of 2012. Internet Explorer, Firefox and Safari render a text field. Chrome renders an up-down control. Opera renders *week control* with a down arrow that, when clicked, brings up a calendar for the current month with the corresponding week numbers listed down the left side.

3.3. input and datalist Elements and autocomplete Attribute

[Figure 3.17](#) shows how to use the new `autocomplete` attribute and `datalist` element.

3.3.1. input Element autocomplete Attribute

The **autocomplete attribute** (line 18) can be used on `input` types to automatically fill in the user's information based on previous input—such as name, address or e-mail. You can enable `autocomplete` for an entire

form or just for specific elements. For example, an online order form might set `autocomplete = "on"` for the name and address inputs and set `autocomplete = "off"` for the credit card and password inputs for security purposes.



ERROR-PREVENTION TIP 3.1

The `autocomplete` attribute works only if you specify a `name` or `id` attribute for the `input` element.

```

1  <!DOCTYPE html>
2
3  <!-- Fig. 3.17: autocomplete.html -->
4  <!-- New HTML5 form autocomplete attribute and datalist element. -->
5  <html>
6    <head>
7      <meta charset="utf-8">
8      <title>New HTML5 autocomplete Attribute and datalist
Element</title>
9    </head>
10
11   <body>
12     <h1>Autocomplete and Datalist Demo</h1>
13     <p>This form demonstrates the new HTML5 autocomplete attri-
bute
14       and the datalist element.
15     </p>
16
17     <!-- turn autocomplete on -->
18     <form method = "post" autocomplete = "on">
19       <p><label>First Name:
20         <input type = "text" id = "firstName"
21           placeholder = "First name" /> (First name)

```

```
22     </label></p>
23 <p><label>Last Name:
24     <input type = "text" id = "lastName"
25         placeholder = "Last name" /> (Last name)
26     </label></p>
27 <p><label>Email:
28     <input type = "email" id = "email"
29         placeholder = "name@domain.com" /> (name@domain.com)
30     </label></p>
31 <p><label for = "txtList">Birth Month:
32     <input type = "text" id = "txtList"
33         placeholder = "Select a month" list = "months" />
34     <datalist id = "months">
35         <option value = "January">
36         <option value = "February">
37         <option value = "March">
38         <option value = "April">
39         <option value = "May">
40         <option value = "June">
41         <option value = "July">
42         <option value = "August">
43         <option value = "September">
44         <option value = "October">
45         <option value = "November">
46         <option value = "December">
47     </datalist>
48 </label></p>
49 <p><input type = "submit" value = "Submit" />
50     <input type = "reset" value = "Clear" /></p>
51 </form>
52 </body>
53 </html>
```

a) Form rendered in Firefox before the user interacts with it

Firefox

New HTML5 autocomplete Attribute an...

file:///C:/books/2011/TW3HTP5/examples/

Autocomplete and Datalist Demo

This form demonstrates the new HTML5 autocomplete attribute and and the datalist element.

First Name: (First name)

Last Name: (Last name)

Email: (name@domain.com)

Birth Month: (Select a month)

b) autocomplete automatically fills in the data when the user returns to a form submitted previously and begins typing in the **First Name** input element; clicking Jane inserts that value in the input

Firefox

New HTML5 autocomplete Attribute an...

file:///C:/books/2011/TW3HTP5/examples/

Autocomplete and Datalist Demo

This form demonstrates the new HTML5 autocomplete attribute and and the datalist element.

First Name: J (First name)

Last Name: (Last name)

Email: (name@domain.com)

Birth Month: (Select a month)

c) autocomplete with a datalist showing the previously entered value (June) followed by all items that match what the user has typed so far; clicking an item in the autocomplete list inserts that value in the input

datalist values filtered by what's been typed so far

Firefox

New HTML5 autocomplete Attribute an...

file:///C:/books/2011/TW3HTP5/examples/

Autocomplete and Datalist Demo

This form demonstrates the new HTML5 autocomplete attribute and and the datalist element.

First Name: Jane (First name)

Last Name: (Last name)

Email: (name@domain.com)

Birth Month: j

June
January
June
July

Fig. 3.17. New HTML5 form `autocomplete` attribute and `datalist` element.

3.3.2. `datalist` Element

The **`datalist` element** (lines 32–47) provides input options for a text input element. At the time of this writing, `datalist` support varies by browser. In this example, we use a `datalist` element to obtain the user's birth month. Using Opera, when the user clicks in the text field, a drop-down list of the months of the year appears. If the user types "M" in the text field, the list on months is narrowed to March and May. When using Firefox, the drop-down list of months appears only after the user begins typing in the text field. If the user types "M", all months containing the letter "M" or "m" appear in the drop-down list—March, May, September, November and December.

3.4. Page-Structure Elements

HTML5 introduces several new page-structure elements ([Fig. 3.18](#)) that meaningfully identify areas of the page as headers, footers, articles, navigation areas, asides, figures and more.

