Core Techniques for Web Content Accessibility Guidelines 1.0

W3C Note 15 September 2000

This version:
http://www.w3.org/WAI/GL/NOTE-WCAG10-CORE-TECHS-20000915
(plain text, postscript, pdf, gzip tar file of HTML, zip archive of HTML)
Latest version:
http://www.w3.org/WAI/GL/WCAG10-CORE-TECHS
Previous version:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-TECHS-19990505
Editors:
Wendy Chisholm, W3C,
Gregg Vanderheiden, Trace R & D Center, University of Wisconsin -- Madison
Ian Jacobs, W3C

Abstract

This document describes general techniques that apply across technologies. i It is intended to help authors of Web content who wish to claim conformance to "Web Content Accessibility Guidelines 1.0" (*[WCAG10] [p. 18]*). While the techniques in this document should help people author Web content that conforms to WCAG 1.0, these techniques are neither guarantees of conformance nor the only way an author might produce conforming content.

This document is part of a series of documents that discuss techniques for a variety of technologies. For information on the rest of the series refer to "Techniques for Web Content Accessibility Guidelines 1.0" *[WCAG10-TECHS] [p. ??]* .

Status of this document

This is a Note in a series of Notes produced and endorsed by the Web Content Accessibility Guidelines Working Group (WCAG WG). This Note has not been reviewed or endorsed by W3C Members. The series of documents supersedes the single document 5 May 1999 W3C Note Techniques for Web Content Accessibility Guidelines 1.0. The topics from the earlier document have been separated into

technology-specific documents that may evolve independently. Smaller technology-specific documents allow authors to focus on a particular technology.

While the "Web Content Accessibility Guidelines 1.0" Recommendation *[WCAG10] [p. 18]* is a stable document, this series of companion documents is expected to evolve as technologies change and content developers discover more effective techniques for designing accessible Web content.

The history of changes to the series of documents as well as the list of open and closed issues are available. Readers are encouraged to comment on the document and propose resolutions to current issues. Please send detailed comments on this document to the Working Group at w3c-wai-gl@w3.org; public archives are available.

The list of known errors in this document is available at "Errata in Web Content Accessibility Guidelines." Please report errors in this document to wai-wcag-editor@w3.org.

The Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C) makes available a variety of resources on Web accessibility. WAI Accessibility Guidelines are produced as part of the WAI Technical Activity. The goals of the WCAG WG are described in the charter.

A list of current W3C Recommendations and other technical documents is available.

Table of Contents

1 Structure vs. Presentation

Checkpoints in this section:

- 2.1 Ensure that all information conveyed with color is also available without color, for example from context or markup. [Priority 1]
- 3.1 When an appropriate markup language exists, use markup rather than images to convey information. [Priority 2]
- 3.3 Use style sheets to control layout and presentation. [Priority 2]
- 3.5 Use header elements to convey document structure and use them according to specification. [Priority 2]
- 3.6 Mark up lists and list items properly. [Priority 2]
- 5.3 Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent (which may be a linearized version). [Priority 2]
- 5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting. [Priority 2]
- 6.4 For scripts and applets, ensure that event handlers are input device-independent. [Priority 2]

When designing a document or series of documents, content developers should strive first to identify the desired structure for their documents before thinking about how the documents will be presented to the user. Distinguishing the structure of a document from how the content is presented offers a number of advantages, including improved accessibility, manageability, and portability.

Identifying what is structure and what is presentation may be challenging at times. For instance, many content developers consider that a horizontal line communicates a structural division. This may be true for sighted users, but to unsighted users or users without graphical browsers, a horizontal line may have next to no meaning. For example, in HTML content developers should use the HTML 4.0 header elements (H1-H6) to identify new sections. These may be *complemented* by visual or other cues such as horizontal rules, but should not be replaced by them.

The inverse holds as well: content developers should not use structural elements to achieve presentation effects. For instance in HTML, even though the BLOCKQUOTE element may cause indented text in some browsers, it is designed to identify a quotation, not create a presentation side-effect. BLOCKQUOTE elements used for indentation confuse users and search robots alike, who expect the element to be used to mark up block quotations.

The separation of presentation from structure in XML documents is inherent. As Norman Walsh states in "A Guide to XML" *[WALSH] [p. 18]* ,

> HTML browsers are largely hardcoded. A first level heading appears the way it does because the browser recognizes the H1 tag. Again, since XML documents have no fixed tag set, this approach will not work. The presentation of an XML document is dependent on a stylesheet.

