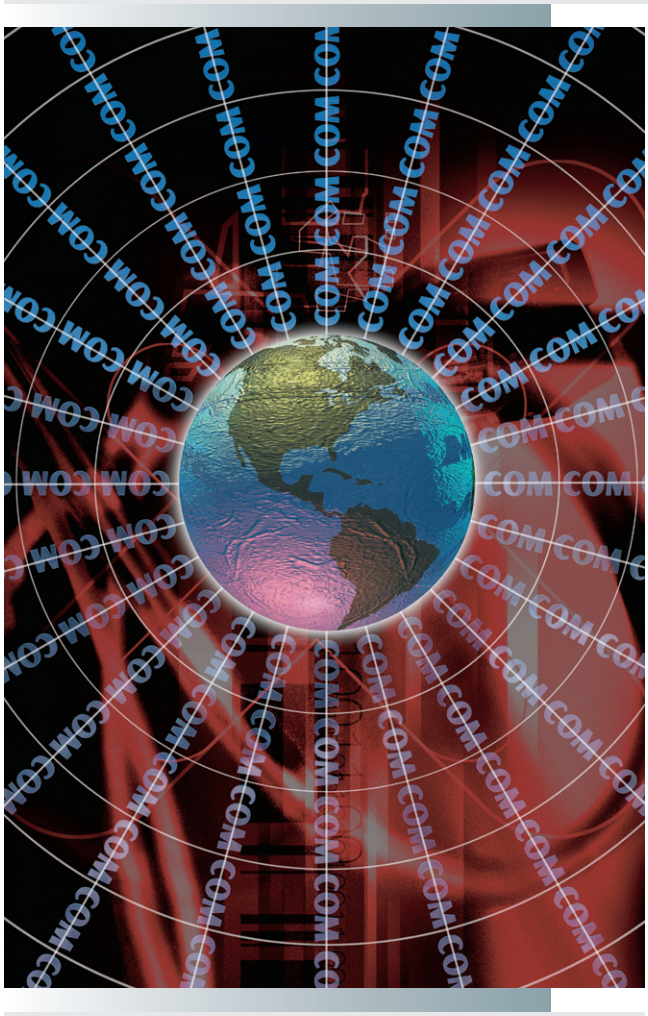

chapter 9

Making Web Pages Accessible

"The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect."

—Tim Berners-Lee, director of the W3C and inventor of the World Wide Web

In this chapter, you will learn how to:

- Define the concept of Web accessibility and list applicable guidelines and standards for making Web sites accessible.
- List the HTML coding practices you must follow to make a Web site meet the Section 508 accessibility standards.
- Identify resources other than HTML pages that must adhere to the accessibility guidelines in order for a Web site to be accessible.
- Define how style sheets can enhance a Web site's accessibility.
- List the tools you can use to assess the extent to which a site follows Web accessibility standards.

ACCESS to the Internet is vitally important for anyone who plans to participate in the twenty-first century information economy. So essential is the Web that to be denied access is to be disenfranchised. To help ensure that all U.S. citizens have access, the U.S. government has enacted laws mandating that certain accessibility features must be built into any Web site that receives public funding or serves constituents of federally funded programs. Because the need is worldwide, the W3C has initiated a Web Accessibility Initiative (WAI) that works with organizations around the world to coordinate efforts to enable all users to access the Web, regardless of disability or special needs.

Making the Web truly accessible to everyone is a great challenge, because users with special needs have many different kinds of disabilities. This chapter begins by making you aware of guidelines you can follow to make your Web pages accessible. Tutorial exercises will step you through the process of implementing HTML coding practices that bring a Web site into compliance with federally mandated accessibility standards. Because a Web site can include many kinds of multimedia documents, however, accessibility goes beyond HTML. In order for a site to be truly accessible, all of its printed, audio, and video formats must also comply with accessibility rules. This chapter will identify multimedia accessibility guidelines that are emerging for non-HTML resources typically found on the Web.

In the long run, however, creating alternate representations may not be the best approach. Rather, the key to achieving true accessibility may be for the computer industry to provide a way for end users to specify the kind of accessibility they need. Imagine how a style sheet, for example, could invoke an XML module to transform a certain class of Web content into a representation suited to the special needs of the viewer. After providing some examples of style sheets that provide new avenues for Web accessibility in the future, this chapter will conclude by providing tools you can use to evaluate the extent to which a Web page follows the guidelines currently in force.

Defining Web Accessibility

Web accessibility is the capability that makes it possible for users with special needs to receive, understand, and navigate content that people without handicaps can process without special assistance. Users with special

needs have many different kinds of handicaps. As you will learn in this chapter, the Web accessibility guidelines currently in force address primarily the needs of seeing- or hearing-impaired users. The current guidelines fall short for other kinds of handicaps, such as physical motion impairments and mental cognitive differences. Making Web sites truly accessible to all users is an ongoing challenge, especially for people with multiple handicaps. After presenting the guidelines currently in force, this chapter will discuss efforts that are underway to build in accessibility to the Web from an architectural perspective known as universal design.

Accessibility Is a Right

In the United States, Web accessibility is a right that is guaranteed by law under **Section 508** of the Rehabilitation Act of 1973, as amended in 1998. According to the Section 508 law, a Web site is accessible when users with special needs can access it as well as people without disabilities. The law requires that all Web sites (as well as other forms of information technology) used, procured, developed, or maintained by government agencies and departments must be accessible. If your school or business receives any kind of federal funding, therefore, the law may require you to follow the Section 508 accessibility standards presented in this chapter. Whether or not the law applies to your particular situation, the Section 508 standards are not difficult to implement. This chapter's tutorial will provide you with step-by-step instructions for creating Web pages that are Section 508 compliant. On behalf of users with special needs, I encourage you to follow these guidelines, regardless of whether the law requires them in your workplace.

W3C Web Accessibility Initiative (WAI)

Before diving into the Section 508 accessibility standards, you need to learn a little historical background on the source of those standards. In 1997, the W3C launched the **Web Accessibility Initiative (WAI)**, which coordinates the Web's official efforts to achieve accessibility. WAI went right to work on HTML version 4.0, which introduced new mechanisms for making Web page elements accessible. To provide Web authors with guidance in using the new accessibility features, the WAI issued a set of guidelines called the WAI Web Content Accessibility Guidelines version 1.0. These guidelines influenced the formulation of the Section 508 guidelines that this chapter will present. The work of the WAI is documented at www.w3.org/WAI.

WAI Web Content Accessibility Guidelines

The **Web Content Accessibility Guidelines (WCAG)** consist of 65 checkpoints organized under 14 general guidelines. Each checkpoint is assigned to one of three priority levels, which define the degree to which the site is accessible. Priority 1 is defined as a checkpoint that must be met, otherwise many users with disabilities will find it impossible to access the material. Priority 2 is a checkpoint that should be met, otherwise users will find it

Logo	What the Logo Means
	Conformance Level A: this page satisfies all Priority 1 checkpoints.
	Conformance Level Double-A: this page satisfies all Priority 1 and 2 checkpoints.
	Conformance Level Triple-A: this page satisfies all Priority 1, 2, and 3 checkpoints.

TABLE 9-1 *W3C Web Content Accessibility Guidelines 1.0 Conformance Logos*

difficult, but not impossible, to access the material. Priority 3 is a checkpoint that may be met, in order to further access to Web documents. The three levels of conformance are designated A, AA, and AAA, respectively. Conformance level A requires that a site pass all Priority 1 checkpoints. Conformance level AA (pronounced double-A) requires the passing of all Priority 1 and 2 checkpoints. Level AAA (triple-A) requires that a site pass all of the checkpoints at Priorities 1, 2, and 3. Depending on the level at which a Web site claims to conform, the site's pages can display one of three conformance logos described in Table 9-1.

Section 508 Accessibility Standards

The Section 508 accessibility standards do not include levels of conformance and as many checkpoints as the WCAG guidelines. Instead, Section 508 includes 16 Web accessibility requirements, all of which must be met in order for a Web site to be considered accessible. To see the standards, go to www.section508.gov and follow the link to the 508 law, which will give you the choice of viewing either a summary or a detailed presentation of the regulations. The next part of this chapter is a tutorial in making Web pages comply with the 16 Web accessibility standards in the Section 508 guidelines.

Coding to the Section 508 Web Accessibility Standards

Table 9-2 lists the 16 rules of the Section 508 Web accessibility standards. These rules are lettered from (a) through (p) in section §1194.21 of the law. The following sections of this chapter are a brief guide to creating accessible Web pages that comply with these rules. The tutorial exercises cover the rules in the order in which they appear in the law. For more detailed examples, go to www.section508.gov and follow the link to 508 Training.

Textual Equivalents for Nontext Elements

Section 508 Web accessibility rule (a) requires that you must provide a textual equivalent for every nontext element onscreen. HTML has two attributes that you can use to create textual equivalents for nontext elements: the **alt attribute** and the **longdesc attribute**. The attribute `alt` stands for

Rule	Web Accessibility Requirement
(a)	A text equivalent for every non-text element shall be provided (e.g., via `alt`, `longdesc`, or in element content).
(b)	Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.
(c)	Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.
(d)	Documents shall be organized so they are readable without requiring an associated style sheet.
(e)	Redundant text links shall be provided for each active region of a server-side image map.
(f)	Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.
(g)	Row and column headers shall be identified for data tables.
(h)	Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.
(i)	Frames shall be titled with text that facilitates frame identification and navigation.
(j)	Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.
(k)	A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.
(l)	When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.
(m)	When a web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with §1194.21(a) through (l).
(n)	When electronic forms are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.
(o)	A method shall be provided that permits users to skip repetitive navigation links.
(p)	When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.

TABLE 9-2 *The 16 Rules of the Section 508 Web Accessibility Standards* ■

alternate; true to its name, you use the `alt` attribute when you want to provide alternate text for a nontext element, such as an image. When a sighted user mouses over the element, the alternate text pops up in a tool-tip window the user can see onscreen. When a seeing-impaired user accesses the page with a screen reader, it speaks the text aloud, thereby providing a way for users with vision impairments to know what is pictured in the image.

When you use the `alt` attribute, the alternate text should be brief. If the textual description is longer than 150 characters, you should use the attribute `longdesc`, which stands for long description.

Using the alt Attribute

Using the `alt` attribute is very straightforward. To provide alternate text for any image onscreen, follow these steps:

1. Use the Notepad to open the file containing the image. In this example, open the *resume.html* page you created in Chapter 6.

2. Click to position your cursor inside the `<img>` tag of the image for which you want to provide alternate text. In this example, position your cursor before the closing bracket of the `<img>` tag that displays your picture onscreen.
3. Using the `alt` attribute, type the alternate text you want the image to have. In this example, type an explanation that identifies who is pictured onscreen:

This is the filename of your image. *Type the alternate text between the quote signs.*

```
<img src="santa.gif" alt="Santa Claus's picture">
```

4. Save the file, and then open it with your Web browser. Use your mouse to hover over the image until the tool-tip window appears. Notice how the alternate text appears in the tool-tip window.