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 3.18: sectionelements.html -->
4 <!-- New HTML5 section elements. -->
5 <html>
6   <head>
7     <meta charset="utf-8">
8     <title>New HTML5 Section Elements</title>
9   </head>
10
11  <body>
12    <header> <!-- header element creates a header for the page -->
13      <img src = "deitellogo.png" alt = "Deitel logo" />
14      <h1>Welcome to the Deitel Buzz Online</h1>
```

```
15
16     <!-- time element inserts a date and/or time -->
17     <time>2012-01-17</time>
18
19 </header>
20
21 <section id = "1"> <!-- Begin section 1 -->
22     <nav> <!-- nav element groups navigation links -->
23         <h2> Recent Publications</h2>
24         <ul>
25             <li><a href = "http://www.deitel.com/books/iw3http5">
26                 Internet & World Wide Web How to Program, 5/e</a></li>
27             <li><a href = "http://www.deitel.com/books/androidfp/">
28                 Android for Programmers: An App-Driven Approach</a>
29             </li>
30             <li><a href = "http://www.deitel.com/books/iphonefp">
31                 iPhone for Programmers: An App-Driven Approach</a></li>
32             <li><a href = "http://www.deitel.com/books/jhttp9/">
33                 Java How to Program, 9/e</a></li>
34             <li><a href = "http://www.deitel.com/books/cpphttp8/">
35                 C++ How to Program, 8/e</a></li>
36             <li>
37                 <a href = "http://www.deitel.com/books/vcsharp2010http">
38                     Visual C# 2010 How to Program, 4/e</a></li>
39             <li><a href = "http://www.deitel.com/books/vb2010http">
40                 Visual Basic 2010 How to Program</a></li>
41         </ul>
42     </nav>
43 </section>
44
45 <section id = "2"> <!-- Begin section 2 -->
46     <h2>How to Program Series Books</h2>
47     <h3><em>Java How to Program, 9/e</em></h3>
48
49     <figure> <!-- figure element describes the image -->
50         <img src = "jhttp.jpg" alt = "Java How to Program, 9/e" />
```

```
51
52      <!-- figurecaption element inserts a figure caption -->
53      <figcaption><em>Java How to Program, 9/e</em>
54          cover.</figcaption>
55  </figure>
56
57  <!--article element represents content from another source -->
58  <article>
59      <header>
60          <h5>From
61              <em>
62                  <a href = "http://www.deitel.com/books/jhttp9/">
63                      Java How to program, 9/e: </a>
64              </em>
65          </h5>
66      </header>
67
68      <p>Features include:
69          <ul>
70              <li>Rich coverage of fundamentals, including
71                  <!-- mark element highlights text -->
72                  <mark>two chapters on control statements.</mark></li>
73              <li>Focus on <mark>real-world examples.</mark></li>
74              <li><mark>Making a Difference exercises set.</mark></li>
75              <li>Early introduction to classes, objects,
76                  methods and strings.</li>
77              <li>Integrated exception handling.</li>
78              <li>Files, streams and object serialization.</li>
79              <li>Optional modular sections on language and
80                  library features of the new Java SE 7.</li>
81              <li>Other topics include: Recursion, searching,
82                  sorting, generic collections, generics, data
83                  structures, applets, multimedia,
84                  multithreading, databases/JDBC&trade;, web-app
85                  development, web services and an optional
86                  ATM Object-Oriented Design case study.</li>
```