Quicktest! To determine if content is structural or presentational, create an outline of your document. Each point in the hierarchy denotes a structural change. Use structural markup to mark these changes and presentational markup to make them more apparent visually and aurally. Notice that horizontal rules will not appear in this outline and therefore are not structural, but presentational. **Note.** This quicktest addresses chapter, section, and paragraph structure. To determine structure within phrases, look for abbreviations, changes in natural language, definitions, and list items.

2 Text equivalents

Checkpoints in this section:

- 1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). *This includes*: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video. [Priority 1]
- 1.2 Provide redundant text links for each active region of a server-side image map. [Priority 1]
- 1.5 Until user agents render text equivalents for client-side image map links, provide redundant text links for each active region of a client-side image map. [Priority 3]
- 12.2 Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone. [Priority 2]

Text is considered accessible to almost all users since it may be handled by screen readers, non-visual browsers, and braille readers. It may be displayed visually, magnified, synchronized with a video to create a caption, etc. As you design a document containing non-textual information (images, applets, sounds, multimedia presentations, etc.), supplement that information with textual equivalents wherever possible.

When a text equivalent is presented to the user, it fulfills essentially the same function (to the extent possible) as the original content. For simple content, a text equivalent may need only describe the function or purpose of content. For complex content (charts, graphs, etc.), the text equivalent may be longer and include descriptive information.

Text equivalents must be provided for logos, photos, submit buttons, applets, bullets in lists, ascii art, and all of the links within an image map as well as invisible images used to lay out a page.

Quicktest! A good test to determine if a text equivalent is useful is to imagine reading the document aloud over the telephone. What would you say upon encountering this image to make the page comprehensible to the listener?

2.1 Overview of technologies

How one specifies a text equivalent depends on the document language.

For example, depending on the element, HTML allows content developers to specify text equivalents through attributes (" alt" or "longdesc") or in element content (the OBJECT element).

Video formats, such as Quicktime, will allow developers to include a variety of alternative audio and video tracks. SMIL (*[SMIL] [p. 18]*) allows developers to synchronize alternative audio and video clips, and text files with each other.

In creating XML DTDs, ensure that elements that might need a description have some way of associating themselves with the description.

Some image formats allow internal text in the data file along with the image information. If an image format supports such text (e.g., Portable Network Graphics, see *[PNG] [p. 18]*) content developers may also supply information there as well.

2.2 Backward Compatibility

Content developers must consider backward compatibility when designing Web pages or sites since:

- Some user agents do not support some HTML features,
- People may use older browsers or video players,
- Compatibility problems may arise between software

Therefore, when designing for older technologies, consider these techniques:

- Provide inline text equivalents. For example, include a description of the image immediately after the image.
- Provide links to long text equivalents either in a different file or on the same page. These are called description links or "d-links". The link text should explain that the link designates a description. Where possible, it should also explain the nature of the description. However, content developers concerned about how the description link will affect the visual appearance of the page may use more discrete link text such as "[D]", which is recommended by NCAM (refer to *[NCAM] [p. 19]*). In this case, they should also provide more information about the link target so that users can distinguish links that share "[D]" as content (e.g., with the "title" attribute in HTML).

3 Alternative pages

Checkpoints in this section:

- 11.4 If, after best efforts, you cannot create an accessible page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent information (or functionality), and is updated as often as the inaccessible

(original) page. [Priority 1]
- 6.5 Ensure that dynamic content is accessible or provide an alternative presentation or page. [Priority 2]

Although it is possible to make most content accessible, it may happen that all or part of a page remains inaccessible. Additional techniques for creating accessible alternatives include:

1. Allow users to navigate to a separate page that is accessible, contains the same information as the inaccessible page, and is maintained with the same frequency as the inaccessible page.
2. Instead of static alternative pages, set up server-side scripts that generate accessible versions of a page on demand.
3. Refer to the examples for Frames and Scripts.
4. Provide a phone number, fax number, e-mail, or postal address where information is available and accessible, preferably 24 hours a day

Here are two techniques for linking to an accessible alternative page:

1. Provide links at the top of both the main and alternative pages to allow a user to move back and forth between them. For example, at the top of a graphical page include a link to the text-only page, and at the top of a text-only page include a link to the associated graphical page. Ensure that these links are one of the first that users will tab to by placing them at the top of the page, before other links.
2. Use meta information to designate alternative documents. Browsers should load the alternative page automatically based on the user's browser type and preferences.