Using the longdesc Attribute

If an image conveys more information than you can describe in 150 characters, you should use the `longdesc` attribute, which tells assistive devices the location of the HTML file that contains the long description. In conjunction with `longdesc`, you should also provide an `alt` attribute, which can identify the topic that the long description describes. The command syntax is as follows:

In this example, the image displays election results that require several paragraphs of text to describe. *The `alt` attribute contains alternate text identifying the topic of the image.*

```
L 9-1    <img src="results.gif" alt="Election Results" longdesc="results.html">
```

The `longdesc` attribute specifies the URL of the file containing the full description of the election results.

Synchronized Alternatives for Multimedia Presentation

Section 508 Web accessibility rule (b) requires that "Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation." In the previous chapter, you learned how to use the XHTML+SMIL timing module to write synchronized text onscreen to caption a video clip from President John F. Kennedy's famous moon challenge speech. If you tried to caption another video following that same technique, you probably found that it can take quite some time to figure out the timing values. To save time, you can use captioning tools that can determine these timings more efficiently.

One of the most popular tools is called **MAGpie**, which you can download from the National Center for Accessible Media (NCAM) at ncam.wgbh.org/webaccess/magpie. The MAG in MAGpie stands for Media Access Generator. Figure 9-1 shows how MAGpie makes it easy to create closed captions for audio tracks and videos recorded for playback by Apple's QuickTime Player, Real Networks' Real Player, and Microsoft's Windows Media Player. MAGpie can also create captions for integration into SMIL presentations. In 2003, MAGpie won an honorable mention for best educational streaming program in *Streaming* magazine's reader's choice awards.

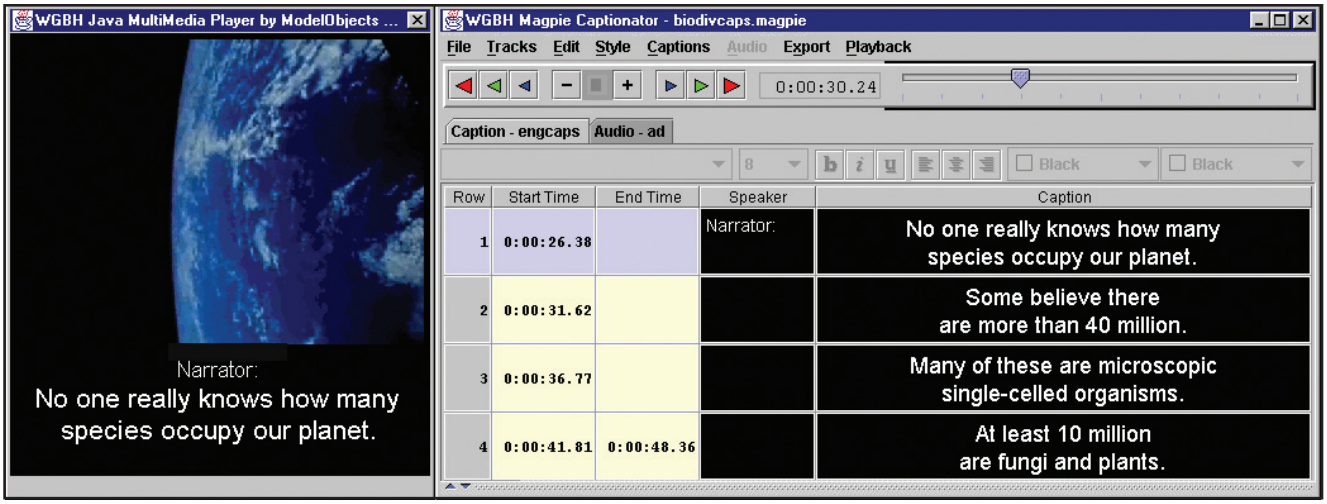

FIGURE 9-1 *MAGpie makes it easy to mark timings and type the captions that will appear onscreen when the clip reaches each time segment. MAGpie was created by the CPB/WGBH National Center for Accessible Media with funding from the Mitsubishi Electric America Foundation and the U.S. Department of Education's National Institute on Disability and Rehabilitation Research.* ■

Conveying Color-Coded Information from Context or Markup

A significant number of users are color blind. About 10 percent of males, for example, are unable to perceive red or green. About half of a percent of females have similar difficulties distinguishing between red and green. That is why rule (c) forbids using color to convey information that cannot be understood in the absence of color. Whenever you are color-coding a chart or a graph, therefore, make sure you provide an alternate way in which someone who is color blind can understand the color-coded information. In the section "Table Row and Column Headers" later in this chapter, for example, you will learn how to create row and column headers in an HTML table. You can use these headers to explain any categorization conveyed by color-coded data cells.

You must also avoid navigational instructions that rely solely on color. Telling the user to press the red button, for example, violates rule (c). To bring such a statement into compliance, you could print the word stop inside the button and tell the user to press the red stop button. Thus, users who cannot see red can identify the button via the word stop.

When printing text on colored backgrounds, you must make sure that your color choices have enough contrast. On the left side of Figure 9-2, you see how people with different kinds of color blindness perceive a color combination which seems to have enough contrast but is lacking in blue hues. On the right side of the figure, you see the same image with more blue hues.

Making Web Pages Readable Without Requiring Style Sheets

The previous chapter touted the fanciful features of style sheets. Note however that Section 508 Web accessibility rule (d) requires that Web pages must be readable without requiring style sheets. Is this a contradiction? It depends on how you are using the style sheet. Rule (d) means that you must not use a style sheet so that it changes the meaning that the page would convey without the style sheet.

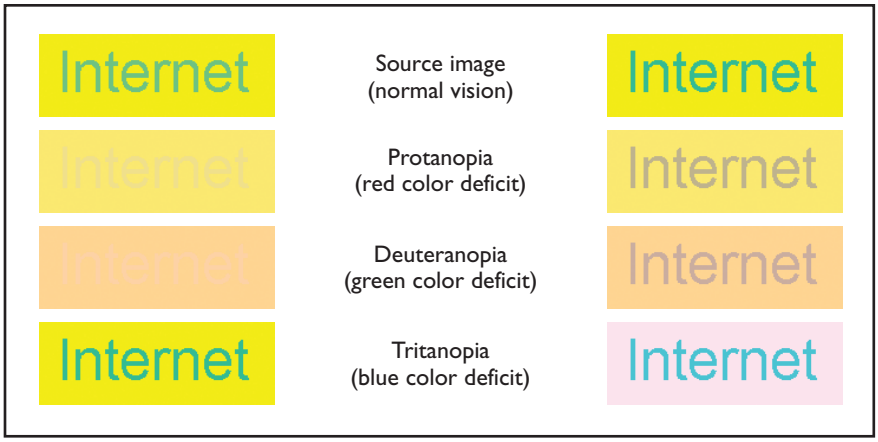

FIGURE 9-2 *Color combinations that seem to have enough contrast for normal vision may cause serious problems for users with certain kinds of color blindness. Notice how the green text in the source image on the left is indecipherable by someone with deuteranopia. The foreground text in the source image on the right contains more blue, thereby creating much better contrast.* ■

Regardless of what style sheets the page may already call upon, the user can always add another style sheet by editing the browser's accessibility settings. Figure 9-3 shows how to do this via the IE browser's Accessibility dialog. Because the style sheet specified in the Accessibility dialog is always last on the cascade, the end user can redefine or override any style on the page. Thus, every user has the right to add a special style sheet to enhance the presentation of any Web page.

Text Links for Active Regions of Server-Side Image Maps

Because the image coordinates in a server-side image map are not processed by the server until the user clicks the mouse, the alternate text attribute does not work with server-side image maps. To provide a textual

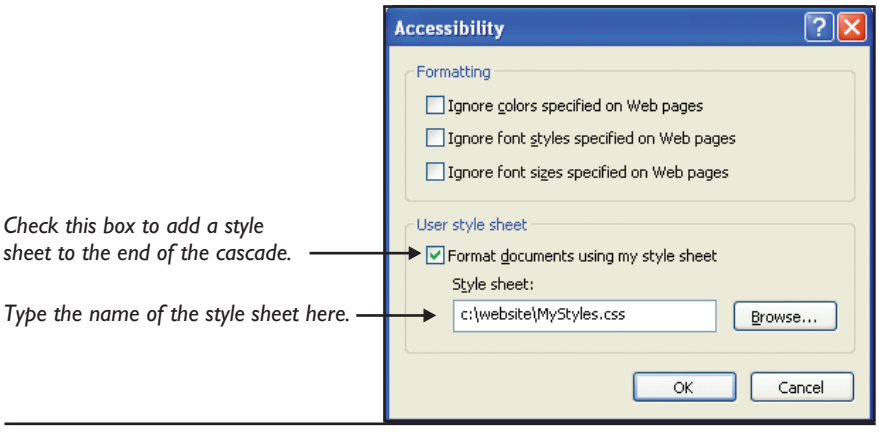

FIGURE 9-3 *In the IE Web browser, the Accessibility dialog enables the end user to add a style sheet to the end of the cascade. To bring up this Accessibility dialog, pull down the IE browser's Tools menu, choose Internet Options, and click the Accessibility button.* ■

alternative to users with special needs, rule (e) of the Section 508 Web accessibility guidelines requires that you provide a text link for each active region of a server-side image map. Depending on the layout of the active regions in the map, printing text links onscreen may or may not work well with your page design. Before you fret about this requirement, read the next section of this chapter, "When to Use Client-Side Versus Server-Side Image Maps," where you will learn that server-side image maps are pretty much a thing of the past.

When to Use Client-Side vs. Server-Side Image Maps

Section 508 Web accessibility rule (f) requires that "Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape." When you studied image maps in Chapter 7, you learned how to create round, rectangular, and polygonal shapes. Anyone who knows their math realizes that because a polygon can have any number of sides with any direction or length, you can define any conceivable shape with a polygon. One could argue, therefore, that there is no situation that requires a server-side image map. Whether or not this is always true, chances are that you will never encounter a situation in which a client-side image map cannot do the job. Rule (f) is saying that if a client-side image map can do it, you are not permitted to use a server-side map instead. In other words, use client-side instead of server-side image maps whenever possible.

To make a client-side image map accessible, you use the `alt` attribute to specify the alternate text for each area in the map. You can also use the `longdesc` attribute for any area that needs further explanation.

Imagine that you want to make accessible the music keyboard image map that you studied in Chapter 7. Figure 9-4 shows the modified code, and Figure 9-5 shows the effect of a user hovering over the first area defined by that code. Screen readers verbalize the alternate text when users with screen readers tab over the image map areas.