```
87      </ul>
88
89      <!-- summary element represents a summary for the -->
90      <!-- content of the details element -->
91      <details>
92          <summary>Recent Edition Testimonials</summary>
93          <ul>
94              <li>"Updated to reflect the state of the
95                  art in Java technologies; its deep and
96                  crystal clear explanations make it
97                  indispensable. The social-consciousness
98                  [Making a Difference] exercises are
99                  something really new and refreshing."
100              <strong>&mdash;Jos&eacute; Antonio
101                  Gonz&aacute;lez Seco, Parliament of
102                  Andalusia</strong></li>
103              <li>"Gives new programmers the benefit of the
104                  wisdom derived from many years of software
105                  development experience."<strong>
106                  &mdash;Edward F. Gehringer, North Carolina
107                  State University</strong></li>
108              <li>"Introduces good design practices and
109                  methodologies right from the beginning.
110                  An excellent starting point for developing
111                  high-quality robust Java applications."
112              <strong>&mdash;Simon Ritter,
113                  Oracle Corporation</strong></li>
114              <li>"An easy-to-read conversational style.
115                  Clear code examples propel readers to
116                  become proficient in Java."
117              <strong>&mdash;Patty Kraft, San Diego State
118                  University</strong></li>
119              <li>"A great textbook with a myriad of examples
120                  from various application domains&mdash;
121                  excellent for a typical CS1 or CS2 course."
122              <strong>&mdash;William E. Duncan, Louisiana
```



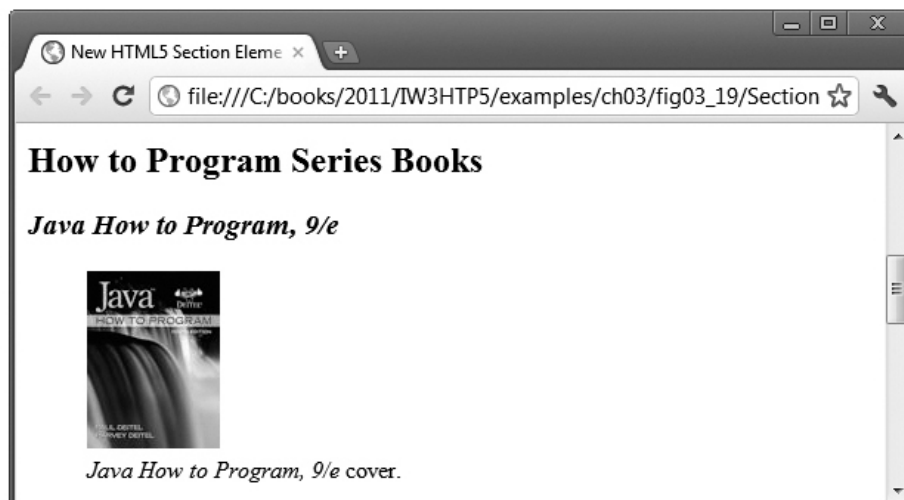
```
123         State University</strong></li>
124     </ul>
125 </details>
126 </p>
127 </article>
128
129 <!-- aside element represents content in a sidebar that's -->
130 <!-- related to the content around the element -->
131 <aside>
132     The aside element is not formatted by the browsers.
133 </aside>
134
135 <h2>Deitel Developer Series Books</h2>
136 <h3><em>Android for Programmers: An App-Driven Approach
137     </em></h3>
138     Click <a href = "http://www.deitel.com/books/androidfp/">
139     here</a> for more information or to order this book.
140
141 <h2>LiveLessons Videos</h2>
142 <h3><em>C# 2010 Fundamentals LiveLessons</em></h3>
143     Click <a href = "http://www.deitel.com/Books/LiveLessons/">
144     here</a> for more information about our LiveLessons videos.
145 </section>
146
147 <section id = "3"> <!-- Begin section 3 -->
148     <h2>Results from our Facebook Survey</h2>
149     <p>If you were a nonprogrammer about to learn Java for the
first
150     time, would you prefer a course that taught Java in the
151     context of Android app development? Here are the results
from
152     our survey:</p>
153
154 <!-- meter element represents a scale within a range -->
155     0 <meter min = "0"
156     max = "54"
```