Checkpoints in this section:

- 9.2 Ensure that any element that has its own interface can be operated in a device-independent manner. [Priority 2]
- 9.3 For scripts, specify logical event handlers rather than device-dependent event handlers. [Priority 2]
- 9.4 Create a logical tab order through links, form controls, and objects. [Priority 3]
- 9.5 Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls. [Priority 3]

Not every user has a graphic environment with a mouse or other pointing device. Some users rely on keyboard, alternative keyboard or voice input to navigate links, activate form controls, etc. Content developers must ensure that users may interact with a page with devices other than a pointing device. A page designed for keyboard access (in addition to mouse access) will generally be accessible to users with other input devices. What's more, designing a page for keyboard access will usually improve its overall design as well.

Keyboard access to links and form controls may be specified in a few ways:

Image map links
: Provide text equivalents for client-side image map areas, or provide redundant text links for server-side image maps. Refer to the image map section for examples.

Keyboard shortcuts
: Provide keyboard shortcuts so that users may combine keystrokes to navigate links or form controls on a page. **Note.** Keyboard shortcuts -- notably the key used to activate the shortcut -- may be handled differently by different operating systems. On Windows machines, the "alt" and "ctrl" key are most commonly used while on a Macintosh, it is the apple or "clover leaf" key. Refer to the Keyboard access for links and Keyboard Access to Forms sections for examples.

Tabbing order
: Tabbing order describes a (logical) order for navigating from link to link or form control to form control (usually by pressing the "tab" key, hence the name). Refer to the Keyboard Access to Forms section for examples.

3.1 Device-independent control for embedded interfaces

Some elements import objects (e.g., applets or multimedia players) whose interfaces cannot be controlled through the markup language. In such cases, content developers should provide alternative equivalents with accessible interfaces if the imported objects themselves do not provide accessible interfaces.

4 Navigation

Checkpoints in this section:

- 14.3 Create a style of presentation that is consistent across pages. [Priority 3]
- 13.4 Use navigation mechanisms in a consistent manner. [Priority 2]
- 13.5 Provide navigation bars to highlight and give access to the navigation mechanism. [Priority 3]
- 13.3 Provide information about the general layout of a site (e.g., a site map or table of contents). [Priority 2]
- 13.7 If search functions are provided, enable different types of searches for different skill levels and preferences. [Priority 3]
- 13.2 Provide metadata to add semantic information to pages and sites. [Priority 2]

A consistent style of presentation on each page allows users to locate navigation mechanisms more easily but also to skip navigation mechanisms more easily to find important content. This helps people with learning and reading disabilities but also makes navigation easier for all users. Predictability will increase the likelihood that people will find information at your site, or avoid it when they so desire.

Examples of structures that may appear at the same place between pages:

1. navigation bars
2. the primary content of a page
3. advertising

A navigation mechanism creates a set of paths a user may take through your site. Providing navigation bars, site maps, and search features all increase the likelihood that a user will reach the information they seek at your site. If your site is highly visual in nature, the structure might be harder to navigate if the user can't form a mental map of where they are going or where they have been. To help them, content developers should describe any navigation mechanisms. It is crucial that the descriptions and site guides be accessible since people who are lost at your site will rely heavily on them.

When providing search functionality, content developers should offer search mechanisms that satisfy varying skill levels and preferences. Most search facilities require the user to enter keywords for search terms. Users with spelling disabilities and users unfamiliar with the language of your site will have a difficult time finding what they need if the search requires perfect spelling. Search engines might include a spell checker, offer "best guess" alternatives, query-by-example searches, similarity searches, etc.

5 Comprehension

Checkpoints in this section:

- 14.1 Use the clearest and simplest language appropriate for a site's content. [Priority 1]
- 13.8 Place distinguishing information at the beginning of headings, paragraphs, lists, etc. [Priority 3]
- 14.2 Supplement text with graphic or auditory presentations where they will facilitate comprehension of the page. [Priority 3]

The following sections discuss techniques for helping comprehension of a page or site.

5.1 Writing style

The following writing style suggestions should help make the content of your site easier to read for everyone, especially people with reading and/or cognitive disabilities. Several guides (including *[HACKER] [p. 18]*) discuss these and other writing style issues in more detail.