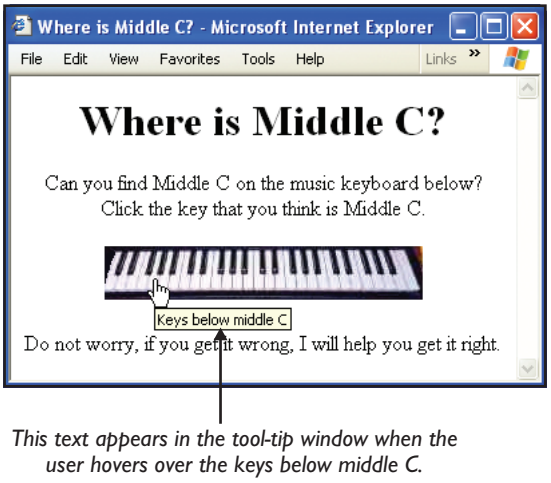

FIGURE 9-5 *When a user hovers over an image map area that has alternate text, the browser displays the text in a tool-tip window. In this example, the user is hovering over the keys below middle C.*

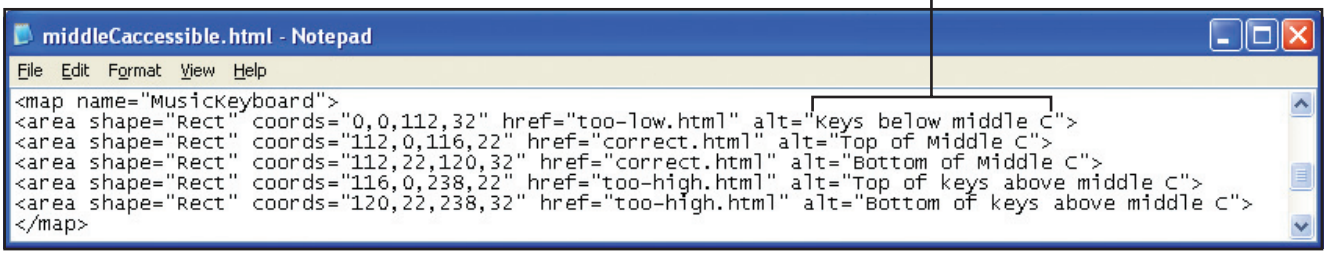

FIGURE 9-4 *You make an image map accessible by providing alternate text for each area in the map. Compare this code to Figure 9-5, which shows a user hovering over the part of the keyboard below middle C.*

Table Row and Column Headers

Section 508 Web accessibility rule (g) requires that data tables must have clearly identified row and column headers. This requirement applies only to tables containing data that, in order to be understood, require that users know what specific row or column they are in. Out on the Web, most of the tables are used for page layout. When you learned how to design Web pages in Chapter 5, you saw an example of a table used purely for layout in Figure 5-14. Tables that are used purely for layout do not need row and column headers.

In Chapter 6, on the other hand, a step-by-step tutorial had you create a table of the world's highest mountains. In order to understand the information in this table, the user must be able to identify the category represented by the row and column of each data cell. Suppose you want to make this table accessible. To define the table headers, you use the HTML table header **<th>** start and **</th>** stop tags. To make the world's highest mountains table use these tags, you can modify the first row of the table. Instead of using the `<td>` and `</td>` tags to create the title cells in the first row, you use the `<th>` start and `</th>` stop tags. These table header tags both create a data cell and define it as a table header. The modified code appears as follows:

The `<th>` start and `</th>` stop tags create a data cell that functions as a table header. This is the table header for the first data column.

L9-2

```
<tr align="center" valign="middle" bgcolor="#CCFFFF">
<th>Mountain</th>
<th>Country</th>
<th>Feet</th>
<th>Meters</th>
</tr>
```

This is the table header for the second data column.

This is the table header for the third data column.

This is the table header for the fourth data column.

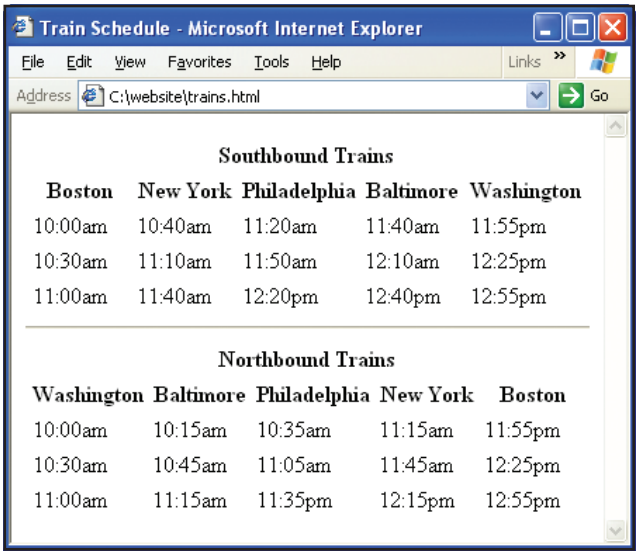

FIGURE 9-6 *In order for assistive devices to make sense of tables that require the user to understand the relationship between multiple headings, you use the id and headers attributes to associate each table cell with its corresponding headings. In this example, the user who is reading 12:20 pm should understand that it indicates the Philadelphia arrival time for a southbound train. See Figure 9-7 for the code that makes it possible for assistive devices to understand this relationship and communicate it to users with special needs.* ■

Related Table Row and Column Headers

Situations can arise in which a data table contains groups of rows or columns that contain related data. The train schedule illustrated in Figure 9-6 has two main groupings: one for northbound and the other for southbound trains. A sighted person can clearly see these groupings onscreen, but imagine the difficulty someone using a screen reader would have trying to follow the information in such a table.

To provide assistive devices with a way to identify the structure of such a table, rule (h) requires that you use HTML markup to associate the data cells with their corresponding header cells. While this may sound complicated, it is easily achieved, thanks to the `id` and `headers` attributes. Whenever you have a table in which the association between the data cells and their corresponding headers is not straightforward, you use the **id attribute** to assign a unique identifier to each table

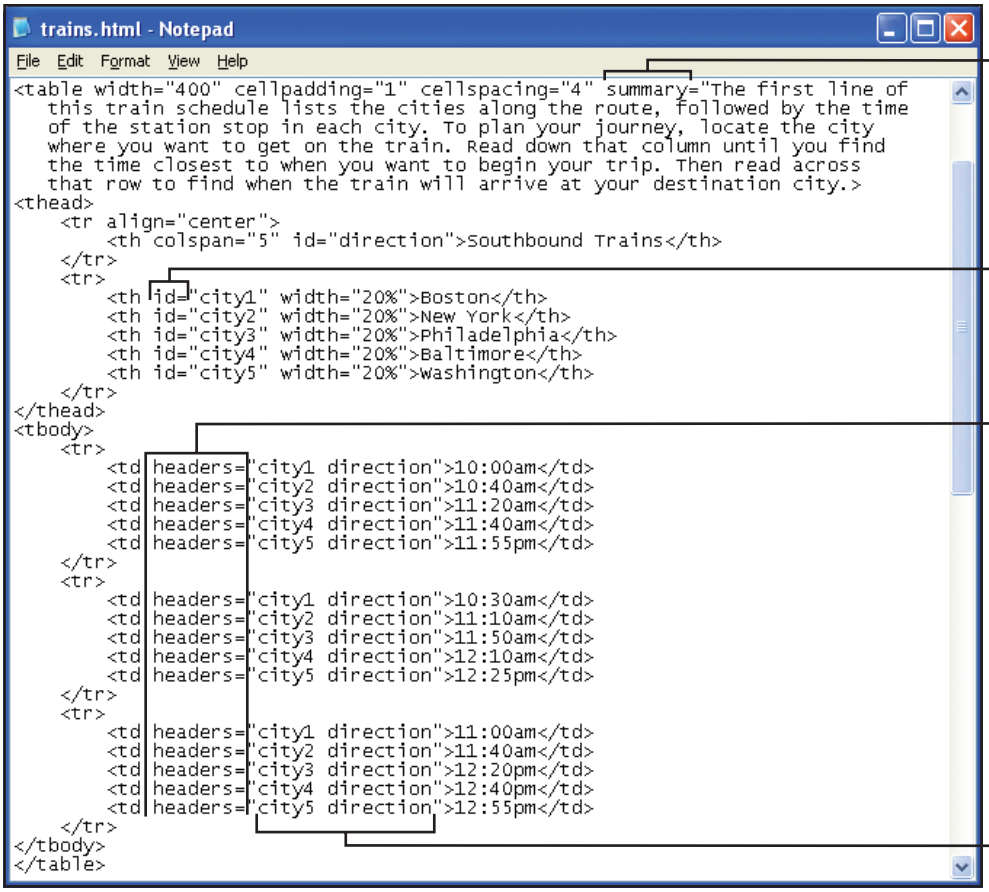

FIGURE 9-7 *Behind the scenes of the train schedule, id and header attributes maintain the association between the train direction and the city. Thus, users with screen readers can work their way down and across the columns, without losing track of these critical row and column relationships.* ■

header. Then you add to each `<td>` tag a **headers attribute** that identifies the header(s) associated with each data cell. To see how this works, study the callouts in Figure 9-7, which displays the code that creates the train schedule illustrated in Figure 9-6.

Frame Titling

Screen readers and other kinds of assistive devices use the text you type in a frame's **title attribute** to identify the frame to users with special needs. Section 508 Web accessibility rule (i) requires that when you create a frameset, you must give each frame a title that identifies the purpose and function of the frame. In addition to the `title` attribute, some assistive devices use the **name attribute** to identify the frames. You should therefore set both the `title` and `name` attributes. Imagine a frameset that has a top frame displaying a banner, a left sidebar containing navigation options, and a main content frame. The following HTML satisfies rule (h) by

using the `title` and `name` attributes to give each frame a title that facilitates frame identification and navigation:

L 9-3

```
<frameset rows="64,*" title="Making Web Sites Accessible">
  <frame title="Top Banner" name="Top Banner" scrolling="no" noresize
target="Sidebar Navigation" src="Banner.html">
  <frameset cols="150,*">
    <frame title="Sidebar Navigation" name="Sidebar Navigation" target="Main Content" src="Navigation.html">
    <frame title="Main Content" name="Main Content" src="Main.html">
  </frameset>
  <noframes>
     <body>
This page uses frames. You can also <a href="noframes.html">use a version of
this page without frames</a>.
     </body>
  </noframes>
</frameset>
```

You should also make sure that each page displayed in a frameset has an appropriately descriptive `<title>` tag in the `<head>` section of the page. Even though these titles do not appear onscreen in the frameset, assistive devices may make use of them. Thus, the `<head>` sections of the three pages in the frameset just described should read as follows:

Main.html: `<title>Main Content</title>`
Banner.html: `<title>Top Banner</title>`
Navigation.html: `<title>Sidebar Navigation</title>`

Avoiding Screen Flicker in the Range of 2 Hz to 55 Hz

Strobing, flashing, blinking, or flickering at a frequency of 2 to 55 times per second can induce seizures in users with certain genetic dispositions. To avoid the possibility of Web sites inducing seizures, Section 508 Web accessibility rule (j) forbids flicker in the range of 2 Hz to 55 Hz. The term Hz stands for **hertz**, which means vibrations per second.

Animations that blink are not particularly desirable at a Web site. Most users quickly tire of the repetition, and people who are easily distracted have a hard time concentrating on a text when something is moving in their field of vision onscreen. To learn more about the risk of inducing seizures from flicker at a Web site, go to usability.gov/web_508/tut-j.html.

