Microsoft



Step by Step





HTML and XHTML

- Learn at your own pace
- Build exactly the skills you need
- Practice with real-world examples

Faithe Wempen

Guide 2
Designing for Accessibility

Accessibility, a subset of usability, refers to a Web site's suitability for use by anyone, regardless of age or disability. Designing for accessibility is not only a nice thing to do, but a smart thing. An estimated 15 percent of the population of the United States has some form of disability, and as the Baby Boomer generation continues to age, that number will only increase. Nobody would intentionally alienate 15 percent of his or her potential audience, but that's exactly what creators of non-accessible Web sites do. A certain level of accessibility might even be required by law if your organization is required to comply with the Americans with Disabilities Act (ADA).

Note Many resources are available online to help Web designers make their sites more accessible. One of the best known is the W3C Web Accessibility Initiative, found at *http://www.w3.org/WAI*. On the WAI site you will find more complete coverage of each of the guidelines presented here, as well as a working draft for a newer version of these guidelines, Web Content Accessibility Guidelines (WCAG) 2.0.

If you have normal sight, vision, and mobility, perhaps you have never thought about the Web surfing challenges faced by people who have difficulty in any of those areas. Here are some of the most common accessibility issues:

- **Mobility limitations.**
  - Users might be limited to keyboard or mouse use only.
  - Users might be using voice recognition software to navigate.
- **Visual limitations.**
  - Users might have difficulty reading on-screen text, especially at its default size.
  - Users might be color-blind or have trouble reading colored text on a colored background.
  - Users might be relying on a program that reads the content of the page aloud.
- **Hearing limitations.**
  - Users might not hear music or narration being played.

To plan for these limitations, W3C has compiled a list of accessibility guidelines for Web designers to follow. The following sections summarize these guidelines; for more complete information about the guidelines, see *http://www.w3.org/TR/WCAG*.

Guideline 1: Provide equivalent alternatives to auditory and visual content.

Provide content that, when presented to the user, conveys essentially the same function or purpose as auditory or visual content.

You don't have to avoid graphics, audio clips, and video clips altogether; they add interest and excitement to your pages, and the majority of visitors can enjoy them. However, you should not deliver any content exclusively in those forms. Here are some ways to satisfy this guideline:

- Include an `alt=` argument for each picture, describing its content and purpose.
- For complex content where the description would be too long to display in an `alt=` argument, use an accompanying text note.
- Provide a transcript of audio and video clips. It doesn't have to be on the page itself; you could create a hyperlink that connects to a separate page containing the transcript.
- Use client-side image maps with `alt=` arguments for each area. Or, for a server-side image map, provide text hyperlink alternatives.
- In a visually based multimedia presentation, provide an audio track that reads or describes any essential information. Ensure that the audio is synchronized with the video.

Guideline 2: Don't rely on color alone.

Ensure that text and graphics are understandable when viewed without color.

Use color freely, but don't use it to convey information without providing an alternative method of conveying the same information. In addition, ensure that foreground and background colors contrast sufficiently so that someone with limited ability to distinguish colors (such as someone who is color-blind) can easily read the information provided.

Guideline 3: Use markup and style sheets, and do so properly.

Mark up documents by using the proper structural elements. Control presentation with style sheets rather than with presentation elements and attributes.

More Web designers have been moving toward using division-based layouts that separate the page's content from its formatting, as you learned in Chapter 13. This approach has many advantages, such as ease of making formatting changes, but one of the best benefits is greater accessibility. Accessibility experts recommend using only style sheet-based layout (that is, a layout with divisions), and not tables or frames. They maintain that tables must be used only for true tabular information, and frames should not be used at all.

Separating the content from the formatting has the side benefit of being able to offer different style sheets for the same content. In "old school" HTML, specific formatting was applied directly to each tag, limiting the way site visitors could modify it in their browsers. In HTML based on cascading style sheets, however, the content and the formatting are independent, so you can provide multiple style sheets and allow site visitors to choose among them by providing buttons that, when clicked, switch to a different version of the page. You might have a regular style sheet applied by default, for example, but also have one with extra-large fonts and high color contrast available for users who can benefit from that.