1. Strive for clear and accurate headings and link descriptions. This includes using link phrases that are terse and that make sense when read out of context or as part of a series of links (Some users browse by jumping from link to link and listening only to link text.) Use informative headers so that users can scan a

page quickly for information rather than reading it in detail.
2. State the topic of the sentence or paragraph at the beginning of the sentence or paragraph (this is called "front-loading"). This will help both people who are skimming visually, but also people who use speech synthesizers. "Skimming" with speech currently means that the user jumps from heading to heading, or paragraph to paragraph and listens to just enough words to determine whether the current chunk of information (heading, paragraph, link, etc.) interests them. If the main idea of the paragraph is in the middle or at the end, speech users may have to listen to most of the document before finding what they want. Depending on what the user is looking for and how much they know about the topic, search features may also help users locate content more quickly.
3. Limit each paragraph to one main idea.
4. Avoid slang, jargon, and specialized meanings of familiar words, unless defined within your document.
5. Favor words that are commonly used. For example, use "begin" rather than "commence" or use "try" rather than "endeavor."
6. Use active rather than passive verbs.
7. Avoid complex sentence structures.

To help determine whether your document is easy to read, consider using the Gunning-Fog reading measure (described in *[SPOOL] [p. 18]* with examples and the algorithm online at *[TECHHEAD] [p. 19]*). This algorithm generally produces a lower score when content is easier to read. As example results, the Bible, Shakespeare, Mark Twain, and TV Guide all have Fog indexes of about 6. Time, Newsweek, and the Wall St. Journal an average Fog index of about 11.

5.2 Multimedia equivalents

For people who do not read well or not at all, multimedia (non-text) equivalents may help facilitate comprehension. Beware that multimedia presentations do not **always** make text easier to understand. Sometimes, multimedia presentations may make it more confusing.

Examples of multimedia that supplement text:

1. A chart of complex data, such as sales figures of a business for the past fiscal year.
2. A translation of the text into a Sign Language movie clip. Sign Language is a very different language than spoken languages. For example, some people who may communicate via American Sign Language may not be able to read American English.
3. Pre-recorded audio of music, spoken language, or sound effects may also help non-readers who can perceive audio presentations. Although text may be generated as speech through speech synthesis, changes in a recorded speaker’s voice can convey information that is lost through synthesis.

6 Content negotiation

Checkpoints in this section:

- 11.3 Provide information so that users may receive documents according to their preferences (e.g., language, content type, etc.) [Priority 3]

There are a variety of strategies to allow users to select the appropriate content:

1. Include links to other versions of content, such as translations. For example, the link "Refer to the French version of this document" links to the French version.
2. Indicate content type or language through markup (e.g., in HTML use "type" and "hreflang").
3. Use content negotiation to serve content per the client request. For example, serve the French version of a document to clients requesting French.

7 Automatic page refresh

Checkpoints in this section:

- 7.4 Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages. [Priority 2]
- 7.5 Until user agents provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects. [Priority 2]

Content developers sometimes create pages that refresh or change without the user requesting the refresh. This automatic refresh can be very disorienting to some users. Instead, in order of preference, authors should:

1. Configure the server to use the appropriate HTTP status code (301). Using HTTP headers is preferable because it reduces Internet traffic and download times, it may be applied to non-HTML documents, and it may be used by agents who requested only a HEAD request (e.g., link checkers). Also, status codes of the 30x type provide information such as "moved permanently" or "moved temporarily" that cannot be given with META refresh.
2. Replace the page that would be redirected with a static page containing a normal link to the new page.

Note. Both checkpoint 7.4 and checkpoint 7.5 address problems posed by legacy user agents. Newer user agents should disable refresh and substitute a link to new information at the top of the page.

Deprecated examples are provided in the HTML Techniques document. A link will be provided in the future.

8 Screen flicker

Checkpoints in this section:

- 7.1 Until user agents allow users to control flickering, avoid causing the screen to flicker. [Priority 1]

A flickering or flashing screen may cause seizures in users with photosensitive epilepsy and content developers should thus avoid causing the screen to flicker. Seizures can be triggered by flickering or flashing in the 4 to 59 flashes per second (Hertz) range with a peak sensitivity at 20 flashes per second as well as quick changes from dark to light (like strobe lights).