Providing Text-Only Page Alternatives for Noncompliant Pages

Rule (k) requires that you must provide a text-only page to substitute for any Web page that you cannot make comply with all of the other rules in the Section 508 Web accessibility guidelines. Furthermore, rule (k) requires that you must update the text any time a change is made to the page for which it substitutes.

Please do not assume, however, that you can use rule (k) as an easy way out any time you cannot meet the rest of the guidelines. Rule (k) is a last resort. Whenever you find yourself resorting to rule (k), you should rethink the design of your page and make an honest effort to make the document accessible.

Describing Scripts with Functional Text

Section 508 Web accessibility rule (l) addresses the problem of making scripts accessible. Scripts can make things happen onscreen that assistive devices cannot interpret. Whenever you have a script doing something that displays content or provides interface elements onscreen, rule (l) requires that you identify this information with functional text that can be read by assistive technology. The HTML codes that you use to provide this text are the **<noscript>** start and **</noscript>** stop tags. You put these tags immediately after the `</script>` end tag of the script that you are describing. The syntax is

L 9-4

```
<script language="javascript">
       //The script goes here.
</script>
<noscript>
      <p>Describe what the script does here, or provide a link to an accessible
      document that provides the data generated by the script.</p>
</noscript>
```

The `<noscript>` start tag must appear immediately after the `</script>` end tag of the script it is describing.

When creating user interface elements with JavaScript, you need to make sure that users can operate them without a mouse, which is easy to test: Put your mouse aside, and make sure you can operate the controls via the computer keyboard.

Avoid using the `onDblClick` event handler, for which there is no keyboard equivalent. If you use the `onMouseDown` event handler, provide an `onKeyDown` handler so the mouse down event has a keyboard equivalent. In like manner, pair `onMouseUp` with `onKeyUp`.

Avoid using popup windows, which can cause users with assistive technology to lose track of where they are.

Applet and Plug-In Accessibility

In Chapter 2, you learned how applets and plug-ins can extend the multimedia capabilities of a Web page. In order for such a page to be accessible, the applets and plug-ins must follow the same accessibility guidelines as the other Web page elements onscreen. That is why Section 508 Web accessibility rule (m) requires that whenever an applet, a plug-in, or another application is called upon to interpret page content on the user's computer, the page must provide a link to a plug-in or an applet that complies with rules (a) through (l).

You should keep in mind three key points with regard to applet and plug-in accessibility. First, users with special needs must be able to navigate the page without a mouse. You can test this by putting your mouse aside and using the keyboard to operate the applet and navigate the page. Second, the user should be able to move from element to element onscreen. Once again, you can test this by putting your mouse aside and pressing the TAB key to make sure it lets the user move from item to item onscreen. Finally, you must remember that for every graphical element that conveys meaning or navigation in an applet or a plug-in, a textual equivalent that can be understood by assistive technologies must also be present.

Form Elements, Directions, and Cues

In Chapter 7, you learned how HTML forms enable the Web author to interact with and receive information from users at a Web site. Rule (n) requires that any time users complete a form, you must present the form in such a way that users with assistive technology can access the field elements, read all of the directions, understand the labeling, and follow the cues to complete and submit the form.

To clarify which instructions and labels go with which form elements onscreen, the W3C invented the **<label>** start and **</label>** stop tags. The `<label>` tag has a **for attribute** that you use to identify the ID of the `<input>` element with which the label is associated. Assistive devices use these attributes to determine which labels go with which form elements and cue the user accordingly. Consider, for example, the form in the *subscribe.html* page you created in Chapter 7. To make that form accessible, you can modify the code by using the `<label>` tag's `for` attribute to associate each label with the ID of its corresponding input. Study the callouts in the following example to see how the `<label>` tag's `for` attribute identifies which input has that label:

The `for="Name"` attribute associates this label with the input field that has `id="Name"`.

L 9-5

```
<p><label for="Name">What is your name?</label><br>
<input id="Name" type="text" name="Name" size="50" maxlength="150"
       style="border-style: inset; border-width: 4">
</p>
<p><label for="Email">What is your e-mail address?</label><br>
<input id="Email" type="text" name="Email" size="50" maxlength="150"
       style="border-style: inset; border-width: 4">
</p>
```

The `for="Email"` attribute associates this label with the input field that has `id="Email"`.

Many users who cannot use the mouse press the TAB key instead to move from field to field within a form. When you create a form, therefore, you should test it by pressing the TAB key repeatedly and observe whether the tab order is logical. If the focus moves illogically from field to field, you can clarify the tab order via the `<input>` field's `tabindex` attribute. To clarify the tab order on the *subscribe.html* page, for example, you would modify the input controls as follows:

The `tabindex` attribute establishes the tab order. You do not need to use this attribute, however, unless the default tab order is illogical.

L 9-6

```
<input tabindex="1" id="Name" type="text" name="Name" size="50" maxlength="150"
       style="border-style: inset; border-width: 4">
<input tabindex="2" id="Email" type="text" name="Email" size="50" maxlength="150"
       style="border-style: inset; border-width: 4">
<input tabindex="3" type="radio" name="Frequency" value="daily" checked>Daily
<input tabindex="4" type="radio" name="Frequency" value="weekly">Weekly
<input tabindex="5" type="radio" name="Frequency" value="monthly">Monthly
<input tabindex="6" type="submit" value="Subscribe">
<input tabindex="7"
       type="reset">
```

A final point about making forms accessible is that you need to avoid the temptation to use JavaScript event handlers to make the ENTER key submit form data without requiring the user to click a Submit button. Users who cannot use the mouse press the ENTER key to input or select information in a form field. That is why these users need a Submit button to press when they are done filling out the form. In order to make forms accessible, therefore, you must provide a Submit button for every form onscreen. Otherwise, many users with assistive devices have no way of submitting the information when they are finished filling out the form.

Skip Navigation

Many Web pages have navigational options across the top and down the left side of the page, with the main content occupying the rest of the page. When you create this kind of a layout with an HTML table, the top navigational options go into the first table row, and the sidebar navigation goes into the first column of the next row onscreen. Only after the sidebar gets populated does the main content flow onto the rest of the screen. Imagine someone accessing such a page with a screen reader. Without a way to skip these navigation options, the user has to tab through all of these navigation links before getting to the main content on the page.

To solve this problem, Section 508 Web accessibility rule (o) requires that you provide a way for the user to skip over these kinds of repetitive navigation links. Most sites do so by placing a **skip navigation link** at the start of such a page. A skip navigation link is a hyperlink that, when clicked, takes the user to a place on the page that is just past the navigation options that typically appear at the top or on the left of the screen. A user with a screen reader can follow the skip navigation link to jump to the main content of the page, just like a sighted user can glance past the navigation link to read the main content onscreen.

To create a skip navigation link, insert the following code at or near the top of the body of the page. Notice how you can make the link be invisible, if you do not want it appearing on the page of a sighted user:

L 9-7
```
<a href="#main content" style="display:none">Skip over navigation</a>
```

Setting the display style to "none" prevents this link from displaying onscreen. Assistive devices, such as screen readers, use this link to provide users with a way to skip the navigation.

After you make the skip navigation link, you need to create the target of that link. The target of a skip navigation link is a named anchor that you type at the spot on the page where the main content begins. The HTML code for such a link appears as follows:

L 9-8
```
<a name="main content" id="main content"></a>
```

In these examples, the name "main content" is arbitrary. You can make that name anything you want. I named the anchor "main content" because it marks the beginning of the main content on the page.

Timed Responses

Web designers generally frown upon timed responses because you normally want to give the user as much time as needed to respond. For security reasons, however, some sites may require users to interact every so often, otherwise the session times out and the user must log in again to start another session. Such a site should make it possible for a user with special needs to request a longer timeout interval. That is why Section 508 Web accessibility rule (p) requires that whenever a Web page requires the user to respond within a certain amount of time, the user must be informed and given an opportunity to request more time.

Making Applets, Helpers, and Plug-Ins Accessible

In Chapter 2, you learned that whenever a browser encounters a multimedia resource that HTML cannot handle, the browser calls upon the appropriate plug-in or helper application to play or display the content onscreen. So much multimedia content is on the Web that almost every site includes resources that HTML cannot handle on its own. That is why this part of the book takes a closer look at how you go about making multimedia content accessible.

Flash Accessibility

Flash is the most popular multimedia plug-in on the Web. A large percentage of the Web's multimedia content, therefore, gets handled by the Flash player. Its parent company, Macromedia, has worked in partnership with Microsoft to make the latest version of the Flash player conform to **Microsoft Active Accessibility (MSAA)**, which is an application programming interface (API) that helps Windows applications interoperate with assistive technology. Flash authors who follow the guidelines can make accessible to screen readers text elements, buttons, input fields, and movie clips. Scalable graphics enable users to zoom in on Flash content, which enables visually impaired users to see small text or graphics. Content controls enable users to stop, fast-forward, rewind, and pause playback. Mouse-free navigation enables assistive technology to provide keyboard controls over all these functions. Synchronized audio tracks provide descriptive audio for users who need it. Customized color swatches enable developers to customize palettes for users who are color blind. Guidelines for making Flash movies comply with Section 508 are at www.macromedia.com/macromedia/accessibility.

Flash movies are inherently more complex than HTML documents. No matter how carefully you follow Macromedia's guidelines, you need to remember that some users will require simpler alternatives. Happily, the W3C kept this in mind when they invented the `<object>` start and `</object>` stop tags, which you use to put Flash movies (and other kinds of multimedia content) on a Web page. Consider the following `<object>` tag, which plays a Flash movie called *DNAsynthesis.swf*.

If the user has the Flash player installed, the movie appears onscreen, and the browser does not display the alternate text. If the user does not have Flash installed, however, the movie cannot play, so the browser continues on to display the alternate text onscreen:

L 9-9

```
<object title="Core DNA Synthesis"
classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab
#version=6,0,0,0" width="100%" height="100%" align="">
     <param name=movie value="coreDNAsynthesis.swf">
     <param name=quality value=high>
     <param name=salign value=LT>
     <param name=bgcolor value=#FFFFFF>
This movie consists of an interactive demonstration of core DNA synthesis.
There is a <a href="coreDNAsynthesis.html">textual description</a> for users
who do not have Flash installed.
</object>
```

You put the alternate text before the `</object>` end tag. The browser displays this text if the user does not have Flash installed.

In their wisdom, the W3C further provided a mechanism whereby you can specify several alternative representations of an object. The browser works down the tree of these objects until it finds one it can play. Objects further down the tree get skipped. The following example shows how this works. Notice how (1) the content gets played by Flash, if the user has Flash installed. If not and the user has an MPEG movie player, (2) the content renders as an MPEG movie. If not, (3) the content appears in an animated GIF image. If all else fails, (4) alternate text appears onscreen.