```
157     value = "14"></meter> 54
158     <p>Of the 54 responders, 14 (green) would prefer to
159     learn Java in the context of Android app development.</p>
160 </section>
161
162 <!-- footer element represents a footer to a section or page, -->
163 <!-- usually containing information such as author name, -->
164 <!-- copyright, etc. -->
165 <footer>
166     <!-- wbr element indicates the appropriate place to break a -->
167     <!-- word when the text wraps -->
168     <h6>&copy; 1992-2012 by Deitel & Associ<wbr>ates, Inc.
169     All Rights Reserved.<h6>
170     <!-- address element represents contact information for a -->
171     <!-- document or the nearest body element or article -->
172     <address>
173         Contact us at <a href = "mailto:deitel@deitel.com">
174         deitel@deitel.com</a>
175     </address>
176 </footer>
177 </body>
178 </html>
```

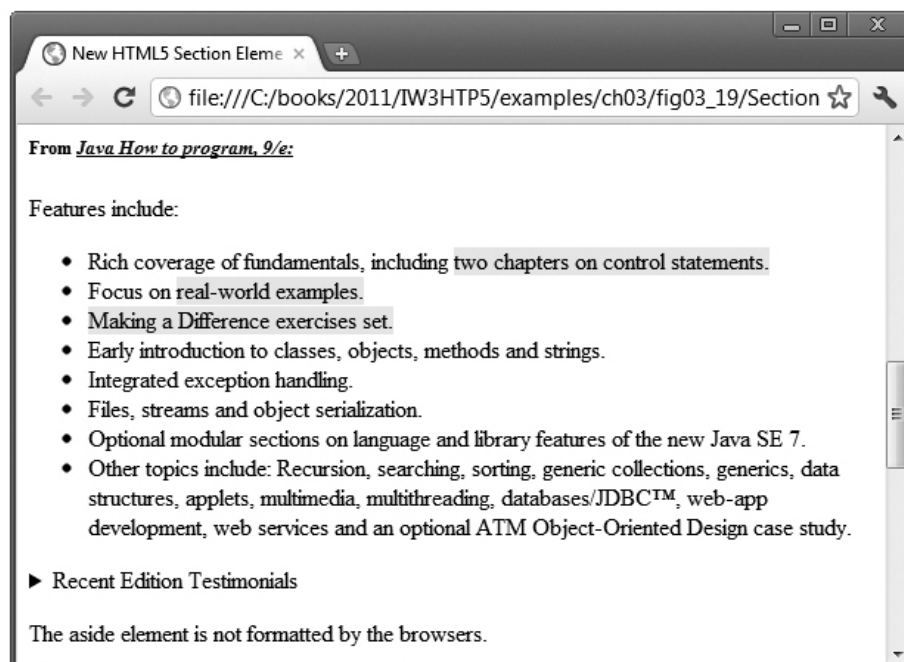
a) Chrome browser showing the header element and a nav element that contains an unordered list of links



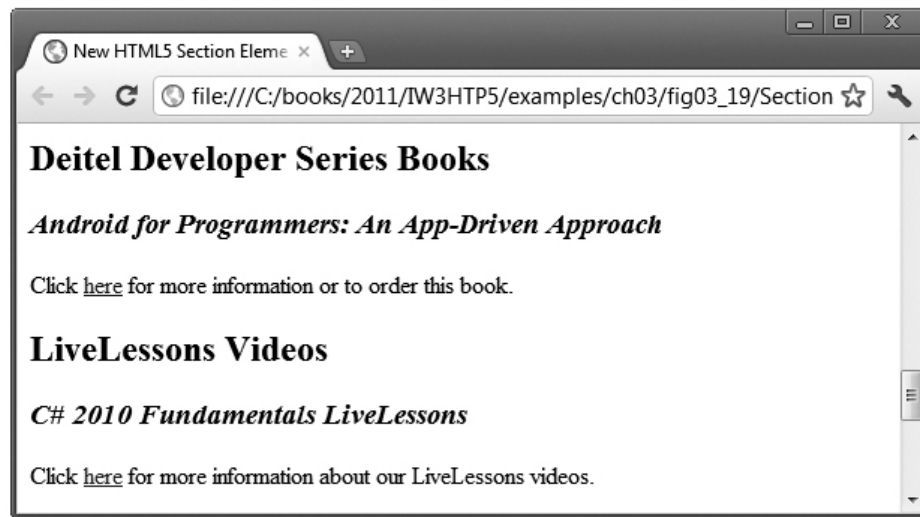
b) Chrome browser showing the beginning of a section containing a figure and a figurecaption



c) Chrome browser showing an article containing a header, some content and a collapsed details element, followed by an aside element



d) Chrome browser showing the end of the section that started in part (b)



e) Chrome browser showing the last section containing a meter element, followed by a footer element



Fig. 3.18. New HTML5 section elements.

3.4.1. header Element

The **header element** (lines 12–19) creates a header for this page that contains both text and graphics. The **header element** can be used multiple times on a page and can include HTML headings (`<h1>` through `<h6>`), navigation, images and logos and more. For an example, see the top of the front page of your favorite newspaper.

time Element

The **time element** (line 17), which does not need to be enclosed in a header , enables you to identify a date (as we do here), a time or both.

3.4.2. nav Element

The **nav element** (lines 22–42) groups navigation links. In this example, we used the heading **Recent Publications** and created a **ul** element with seven **li** elements that link to the corresponding web pages for each book.

3.4.3. figure Element and figcaption Element

The **figure element** (lines 49–55) describes a figure (such as an image, chart or table) in the document so that it could be moved to the side of the page or to another page. The **figure** element does not include any styling, but you can style the element using CSS. The **figcaption element** (lines 53–54) provides a caption for the image in the **figure** element.

3.4.4. article Element

The **article** element (lines 58–127) describes standalone content that could potentially be used or distributed elsewhere, such as a news article, forum post or blog entry. You can nest **article** elements. For example, you might have reader comments about a magazine nested as an **article** within the magazine **article**.

3.4.5. summary Element and details Element

The **summary element** (line 92) displays a right-pointing arrow next to a summary or caption when the document is rendered in a browser ([Fig. 3.19](#)). When clicked, the arrow points downward and reveals the content in the **details element** (lines 91–125).

3.4.6. section Element

The **section element** describes a section of a document, usually with a heading for each section—these elements can be nested. For example, you could have a **section** element for a book, then nested **sections** for each chapter name in the book. In this example, we broke the document into three **sections**—the first is **Recent Publications** (lines 21–43). The **section** element may also be nested in an article.

3.4.7. aside Element

The **aside element** (lines 131–133) describes content that’s related to the surrounding content (such as an `article`) but is somewhat separate from the flow of the text. For example, an `aside` in a news story might include some background history. A print advertisement might include an `aside` with product testimonials from users.

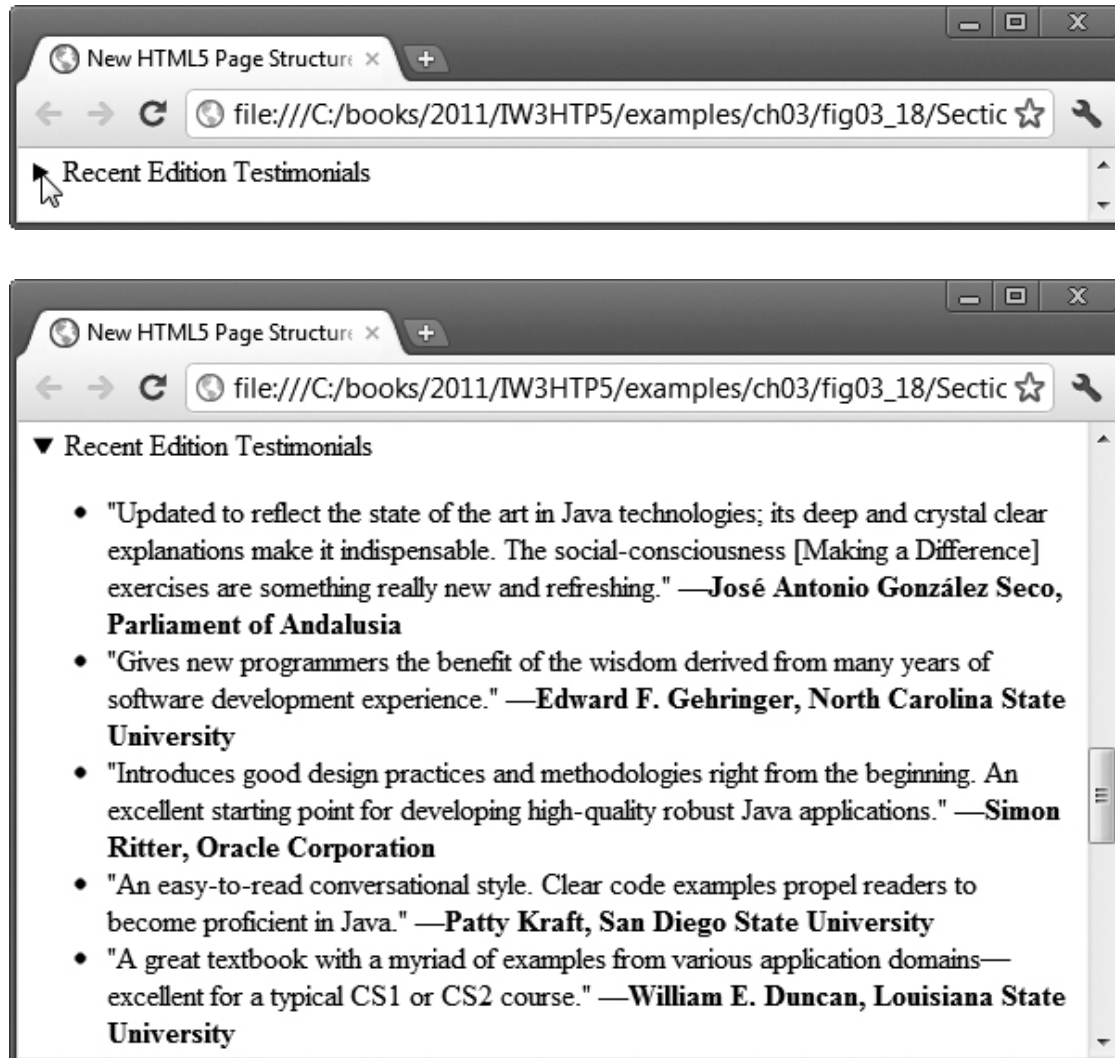


Fig. 3.19. Demonstrating the `summary` and `detail` elements.

3.4.8. `meter` Element

The **`meter` element** (lines 155–157) renders a visual representation of a measure within a range (Fig. 3.20). In this example, we show the results of a recent web survey we did. The `min` attribute is `"0"` and a `max` attribute is `"54"`—indicating the total number of responses to our survey. The `value` attribute is `"14"`, representing the total number of people who responded “yes” to our survey question.



Fig. 3.20. Chrome rendering the `meter` element.

3.4.9. footer Element