9 Bundled documents

Checkpoints in this section:

- 13.9 Provide information about document collections (i.e., documents comprising multiple pages.). [Priority 3]

Bundled documents can facilitate reading offline. To create a coherent package:

- Use metadata to describe the relationships between components of the package (refer to link metadata for HTML).
- Use archiving tools such as zip, tar and gzip, and stuffit to create the package.

10 Validation

This section discusses strategies and techniques for testing Web documents to determine accessibility issues that have been resolved and those that haven't. These tests should highlight major access issues, are valuable in reducing a number of accessibility barriers. However, some of these testing scenarios only replicate conditions caused by a disability; they do not simulate the full experience a user with a disability might have. In real-life settings, your pages may be less usable than you expected. Thus, one of the strategies recommends that content developers observe people with different disabilities as they attempt to use a page or site.

If, after completing the following tests and adjusting your design accordingly, you find that a page is still not accessible, it is likely that you should create an alternative page [p. 6] that is accessible.

Note. Passing these tests does not guarantee conformance to the "Web Content Accessibility Guidelines 1.0".

10.1 Automatic validators

A validator can verify the syntax of your pages (e.g., HTML, CSS, XML). Correct syntax will help eliminate a number of accessibility problems since software can process well-formed documents more easily. Also, some validators can warn you of some accessibility problems based on syntax alone (e.g., a document is missing an attribute or property that is important to accessibility). Note, however, that correct syntax does not guarantee that a document will be accessible. For instance, you may provide a text equivalent for an image according to the language's specification, but the text may be inaccurate or insufficient. Some validators will therefore ask you questions and step you through more subjective parts of the analysis. Some examples of automatic validators include:

1. An automated accessibility validation tool such as Bobby (refer to *[BOBBY] [p. 19]*).
2. An HTML validation service such as the W3C HTML Validation Service (refer to *[HTMLVAL] [p. 19]*).
3. A style sheets validation service such as the W3C CSS Validation Service (refer to *[CSSVAL] [p. 19]*).

10.2 Repair tools

Validators usually report what issues to solve and often give examples of how to solve them. They do not usually help an author walk through each problem and help the author modify the document interactively. The WAI Evaluation and Repair Working Group (*[WAI-ER] [p. 19]*) is working to develop a suite of tools that will help authors not only identify issues but solve them interactively.

10.3 User scenarios

Keep in mind that most user agents (browsers) and operating systems allow users to configure settings that change the way software looks, sounds, and behaves. With the variety of user agents, different users will have very different experiences with the Web. Therefore:

1. Test your pages with a text-only browser such as Lynx (*[LYNX] [p. 19]*) or a Lynx emulator such as Lynx Viewer (*[LYNXVIEW] [p. 19]*) or Lynx-me (*[LYNXME] [p. 19]*).
2. Use multiple graphic browsers, with:
   - sounds and images loaded,
   - images not loaded,
   - sounds not loaded,
   - no mouse,
   - frames, scripts, style sheets, and applets not loaded.
3. Use several browsers, old and new. **Note.** Some operating systems or browsers do not allow multiple installations of the browser on the same machine. It may also be difficult to locate older browser software.

4. Use other tools such as a self-voicing browser (e.g., *[PWWEBSPEAK] [p. 19]* and *[HOMEPAGEREADER] [p. 19]*), a screen reader (e.g., *[JAWS] [p. 19]* and *[WINVISION] [p. 19]*), magnification software, a small display, an onscreen keyboard, an alternative keyboard, etc. **Note.** If a Web site is usable with one of these products it does not ensure that the site will be usable by other products. For a more detailed list of assistive technologies used to access the Web refer to (*[ALTBROWSERS] [p. 19]*).

10.4 Spell and grammar checks

A person reading a page with a speech synthesizer may not be able to decipher the synthesizer's best guess for a word with a spelling error. Grammar checkers will help to ensure that the textual content of your page is correct. This will help readers for whom your document is not written in their native tongue, or people who are just learning the language of the document. Thus, you will help increase the comprehension of your page.

11 Browser Support

Note. *At the time of this writing, not all user agents support some of the (new) HTML 4.0 attributes and elements that may significantly increase accessibility of Web pages.*

Please refer to the W3C Web site (*[WAI-UA-SUPPORT] [p. 19]*) for information about browser and other user agent support of accessibility features.

In general, please note that HTML user agents ignore attributes they don't support and they render the content of unsupported elements.