L 9-10

```
                    <!-- First, try the Flash movie -->
<object title="Core DNA Synthesis"
classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab
#version=6,0,0,0" width="100%" height="100%" align="">
<param name=movie value="coreDNAsynthesis.swf"><param name=quality value=high>
<param name=salign value=LT><param name=bgcolor value=#FFFFFF>
                    <!-- Else, try the MPEG video -->
   <object data=" coreDNAsynthesis.mpeg" type="application/mpeg">
                    <!-- Else, try the GIF animation -->
      <object data=" coreDNAsynthesis.gif" type="image/gif">
                    <!-- Else render the text -->
This movie consists of an interactive demonstration of core DNA synthesis.
There is a <a href="coreDNAsynthesis.html">textual description</a> for users
who do not have Flash installed.
      </object>
   </object>
</object>
```

When objects are nested, the browser works its way down through them, executing the first object it can handle and skipping the rest.

PDF Accessibility

Partly due to its televised mass-market ad campaign, Adobe's Portable Document Format (PDF) is the most popular file format for sharing non-

HTML documents over the Web. So flexible is this format that the product's family name Adobe Acrobat is very befitting this product. For every major word processor, for example, an Adobe Acrobat plug-in enables you to "print" to the Adobe distiller, which creates the PDF files. Thus, anyone who can use a word processor can create a PDF file. Therein lies the problem of accessibility: Can anyone who can word process make the file accessible?

The key to making PDF files accessible is to structure the document properly with your word processor prior to converting it into a PDF file. Type meaningful headings and subheadings at the beginning of each section and subsection of the document. Use the word processor's Style menu to give each heading or subheading the appropriate heading style. Microsoft Word, for example, has heading styles from Heading1 through Heading 6. These are equivalent to the HTML heading tags `<h1>` through `<h6>`. When you convert an MS Word document into a PDF file, Adobe Acrobat uses these headings to create structural tags in the PDF file. Screen readers and other assistive devices use the tagged PDF document to provide navigational options that make it easy for users with special needs to go to different sections in the document. If you do not use the heading styles, on the other hand, the PDF document will not contain these tags, and the document will not be accessible. Therefore, you should form the habit of always using heading styles to mark the sections and subsections of a word-processed document.

To be considered accessible, images in word-processed documents must have alternate text descriptions. To provide alternate text for an image in an MS Word document, you (1) right-click the image to pop out the quick menu, (2) choose Format Picture to display the Format Picture dialog, and (3) click the Web tab to display the Alternative text box. Figure 9-8 shows how you type the alternate text. Later, when you convert the document to a PDF file, Adobe Acrobat will make this alternate text available to users with screen readers and other kinds of assistive technology.

By default, the accessibility options in Adobe Acrobat are turned on. These include the options to embed tags in PDF and to create PDF bookmarks for the heading styles that mark the structure of your document. After you create the PDF file, it opens in Acrobat. To test the file for accessibility, run Acrobat's built-in Accessibility Checker, which checks the document for Section 508 compatibility and advises you of any problems. For more on creating accessible PDFs, go to www.adobe.com/products/acrobat/solutionsacc.html.

Multimedia Accessibility Showcase

The National Center for Accessible Media (NCAM) at the WGBH Boston public television station is an excellent source for learning about best practices of accessibility in all things media. By following the links to the rich media accessibility showcase at ncam.wgbh.org/richmedia, you can read about the latest innovations and best practices. The showcase contains examples of accessible media delivered in the following formats: Director, Flash, PDF, QuickTime, Real Media, Scalable Vector Graphics, SMIL, Windows Media, and XHTML+SMIL. Accessibility strategies include

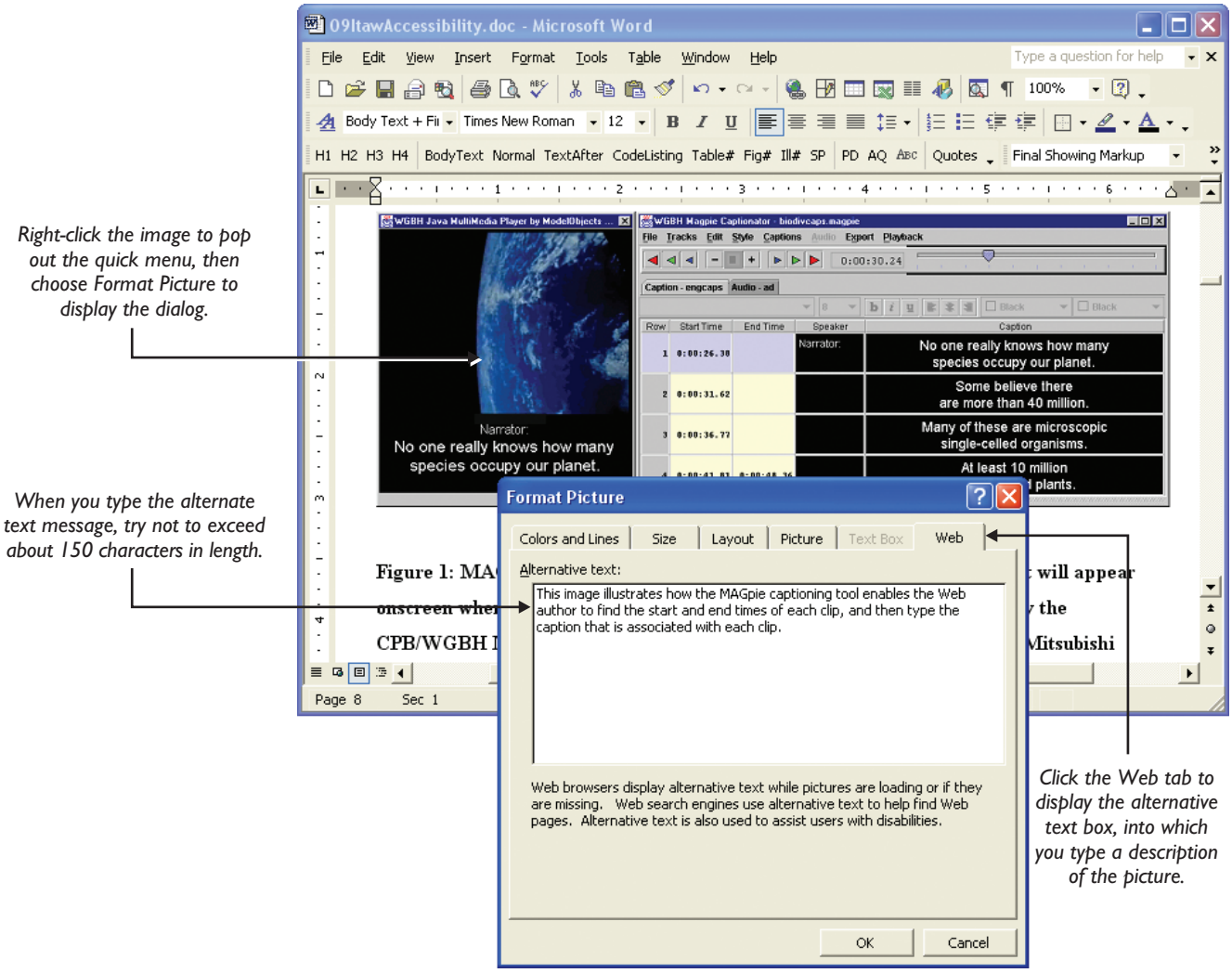

FIGURE 9-8 *Before you convert an MS Word document into a PDF file, you should provide alternative text for each image in the document. In this example, I am typing alternative text that explains the MAGpie illustration. The PDF version of this document makes this alternative text available to screen readers and other kinds of assistive devices.* ■

assistive devices, audio description, captions, extended audio description, keyboard access, self-voicing, and text transcript. You can search the showcase by format and strategy to see or hear examples of all these combinations.

Designing Style Sheets for Accessibility

In Chapter 8, you learned that instead of hard-coding presentation elements into the HTML of a Web page, it is better to separate content from style by assigning the HTML elements to style classes. At runtime, the browser looks to the style sheet for the rules that are associated with these classes. Thus, the style classes determine the look and feel of the HTML elements onscreen.

Separating content from style in this manner has an important advantage for users with special needs. If the style sheets you provide do not

handle the content appropriate to the user's needs, the user can use the browser's accessibility controls to add another style sheet onto the cascade. By being last on the cascade, this style sheet can redefine any of the style classes that the user needs to have interpreted differently.

Although users with special needs have this ability to add a style sheet to the cascade, you should always try your best when authoring Web pages to create styles that are maximally accessible. The following sections recommend some best practices for creating accessible styles.

Font Selection and Spacing

Most browsers default to the Times Roman font. If you do not specify a different font, Times Roman is what you get. Studies have shown that the Times Roman font is one of the best choices if you are creating a document for printing on paper. The serifs, which are the angled decorations on the endpoints of the characters, enhance the flow of the text on the printed page. It is ironic that on the computer screen the serifs, which make Times Roman look cool on paper, can interfere with readability onscreen. Fonts without serifs, such as the Arial font, work better on the computer screen. This may be because the computer screen has far less resolution than the printed page. This book, for example, is printed at a resolution of 4,800 dots per inch (DPI). The typical computer screen, on the other hand, has a resolution of about 50 DPI. There is not enough resolution at 50 DPI to give the serifs the smooth flow they have on paper. Instead, serifs look jagged onscreen.

In an article published in the *International Journal of Human-Computer Studies*, Bernard et al. (2003) conducted a controlled experiment that compared Times New Roman and Arial typefaces in 10- and 12-point, dot-matrix and anti-aliased (i.e., edge-softened) versions. In terms of overall readability and legibility, the 12-point Arial dot matrix typeface was preferred to all of the other typefaces. The reason for this may be because Arial simply looks cleaner and neater onscreen. On paper, however, Times New Roman rules. In 2004, for example, the U.S. State Department issued an edict declaring Times New Roman as the font for all diplomatic notes (*ABC News Online*, January 30, 2004).

Besides choosing Arial to use on a Web page, you want the text onscreen to appear large enough for the user to read. Text that is large enough for one user, however, may be too small for another. Instead of hard-coding the fonts to a specific size, therefore, you should use relative sizing. The W3C has created two ways for you to do this. First, you can use the adjectives small, smaller, x-small, xx-small, medium, large, larger, x-large, and xx-large. Or you can specify the font size in a unit called an **em**, which is pronounced like the letter *m*. The setting 3em, for example, is three times the width of the letter *m*. Before printing such a font onscreen, the browser looks to the default text size that the user has specified in the browser's View menu. Thus, the font scales automatically to the visual needs of the user. Figure 9-9 provides examples of font size settings that scale in this manner, as well as examples that fail to scale. To create pages that are maximally acceptable, you should choose font settings that scale.