The **footer element** (lines 165–176) describes a *footer*—content that usually appears at the bottom of the content or `section` element. In this example, we use the `footer` to describe the copyright notice and contact information. You can use CSS3 to style the `footer` and position it on the page.

3.4.10. Text-Level Semantics: mark Element and wbr Element

The **mark element** (lines 72–74) highlights the text that’s enclosed in the element. The **wbr element** (line 168) indicates the appropriate place to break a word when the text wraps to multiple lines. You might use `wbr` to prevent a word from breaking in an awkward place.

Summary


Section 3.2 New HTML5 Form input Types

- HTML5 introduces several new form `input` types and attributes. These are not yet universally supported by all browsers.
- The Opera browser offers robust support of the new `input` types.
- We provide sample outputs from a variety of browsers so that you can see how the `input` types behave differently in each.

Section 3.2.1 input Type color

- The `color` input type ([p. 80](#)) enables the user to enter a color.
- Most browsers render the `color` input type as a text field in which the user can enter a hexadecimal code.
- In the future, when the user clicks a color input, browsers will likely display a dialog from which the user can select a color.
- The `autofocus` attribute ([p. 80](#))—which can be used in only one input element on a form—places the cursor in the text field after the browser loads and renders the page. You do not need to include `autofocus` in your forms.
- The new HTML 5 input types self validate on the client side, eliminating the need to add JavaScript code to validate user input and reducing the amount of invalid data submitted.
- When a user enters data into a form then submits the form, the browser immediately checks that the data is correct.
- If you want to bypass validation, you can add the `formnovalidate` attribute ([p. 82](#)) to input type `submit`.
- Using JavaScript, we can customize the validation process.

Section 3.2.2 input Type date

- The `date` input type ([p. 82](#)) enables the user to enter a date in the format `yyyy-mm-dd`.
- Firefox and Internet Explorer all display a text field in which a user can enter a date such as `2012-01-27`.
- Chrome and Safari display a spinner control ([p. 82](#))—a text field with an up-down arrow () on the right side—allowing the user to select a date by clicking the up or down arrows.

- Opera displays a calendar.

Section 3.2.3 input Type datetime

- The `datetime` input type ([p. 82](#)) enables the user to enter a date (year, month, day), time (hour, minute, second, fraction of a second) and the time zone set to UTC (Coordinated Universal Time or Universal Time, Coordinated).

Section 3.2.4 input Type datetime-local

- The `datetime-local` input type ([p. 82](#)) enables the user to enter the date and time in a *single* control.
- The date is entered as year, month, day, hour, minute, second and fraction of a second.

Section 3.2.5 input Type email

- The `email` input type ([p. 83](#)) enables the user to enter an e-mail address or list of e-mail addresses separated by commas.
- If the user enters an invalid e-mail address (i.e., the text entered is not in the proper format) and clicks the **Submit** button, a callout asking the user to enter an e-mail address is rendered pointing to the `input` element.
- HTML5 does not validate whether an e-mail address entered by the user actually exists—rather it just validates that the information is in the *proper format*.
- The `placeholder` attribute ([p. 83](#)) allows you to place temporary text in a text field. Generally, `placeholder` text is light gray and provides an example of the text and text format the user should enter. When the focus is placed in the text field (i.e., the cursor is in the text field), the `placeholder` text disappears—it's not “submitted” when the user clicks the **Submit** button (unless the user types the same text).
- Add descriptive text to the right of each `input` element in case the user's browser does not support `placeholder` text.

- The `required` attribute ([p. 84](#)) forces the user to enter a value before submitting the form.
- You can add `required` to any of the `input` types. If the user fails to fill enter a required item, a callout pointing to the empty element appears, asking the user to enter the information.

Section 3.2.6 `input Type month`

- The `month` `input` type ([p. 84](#)) enables the user to enter a year and month in the format `yyyy-mm`, such as `2012-01`.
- If the user enters a month in an improper format and clicks the **Submit** button, a callout stating that an invalid value was entered appears.

Section 3.2.7 `input Type number`

- The `number` `input` type ([p. 84](#)) enables the user to enter a numerical value.
- The `min` attribute sets the minimum valid number, in this case `"0"`.
- The `max` attribute sets the maximum valid number, which we set to `"7"`.