Here are the guidelines for ensuring that your code is accessible from a structural perspective:

- Use HTML tags and text rather than graphics wherever possible. For example, for a math formula, use text rather than a graphic of it.
- Use document type declarations at the beginning of the HTML file, as you learned to do in Chapter 2, and ensure that the type you declare is valid.
- Use style sheets rather than formatting tags to control layout and presentation.
- Use relative rather than absolute units of measurement when describing the formatting properties of an item or class. For example, you might use percentages rather than inches or centimeters to describe an item.
- Nest headings, starting with <h1> for the top-level headings, <h2> for headings within an H1 section, and so on. Do not choose a heading style simply because you like its default formatting; instead, use the next logical heading level and then format it in the style sheet to look like you want.
- Ensure that nested lists are properly marked. For example, if you have an <ol> within a <ul>, ensure you close the <ol> before you close the <ul>.
- Format quotations by using the <q> or <blockquote> tag, not simply by italicizing or indenting them.

Guideline 4: Clarify natural language usage.

Use markup that facilitates pronunciation or interpretation of abbreviated or foreign text.

When a visitor is using a screen reading program to read a page, the software that reads the text aloud can have difficulty reading foreign words and abbreviations.

Note *Markup* in this context means *HTML code*.

Sometimes such software can switch to a different mode if you alert it to the change in language by using the `lang=` argument. If there's no existing tag where the language changes, surround the word with a tag. You can also identify the primary natural language of the document in the opening <html> tag, but if the language is English, most reader software will assume it is even if you don't declare it.

You can use the <abbr> or <acronym> tag to mark an abbreviation or acronym. Even though Microsoft Internet Explorer does not support those tags directly, the screen reader recognizes them and signals their presence to the user. At the first usage of an abbreviation or acronym, you should spell out the full word or phrase, and use the shortened version only for subsequent occurrences on the same page.

Guideline 5: Create tables that transform gracefully.

Ensure that tables have necessary markup to be transformed by accessible browsers and other user agents.

This guideline states that tables should be used only for tabular information and not for layout because tables are difficult for screen reading software to interpret.

When you do use tables, it suggests using some additional tags that you didn't learn in this book to clarify the purposes of various cells. For example, use <td> for data cells, but use <th> for headers. In addition, for tables with two or more logical levels of row and column headers, use column groups to organize them.

If you do use tables for layout, ensure that the information would still make sense if the table tags were stripped out and the information was presented as plain text. Avoid using table elements strictly for visual formatting; for example, the <th> tag makes the text in a table cell centered and bold, but do not use <th> simply to achieve that formatting.

Guideline 6: Ensure that pages featuring new technologies transform gracefully.

Ensure that pages are accessible even when newer technologies are not supported or are turned off.

This guideline states that pages must not rely on new technologies, such as cascading style sheets, XML, JavaScript, Flash, Shockwave, and so on, to deliver their content. It's okay to use those techniques, as long as you provide alternatives, such as the following:

- Ensure that all pages are still readable when the style sheets are not available.
- Make text-only equivalents available for dynamic content, and ensure that the text is updated when the dynamic content changes.
- Ensure that pages still load even when scripts, applets, or other programmatic objects are turned off or not supported. If that's not possible, provide equivalent information on an alternative accessible page.

Guideline 7: Ensure user control of time-sensitive content changes.

Ensure that moving, blinking, scrolling, or auto-updating objects or pages may be paused or stopped.

This guideline states that whenever there is sound or movement on a page, the visitor should be able to control it. Here are some tips:

- Don't use background sounds that the visitors can't control. For example, don't use the <bgsound> tag, which you learned about in Chapter 16.
- Provide controls for all audio and video clips, so the visitor can pause, stop, and restart the clip.
- Avoid flickering, scrolling, or blinking elements. For example, do not use the blink or marquee elements (which are both non-standard and deprecated anyway).
- Don't allow pages to automatically refresh themselves unless there is a way for the visitor to stop the page from refreshing.
- If possible, do not use HTML to redirect pages automatically; instead configure the server to perform redirection.

Guideline 8: Ensure direct accessibility of embedded user interfaces.

Ensure that the user interface follows the principles of accessible design: device-independent access to functionality, keyboard operability, self-voicing, etc.

When an embedded object has its own interface, such as a Java applet that plays a game or performs a test, the interface must be accessible, just like the page itself. If this is not possible, provide an alternative, accessible page.

Guideline 9: Design for device independence.

Use features that enable activation of page elements through a variety of input devices.