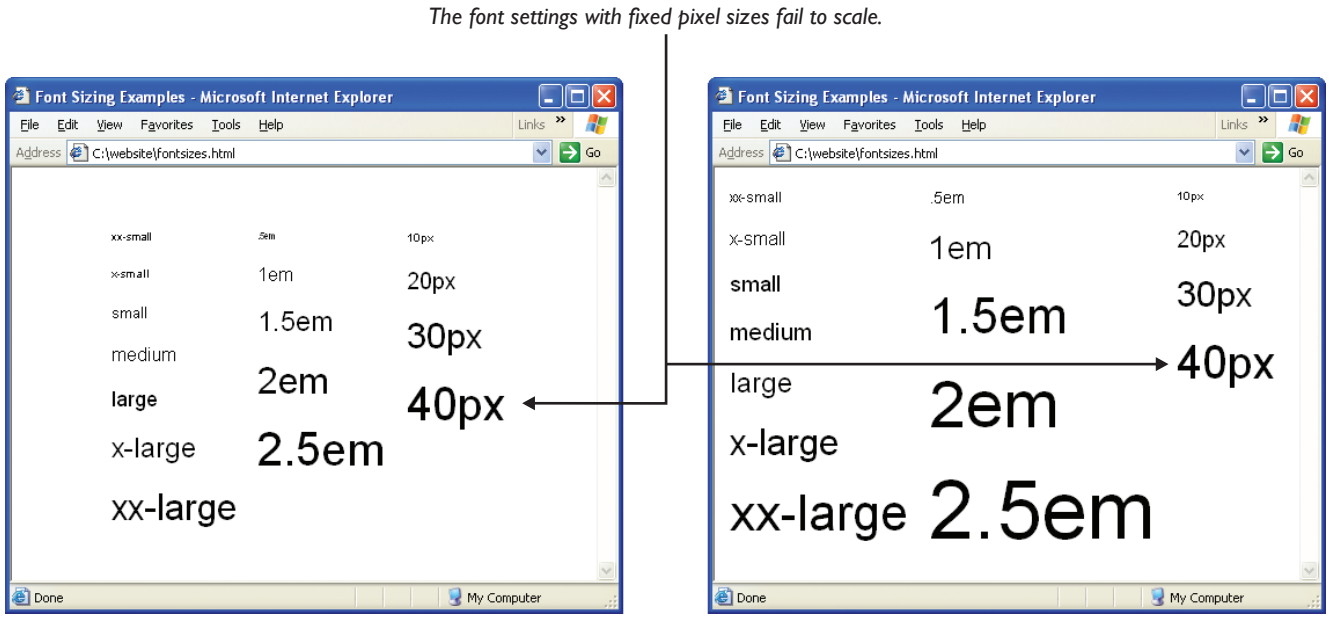

FIGURE 9-9 *On the left are font size examples displayed at the browser's default setting. On the right, the same examples appear with the browser's View | text-size setting configured to increase the font size. Notice how the font settings with relative values scale, while the settings that have fixed pixel sizes fail to scale, thereby defeating the browser's text-size feature.* ■

Color and Contrast

Without sufficient contrast between foreground text and background colors, users have trouble reading Web content, no matter how you style the font. That is why you must always use colors that are high in contrast. Figure 9-10 illustrates recommended style sheet settings that create good contrast onscreen. These examples adhere to the color contrast algorithm that the W3C recommends for determining whether there is enough contrast. According to this algorithm, you determine color brightness by the formula:

((Red value × 299) + (Green value × 587) + (Blue value × 114)) / 1000

The difference between the brightness of the text and background colors must be greater than 125. If they are, you proceed to determine color difference by the formula:

(maximum (Red value 1, Red value 2) – minimum (Red value 1, Red value 2)) +
(maximum (Green value 1, Green value 2) – minimum (Green value 1, Green value 2)) +
(maximum (Blue value 1, Blue value 2) – minimum (Blue value 1, Blue value 2))

If the color difference is greater than 500, the colors are sufficiently high in contrast. A tool you can use to perform these calculations is at www.juicystudio.com/services/colourcontrast.asp. For more on the W3C color contrast algorithm, go to www.w3.org/TR/AERT#color-contrast.

FIGURE 9-10 *These style sheet settings satisfy the W3C recommendation that the difference between the brightness of the text and background colors must be greater than 125 and the color difference must be more than 500.* ■

Try This!

Web Page Color Blindness Simulator

Vischeck is a tool that simulates human vision. The simulation models three kinds of color blindness, namely, protonopia, a red color deficit; deuteranopia, a green color deficit; and tritanopia, a very rare blue color deficit. To run the Vischeck simulation and see how your Web page will appear to users with protonopia, deuteranopia, or tritanopia, follow these steps:

1. Go to www.vischeck.com, follow the link to Vischeck, and choose the option to Run Web pages.
2. Vischeck asks you to select the type of color vision you want to simulate. The choices are deuteranope, protanope, and tritanope. In this example, click deuteranope.
3. Vischeck will prompt you to type the URL of a Web page. In this example, type www.google.com.
4. Click the Run Vischeck button. You will get a message telling you to wait while Vischeck processes the page. Wait patiently while Vischeck prepares the results.
5. Vischeck will display links to the simulated page and its original version. In this example, follow the link to the Deuteranope simulation to see how the page appears to someone who has a green color deficit.
6. Repeat this process for the other two forms of color blindness, namely, protanope and tritanope. You will probably observe some striking differences. Figure 9-11, for example, shows how the Smithsonian home page appears through the eyes of a user with deuteranope. Notice the lack of greens as compared to Figure 9-12, which pictures the original page in full color.

FIGURE 9-11 *The way the Smithsonian home page appears through the eyes of a user with deuteranopia, which is a green color deficit. Compare this to Figure 9-12, which displays the original page in full color.* ■

FIGURE 9-12 *This is the unaltered, full-color version of the Smithsonian home page. Notice especially the green colors. Compare this full-color version to Figure 9-11, which pictures the same page viewed through the eyes of a user with deuteranopia.* ■

Layering and CSS Page Layout

Layout is one of the most important issues in making Web pages accessible. At the moment, most of the Web's pages use HTML tables to lay out content onscreen. This works fine for sighted users, who can quickly glance from place to place onscreen. Users with screen readers and other kinds of assistive devices, however, run into trouble because the order of the content in the HTML file is not necessarily the order in which the user wants to read the material.

Linearization is the process of thinking of a Web page in the order in which the elements occur in the HTML file. Earlier in this chapter, you learned how users with special needs have difficulty with the order in which Web page elements are presented in the most common kind of navigational layout, in which the navigation options appear at the top and down the left-hand column of the screen. Consider how such a design linearizes when created with tables: The navigational links come first in the file, followed by the page content that the user would most likely want to read first. Section 508 Web accessibility rule (o) addresses this problem by requiring that Web pages have a skip navigation link if the document begins with these kinds of navigational options. There is a more elegant way to linearize content on a Web page, however.

In the absolute positioning section of the previous chapter, you learned how the z-index style sheet property enables you to specify the order in which the browser prints layers onscreen. Imagine using the z-index property to tell the browser the order in which to print the main content layer, the left sidebar, and the top banner information onscreen. Because the z-index property determines the display order, you can put the most important layer first in the file, thereby enabling screen readers and other assistive devices to receive the main content first. To provide a way to divide the content of a page in this manner, the W3C invented the ** ** start and ** ** stop tags. `<div>` is an abbreviation for division. You use the `<div>` tag to create structural divisions onscreen.

Figure 9-13 shows a page that has three divisions onscreen: a top banner, a left sidebar, and a main content layer. When created with tables, this layout positions the content 190 lines below the start of the file. If you use `<div>` tags to lay out this page with absolute positioning, on the other hand, you can put the content at the very beginning of the document's

FIGURE 9-13 *HTML tables create the layout of this Web page. Because table-driven layout requires that the banner and sidebar precede the main content in this HTML file, screen readers must wade past more than 190 lines of code to get to the main content on this page.* ■

body as shown in Figure 9-14. For sighted users, the page appears onscreen as illustrated in Figure 9-13. Users with special needs, on the other hand, get the main content first. The only problem is that some browsers do not yet properly support absolute positioning. Because of this uneven support for absolute positioning, you cannot rely on it for public use. That is why most sites continue to use tables to layout the pages. In future years, when CSS rolls out more broadly, absolute positioning may become the preferred way of laying out Web pages. If that happens, absolute positioning will obviate the need for rule (o), which requires that pages like this one must have a skip navigation link.

Tools for Assessing Web Site Accessibility

This chapter has presented many rules and suggestions for ways in which you can increase the accessibility of a Web site. You can spend many hours working to meet these guidelines. When do you know, however, that you have done enough for the site to be considered compliant?

Happily, you can use tools to determine the extent to which a Web site complies with the accessibility guidelines. These tools can analyze the code of any Web page, report specific rules and guidelines that the page may be violating, and suggest improvements you can make to bring the page into compliance.

Absolute positioning enables you to put the main content first in the document's body, even though the banner and the sidebar appear first onscreen.

These three style classes use absolute positioning to create the layout onscreen.