- The `step` attribute determines the increment in which the numbers increase. For example, if we set the `step` to `"2"`, the number in the spinner control will increase or decrease by two each time the up or down arrow, respectively, in the spinner control is clicked.
- The `value` attribute sets the initial value displayed in the form.
- The spinner control includes only the valid numbers. If the user attempts to enter an invalid value by typing in the text field, a callout pointing to the `number` `input` element will instruct the user to enter a valid value.

Section 3.2.8 `input Type range`

- The `range` `input` type ([p. 85](#)) appears as a *slider* control in Chrome, Safari and Opera.

- You can set the minimum and maximum and specify a value.
- The slider appears at the value in the range when the HTML5 document is rendered.
- The `range` input type is inherently self-validating when it's rendered by the browser as a slider control, because the user is unable to move the slider outside the bounds of the minimum or maximum value.

Section 3.2.9 input Type search

- The `search` input type ([p. 85](#)) provides a search field for entering a query and is functionally equivalent to an input of type `text`.
- When the user begins to type in the search field, Chrome and Safari display an **X** that can be clicked to clear the field.

Section 3.2.10 input Type tel

- The `tel` input type ([p. 86](#)) enables the user to enter a telephone number.
- At the time of this writing, the `tel` input type is rendered as a text field in all of the browsers.
- The length and format of telephone numbers varies greatly based on location, making validation quite complex. HTML5 does not self validate the `tel` input type. To ensure that the user enters a phone number in a proper format, you can use the `pattern` attribute.
- When the user enters a phone number in the wrong format, a callout requesting the proper format appears, pointing to the `tel` input element.

Section 3.2.11 input Type time

- The `time` input type ([p. 86](#)) enables the user to enter an hour, minute, second and fraction of a second.

Section 3.2.12 input Type url

- The `url` input type ([p. 87](#)) enables the user to enter a URL. The element is rendered as a text field. If the user enters an improperly formatted URL, it will not validate. HTML5 does not ensure that the URL entered actually exists.

Section 3.2.13 input Type week

- The `week` input type ([p. 87](#)) enables the user to select a year and week number in the format `yyyy-Wnn`.
- Opera renders week control with a down arrow that, when clicked, brings up a calendar control.

Section 3.3.1 input Element autocomplete Attribute

- The `autocomplete` attribute ([p. 87](#)) can be used on input types to automatically fill in the user's information based on previous input.
- You can enable `autocomplete` for an entire form or just for specific elements.

Section 3.3.2 datalist Element

- The `datalist` element ([p. 90](#)) provides input options for a text input element. The browser can use these options to display `autocomplete` options to the user.

Section 3.4 Page-Structure Elements

- HTML5 introduces several new page structure elements.

Section 3.4.1 header Element

- The `header` element ([p. 96](#)) creates a header for the page that contains text, graphics or both.
- The `header` element may be used multiple times on a page and often includes HTML headings.

- The `time` element ([p. 96](#)) enables you to identify a date, a time or both.

Section 3.4.2 `nav` Element

- The `nav` element ([p. 96](#)) groups navigation links.

Section 3.4.3 `figure` Element and `figcaption` Element

- The `figure` element ([p. 96](#)) describes an image in the document so that it could be moved to the side of the page or to another page.
- The `figcaption` element ([p. 96](#)) provides a caption for the image in the `figure` element.

Section 3.4.4 `article` Element

- The `article` element ([p. 96](#)) describes content that's separate from the main content of the page and might be used or distributed elsewhere, such as a news article, forum post or blog entry.
- `article` elements can be nested.

Section 3.4.5 `summary` Element and `details` Element

- The `summary` element ([p. 96](#)) displays a right-pointing arrow next to a summary or caption when the document is rendered in a browser. When clicked, the arrow points downward and reveals the content in the `details` element ([p. 96](#)).

Section 3.4.6 `section` Element

- The `section` element ([p. 96](#)) describes a section of a document, usually with a heading for each section.
- `section` elements can be nested.

Section 3.4.7 `aside` Element

- The `aside` element ([p. 96](#)) describes content that's related to the surrounding content (such as an `article`) but that's somewhat separate

from the flow of the text.

- `nav` elements can be nested in an `aside` element.