12 Accessibility Reviewed Technologies

Checkpoints in this section:

- 11.1 Use W3C technologies when they are available and appropriate for a task and use the latest versions when supported. [Priority 2]

WCAG 1.0 suggests using W3C Technologies since they have been reviewed for accessibility issues and therefore have accessibility features built-in. The latest W3C technologies are available from the W3C Technical Reports and Publications page.

Brief overview of current W3C technologies:

- MathML for mathematical equations
- HTML, XHTML, XML for structured documents
- RDF for meta data
- SMIL to create multimedia presentations
- CSS and XSL to define style sheets
- XSLT to create style transformations

- PNG for graphics (although some are best expressed in JPG, a non-w3c spec)
- Are there others that we need to discuss? WebCGM?

13 Audio and Video

14 Audio information

Checkpoints in this section:

- 1.4 For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation. [Priority 1]
- 6.5 Ensure that dynamic content is accessible or provide an alternative presentation or page. [Priority 2]

Auditory presentations must be accompanied by *text transcripts*, textual equivalents of auditory events. When these transcripts are presented synchronously with a video presentation they are called *captions* and are used by people who cannot hear the audio track of the video material.

Some media formats (e.g., QuickTime 3.0 and SMIL) allow captions and video descriptions to be added to the multimedia clip. SAMI allows captions to be added. The following example demonstrates that captions should include speech as well as other sounds in the environment that help viewers understand what is going on.

Example.

Captions for a scene from "E.T." The phone rings three times, then is answered.

[phone rings]

[ring]

[ring]

Hello?"

End example.

Until the format you are using supports alternative tracks, two versions of the movie could be made available, one with captions and descriptive video, and one without. Some technologies, such as SMIL and SAMI, allow separate audio/visual files to be combined with text files via a synchronization file to create captioned audio and movies.

Some technologies also allow the user to choose from multiple sets of captions to match their reading skills. For more information see the SMIL 1.0 (*[SMIL] [p. 18]*) specification.

Equivalents for sounds can be provided in the form of a text phrase on the page that links to a text transcript or description of the sound file. The link to the transcript should appear in a highly visible location such as at the top of the page. However, if a script is automatically loading a sound, it should also be able to automatically load a visual indication that the sound is currently being played and provide a description or transcript of the sound.

Note. Some controversy surrounds this technique because the browser should load the visual form of the information instead of the auditory form if the user preferences are set to do so. However, strategies must also work with today's browsers.

For more information, please refer to *[NCAM] [p. 19]* .

15 Visual information and motion

Checkpoints in this section:

- 1.3 Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation. [Priority 1]
- 7.1 Until user agents allow users to control flickering, avoid causing the screen to flicker. [Priority 1]
- 7.3 Until user agents allow users to freeze moving content, avoid movement in pages. [Priority 2]

Auditory descriptions of the visual track provide narration of the key visual elements without interfering with the audio or dialogue of a movie. Key visual elements include actions, settings, body language, graphics, and displayed text. Auditory descriptions are used primarily by people who are blind to follow the action and other non-auditory information in video material.

Example.

Here's an example of a collated text transcript [p. 17] of a clip from "The Lion King" (available at *[DVS] [p. 19]*). Note that the Describer is providing the auditory description of the video track and that the description has been integrated into the transcript.

> Simba: Yeah!
>
> Describer: Simba races outside, followed by his parents. Sarabi smiles and nudges Simba gently toward his father. The two sit side-by-side, watching the golden sunrise.
>
> Mufasa: Look Simba, everything the light touches is our kingdom.
>
> Simba: Wow.

End example.

Note. If there is no important visual information, for example, an animated talking head that describes (through prerecorded speech) how to use the site, then an auditory description is not necessary.

For movies, provide auditory descriptions that are synchronized with the original audio. Refer to the section on audio information [p. 15] for more information about multimedia formats.

16 Collated text transcripts

Collated text transcripts allow access by people with both visual and hearing disabilities. They also provide everyone with the ability to index and search for information contained in audio/visual materials.

Collated text transcripts include spoken dialogue as well as any other significant sounds including on-screen and off-screen sounds, music, laughter, applause, etc. In other words, all of the text that appears in captions as well as all of the descriptions provided in the auditory description.

---

17 References

For the latest version of any W3C specification please consult the list of W3C Technical Reports at http://www.w3.org/TR.