cssLayout.html - Notepad

File Edit Format View Help

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head><title>CSS Layout Example</title></head>
<style>
HTML{margin:0; padding:0}
BODY{margin:0; padding:0}
DIV.TopBanner{position:absolute; top:0; left:0; padding:0; background-color:white; z-index:10}
DIV.LeftSidebar{position:absolute; top:138px; left:0; background-color:white; z-index:11}
DIV.MainContent{position:absolute; top:138px; left:212px; width: 390px; font-family:Arial; z-index: 12}
</style>
<body>
<div class="MainContent">
<p style="font-size:1.5em; font-weight:bold">Welcome</p>
<P>
This Web site is brought to you from the Library of Congress in Washington, D.C., the largest
library in the world and the nation's library.
</P>
<P><IMG SRC="/assets/about/circleright.jpg" WIDTH="130" HEIGHT="132" ALIGN="right" HSPACE="15">
We hope you will find this Web site entertaining and fun to use. And, of course, we hope you will
learn something from it. The site was designed especially with young people in mind, but there are
great stories for people of all ages, and we hope children and their families will want to explore
this site together.
</P>
<P>
Here, you can discover what Abraham Lincoln had in his pockets on the night he was assassinated.
(You will be surprised.) Or you can read about other &quot;<a href="/cgi-bin/page.cgi/aa">Amazing
Americans</a>&quot; such as Buffalo Bill Cody and his &quot;Wild West&quot; show; the heroism of
Harriet Tubman, who helped many slaves escape bondage; the music of jazz great Duke Ellington; or
the inventions of Thomas Edison. (You will even be able to see his first motion picture!)
</P>
```

FIGURE 9-14 *This code uses absolute positioning to create the page layout illustrated in Figure 9-13. The cascading style sheet defines three classes of divisions entitled TopBanner, LeftSidebar, and MainContent, respectively. In the body of the page, tags use the style classes to position the divisions onscreen. Because the style classes do the positioning, the file can contain the content in the optimal reading order for users with special needs. In the meantime, browsers continue to display the page as illustrated in Figure 9-13.* ■

Bobby

Created by the Center for Applied Special Technology (CAST), **Bobby** is a comprehensive Web accessibility tool that can analyze a single page or an entire Web site. Recently acquired by Watchfire Corporation, Bobby exposes barriers to accessibility, makes recommendations for necessary repairs, and encourages compliance with existing guidelines. For a demonstration of Bobby in action, go to bobby.watchfire.com and follow the onscreen instructions to enter the URL of a Web page you would like Bobby to analyze. The options onscreen let you choose whether to test for compliance with the U.S. Section 508 guidelines or the W3C's Web Content Accessibility Guidelines.

The free version of Bobby lets you evaluate only one page at a time, and it limits you to check no more than one Web page per minute. From Watchfire, you can purchase a copy of Bobby to run on your own server, where all of the Web developers at your site can evaluate pages as often as they want. Web pages that pass the Bobby test are entitled to display the appropriate Bobby conformance logo. Table 9-3 displays the Bobby conformance logos and explains what they signify.

LIFT

Produced by UsableNet and available at www.usablenet.com, **LIFT** is a suite of products that can test, monitor, report, and repair Web accessibility problems. Dreamweaver and FrontPage versions help authors create Web pages that are Section 508 or WCAG compliant. To try these tools before you buy, follow the links to request a demo at www.usablenet.com, where you can get a free accessibility test of the Web page of your choice.

WebKing

Produced by Parasoft, **WebKing** is a Web verification tool that integrates with IBM's WebSphere Studio Application Developer. WebKing performs (1) static analysis, (2) functional testing, and (3) load testing. As part of static analysis, WebKing checks to make sure your files do not contain any broken links or navigational problems. Along the way, WebKing checks

Logo	What the Logo Means
	The Web page passes at least the WCAG Priority 1 checkpoints.
	The Web page contains no Section 508 accessibility errors.
	Conformance Level A: This page satisfies all WCAG Priority 1 checkpoints.
	Conformance Level Double-A: This page satisfies all WCAG Priority 1 and 2 checkpoints.
	Conformance Level Triple-A: This page satisfies all WCAG Priority 1, 2, and 3 checkpoints.

TABLE 9-3 *Bobby Conformance Logos* ■

your HTML and CSS code for Section 508 and WCAG accessibility. During the functional analysis phase, WebKing follows the user click paths to make sure they execute correctly. Under load testing, WebKing simulates virtual users and verifies the number of simultaneous users that the site can sustain under different kinds of usage patterns.

You can get a demonstration and download free evaluation software by following the link to WebKing at www.parasoft.com.

STEP508

Created for the federal government by the WGBH National Center for Accessible Media, STEP508 stands for **Simple Tool for Error Prioritization (STEP)** for Section 508 compliance. The STEP508 tool compares and analyzes the output of compliance tools, including Bobby, LIFT, and WebKing. Based on this analysis, STEP508 determines the severity of the errors and prioritizes the repairs needed to bring a Web site into compliance with Section 508. After computing a baseline compliance score, STEP508 provides a metric to track progress in improving the site's accessibility over time. Figure 9-15 shows how you can download STEP508 from the www.section508.gov Web site.

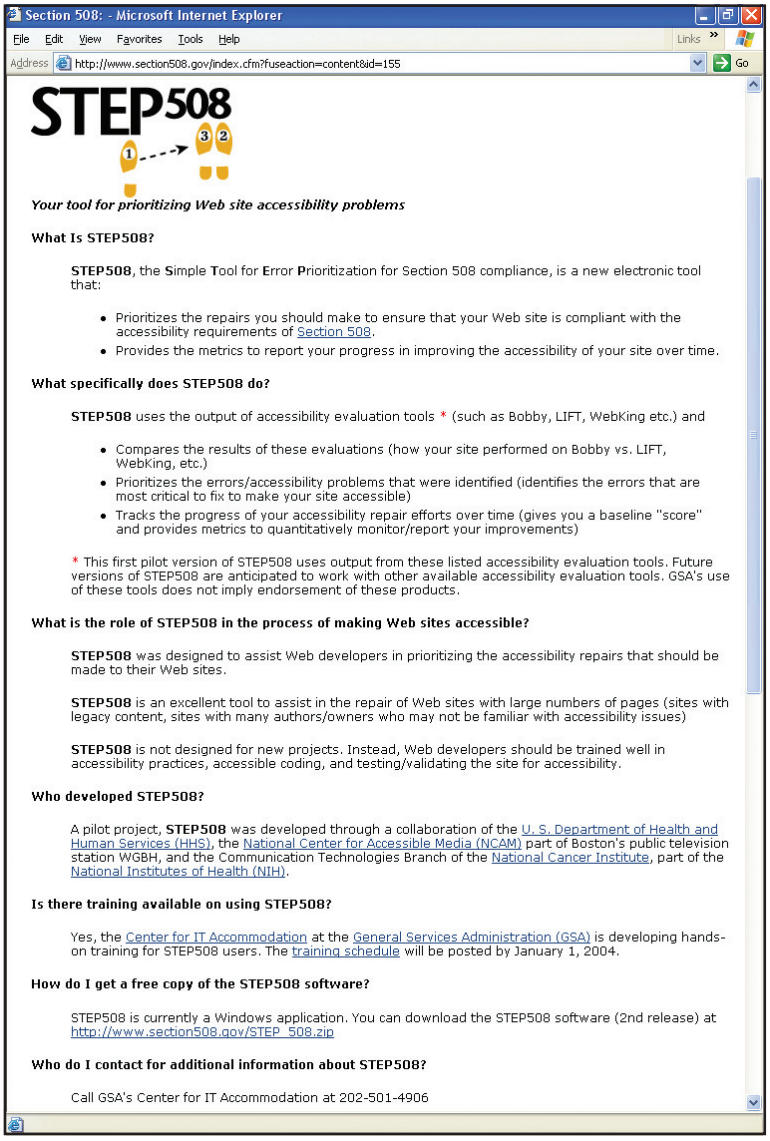

FIGURE 9-15 *STEP508 is a tool for prioritizing the repairs needed to bring a Web site into compliance with Section 508 Web accessibility guidelines. For the latest version, follow the link to STEP508 at www.section508.gov.* ■

Learning More about Web Accessibility

If you would like to learn more about Web accessibility, I recommend an excellent book by Dr. John M. Slatin and Sharron Bush. Entitled *Maximum Accessibility*, this book presents, discusses, and demonstrates issues and techniques for making Web sites accessible. Published by Addison-Wesley, *Maximum Accessibility* does a very good job of explaining the relationship between the Section 508 rules and the W3C's Web Content Accessibility Guidelines.

An excellent online source of information regarding Web accessibility is the National Center for Accessible Media (NCAM) site at ncam.wgbh.org. Another good source is the Section 508 site at www.section508.gov. A Section 508 tutorial is at usability.gov/web_508/tutorial.html.

For the latest on Web accessibility, go to the W3C's Web Accessibility Initiative (WAI) at www.w3.org/WAI.

Chapter 9 Review

Chapter Summary

After reading this chapter and completing the step-by-step tutorials and Try This! exercises, you should understand the following facts about creating accessible Web pages:

Defining Web Accessibility

- Web accessibility is the capability that results from the process of making it possible for users with special needs to receive, understand, and navigate content that people without handicaps can process in lieu of such special assistance.
- In the United States, Web accessibility is a right that is guaranteed by law under Section 508 of the Rehabilitation Act of 1973, as amended in 1998. Section 508 requires that all Web sites (as well as other forms of information technology) used, procured, developed, or maintained by government agencies and departments must be accessible.
- In 1997, the W3C launched the Web Accessibility Initiative (WAI), which coordinates the Web's official efforts to achieve accessibility. This initiative's Web Content Accessibility Guidelines (WCAG) consists of 65 checkpoints organized under 14 general guidelines. The three levels of conformance are called A, AA (pronounced double-A), and AAA (pronounced triple-A), respectively.
- The Section 508 accessibility standards do not contain as many checkpoints as the WCAG guidelines, and there are no levels of conformance. Instead, Section 508 includes 16 Web accessibility rules, all of which must be met in order for a Web site to be considered accessible. The Section 508 rules are online at www.section508.gov.

Coding to the Section 508 Web Accessibility Standards

- You must provide a textual equivalent for every nontext element onscreen. HTML has two attributes that you can use to create textual equivalents for nontext elements, namely, `alt` and `longdesc`.
- You may not use color to convey information that cannot be understood in the absence of color.
- You must not use a style sheet in such a way that it changes the meaning that the page would convey without the style sheet.
- To make an image map accessible, use the `alt` attribute to specify the alternate text for each area in the map. You can also use the `longdesc` attribute for any area in the map for which further explanation is required.
- Data tables must have clearly identified row and column headers, which you create via the `<th>` start and `</th>` stop tags. Tables that are used purely for layout, on the other hand, do not require header tags.
- In complex (i.e., nested) data tables, you must use the `headers` attribute to identify the header(s) that are associated with each data cell.
- When you create a frameset, you must give each frame a title that identifies the purpose and function of the frame. In addition to the `title` attribute, some assistive devices use the `name` attribute to identify the frames. You should therefore set both the `title` and `name` attributes for each frame.
- Scripts can make things happen onscreen that assistive devices cannot interpret. Whenever you have a script doing something that displays content or provides interface elements onscreen, you must use the `<noscript>` start and `</noscript>` stop tags to identify this information with functional text that can be read by assistive technology.
- To make forms accessible, you must use the `<label>` start and `</label>` stop tags to identify the ID of the `<input>` field with which the label is associated.
- When a Web page begins with repetitive navigation links, you must provide a way for the user to skip over them. You can accomplish this by creating skip navigation links.

Making Applets, Helpers, and Plug-Ins Accessible

- Flash is the most popular multimedia plug-in on the Web. A large percentage of the Web's multimedia content, therefore, gets handled by the Flash player. Guidelines for making Flash movies comply with Section 508 are at www.macromedia.com/macromedia/accessibility.
- Adobe's Portable Document Format (PDF) is the most popular file format for sharing documents over the Web. Guidelines for creating accessible PDFs are at www.adobe.com/products/acrobat/solutionsacc.html.
- The rich media accessibility showcase at ncam.wgbh.org/richmedia contains examples of accessible media delivered in the following formats: Director, Flash, PDF, QuickTime, Real Media, Scalable Vector Graphics, SMIL, Windows Media, and XHTML+SMIL.

Designing Style Sheets for Accessibility

- Instead of hard-coding presentation elements into the HTML of a Web page, it is better to separate content from style by assigning the HTML elements to style classes. At runtime, the browser looks to the style sheet for the rules that are associated with these classes. Thus, the style classes determine the look and feel of the HTML elements onscreen.
- Separating content from style in this manner has an important advantage for users with special needs. If the style sheets you provide do not handle the content appropriate to the user's needs, the user can use the browser's accessibility controls to add another style sheet onto the cascade. By being last on the cascade, this style sheet can redefine any of the style classes that the user needs to have interpreted differently.
- Instead of hard-coding fonts to a specific size, you should use relative sizes. The W3C has created two ways for you to do this. First, you can use the adjectives small, smaller, x-small, xx-small, medium, large, larger, x-large, and xx-large. Or you can specify the font size in a unit called an em, which is pronounced like the letter *m*. The setting 3em, for example, is three times the width of the letter *m*.
- Layout is one of the largest issues in making Web pages accessible. At the moment, most of the Web's pages use HTML tables to lay out the content onscreen. This works fine for sighted users, who can quickly move their gaze from place to place onscreen. Users with screen readers and other kinds of assistive devices, however, can run into trouble because the order of the content in the HTML file is not necessarily the order in which the user would want to read the material. You can solve this problem by using CSS to create the layout through absolute positioning, which can place the document's structural divisions at the desired locations onscreen, regardless of the order in which the divisions appear in the file.