Section 3.4.8 `meter` Element

- The `meter` element ([p. 97](#)) renders a visual representation of a measure within a range.
- Useful `meter` attributes are `min`, `max` and `value`.

Section 3.4.9 `footer` Element

- The `footer` element ([p. 98](#)) describes a *footer*—content that usually appears at the bottom of the content or `section` element.
- You can use CSS3 to style the footer and position it on the page.

Section 3.4.10 Text-Level Semantics: `mark` Element and `wbr` Element

- The `mark` element ([p. 98](#)) enables you to highlight text.
- The `wbr` element ([p. 98](#)) indicates the appropriate place to break a word when the text wraps to multiple lines. You might use `wbr` to prevent a word from breaking in an awkward place.

Self-Review Exercises

3.1 Fill in the blanks in each of the following:

- The `color` `input` type enables the user to enter a color. At the time of this writing, most browsers render the `color` `input` type as a text field in which the user can enter a _____.
- The _____ attribute allows you to place temporary text in a text field.
- If you want to bypass validation, you can add the `formnovalidate` attribute to `input` type _____.

- d. The _____ attribute forces the user to enter a value before submitting the form.
- e. The _____ control is typically displayed for the `number` `input` type and includes only the valid numbers.
- f. The _____ `input` type enables the user to enter an hour, minute, second and fraction of second.
- g. The _____ element provides input options for a `text` `input` element.
- h. The _____ element describes content that's separate from the main content of the page and could potentially be used or distributed elsewhere, such as a news article, forum post or blog entry.
- i. The _____ element describes the text that usually appears at the bottom of the content or the bottom of a `section` element.
- j. The _____ element indicates the appropriate place to break a word when the text wraps to multiple lines.

3.2 State whether each of the following is *true* or *false*. If *false*, explain why.

- a. Any particular HTML5 form `input` types must render identically in every HTML5-compliant browser.
- b. When the focus is placed in the text field (i.e., the cursor is in the text field), the `placeholder` text is submitted to the server.
- c. You do not need to include `autofocus` in your forms.
- d. The new HTML 5 `input` types are self validating on the client side, eliminating the need to add complicated scripts to your forms to validate user input and reducing the amount of invalid data submitted.
- e. The `range` `input` type is inherently self-validating when it's rendered by the browser as a slider control, because the user is unable to move the slider outside the bounds of the minimum or maximum value.

- f.** HTML5 self validates the `tel` input type.
- g.** If the user enters an improperly formatted URL in a `url` input type, it will not validate. HTML5 does not validate that the URL entered actually exists.
- h.** The `nav` element displays a drop-down menu of hyperlinks.
- i.** The `header` element may be used only one time on a page.
- j.** `nav` elements can be nested in an `aside` element.
- k.** You might use the `brk` to prevent awkward word breaks.

Answers to Self-Review Exercises

3.1

- a.** hexadecimal code.
- b.** placeholder .
- c.** submit .
- d.** required .
- e.** spinner.
- f.** time .
- g.** datalist .
- h.** article .
- i.** footer .
- j.** wbr .

3.2

- a. False. The rendering of `input` types can vary among browsers.
- b. False. When the focus is placed in the text field, the `placeholder` text disappears. It's not "submitted" when the user clicks the **Submit** button (unless the user types the same text).
- c. True.
- d. True.
- e. True.
- f. False. The length and format of telephone numbers varies greatly based on location, making validation quite complex, so HTML5 does not self validate the `tel` input type. To ensure that the user enters a phone number in a proper format, we use the `pattern` attribute.
- g. True.
- h. False. The `nav` element groups navigation links.
- i. False. The `header` element may be used multiple times on a page and often includes HTML headings (`<h1>` through `<h6>`)
- j. True.
- k. False. You might use the `wbr` to prevent awkward word breaks.

Exercises

3.3 Fill in the blanks in each of the following:

- a. The _____ attribute—used in a single `input` element on a form—automatically highlights the `input` element and, if appropriate, places the cursor in the text field after the browser loads and renders the page.
- b. The new HTML 5 `input` types are _____ on the client side.
- c. The _____ `input` type enables the user to enter a numerical value.

- d. The _____ `input` type is inherently self-validating when it's rendered by the browser as a slider control, because the user is unable to move the slider outside the bounds of the minimum or maximum value.