[PNG]
"PNG (Portable Network Graphics) Specification", T. Boutell, ed., T. Lane, contributing ed., 1 October 1996. The latest version of PNG 1.0 is available at http://www.w3.org/TR/REC-png.

[SMIL]
"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", P. Hoschka, ed., 15 June 1998. This SMIL 1.0 Recommendation is http://www.w3.org/TR/1998/REC-smil-19980615. The latest version of SMIL 1.0 is available at http://www.w3.org/TR/REC-smil.

[WCAG10]
"Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds., 5 May 1999. This WCAG 1.0 Recommendation is http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505.

18 Resources

Note: *W3C does not guarantee the stability of any of the following references outside of its control. These references are included for convenience. References to products are not endorsements of those products.*

18.1 Other guidelines

[HACKER]
Hacker, Diana. (1993). A Pocket Style Manual. St. Martin’s Press, Inc. 175 Fifth Avenue, New York, NY 10010.

[SPOOL]
Spool, J.M., Sconlong, T., Schroeder, W., Snyder, C., DeAngelo, T. (1997). Web Site Usability: A Designer’s Guide User Interface Engineering, 800 Turnpike St, Suite 101, North Andover, MA 01845.

[UWSAG]
"The Unified Web Site Accessibility Guidelines", G. Vanderheiden, W. Chisholm, eds. The Unified Web Site Guidelines were compiled by the Trace R & D Center at the University of Wisconsin under funding from the National Institute on Disability and Rehabilitation Research (NIDRR), U.S. Dept. of Education.

[WALSH]
Walsh, Norman. (1997). "A Guide to XML." In "XML: Principles, Tools, and Techniques." Dan Connolly, Ed. O’Reilly & Associates, 101 Morris St, Sebastopol, CA 95472. pp 97-107.

18.2 User agents and other tools

A list of alternative Web browsers (assistive technologies and other user agents designed for accessibility) is maintained at the WAI Web site.

[ALTBROWSERS]
"Alternative Web Browsing". This page documents known support by user agents (including assistive technologies) of some accessibility features listed in this document. The page is available at http://www.w3.org/WAI/References/Browsing.

[BOBBY]
Bobby is an automatic accessibility validation tool developed by Cast.

[CSSVAL]
The W3C CSS Validation Service.

[HOMEPAGEREADER]
IBM's Home Page Reader.

[HTMLVAL]
The W3C HTML Validation Service.

[JAWS]
Henter-Joyce's Jaws screen reader.

[LYNX]
Lynx is a text-only browser.

[LYNXME]
Lynx-me is a Lynx emulator.

[LYNXVIEW]
Lynx Viewer is a Lynx emulator.

[PWWEBSPEAK]
The Productivity Works' pwWebSpeak.

[WAI-UA-SUPPORT]
User Agent Support for Accessibility

[WINVISION]
Artic's WinVision.

18.3 Accessibility resources

[DVS]
DVS Descriptive Video Services.

[NCAM]
The National Center for Accessible Media includes information about captioning and audio description on the Web.

[TECHHEAD]
Tech Head provides some information about the Fog index described in [SPOOL] [p. 18] .

[WAI-ER]
The WAI Evaluation and Repair Working Group

19 Acknowledgments

Web Content Guidelines Working Group Co-Chairs:

Jason White, University of Melbourne

Gregg Vanderheiden, Trace Research and Development

W3C Team contact:

Wendy Chisholm

We wish to thank the following people who have contributed their time and valuable comments to shaping these guidelines:

Harvey Bingham, Kevin Carey, Chetz Colwell, Neal Ewers, Geoff Freed, Al Gilman, Larry Goldberg, Jon Gunderson, Eric Hansen, Phill Jenkins, Leonard Kasday, George Kerscher, Marja-Riitta Koivunen, Josh Krieger, Chuck Letourneau, Scott Luebking, William Loughborough, Murray Maloney, Charles McCathieNevile, MegaZone (Livingston Enterprises), Masafumi Nakane, Mark Novak, Charles Oppermann, Mike Paciello, David Pawson, Michael Pieper, Greg Rosmaita, Liam Quinn, Dave Raggett, T.V. Raman, Robert Savellis, Jutta Treviranus, Steve Tyler, and Jaap van Lelieveld

The original draft of this document is based on "The Unified Web Site Accessibility Guidelines" *[UWSAG] [p. 18]* compiled by the Trace R & D Center at the University of Wisconsin. That document includes a list of additional contributors.