Device independence means that visitors can interact with the page by using whatever input device they are most comfortable with: keyboard, mouse, voice, and so on. Someone with a movement-related disability might be limited to only one of those inputs.

Device independence can be an issue with non-text elements on a page, such as embedded user interfaces and image maps. Client-side image maps are better than server-side ones because they are easier to navigate without a mouse.

HTML forms can be made more device-independent by the use of keyboard shortcuts (`accesskey=` argument) and by setting a logical tab order for links, form controls, and objects. For example, you can add a `tabindex=` argument for each form control and set its value to a number representing the order in which the tab key should move a user through the fields.

Guideline 10: Use interim solutions.

Use interim accessibility solutions so that assistive technologies and older browsers will operate correctly.

User agents and other assistive technologies are being developed to enable users with disabilities to more easily view Web pages that employ the newest features, but until user agents are widely available to all visitors who need them, Web designers must be creative and employ interim accessibility solutions—basically, workarounds—ensuring that the pages are accessible to all.

Here are some tips for avoiding Web design elements that cause problems for many users:

- Don't cause pop-up windows or other windows to appear automatically. For example, avoid using a frame whose target is a new window.
- Don't change the current window without informing the user.

- For all form fields, ensure that the text label describing the field is positioned to the left of the field, so that a screen reader would first read the label, and then move on to the field immediately afterward. Do not position the field labels above the fields (in a previous row of a table, for example), or to the right of the field.
- Include place-holding characters in empty text areas and input form controls. (The most popular one is the non-breaking space: ` `.) Some older browsers do not allow users to navigate to empty edit boxes.
- Include non-link, printable characters between adjacent hyperlinks, surrounded by spaces. Some older screen readers read lists of consecutive lines as one link.

Guideline 11: Use W3C technologies and guidelines.

Use W3C technologies (according to specification), and follow accessibility guidelines. Where it is not possible to use a W3C technology, or doing so results in material that does not transform gracefully, provide an alternative version of the content that is accessible.

The current guidelines recommend the use of standardized HTML/XHTML coding wherever possible; that's the type of coding you've learned about in this book. Some non-W3C formats, such as PDF and Shockwave, require plug-ins or stand-alone external applications, and these formats sometimes can not be viewed or navigated easily with screen readers and other assistive technologies.

Guideline 12: Provide context and orientation information.

Provide context and orientation information to help users understand complex pages or elements.

When a page has a complex structure, it can be difficult for users to understand it using screen readers or other assistive technologies. Here are some ways to help:

- If you are using a frameset, ensure that each frame has a title. (Use the `title=` argument.)
- For each frame, if it is not obvious what the frame's purpose is and how it relates to the other frames, include a `longdesc=` argument containing that information.
- Divide blocks of information into manageable groups where natural and appropriate. For example, you can create option groups to organize options.
- Associate labels with form controls by using the `label=` argument.

Guideline 13: Provide clear navigation mechanisms.

Provide clear and consistent navigation mechanisms–orientation information, navigation bars, a site map, and so on–to increase the likelihood that a person will find what they are looking for at a site.

Throughout the book, I have encouraged you to use clear and consistent navigational aids, but these are especially critical for visitors with disabilities. Here are some tips for making your site easier to navigate:

- Ensure that each hyperlink's target is clearly identifiable. The underlined text in a hyperlink should describe the target page, not simply be an instruction such as "Click here".
- Keep hyperlink text brief–a few words at most.
- Provide metadata to add semantic information to pages and sites. For example, you can use the Resource Description Framework (RDF) to identify a document's author and content type. (For more information about RDF, see *http://www.w3.org/RDF*.)
- Provide a site map or table of contents. Include a description of the available accessibility features.
- Ensure that navigational elements are consistent among pages.
- Use navigation bars.
- Group related items together.
- If you provide a search function, enable different types of searches for different skill levels and preferences (for example, a basic search and an advanced search).
- Place descriptive information at the beginning of headings, paragraphs, lists, and so on.
- Provide a means of skipping over multi-line ASCII art.

Guideline 14: Ensure that documents are clear and simple.

Ensure that documents are clear and simple so they can be more easily understood.

This guideline is fairly self-explanatory: keep it simple. Use consistent page layout, recognizable graphics, and easy-to-understand language. All users appreciate this, not just those with disabilities. Use the clearest and simplest language possible, and supplement it with graphics or audio clips only when they help users understand the site better.