- e. The _____ attribute can be used on `input` types to automatically fill in the user's information based on previous input.
- f. The _____ element provides a caption for the image in the `figure` element.
- g. The `summary` element displays a right-pointing arrow next to a summary or caption when the document is rendered in a browser. When clicked, the arrow points downward and reveals the content in the _____ element.
- h. The `mark` element enables you to _____.

3.4 State whether each of the following is *true* or *false*. If *false*, explain why.

- a. Browsers that render the `color` `input` type as a text field require the user to enter a color name.
- b. When a user enters data into a form then submits the form (typically, by clicking the **Submit** button), the browser immediately checks that the data is correct.
- c. HTML5 can validate whether an e-mail address entered by the user actually exists.
- d. You can add `required` to any of the `input` types.
- e. You can enable `autocomplete` only for specific `input` elements.
- f. The `time` element enables you to indentify a date, a time or both.
- g. The `caption` element provides a caption for the image in a `figure` element.

- h.** The `details` element displays a right-pointing arrow next to a summary or caption when the document is rendered in a browser. When clicked, the arrow points downward and reveals the content in the `summary` element.
- i.** The `footer` element describes content that usually appears at the bottom of the content or `section` element.
- j.** The `highlight` element enables you to highlight text.

3.5 Write an HTML5 element (or elements) to accomplish each of the following tasks:

- a.** Students were asked to rate the food in the cafeteria on a scale of 1 to 10. Use a `meter` element with text to its left and right to indicate that the average rating was 7 out of 10.
- b.** Create a `details` element that displays the `summary` text "Survey Results" for Part (a). When the user clicks the arrow next to the `summary` text, an explanatory paragraph about the survey should be displayed.
- c.** Create a `text input` element for a first name. The element should automatically receive the focus when the form is rendered in a browser.
- d.** Modify Part (c) to eliminate the `label` element and use placeholder text in the `input` element.
- e.** Use a `datalist` to provide an `autocomplete` list for five states.
- f.** Create a `range input` element that allows the user to select a number from 1 to 100.
- g.** Specify that `autocomplete` should not be allowed for a form. Show only the form's opening tag.
- h.** Use a `mark` element to highlight the second sentence in the following paragraph.

<p>Students were asked to rate the food in the cafeteria
on a scale of 1 to 10. The average result was 7.</p>

3.6 (Website Registration Form with Optional Survey) Create a website registration form to obtain a user's first name, last name and e-mail address. In addition, include an optional survey question that asks the user's year in college (e.g., Freshman). Place the optional survey question in a `details` element that the user can expand to see the question.

3.7 (Creating an Autocomplete Form) Create a simple search form using a `search input` element in which the user can enter a search query. Using the Firefox web browser, test the form by entering `January` and submitting the form. Then enter a `J` in the input element to see previous entries that started with `J` — `January` should be displayed below the input element. Enter `June` and submit the form again. Now enter a `J` in the `input` element to see previous entries that started with `J` — `January` and `June` should be displayed below the `input` element. Try this with your own search queries as well.

3.8 (Creating an Autocomplete Form with a `datalist`) Create an `autocomplete input` element with an associated `datalist` that contains the days of the week.

3.9 (Laying Out Book Pages in HTML5: Creating the Sections) Mark up the paragraph text from [Section 3.2.1](#) of this chapter as a web page using page-structure elements. The text is provided in the `exerciseTextAndImages` folder with this chapter's examples. Do not include the figures in this exercise.

3.10 (Laying Out Book Pages in HTML5: Adding Figures) Modify your solution to Exercise 3.9 to add the section's graphics as figures. The images are provided in the `exerciseTextAndImages` folder with this chapter's examples.

3.11 (Laying Out Book Pages in HTML5: Adding a `details` Element) Modify your solution to Exercise 3.10 to add the table in [Fig. 3.5](#). Use the figure caption as the `summary` and format the table as an HTML table element inside the `details` element.

[Support](#) [Sign Out](#)

©2022 O'REILLY MEDIA, INC. [TERMS OF SERVICE](#) [PRIVACY POLICY](#)