Tools for Assessing Web Site Accessibility

- Bobby is a comprehensive Web accessibility tool that can analyze a single page or an entire Web site. Bobby exposes barriers to accessibility, makes recommendations for necessary repairs, and encourages compliance with existing guidelines. For a demonstration of Bobby in action, go to bobby.watchfire.com.
- LIFT is a suite of products that can test, monitor, report, and repair Web accessibility problems. Dreamweaver and FrontPage versions help authors create Web pages that are Section 508 or WCAG compliant. To try the LIFT tools before you buy, follow the links at www.usablenet.com.
- WebKing is a Web verification tool that integrates with IBM's WebSphere Studio Application Developer. WebKing performs (1) static analysis, (2) functional testing, and (3) load testing. As part of static analysis, WebKing checks your HTML and CSS code for Section 508 and WCAG accessibility. You can get a demonstration and download free evaluation software by following the link to WebKing at www.parasoft.com.
- STEP508 stands for Simple Tool for Error Prioritization (STEP) for Section 508 compliance. The STEP508 tool compares and analyzes the output of compliance tools, including Bobby, LIFT, and WebKing. Based on this analysis, STEP508 (1) determines the severity of the errors, (2) prioritizes the repairs needed to bring a Web site into compliance with Section 508, and (3) provides a metric to track progress in improving the site's accessibility over time. You can download STEP508 from www.section508.gov.

Key Terms

alt attribute *()*
Bobby *()*
** ** *()*
em *()*
for attribute *()*
headers attribute *()*
hertz (Hz.) *()*
id attribute *()*
<label> </label> *()*
LIFT *()*
linearization *()*
longdesc attribute *()*
MAGpie *()*
Microsoft Active Accessibility (MSAA) *()*
name attribute *()*
<noscript> </noscript> *()*
Section 508 *()*
Simple Tool for Error Prioritization (STEP) *()*
skip navigation link *()*
<th> </th> *()*
title attribute *()*
Web accessibility *()*
Web Accessibility Initiative (WAI) *()*
Web Content Accessibility Guidelines (WCAG) *()*
WebKing *()*

Key Terms Quiz

1. _____ is the capability that results from the process of making it possible for users with special needs to receive, understand, and navigate content that people without handicaps can process in lieu of such special assistance.
2. In the United States, Web accessibility is a right that is guaranteed by law under _____ of the Rehabilitation Act of 1973, as amended in 1998.
3. In 1997, the W3C launched the _____, which coordinates the Web's official efforts to achieve accessibility. This initiative's Web Content Accessibility Guidelines (WCAG) consists of 65 checkpoints organized under 14 general guidelines.
4. In HTML image tags, you create textual equivalents that are less than 150 characters via the _____. You should use the _____ for longer text descriptions.
5. Data tables must have clearly identified row and column headers, which you create via the <____> start and <____> stop tags.
6. In complex (i.e., nested) data tables, you must use the _____ to identify the header(s) that are associated with each data cell.
7. Whenever you have a script doing something that displays content or provides interface elements onscreen, you must use the <____> start and <____> stop tags to identify this information with functional text that can be read by assistive technology.
8. When a Web page begins with repetitive navigation links, you must provide a way for the user to skip over them. You can accomplish this by creating a(n) _____.
9. Available from watchfire.com, _____ is a Web accessibility tool that can analyze a single page or an entire Web site, expose barriers to accessibility, and make recommendations for necessary repairs in order to comply with accessibility guidelines.
10. In the STEP508 accessibility tool, STEP stands for _____.

Multiple-Choice Quiz

1. In the W3C's Web Content Accessibility Guidelines, what is a checkpoint that should be met, otherwise users will find it difficult, but not impossible, to access the material?
   a. Priority 1
   b. Priority 2
   c. Priority 3
   d. Priority 4
2. A Web site that passes all Priority 1, 2, and 3 checkpoints is entitled to display which W3C conformance logo?
   a. 

   b. 

   c. 

3. How many rules are there in the Section 508 Web accessibility standards?
   a. 14
   b. 16
   c. 65
   d. 508
4. What does the MAG in MAGpie stand for?
   a. Maximum accessibility guidelines
   b. Media access generator
   c. Multimedia accessibility group
   d. Multiple accessibility guidelines
5. About how many people are color blind?
   a. 0.5 percent of males and females
   b. 10 percent of males and females
   c. 10 percent of females and 0.5 percent of males
   d. 10 percent of males and 0.5 percent of females
6. Regardless of which style sheets the page may already call upon, the user can always add another style sheet by editing the browser's accessibility settings. Such a style sheet is always:
   a. First on the cascade
   b. Second on the cascade
   c. Third on the cascade
   d. Last on the cascade
7. Which frequency range of screen flicker is forbidden by the Section 508 guidelines?
   a. 0 to 1.5 Hz
   b. 2 to 55 Hz
   c. 55 to 110 Hz
   d. 110 to 440 Hz
8. Which HTML code(s) do you use to describe functional text information displayed onscreen by a script?
   a. `alt`
   b. `longdesc`
   c. `title`
   d. `<noscript> </noscript>`
9. When should you take advantage of Section 508 rule (k) and provide a text-only page to substitute for a Web page that you cannot bring into compliance with all of the other Section 508 rules?
   a. As often as possible
   b. Only as a last resort
   c. When you are in a hurry
   d. When you want an easy way out
10. Which key do you press to move from element to element as an alternative to clicking the desired element with a mouse?
    a. CTRL
    b. ESC
    c. ENTER
    d. TAB

Essay Quiz

1. The Section 508 Web site explains the relationship between the 16 Section 508 Web accessibility rules and the W3C's Web Content Accessibility Guidelines (WCAG). In your own words, describe how the Section 508 rules compare to the WCAG. If you have trouble finding this information, go to section508.gov, set the search option to search the section 508 Web site, and search for the keyword **WCAG**.
2. In the Macintosh operating system, the buttons for closing, minimizing, or maximizing a window follow a traffic light metaphor according to which these options are color-coded red, yellow, and green:

Ill 9-4 

When the user moves the mouse over these options, icons appear inside the buttons:

Ill 9-5 

In your opinion, does this color coding violate Section 508 rule (c), which forbids using color to convey information that cannot be understood in the absence of color? Explain the reasons why you feel Apple complied with or violated Section 508 rule (c).

3. Many users who cannot use a mouse rely on the keyboard to navigate from element to element on a Web page. In an HTML form, why should a Web author not program the ENTER key to submit the form's data?
4. What cascading style sheet rule can you use to hide from sighted users hyperlinks that you want users with screen readers to receive? To help answer this question, review and reflect on the section entitled "Skip Navigation" earlier in this chapter.
5. In your own words, explain why it is important to use heading styles when you word process a document for which you plan to create a PDF file. To help answer this question, reflect on the kind of navigational element that will be available to users with special needs as a result of the heading styles.

Lab Projects

• Lab Project 9-1: Designing Accessible Web Sites

Imagine that the government notified your school or company that federal funding will be discontinued in six months if you do not bring your Web site into compliance with the Section 508 Web accessibility rules. In response to this warning, your institution went through the process of bringing the site into compliance. Your employer is concerned, however, that the site will fall back out of compliance if continued work does not comply with accessibility guidelines. To address this concern, your employer has assigned you the job of creating an accessibility checklist that Web developers must follow at your site whenever they create or update site content. Use your word processor to create this checklist. In deciding what to include on the checklist, consider the following issues:

- **Textual Description for Nontext Elements** Section 508 requires that you must provide a textual description for every nontext element onscreen. Consider the kinds of nontext elements at your Web site, and create the appropriate checkpoints.

- **Forms** You must code forms in such a manner that users who do not have a mouse can use the computer keyboard to navigate, fill in, and submit the form. Be sure to include checkpoints for the `<label>` tags, `for` attributes, and Submit button that every form must have.
- **PDF Accessibility** Include a checkpoint that reminds your fellow employees to use heading styles when word processing documents that will be mounted at your site as PDF files. In the PDF file, these headings are tagged as bookmarks, which enable users with assistive devices to jump to different sections and subsections in the document.
- **Accessibility by Design** As you learned in this chapter, good Web design principles can obviate the need for some of the Section 508 rules. By requiring your authors to use client-side image maps, for example, you can avoid needing to create a checkpoint requiring redundant text links for server-side maps.

If your instructor asked you to hand in the checklist, make sure you put your name at the top of the document, then copy it onto a disk or follow the other instructions you may have been given for submitting this assignment.

• Lab Project 9-2: Evaluating Web Site Accessibility

Imagine that your employer has heard about Web accessibility tools that can automatically scan all of the pages at a Web site and report violations of Section 508 or WCAG accessibility guidelines. Your employer wants to get one of these tools to help ensure that the pages at your school or company Web site are compliant. The stakes are high because of the federal funding that your organization stands to lose if the site violates the Section 508 rules. Your employer has asked you to recommend which tool your organization should adopt for periodically scanning the site for accessibility violations. Use your word processor to write an essay in which you discuss the alternatives and recommend how your organization should go about evaluating Web site accessibility. In developing this recommendation, consider the following issues:

- **Alternative Tools** In this chapter, you learned about three tools for assessing the accessibility of a Web site—namely, Bobby, LIFT, and WebKing. There are about thirty other tools you can consider. The W3C keeps track of these tools at www.w3.org/WAI/ER/existingtools.html. Go there to read the summaries of what these tools do, and make a list of the tools you want to consider. In your essay, list the tools you considered, explain why you chose the tool you decided to recommend, and state the reasons why you rejected the others.
- **Authoring Environments** Some of the accessibility tools plug into work with certain Web development tools. In this chapter, for example, you learned that versions of LIFT for Dreamweaver and FrontPage are available, while WebKing plugs into IBM's WebSphere Studio Application Developer. Read the tool summaries at www.w3.org/WAI/ER/existingtools.html to find out whether other accessibility tools exist that plug into specific Web development environments. If you find plug-ins for the Web-authoring tools used at your site, include them on your list of alternatives to be considered.
- **Trial Versions** As you learned in this chapter, you can get trial versions of Bobby, LIFT, and WebKing. Find out whether trial versions exist for other alternatives you are considering. In your recommendation, consider proposing a trial period during which you test the tool before adopting it for production use at your Web site.
- **Prioritizing Errors** Go to www.section508.gov and read about the STEP508 tool. Consider whether STEP508 would be useful at your Web site. In your recommendation, state the reasons why you decided to include or forego STEP508 at this time.

If your instructor asked you to hand in the recommendation, make sure you put your name at the top of the essay, then copy it onto a disk or follow the other instructions you may have been given for submitting this assignment.